Complementation of Emerson-Lei Automata

Vojtěch Havlena, Ondřej Lengál, and Barbora Šmahlíková

Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic

Abstract. We give new constructions for complementing subclasses of Emerson-Lei automata using modifications of rank-based Büchi automata complementation. In particular, we propose a specialized rank-based construction for a Boolean combination of Inf acceptance conditions, which heavily relies on a novel way of a run DAG labelling enhancing the ranking functions with models of the acceptance condition. Moreover, we propose a technique for complementing generalized Rabin automata, which are structurally as concise as general Emerson-Lei automata (but can have a larger acceptance condition). The construction is modular in the sense that it extends a given complementation algorithm for a condition φ in a way that the resulting procedure handles conditions of the form $\mathsf{Fin} \wedge \varphi$. The proposed constructions give upper bounds that are exponentially better than the state of the art for some of the classes.

1 Introduction

Complementation of ω-automata is an important operation in formal verification with various applications, for example in model checking wrt expressive temporal logics such as QPTL [25] or HyperLTL [10]; testing language inclusion of ω-automata, or in decision procedures of various logics [6,21]. For Büchi automata (BAs)—i.e., ω-automata with the simplest acceptance condition—complementation has been, from the theoretical point of view, thoroughly explored, starting with constructions having the $2^{2^{O(n)}}$ state complexity [6] coming down to constructions asymptotically matching the lower bound $\Omega((0.76n)^n)$ (modulo a polynomial factor) [38,1]. Over the years, ω-automata with more complex acceptance conditions (such as generalized Büchi (GBAs), (generalized) Rabin/Streett, parity) have found uses in practice. The most general acceptance condition used is the so-called *Emerson-Lei* condition [11], which is an arbitrary Boolean formula consisting of Fin and Inf atoms. $\mathsf{Fin}(c)$ denotes that all transitions labeled with c must occur only finitely often in an accepting run and $\mathsf{Inf}(c)$ denotes that there must be a transition labeled with c occurring infinitely often. There are two main reasons for using more complex acceptance conditions: (i) more compact representation of automata and (ii) the ability to determinize (deterministic BAs are strictly less expressive than BAs).

From the theoretical point of view, precise bounds on complementation of automata with more complex acceptance condition is much less researched, demonstrated by the best upper bound for (transition-based) Emerson-Lei automata (TELAs) being $2^{2^{O(n)}}$ [37] states. Here, the O in the exponent can hide a linear (or constant) factor, which would have a doubly-exponential effect, giving little information about the actual complexity. In this paper, we present complementation algorithms for several subclasses of TELAs and thoroughly study their complexity, giving better upper bounds than the currently-best known algorithms.

Our contributions can be summarized as follows:

1. We propose a rank-based complementation algorithm for Inf-TELAs, i.e., TELAs where the acceptance condition does not contain any Fin atom, with the state complexity $O(n(0.76nk)^n)$ where n is the number of states and k is the number of *minimal models* of the acceptance condition.
2. By instantiating the previously mentioned algorithm, we obtain a complementation algorithm for generalized Büchi automata with k colours constructing a BA with the state complexity $O(n(0.76nk)^n)$, which is, to the best of our knowledge, better than the best previously known algorithms.
3. We propose a modular procedure for complementing TELAs with the acceptance condition $\mathsf{Fin}(c) \wedge \varphi$ given a compatible complementation procedure for formula φ.
4. Next, we instantiate the modular procedure to handle Rabin pairs ($\mathsf{Fin}(0) \wedge \mathsf{Inf}(1)$) and, in turn, obtain an algorithm for complementing Rabin automata with k Rabin pairs with the complexity $O(n^k(0.76n)^{nk})$, which is, again, better than any other algorithm that we know of.
5. Finally, we instantiate the procedure also for generalized Rabin pairs ($\mathsf{Fin}(0) \wedge \mathsf{Inf}(1) \wedge \ldots \wedge \mathsf{Inf}(\ell)$) and obtain complementation constructions for generalized Rabin automata and TELAs with the upper bound $O(n^{2^k}(0.76nk)^{n^{2^k}})$, which is the best upper bound for complementation of general TELAs that we are aware of.

An extended version of the paper with missing proofs can be found at [20].

2 Preliminaries

We fix a finite non-empty alphabet Σ and the first infinite ordinal ω. For $k \in \omega$, we use $\llbracket k \rrbracket$ to represent the largest even number less than or equal to k, e.g., $\llbracket 43 \rrbracket = \llbracket 42 \rrbracket = 42$. An (infinite) word w is a function $w : \omega \to \Sigma$ where the i-th symbol is denoted as w_i. Sometimes, we represent w as an infinite sequence $w = w_0 w_1 \ldots$ We denote the set of all infinite words over Σ as Σ^ω; an *ω-language* is a subset of Σ^ω. We use $\cdot$ for ellipsis, e.g., if interested only in the second component of a triple, we may write the triple as $(\cdot, x, \cdot)$.

2.1 Emerson-Lei Acceptance Conditions

Given a set $\Gamma = \{0, \ldots, k-1\}$ of k *colours* (often depicted as 0, 1, etc.), we define the set of *Emerson-Lei acceptance conditions* $\mathbb{EL}(\Gamma)$ as the set of formulae constructed according to the following grammar:

$$\alpha ::= tt \mid ff \mid \mathsf{Inf}(c) \mid \mathsf{Fin}(c) \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha)$$

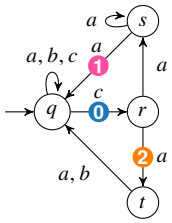
for $c \in \Gamma$. The *satisfaction* relation $\models$ for a set of colours $M \subseteq \Gamma$ and a condition α is defined inductively as follows (for $c \in \Gamma$):

$$M \models tt, \quad M \models \mathsf{Fin}(c) \text{ iff } c \notin M, \quad M \models \alpha_1 \vee \alpha_2 \text{ iff } M \models \alpha_1 \text{ or } M \models \alpha_2,$$
$$M \not\models ff, \quad M \models \mathsf{Inf}(c) \text{ iff } c \in M, \quad M \models \alpha_1 \wedge \alpha_2 \text{ iff } M \models \alpha_1 \text{ and } M \models \alpha_2.$$

If $M \models \alpha$, we say that M is a *model* of α We denote by $|\alpha|$ the number of atomic conditions contained in α, where multiple occurrences of the same atomic condition are counted multiple times.

2.2 Emerson-Lei Automata

A (nondeterministic) transition-based[1] *Emerson-Lei automaton* (TELA) over Σ is a tuple $\mathcal{A} = (Q, \delta, I, \Gamma, \mathsf{p}, \mathsf{Acc})$, where Q is a finite set of *states* (we often use n to denote $|Q|$), $\delta \subseteq Q \times \Sigma \times Q$ is a set of *transitions*[2], $I \subseteq Q$ is the set of *initial* states, Γ is the set of *colours*, $\mathsf{p}\colon \delta \to 2^\Gamma$ is a *colouring* of transitions, and $\mathsf{Acc} \in \mathbb{EL}(\Gamma)$. We use $p \xrightarrow{a} q$ to denote that $(p, a, q) \in \delta$ and sometimes treat δ as a function $\delta\colon Q \times \Sigma \to 2^Q$. Moreover, we extend δ to sets of states $P \subseteq Q$ as $\delta(P, a) = \bigcup_{p \in P} \delta(p, a)$. See Fig. 1 for an example TELA $\mathcal{A}_{ex}$ over $\Sigma = \{a, b, c\}$ with 3 colours $\Gamma = \{0, 1, 2\}$ and the acceptance condition $\mathsf{Inf}(0) \wedge \mathsf{Inf}(1)$. We define $|\mathcal{A}| = |Q|$.



Inf(0) ∧ Inf(1)

Fig. 1: $\mathcal{A}_{ex}$

A *run* of $\mathcal{A}$ from $q \in Q$ on an input word w is an infinite sequence $\rho\colon \omega \to Q$ that starts in q and respects δ, i.e., $\rho(0) = q$ and $\forall i \geq 0\colon \rho(i) \xrightarrow{w_i} \rho(i{+}1) \in \delta$. Let $\mathit{inf}(\rho) \subseteq \delta$ denote the set of transitions occurring in ρ infinitely often and $\mathit{inf}_\Gamma(\rho) = \bigcup\{\mathsf{p}(x) \mid x \in \mathit{inf}(\rho)\}$ be the set of infinitely often occurring colours. A run ρ is *accepting* wrt an acceptance condition α, written as $\rho \models \alpha$, iff $\mathit{inf}_\Gamma(\rho) \models \alpha$ and ρ is accepting in $\mathcal{A}$ iff $\rho \models \mathsf{Acc}$. The *language* of $\mathcal{A}$, denoted as $\mathcal{L}(\mathcal{A})$, is defined as the set of words $w \in \Sigma^\omega$ for which there exists an accepting run in $\mathcal{A}$ starting with some state in I. Classical acceptance conditions can be in this more general framework described as follows (we only provide those used later in the paper and include their abbreviations):

- *Büchi* (BA): $\mathsf{Acc} = \mathsf{Inf}(0)$,
- *co-Büchi* (CBA): $\mathsf{Acc} = \mathsf{Fin}(0)$,
- *Generalized Büchi* (GBA): $\mathsf{Acc} = \bigwedge_{0 \leq j < k} \mathsf{Inf}(j)$,
- *Generalized co-Büchi* (GCBA): $\mathsf{Acc} = \bigvee_{0 \leq j < k} \mathsf{Fin}(j)$,
- *Rabin*: $\bigvee_{0 \leq j < k} \mathsf{Fin}(B_j) \wedge \mathsf{Inf}(G_j)$,
- *Generalized Rabin*: $\bigvee_{0 \leq j < k} (\mathsf{Fin}(B_j) \wedge \bigwedge_{0 \leq \ell < m_j} \mathsf{Inf}(G_{j,\ell}))$, and
- *Parity*[3]: $\mathsf{Fin}(0) \wedge (\mathsf{Inf}(1) \vee (\mathsf{Fin}(2) \wedge (\mathsf{Inf}(3) \vee (\mathsf{Fin}(4) \wedge \ldots))))$,

where $B_j, G_j, G_{j,\ell} \in \Gamma$ for all j, ℓ. Further, we use Inf-TELA to denote a TELA where the acceptance condition contains no Fin atoms. We also use the syntactic sugar $\mathcal{A} = (Q, \delta, I, F)$ to denote a (transition-based) BA that would be defined using the TELA definition above as $(Q, \delta, I, \{0\}, \{t \mapsto \emptyset \mid t \in \delta \setminus F\} \cup \{t \mapsto \{0\} \mid t \in F\}, \mathsf{Inf}(0))$.

2.3 Run DAGs

In this section, we recall the terminology from [19] (which is a minor modification of the terminology from [26] and [38]) used heavily in the paper. Let $\mathcal{A} = (Q, \delta, I, \Gamma, \mathsf{p}, \mathsf{Acc})$ be a TELA. We fix the definition of the *run DAG* of $\mathcal{A}$ over a word w to be a DAG (directed acyclic graph) $\mathcal{G}_w = (V, E)$ of vertices V and edges E where

[1] We only consider transition-based acceptance in order to avoid cluttering the paper by dealing with accepting states *and* accepting transitions. Extending our approach to state/transition-based (or just state-based) automata is straightforward.

[2] Note that there is also a more general definition of TELAs with $\delta \subseteq Q \times \Sigma \times 2^\Gamma \times Q$; in this paper, we use the simpler one.

[3] We consider the so-called *parity min odd* condition; any parity condition from the set $\{\min, \max\} \times \{\text{even}, \text{odd}\}$ can be easily translated to it.

– $V \subseteq Q \times \omega$ s.t. $(q,i) \in V$ iff there is a run ρ of $\mathcal{A}$ from I over w with $\rho_i = q$,
– $E \subseteq V \times V$ s.t. $((q,i),(q',i')) \in E$ iff $i' = i+1$ and $q' \in \delta(q, w_i)$.

See Fig. 2 for an example of a run DAG of $\mathcal{A}_{ex}$ from Fig. 1 over the word $caa(cab)^\omega \notin \mathcal{L}(\mathcal{A}_{ex})$ (we will return to the additional labels in the figure later). Given a DAG $\mathcal{G} = (V, E)$, we often identify $\mathcal{G}$ with V, for instance, we will write $(p,i) \in \mathcal{G}$ to denote that $(p,i) \in V$. For a vertex $v \in \mathcal{G}$, we denote the set of vertices of $\mathcal{G}$ reachable from v (including v itself) as $reach_{\mathcal{G}}(v)$ or just $reach(v)$ if $\mathcal{G}$ is clear from the context. A vertex $v \in \mathcal{G}$ is *finite* iff $reach(v)$ is finite and *infinite* if it is not finite. In Fig. 2, the vertices $(s,2),(s,3),(s,5),\ldots$ are finite and all other vertices are infinite. Moreover, for a colour $c \in \Gamma$, an edge $((q,i),(q',i+1)) \in E$ is a c-edge if $c \in \mathsf{p}(q \xrightarrow{w_i} q')$ and a vertex $v \in V$ is c-*endangered* iff it cannot reach any c-edge. For a set of colours $C \subseteq \Gamma$, v is *C-endangered* iff it is c-endangered for every $c \in C$. For example, in Fig. 2, the vertices $(q,1)$ and $(t,2)$ are $\{1\}$-endangered but they are not $\{0,1\}$-endangered. A pair of vertices $v_1, v_2 \in V$ is *converging* iff $reach(v_1) \cap reach(v_2) \neq \emptyset$ (v_1 and v_2 *converge*). A function $r : V \to \omega$ is a *run DAG ranking* if for every $v \in V$ it holds that $\forall u \in reach(v) : r(u) \leq r(v)$. We use $\max(r)$ to denote the *rank* of r, i.e., the maximum value from $\{r(u) \mid u \in V\}$ if it exists and ∞ otherwise. A ranking r of $\mathcal{G}$ is called *tight* iff there exists a level i such that (i) $m = \max\{r((q,i)) \mid q \in Q\}$ is odd and (ii) for all levels $j \geq i$ it holds that $\{1,3,\ldots,m\} \subseteq \{r((q,j)) \mid q \in Q\}$.

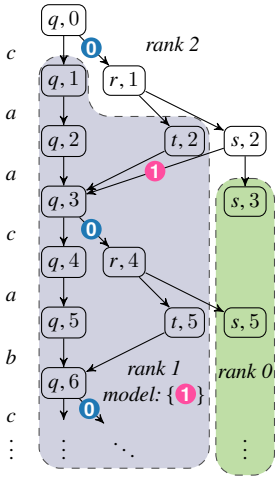


Fig. 2: A labelled run DAG of $\mathcal{A}_{ex}$ over the word $caa(cab)^\omega \notin \mathcal{L}(\mathcal{A}_{ex})$

3 Complementation of Inf-TELAs

In this section, we describe a complement construction for Inf-TELAs. Our approach is an extension of rank-based BA complementation algorithms [26,14,38,24,9,16,19], which construct a BA whose runs simulate a run DAG ranking procedure. We start with giving the run DAG ranking procedure (which extends the ranking procedure from [26] with the introduction of models) and then proceed to the complementation algorithm itself. One can see our algorithm also as an improvement of the algorithm for complementing GBAs in [28] by (i) introducing model assignments, (ii) getting better complexity through the use of tight rankings, and (iii) generalizing the construction from GBAs to arbitrary Inf-TELAs.

3.1 Inf-TELA Run DAG Labelling

Let $\mathcal{A} = (Q, \delta, I, \Gamma, \mathsf{p}, \mathsf{Acc})$ be an Inf-TELA. We use $\overline{\mathsf{Acc}}$ to denote the propositional formula obtained from Acc by replacing conjunctions by disjunctions and vice versa, and substituting atoms of the form Inf(j) by j (this can be viewed as negating Acc, transforming it into the negation normal form, substituting $\neg$Inf(j) by

Fin(j), and denoting each Fin(j) just by j). Let $\mathcal{M}_{\overline{\mathsf{Acc}}}$ be the set of models of $\overline{\mathsf{Acc}}$ where the colours j are interpreted as propositional variables. For example, if $\mathsf{Acc} = \mathrm{Inf}(0) \wedge (\mathrm{Inf}(1) \vee \mathrm{Inf}(2))$, then $\overline{\mathsf{Acc}} = 0 \vee (1 \wedge 2)$ and $\mathcal{M}_{\overline{\mathsf{Acc}}} = \{\{0\}, \{1,2\}, \{0,1\}, \{0,2\}, \{0,1,2\}\}$ ($\mathcal{M}_{\overline{\mathsf{Acc}}}$ can be interpreted as saying which combinations of Inf-conditions need to be broken in order to break Acc; in the example above, we can, e.g., break Inf(0), we can break both Inf(1) and Inf(2), etc.). Furthermore, we use $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ to denote the set of *minimal models* of $\overline{\mathsf{Acc}}$, i.e., $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ is the set where (i) for every model $m \in \mathcal{M}_{\overline{\mathsf{Acc}}}$, there exists a model $m' \in \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ such that $m' \subseteq m$, and (ii) there are no $m, m' \in \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ such that $m \subset m'$. We note that $\mathcal{M}_{\overline{\mathsf{Acc}}}$ can be obtained as the upward closure of $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ (and $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ is an *antichain*). For the example acceptance condition above, $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}} = \{\{0\}, \{1,2\}\}$. Moreover, we use **lex-min**($\overline{\mathsf{Acc}}$) to denote the lexicographically smallest model from $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ (w.l.o.g., we assume $\mathcal{M}_{\overline{\mathsf{Acc}}} \neq \emptyset$). **lex-min**($\overline{\mathsf{Acc}}$) is used to pinpoint one model when any model can be used.

Let $\mathcal{G} = (V, E)$ be a run DAG of $\mathcal{A}$ over w. For a set of vertices $U \subseteq V$, a mapping $\eta \colon U \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ is called *endangered in* $\mathcal{G}$ if

1. U is finite and nonempty,
2. each $v \in U$ is $\eta(v)$-endangered in $\mathcal{G}$, and
3. for each pair of vertices $v_1, v_2 \in U$ converging in $\mathcal{G}$, we have $\eta(v_1) = \eta(v_2)$.

A function m with the signature $m \colon V \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ is called a *model assignment*. For instance, for $\mathcal{A}_{ex}$ in Fig. 1, we have $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}} = \{\{0\}, \{1\}\}$ since $\mathcal{A}_{ex}$ is a GBA. In addition, for the run DAG in Fig. 2 and a set $\{(q,1), (t,2)\}$, the mapping $\{(q,1) \mapsto \{1\}, (t,2) \mapsto \{1\}\}$ is endangered in $\mathcal{G}$. On the other hand, there exists no endangered mapping for the set $\{(s,2)\}$ in $\mathcal{G}$, as $(s,2)$ can reach both a 0-edge as well as a 1-edge.

In Algorithm 1, we give a (nondeterministic) ranking procedure that assigns ranks and minimal models of $\overline{\mathsf{Acc}}$ to each vertex of $\mathcal{G}$. Intuitively, the algorithm starts by giving all initially finite vertices the rank 0 and assigning their model to **lex-min**($\overline{\mathsf{Acc}}$) (Line 4). Then, it proceeds in iterations, each starting with the DAG $\mathcal{G}^i$ and consisting of two steps:

1. First, the algorithm tries to find a model assignment $\eta \colon U \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ for a finite nonempty set of vertices U of $\mathcal{G}^i$ s.t. for all $u \in U$, if $\eta(u) = \{c_1, \ldots, c_\ell\}$, then every path in $\mathcal{G}^i$ starting in u satisfies the condition $\bigwedge_{1 \leq j \leq \ell} \mathrm{Fin}(c_j)$ (the path breaks all the Inf(c_j) conditions, i.e., η is endangered). If such a model assignment exists (the choice is nondeterministic), the algorithm assigns rank $i+1$ to all vertices reachable from U and removes them from the DAG, creating DAG $\mathcal{G}^{i+1}$ (Lines 7–9).
2. Second, the algorithm assigns rank $i+2$ to all vertices in $\mathcal{G}^{i+1}$ that became finite (after the previous step) and removes them from the DAG, creating DAG $\mathcal{G}^{i+2}$ (Lines 10–12). The counter i is incremented by two and the next iteration continues.

The iterations end when $\mathcal{G}^i$ is empty or when no suitable model assignment η is found (which happens when w is accepted by $\mathcal{A}$). Note that due to the nondeterminism within

Algorithm 1: Inf-TELA run DAG labelling

Input: A run DAG $\mathcal{G}_w$ of $\mathcal{A}$ over w, acceptance condition Acc
Output: A run DAG ranking r and a model assignment m if $w \notin \mathcal{L}(\mathcal{A})$, else $\bot$

1 $i \leftarrow 0, r \leftarrow \emptyset, m \leftarrow \emptyset$; // $i \in \omega$, $r\colon V \rightharpoonup \{0, \ldots, 2|Q|\}$, $m\colon V \rightharpoonup \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$
2 $\mathcal{G}^0 = (V^0, E^0) \leftarrow \mathcal{G}_w$ without finite vertices;
3 **foreach** $v \in \mathcal{G}_w$ *s.t.* v *is finite* **do**
4 $\quad r(v) \leftarrow 0, m(v) \leftarrow$ **lex-min**$(\overline{\mathsf{Acc}})$;
5 **while** $\mathcal{G}^i \neq \emptyset$ **do**
6 $\quad$ **if** $\exists(\eta\colon U \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}})$ *s.t.* $U \subseteq V^i$ *and* η *is endangered in* $\mathcal{G}^i$ **then**
7 $\quad\quad$ **foreach** $v \in U$ *and* $u \in reach_{\mathcal{G}^i}(v)$ **do**
8 $\quad\quad\quad r(u) \leftarrow i+1, m(u) \leftarrow \eta(v)$;
9 $\quad\quad \mathcal{G}^{i+1} \leftarrow \mathcal{G}^i$ without vertices with the rank $i+1$;
10 $\quad\quad$ **foreach** $v \in \mathcal{G}^{i+1}$ *s.t.* v *is finite in* $\mathcal{G}^{i+1}$ **do**
11 $\quad\quad\quad r(v) \leftarrow i+2, m(v) \leftarrow$ **lex-min**$(\overline{\mathsf{Acc}})$;
12 $\quad\quad \mathcal{G}^{i+2} \leftarrow \mathcal{G}^{i+1}$ without vertices with the rank $i+2$;
13 $\quad\quad i \leftarrow i+2$;
14 $\quad$ **else**
15 $\quad\quad$ **return** $\bot$;
16 **return** (r, m);

the algorithm, it may be possible to obtain, in two different runs of the algorithm on the same run DAG, two different pairs (r_1, m_1) and (r_2, m_2) with $\max(r_1) \neq \max(r_2)$.

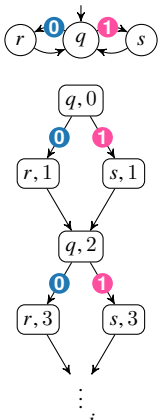
Example 1. See Fig. 2 for a possible labelling of the run DAG of $\mathcal{A}_{ex}$ over the word $caa(cab)^\omega$. The ranking procedure proceeds in the following steps:

1. ($i = 0$) First, all finite vertices, which are in this example vertices of the form $(s, 3)$, $(s, 5), \ldots, (s, 3j+2)$ for all $1 \leq j$, are assigned rank 0 and model **lex-min**$(\overline{\mathsf{Acc}})$, and $\mathcal{G}^0$ is set to be $\mathcal{G}^w$ without those vertices. (Lines 2–4)
2. Second, we set η_1 to the mapping $\eta_1 = \{(q,1) \mapsto \{❶\}, (t,2) \mapsto \{❶\}\}$. The mapping η_1 is endangered in $\mathcal{G}^0$ because the following conditions hold:
   (a) η_1 is finite and nonempty,
   (b) neither $(q,1)$ nor $(t,2)$ can reach a ❶ transition, and
   (c) $(q,1)$ and $(t,2)$ converge (in $(q,3)$) and they are both assigned the same model $(\eta_1((q,1)) = \eta_1((t,2)) = \{❶\})$.
   In particular, η_1 is the endangered mapping that gives the largest number of vertices of $\mathcal{G}^0$ rank 1. (Line 6)
3. Third, we assign every vertex in $\mathcal{G}^0$ reachable from $(q,1)$ or $(t,2)$ the rank 1 and model $\{❶\}$. (Line 7)
4. Fourth, we obtain $\mathcal{G}^1$ from $\mathcal{G}^0$ by removing vertices with rank 1. (Line 9)
5. $\mathcal{G}^1$ contains three vertices $(\{(q,0), (r,1), (s,2)\})$, which all get rank 2 (Line 10) and are removed in $\mathcal{G}^2$ (Line 12). The ranking procedure finishes. □

Lemma 2. *If Algorithm 1 returns* $\bot$*, then* $w \in \mathcal{L}(\mathcal{A})$.

Proof. Let Acc' be a formula in the disjunctive normal form (DNF) equivalent to Acc, i.e., $\mathsf{Acc}' = \bigvee_{j=1}^{\ell} \varphi_j$ where $\varphi_j = \mathrm{Inf}(c_1^j) \wedge \cdots \wedge \mathrm{Inf}(c_{k_j}^j)$ for some ℓ and $k_1, \ldots, k_\ell$. Note that $\mathcal{M}_{\overline{\mathsf{Acc}}}^{\min} = \mathcal{M}_{\overline{\mathsf{Acc}'}}^{\min}$ contains sets of colours $M \subseteq \Gamma$, each of them with at least one colour from φ_1, at least one colour from φ_2, etc. In order for Algorithm 1 to return $\bot$, it needs to hold that there is some $i \geq 0$ such that $\mathcal{G}^i$ is nonempty and there does not exist any mapping $\eta\colon U \to \mathcal{M}_{\overline{\mathsf{Acc}}}^{\min}$, with $U \subseteq V^i$, that would be endangered in $\mathcal{G}^i$. In particular, such a (nonempty) mapping η does not exist iff no vertex $v \in \mathcal{G}^i$ satisfies point (2) of the definition of an endangered mapping (i.e., when we can find an accepting path from all vertices remaining in $\mathcal{G}^i$). Therefore, it follows that no vertex $v \in \mathcal{G}^i$ is M-endangered for any $M \in \mathcal{M}_{\overline{\mathsf{Acc}}}^{\min}$, i.e., in other words,

> for every vertex $v \in \mathcal{G}^i$ there is some clause φ_j such that v can in $\mathcal{G}^i$ reach a c_p^j-edge for each $1 \leq p \leq k_j$. (Reach)



We will now construct an accepting path π in $\mathcal{G}_w$. Note that not all paths in $\mathcal{G}^i$ are necessarily accepting (consider the TELA over a unary alphabet and the run DAG in the right, with the acceptance condition $\mathrm{Inf}(0) \wedge \mathrm{Inf}(1)$; there are many non-accepting paths from $(q, 0)$—e.g., a path that alternates between a q-vertex and an r-vertex and never touches any s-vertex). While constructing π, for every clause φ_j we will be tracking the information about which atom of φ_j we should see next in order to satisfy φ_j on the path. In particular, we will start from a vertex v_0 that is a root vertex of $\mathcal{G}^i$ and we will use the tuple $t_0 = (c_1^1, \ldots, c_1^\ell)$ to keep track of the colours. Using (Reach), it follows that there is a clause φ_j s.t. v_0 can reach a c_1^j-edge e_1. We will set $t_1 = (c_1^1, \ldots, c_2^j, \ldots, c_1^\ell)$ and continue in a similar way: from every vertex we encounter, we use (Reach) to obtain an edge that is a c-edge for some c in t_i. In the case we need to increment some component of t_i from $c_{k_j}^j$, we set the new value to c_1^j. The path π is then constructed as an infinite path that goes through the infinite sequence $v_0, e_1, e_2, \ldots$ Note that because the sequence $v_0, e_1, e_2 \ldots$ is infinite, due to the pigeonhole principle there will be a clause φ_j s.t. the sequence $t_0, t_1, \ldots$ infinitely often increments the j-th component and so π is accepting. From π, we can now extract the accepting run of $\mathcal{A}$ on w. □

Lemma 3. *Algorithm 1 always terminates with $i \leq 2n$.*

Proof. Consider a run DAG $\mathcal{G}_w$ for a word w. First observe that at the end of the main loop of Algorithm 1 (Line 13), $\mathcal{G}^i$ has no finite vertices (all of them were removed). Due to Line 2, $\mathcal{G}^i$ at the beginning of the main loop (Line 6) also has no finite vertices. Let $\mathcal{G}_m^i$ be the DAG $(V_m^i, E^i \cap (V_m^i \times V_m^i))$ where $V_m^i = \{(q, j) \in V^i \mid j \geq m\}$, i.e., the projection of $\mathcal{G}^i$ from level m down, and $width(\mathcal{G}_m^i)$ be the maximum number of vertices on any level of the run DAG below level m, formally, $width(\mathcal{G}_m^i) = \max\{|\{(q, j) : (q, j) \in V_m^i\}| : j \geq m\}$. From the definition of endangered mapping and the loop on Line 7, we have that if the condition on Line 6 holds, there is some $m \in \omega$ s.t. $width(\mathcal{G}_m^{i+1}) < width(\mathcal{G}_m^i)$. This holds because if the mapping η is non-empty, then

there is at least one infinite path in $\mathcal{G}^i$ that is completely removed in the next step, i.e., from some level m, the width of all levels below get decreased by at least one. If the condition on Line 6 does not hold, the algorithm terminates and we are done. From the previous claim we have that in each successful iteration of the main loop, the width of $\mathcal{G}^{i+2}$ in the limit is at most the one of $\mathcal{G}^i$ minus one. Since the maximum width of $\mathcal{G}_w$ is n, then, if $w \notin \mathcal{L}(\mathcal{A})$, at latest $\mathcal{G}_m^{2n-1}$ is empty for some $m \in \omega$, and hence $\mathcal{G}^{2n}$ is empty and the algorithm terminates. □

Lemma 4. *If $w \in \mathcal{L}(\mathcal{A})$, then Algorithm 1 terminates with $\bot$.*

Proof. Consider some $w \in \mathcal{L}(\mathcal{A})$. Then, there is an accepting run ρ on w in $\mathcal{A}$. We have $(\rho_j, j) \in \mathcal{G}_w$ for all $j \in \omega$; we show that (ρ_j, j) is not M-endangered for every $M \in \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$. The fact that no ρ_j is finite follows from the fact that ρ is infinite. Observe that for each $M \in \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$, there is some $c \in M$ s.t. $c \in \mathrm{Inf}(\rho)$ (otherwise, w would not be accepted by $\mathcal{A}$). Therefore, (ρ_j, j) is not M-endangered. Hence, in every iteration of Algorithm 1, all vertices (ρ_j, j) stay in $\mathcal{G}^i$. From Lemma 3 we have that Algorithm 1 always terminates, but $\mathcal{G}^i \neq \emptyset$ for each i. Therefore, the algorithm terminates with $\bot$. □

Corollary 5. *$w \notin \mathcal{L}(\mathcal{A})$ iff Algorithm 1 on $\mathcal{G}_w$ terminates with (r, m).*

The following lemma about the ranking procedure will be useful later.

Lemma 6. *If Algorithm 1 terminates with (r, m), then $\max(r) \leq 2n$ and, moreover, either $\max(r) = 0$ or r is tight.*

Proof. The first part ($\max(r) \leq 2n$) follows directly from Lemma 3. For the second part, there are two options: either $\mathcal{G}_w$ is finite (i.e., there is no infinite run of $\mathcal{A}$ on w), in which case Algorithm 1 assigns all vertices in $\mathcal{G}_w$ rank 0 and does not even enter the loop at Line 5. In the other case ($\mathcal{G}$ is infinite), let $k = \max(r)$ if $\max(r)$ is odd and $k = \max(r) - 1$ otherwise (from the previous case, we know that $k \geq 1$). We know that for every $\ell \in \{1, 3, \ldots, k\}$, there is a vertex $v_\ell = (q_\ell, i_\ell) \in \mathcal{G}_w$ with $r(v_\ell) = \ell$ (this is because the mapping at Line 6 in the algorithm needs to be non-empty) and that such a vertex is the beginning of an infinite path of vertices with rank ℓ. Therefore, there needs to be a level i containing vertices with all ranks $\{1, 3, \ldots, k\}$. From the previous, all levels $j > i$ will also have all of the odd ranks up to k. Choosing i large enough will prevent level i having a vertex with an even rank higher than k. Therefore, r is tight. □

3.2 Inf-TELA Complement Construction

Let $\mathcal{A} = (Q, \delta, I, \Gamma, \mathsf{p}, \mathsf{Acc})$ be an Inf-TELA and $n = |Q|$. We define a *(level) ranking* to be a function $f : Q \to \{0, \ldots, 2n\}$. The *rank* of f is defined as $f = \max\{f(q) \mid q \in Q\}$. We call a mapping $\mu : Q \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ a *level model*. We say that μ is *consistent* wrt f if $\mu(q) = \textbf{lex-min}(\overline{\mathsf{Acc}})$ whenever $f(q)$ is even. We denote the set of all level models by LM. For a set of states $S \subseteq Q$ and a level model μ, f is (S, μ)-*tight* if

(i) it has an odd rank r,
(ii) $f(S) \supseteq \{1, 3, \ldots, r\}$,
(iii) $f(Q \setminus S) = \{0\}$, and
(iv) μ is consistent wrt f.

A ranking is μ-*tight* if it is (Q, μ)-tight; we use $\mathcal{T}$ to denote the set of all μ-tight rankings for some level model μ.

For two level rankings f, f' and two level models μ, μ', we say that (f', μ') is a *consistent successor* of (f, μ) over $a \in \Sigma$, denoted as $(f, \mu) \overset{a}{\underset{\delta}{\multimap}} (f', \mu')$, iff

(i) μ and μ' are consistent wrt f and f', respectively, and
(ii) for all $q \in Q$ and $q' \in \delta(q, a)$ the following holds:
  (a) $f'(q') \leq f(q)$,
  (b) $(\mathsf{p}(q \xrightarrow{a} q') \cap \mu(q) \neq \emptyset) \Rightarrow f'(q') \leq \lfloor\!\lfloor f(q) \rfloor\!\rfloor$, and
  (c) $\mu'(q') \neq \mu(q) \Rightarrow f'(q') \leq \lfloor\!\lfloor f(q) \rfloor\!\rfloor$.

Intuitively, the rankings guess the ranks of states in the run DAG and the level models guess the models assigned to states in the labelling procedure from Section 3.1. Consistent successors respect the labelling procedure. On every path in a run DAG, the ranks are nonincreasing. If some odd-ranked vertex v has an outgoing c-edge to v' and c is in the model assigned to v, the vertex v' has to have a lower rank than v, because when v is removed from $\mathcal{G}_w^i$, there is no reachable c-edge in $\mathcal{G}_w^i$. Moreover, if the model is changed between v and v' and the rank is odd, then the rank also has to be decreased.

The complement of $\mathcal{A}$ is given as the BA $\textsc{CInfTela}(\mathcal{A}) = (Q', \delta', I', F')$ whose components are defined as follows:

- $Q' = Q_1 \cup Q_2$ where
  - $Q_1 = 2^Q$ and
  - $Q_2 = \{(S, O, f, i, \mu) \in 2^Q \times 2^Q \times \mathcal{T} \times \{0, 2, \ldots, 2n-2\} \times \mathsf{LM} \mid f \text{ is } (S, \mu)\text{-tight}, O \subseteq S \cap f^{-1}(i)\}$,
- $I' = \{I\}$,
- $\delta' = \delta_1 \cup \delta_2 \cup \delta_3$ where
  - $\delta_1 \colon Q_1 \times \Sigma \to 2^{Q_1}$ such that $\delta_1(S, a) = \{\delta(S, a)\}$,
  - $\delta_2 \colon Q_1 \times \Sigma \to 2^{Q_2}$ s.t. $\delta_2(S, a) = \{(S', \emptyset, f, 0, \mu) \mid S' = \delta(S, a)\}$, and
  - $\delta_3 \colon Q_2 \times \Sigma \to 2^{Q_2}$ such that $(S', O', f', i', \mu') \in \delta_3((S, O, f, i, \mu), a)$ iff
    - $S' = \delta(S, a)$,
    - $(f, \mu) \overset{a}{\underset{\delta}{\multimap}} (f', \mu')$,
    - $rank(f) = rank(f')$,
    - and
      - $i' = (i + 2) \mod (rank(f') + 1)$ and $O' = f'^{-1}(i')$ if $O = \emptyset$ or
      - $i' = i$ and $O' = \delta(O, a) \cap f'^{-1}(i)$ if $O \neq \emptyset$, and
- $F' = \{\emptyset \xrightarrow{a} \emptyset \in \delta_1 \mid a \in \Sigma\} \cup \{M_1 \xrightarrow{a} M_2 \in \delta_3 \mid M_1 = (\cdot, \emptyset, \cdot, \cdot, \cdot), a \in \Sigma\}$

Intuitively, a run of $\textsc{CInfTela}(\mathcal{A})$ on a word w tries to construct the run DAG $\mathcal{G}_w$ of $\mathcal{A}$ on the same word, with rankings encoded within the states. The restrictions on δ_3 encode the rules from Algorithm 1. The partitioning of Q' into Q_1 and Q_2 allows us to consider only tight rankings, as in [14]. Moreover, the i-component of a macrostate allows us to further decrease the number of states in the same way as in [38] (we know that all states in O have the same rank i).

Theorem 7. *Let $\mathcal{A}$ be an Inf-TELA. Then, $\mathcal{L}(\textsc{CInfTela}(\mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.*

Proof. ($\subseteq$) We use Boolean laws and prove an equivalent statement $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega \setminus \mathcal{L}(\textsc{CInfTela}(\mathcal{A}))$. Let $w \in \mathcal{L}(\mathcal{A})$ be a word and ρ be an accepting run of $\mathcal{A}$ on w. First, let ρ' be the run $\rho' = S_0 S_1 \dots$ with $S_0 = I$ and $S_{i+1} = \delta_1(S_i, w(i))$ for all $i \in \omega$ (i.e., ρ' stays in Q_1). The run ρ' cannot be accepting in $\textsc{CInfTela}(\mathcal{A})$, because $\rho(i) \in S_i$ and so $S_i \neq \emptyset$ for any $i \in \omega$ (in Q_1, the only accepting transitions are those from state $\emptyset$ to state $\emptyset$). Second, let

$$\rho'' = S_0 S_1 \dots S_p (S_{p+1}, O_{p+1}, f_{p+1}, i_{p+1}, \mu_{p+1})(S_{p+2}, O_{p+2}, f_{p+2}, i_{p+2}, \mu_{p+2}) \dots$$

be a run of $\textsc{CInfTela}(\mathcal{A})$ on w (ρ'' jumps to Q_2 at position p). From the construction, it holds that $(f_j, \mu_j) \twoheadrightarrow^a_\delta (f_{j+1}, \mu_{j+1})$ for all $j > p$. Since ρ is accepting in $\mathcal{A}$, eventually there will be a position $k > p$ such that $f_k(\rho(k))$, $f_{k+1}(\rho(k+1))$, $f_{k+2}(\rho(k+2)), \dots$ are all even (because there is no model satisfying ρ in $\mathcal{M}^{\min}_{\mathsf{Acc}}$, so points (iib) and (iic) from the definition of $\twoheadrightarrow^a_\delta$ will enforce this). For the sake of contradiction, assume that ρ'' is accepting. Then for some position $\ell > k$, because the i-component of a macrostate rotates over all even ranks, it holds that $i_\ell = f_\ell(\rho(\ell))$ and $\rho(\ell) \in O_\ell = f_\ell^{-1}(\rho(\ell))$. We can easily show by induction that for all $j \geq \ell$, it holds that $\rho(j) \in O_j \neq \emptyset$, which is in contradiction with the assumption that ρ'' is accepting.

($\supseteq$) Consider any word $w \notin \mathcal{L}(\mathcal{A})$. From Corollary 5 and Lemma 6 it follows that the run DAG $\mathcal{G}_w$ has a bounded rank. If all vertices of $\mathcal{G}_w$ are finite, then there is an accepting run ρ' on $\textsc{CInfTela}(\mathcal{A})$ where $\rho' = S_0 S_1 \dots$ with $S_0 = I$ and $S_{i+1} = \delta(S_i, w_i)$ for all $i \in \omega$. Otherwise, Algorithm 1 terminates with a tight ranking r and a model m. From the definition of $\twoheadrightarrow^a_\delta$, there is a run

$$\rho'' = S_0 S_1 \dots S_p (S_{p+1}, O_{p+1}, f_{p+1}, i_{p+1}, \mu_{p+1})(S_{p+2}, O_{p+2}, f_{p+2}, i_{p+2}, \mu_{p+2}) \dots$$

such that $f_k(q) = r((q,k))$ and $\mu_k(q) = m((q,k))$ for all $k > p$. In order to show that ρ'' is acepting, we need to show that the O-component of the macrostates on the run is empty infinitely often. Assume by contradiction that there is an index $\ell > p$ such that O_j is non-empty for all $j \geq \ell$. Then, there is a vertex $(q, \ell) \in \mathcal{G}_w$ s.t. $r((q,\ell))$ is even and there are infinitely many vertices reachable from (q,ℓ) with the same even rank, which is a contradiction with the construction of r in Algorithm 1, which would give some of the vertices odd ranks. □

For the complexity analysis, we use $tight(n)$ to denote the number of μ-tight level rankings for an automaton with n states (μ-tight rankings for Inf-TELAs correspond to *tight* rankings for BAs). It holds that $tight(n) \approx (0.76n)^n$ [14,38].

Theorem 8. *The number of states of* $\textsc{CInfTela}(\mathcal{A})$ *is in* $O(k^n \cdot tight(n+1)) = O(n(0.76nk)^n)$ *for* $k = |\mathcal{M}^{\min}_{\mathsf{Acc}}|$.

Proof. The set of macrostates Q_1 is obtained by a simple subset construction, therefore $|Q_1| \in O(2^n)$. That is much smaller than $O(k^n \cdot tight(n+1))$, so it is sufficient to count only the number of macrostates of the form (S, O, f, i, μ). For this, we uniquely encode each macrostate as a pair (h, i) where $h \colon Q \to \{-2, -1, \dots, 2n-1\} \times \mathcal{M}^{\min}_{\mathsf{Acc}}$ is defined as follows:

$$h(q) = \begin{cases} (-1, \mu) & \text{if } q \in O, \\ (-2, \mu) & \text{if } q \in Q \setminus S, \\ (f(q), \mu) & \text{otherwise.} \end{cases} \tag{1}$$

We compute the number of encodings h for a fixed i. We divide all encodings into four groups according to the set $\mathrm{img}(h)_0 \cap \{-2,-1\}$ where $\mathrm{img}(h)_0$ denotes the set of first elements of the pairs in $\mathrm{img}(h)$. We show that we can obtain the bound $O(k^n \cdot \mathit{tight}(n))$ for each of the groups. The groups are denoted by g_M with $M \subseteq \{-2,-1\}$. For $h(q) = (m,\mu)$, we use $h(q)_m$ and $h(q)_\mu$ to denote m and μ.

$g_\emptyset$: from the definition, f is μ-tight. The level model μ is of the form $\mu\colon Q \to \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$, so there are k possible assignments for every state from Q. The number of level models is therefore k^n and $|g_\emptyset| = O(k^n \cdot \mathit{tight}(n))$.

$g_{\{-1\}}$: since there is at least one state q with $h(q)_m = -1$, this means that $q \in O$ so q has an even rank. As a consequence, at least one of the positive odd ranks of h (up to $2n-1$) will not be taken, so we can infer that $h\colon Q \to \{-1,\dots,2n-3\} \times \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$. We can therefore uniquely represent h by a mapping h' by incrementing all ranks of h by two, so $h'\colon Q \to \{0,\dots,2n-1\} \times \mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$. But then $h' \in \mathcal{T}(n)$ and the number of all level models is k^n, so $|g_{\{-1\}}| \in O(k^n \cdot \mathit{tight}(n))$.

$g_{\{-2,-1\}}$: similarly as for $g_{\{-1\}}$ we get that $|g_{\{-2,-1\}}| \in O(k^n \cdot \mathit{tight}(n))$.

$g_{\{-2\}}$: the reasoning is similar to the one for $g_{\{-1\}}$, with the exception that now, we know that there is a state $q \in Q \setminus S$, which is, according to the definition of a ranking, assigned the rank 0. This means that one positive odd rank of h is, again, not taken, so we increment all non-negative ranks of h by two and map all states in $Q \setminus S$ to 1, obtaining a tight ranking $h' \in \mathcal{T}(n)$. The number of level models is k^n, therefore, $|g_{\{-2\}}| \in O(k^n \cdot \mathit{tight}(n))$.

Since the size of all groups is bounded by $O(k^n \cdot \mathit{tight}(n))$, for a fixed i, the total number of these encodings is still $O(k^n \cdot \mathit{tight}(n))$. When we sum the encodings for all i's, we obtain that the number is bounded by $O(k^n \cdot \mathit{tight}(n+1))$, since $O(n \cdot \mathit{tight}(n)) = O(\mathit{tight}(n+1))$ [38]. The rest follows from the approximation of $\mathit{tight}(n)$. □

Corollary 9. *Let $\mathcal{A}$ be an Inf-TELA with n states and k colours Γ. The number of states of $\textsc{CInfTela}(\mathcal{A})$ is in $O(\binom{k}{\lfloor k/2 \rfloor}^n \cdot \mathit{tight}(n+1)) = O(n \cdot (\binom{k}{\lfloor k/2 \rfloor} \cdot 0.76n)^n) \subseteq O(n(2^k \cdot 0.76n)^n)$.*

Proof. The proof of the more precise bound follows directly from Theorem 8 and the fact that the size of $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}$ is bounded by the size of the largest antichain in 2^Γ, which is at most $\binom{k}{\lfloor k/2 \rfloor}$ by Sperner's theorem. □

Corollary 10. *Let $\mathcal{A}$ be a GBA with n states and k colours. Then the number of states of $\textsc{CInfTela}(\mathcal{A})$ is in $O(k^n \cdot \mathit{tight}(n+1)) = O(n(0.76nk)^n)$.*

Proof. The proof follows directly from Theorem 8. For a GBA it holds that $\overline{\mathsf{Acc}} = \bigvee_{0 \le j < k} \mathit{j}$. The formula is in DNF, hence $\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}} = \{\{\mathit{j}\} \mid 0 \le j < k\}$ and $|\mathcal{M}^{\min}_{\overline{\mathsf{Acc}}}| = k$. The number of all level models is k^n. □

We note that to the best of our knowledge, our bound on the complementation of GBAs is better than other bounds in the literature. In particular, it is clearly better than the bound $O(k^n(2n+1)^n)$ from [28], which is the best upper bound for complementing GBAs that we are aware of. It is also better than an approach that would go through

determinization by using the procedure in [39], which outputs a deterministic Rabin automaton with the number states bounded by $(1.47nk)^n$ for large k and $2^n - 1$ accepting pairs, which can be complemented easily into a Streett automaton.

4 Modular Complementation of Fin(c) ∧ φ TELAs

In this section, we propose a modular algorithm FinCompl for complementation of TELAs with the acceptance condition Fin(c) ∧ φ for any φ, parameterized by an algorithm for complementing TELAs with the condition φ. In Section 5, we will then instantiate the algorithm for some common acceptance conditions, eventually obtaining an efficient complementation algorithm for general TELAs.

Let us fix a TELA $\mathcal{A} = (Q, \delta, I, \Gamma, \mathsf{p}, \mathrm{Fin}(c) \wedge \varphi)$ and let Δ be δ without transitions whose label contains c. For a word $w \in \Sigma^\omega$, we define a *relaxed run DAG* (RRDAG) over w, denoted by $\mathcal{G}_w^\Delta$, as any sequence of states $\mathcal{G}_w^\Delta = (S_0, S_1, \dots)$ where $S_i \subseteq Q$ and $\Delta(S_i, w_i) \subseteq S_{i+1}$. Intuitively, an RRDAG over a word may contain more states on each level than it is necessary from the reachability of Δ. Note that this definition of RRDAGs is equivalent to having vertices of the form (q, i), where $q \in S_i$ with edges given implicitly by Δ. We use these definitions interchangeably. Clearly, there may be multiple RRDAGs over a single word, they are all, however, subgraphs of the (standard) run DAG $\mathcal{G}_w$. We say that $\mathcal{G}_w^\Delta = (S_0, S_1, \dots)$ is *accepting* wrt φ, written as $\mathcal{G}_w^\Delta \models \varphi$, if there is a run $\rho = q_k q_{k+1} \dots$ for $k \geq 0$ in Δ such that for every $i \geq k$ it holds that $q_i \in S_i$ and $q_{i+1} \in \Delta(q_i, w_i)$, and, moreover, $\rho \models \varphi$ (i.e., the accepting run does not need to start at the beginning of $\mathcal{G}_w^\Delta$). The reason for introducing RRDAGs is that the algorithm for condition φ will construct a BA that runs over RRDAGs constructed using the restricted transition relation Δ. The relaxation allows us to introduce new vertices (not connected to the root of the RRDAG) at any level that represent runs that have seen finitely many times a c transition in δ.

Our definition of the modular procedure FinCompl for Fin(c) ∧ φ is given wrt a *subprocedure* for complementing a TELA with condition φ. The subprocedure is given as a tuple $\mathbb{S}_\Delta^\varphi = (\mathcal{M}, \mathcal{M}_0, \mathsf{SuccAct}_\Delta, \mathsf{SuccTrack}_\Delta, \mathsf{EmptyBreak})$, where

(i) $\mathcal{M}$ is a set of *macrostates*,
(ii) $\mathcal{M}_0 \subseteq \mathcal{M}$ is a set of *initial macrostates*,
(iii) $\mathsf{SuccAct}_\Delta \colon 2^Q \times \Sigma \times \mathcal{M} \to 2^\mathcal{M}$ is an *active transition function*,
(iv) $\mathsf{SuccTrack}_\Delta \colon 2^Q \times \Sigma \times \mathcal{M} \to 2^\mathcal{M}$ is a *tracking transition function*, and
(v) $\mathsf{EmptyBreak} \subseteq \mathcal{M}$ is an *empty-breakpoint* predicate.

We use Succ_Δ to denote $\mathsf{SuccAct}_\Delta \cup \mathsf{SuccTrack}_\Delta$ (when treated as relations). Intuitively, $\mathcal{M}$ is a set of macrostates given by the subprocedure for φ. EmptyBreak is a condition that has to hold for a macrostate to be accepting in $\mathbb{S}_\Delta^\varphi$. The transitions between macrostates of $\mathcal{M}$ are described using transition functions $\mathsf{SuccAct}_\Delta$ and $\mathsf{SuccTrack}_\Delta$. In particular, $\mathsf{M}' \in \mathsf{Succ}_\Delta(P', a, \mathsf{M})$ is computed by taking the successor of the macrostate M over a, but also while taking into account the set P' of states (M corresponds to index i of the run while M' and P' correspond to index $i + 1$) provided by FinCompl, which represent breaking the Fin(c) condition. The reason for using two transition functions ($\mathsf{SuccAct}_\Delta$ and $\mathsf{SuccTrack}_\Delta$) is that some subprocedures that we will introduce later will use two

types of macrostates: active and tracking. For instance, if $\mathbb{S}^{\varphi}_{\Delta}$ is a rank-based procedure (cf. Section 5.2), active macrostates will contain breakpoints, which the construction will try to empty, and once a breakpoint is seen, FinCompl will add some more runs to the rank-based algorithm. The new runs might not be tight at the given point, so we switch into the tracking mode and wait for newly added runs to become tight before switching into the active mode again.

Let w be a word and $\mathcal{G}^{\Delta}_{w} = (S_0, S_1, \dots)$ be an RRDAG over w. A *Fin-run* R of $\mathbb{S}^{\varphi}_{\Delta}$ over $\mathcal{G}^{\Delta}_{w}$ is a sequence $(\mathsf{M}_0, \mathsf{M}_1, \dots)$ s.t. $\mathsf{M}_0 \in \mathcal{M}_0$ and $\mathsf{M}_{i+1} \in \mathsf{Succ}_{\Delta}(S_{i+1}, w_i, \mathsf{M}_i)$ for all $i \geq 0$. R is *accepting* if $\mathsf{EmptyBreak}(\mathsf{M}_i)$ holds for infinitely many i's. We say that the subprocedure $\mathbb{S}^{\varphi}_{\Delta}$ is *correct for* φ if for each word w and every RRDAG $\mathcal{G}^{\Delta}_{w}$ over w it holds that $\mathcal{G}^{\Delta}_{w}$ is not accepting wrt φ iff there is an accepting Fin-run R of $\mathbb{S}^{\varphi}_{\Delta}$ over $\mathcal{G}^{\Delta}_{w}$.

Let us now move to the definition of FinCompl. For subprocedure $\mathbb{S}^{\varphi}_{\Delta}$ and TELA $\mathcal{A}$ given above, the algorithm will construct the BA $\mathsf{FinCompl}(\mathbb{S}^{\varphi}_{\Delta}, \mathcal{A}) = (Q', I', \delta', F')$ defined as follows:

- $Q' = \{(S, P, \mathsf{M}) \in 2^{Q} \times 2^{Q} \times \mathcal{M}\}$,
- $I' = \{(I, I, \mathsf{M}_0) \mid \mathsf{M}_0 \in \mathcal{M}_0\}$,
- $\delta' = \delta_1 \cup \delta_2$ where
  - $\delta_1 \colon Q' \times \Sigma \to 2^{Q'}$ such that $(S', P', \mathsf{M}') \in \delta_1((S, P, \mathsf{M}), a)$ iff
    - $S' = \delta(S, a)$,
    - if $\mathsf{EmptyBreak}(\mathsf{M})$: $P' = S'$,
    - if $\neg\mathsf{EmptyBreak}(\mathsf{M})$: $P' = \Delta(P, a)$,
    - $\mathsf{M}' \in \mathsf{SuccAct}_{\Delta}(P', a, \mathsf{M})$,
  - $\delta_2 \colon Q' \times \Sigma \to 2^{Q'}$ such that $(S', P', \mathsf{M}') \in \delta_2((S, P, \mathsf{M}), a)$ iff
    - $S' = \delta(S, a)$,
    - $P' = \Delta(P, a)$,
    - $\mathsf{M}' \in \mathsf{SuccTrack}_{\Delta}(P', a, \mathsf{M})$, and
- $F' = \{(S, P, \mathsf{M}) \xrightarrow{a} (S', P', \mathsf{M}') \in \delta' \mid a \in \Sigma, \mathsf{EmptyBreak}(\mathsf{M}')\}$.

Intuitively, the construction executes $\mathbb{S}^{\varphi}_{\Delta}$ on the restricted transition relation Δ, while also keeping track of all runs (in S) and runs that either need to terminate or see a C-transition (in P). Whenever $\mathbb{S}^{\varphi}_{\Delta}$ clears its breakpoint, P is re-sampled (and some new runs can be added to $\mathbb{S}^{\varphi}_{\Delta}$).

Theorem 11. *For a correct subprocedure* $\mathbb{S}^{\varphi}_{\Delta}$, $\mathcal{L}(\mathsf{FinCompl}(\mathbb{S}^{\varphi}_{\Delta}, \mathcal{A})) = \Sigma^{\omega} \setminus \mathcal{L}(\mathcal{A})$.

The overhead of the procedure over the subprocedure $\mathbb{S}^{\varphi}_{\Delta}$ is at most 3^n-times.

Theorem 12. *Suppose* $\mathbb{S}^{\varphi}_{\Delta} = (\mathcal{M}, \cdot, \cdot, \cdot)$. *Then* $|\mathsf{FinCompl}(\mathbb{S}^{\varphi}_{\Delta}, \mathcal{A})| \in O(3^n \cdot |\mathcal{M}|)$.

Proof. Since in (S, P, M), it always holds that $P \subseteq S$, each state of $\mathcal{A}$ can be in one of the three following sets: (i) $Q \setminus S$, (ii) $S \cap P$, and (iii) $S \setminus P$. □

5 Complementation of TELAs and their Subclasses

We proceed by instantiating the modular algorithm FinCompl from the previous section for several common automata classes—co-Büchi automata, Rabin automata, parity automata, generalized Rabin automata, and, eventually, TELAs.

5.1 Co-Büchi Automata

As a simple demonstration of instantiation of FinCompl, we use it to create a complementation algorithm for co-Büchi automata. The acceptance condition for co-Büchi automata is $\mathsf{Fin}(\textcircled{0}) = \mathsf{Fin}(\textcircled{0}) \wedge tt$, we therefore need to provide a trivial subprocedure $\mathbb{S}^{tt} = (\mathcal{M}^{tt}, \mathcal{M}_0^{tt}, \mathsf{SuccAct}_\Delta^{tt}, \emptyset, \mathsf{EmptyBreak}^{tt})$ that is correct for tt (notice that $\mathsf{SuccTrack}_\Delta^{tt}$ is empty). In the subprocedure, $\mathcal{M}^{tt} = 2^Q$, $\mathcal{M}_0^{tt} = \{I\}$, and the remaining components are given as follows:

$$\mathsf{SuccAct}_\Delta^{tt}(P, a, S) = \{P\} \qquad \text{and} \qquad \mathsf{EmptyBreak}^{tt}(P) \iff P = \emptyset.$$

Intuitively, the instantiated procedure works with macrostates (S, P, P) (i.e., to adhere to the formal definition of FinCompl, P is there twice) where S tracks all runs and P is a breakpoint that contains runs that yet need to either terminate or see $\textcircled{0}$. To accept, P needs to be emptied infinitely often. One can observe that $\mathsf{FinCompl}(\mathbb{S}^{tt}, \mathcal{A})$ resembles the well-known Miyano-Hayashi construction [34] for complementation of co-Büchi automata.

Lemma 13. *The subprocedure $\mathbb{S}^{tt}$ is correct for the acceptance condition tt.*

Corollary 14. *For a co-Büchi automaton $\mathcal{A}$, $\mathcal{L}(\mathsf{FinCompl}(\mathbb{S}^{tt}, \mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.*

Proof. Follows from Lemma 13 and Theorem 11. □

Since the result of the construction can be mapped to the Miyano-Hayashi's algorithm [34], the complexities also match.

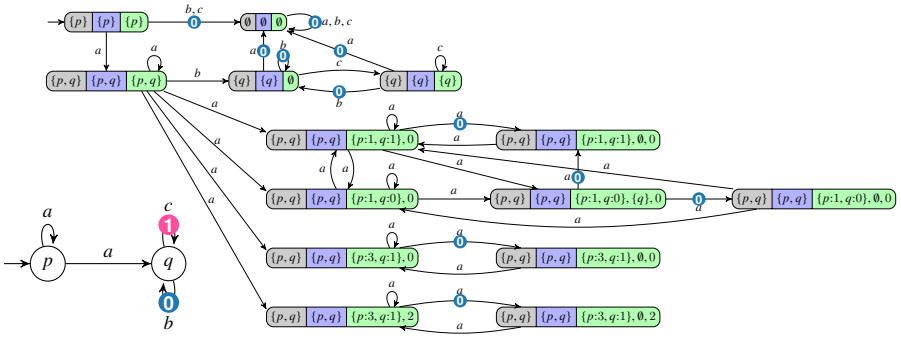
Corollary 15. $|\mathsf{FinCompl}(\mathbb{S}^{tt}, \mathcal{A})| \in O(3^n)$.

5.2 Rabin Automata

In this section, we give an instantiation of FinCompl with subprocedure $\mathbb{S}^{\mathsf{inf}} = (\mathcal{M}^{\mathsf{inf}}, \mathcal{M}_0^{\mathsf{inf}}, \mathsf{SuccAct}_\Delta^{\mathsf{inf}}, \mathsf{SuccTrack}_\Delta^{\mathsf{inf}}, \mathsf{EmptyBreak}^{\mathsf{inf}})$ for $\mathsf{Inf}(\textcircled{1})$, which will allow us to complement TELAs where the acceptance condition is a single Rabin pair. The algorithm is based on the optimal rank-based BA complementation algorithm from [38] adjusted to the needs of the modular construction. The macrostates of the instantiation are given as

$$\mathcal{M}^{\mathsf{inf}} = \overbrace{2^Q \cup (\mathcal{T} \times 2^Q \times \{0, 2, \ldots, 2n-2\})}^{\mathcal{M}_{\mathsf{Act}}^{\mathsf{inf}}} \cup \overbrace{(\mathcal{T} \times \{0, 2, \ldots, 2n-2\})}^{\mathcal{M}_{\mathsf{Track}}^{\mathsf{inf}}}$$

where $\mathcal{M}_0^{\mathsf{inf}} = \{I\}$. Notice that *active macrostates* ($\mathcal{M}_{\mathsf{Act}}^{\mathsf{inf}}$) are either sets of states (from 2^Q, just keeping track of all runs) or states of the form (f, O, i) (representing tight runs). On the other hand, *tracking macrostates* ($\mathcal{M}_{\mathsf{Track}}^{\mathsf{inf}}$) are of the form (f, i); these are used to wait for newly arrived runs to become tight. The remaining components are then defined as follows:



(a) Example of a Rabin automaton with the acceptance condition Fin(0) ∧ Inf(1).

(b) The resulting complement automaton with the acceptance condition Inf(0). The macrostates are depicted in the form S (grey), P (blue), M (green).

Fig. 3: Example of FinCompl instantiated with $\mathbb{S}^{\mathsf{inf}}$ for complementation of automata with the acceptance condition containing a single Rabin pair.

- $(f', O', i') \in \mathsf{SuccAct}^{\mathsf{inf}}_{\Delta}(P, a, (f, O, i))$ iff
  - $f \sqsubseteq^{a}_{\Delta} f'$ and $rank(f) = rank(f')$,
  - $\mathrm{dom}(f') = P$,
  - $O \neq \emptyset$,
  - $i' = i$,
  - $O' = \Delta(O, a) \cap f'^{-1}(i)$
- $(f', i') \in \mathsf{SuccAct}^{\mathsf{inf}}_{\Delta}(P, a, (f, O, i))$ iff
  - $f \sqsubseteq^{a}_{\Delta} f'$ and $rank(f) = rank(f')$,
  - $O = \emptyset$,
  - $i' = (i+2) \mod (rank(f') + 1)$
- $P' \in \mathsf{SuccAct}^{\mathsf{inf}}_{\Delta}(P, a, P)$ iff
  - $P' = P$
- $(f', i') \in \mathsf{SuccTrack}^{\mathsf{inf}}_{\Delta}(P, a, P)$ iff
  - f' is P-tight
  - $i' = 0$
- $\{(f', i'), (f', O', i')\} \subseteq \mathsf{SuccTrack}^{\mathsf{inf}}_{\Delta}(P, a, (f, i))$ iff
  - $f \sqsubseteq^{a}_{\Delta} f'$ and $rank(f) = rank(f')$,
  - $O' = f'^{-1}(i')$,
  - $i' = i$
- $\mathsf{EmptyBreak}^{\mathsf{inf}}((f, O, i)) \iff O = \emptyset$
- $\mathsf{EmptyBreak}^{\mathsf{inf}}(P) \iff P = \emptyset$
- $\mathsf{EmptyBreak}^{\mathsf{inf}}((f, i)) \iff \mathit{false}$

An example of the construction is shown in Fig. 3. The correctness of the instantiation is then summarized by the following lemma.

Lemma 16. *The subprocedure $\mathbb{S}^{\mathsf{inf}}$ is correct for the acceptance condition Inf(1).*

Proof (Sketch). In order to show that the subprocedure $\mathbb{S}^{\mathsf{inf}}$ is correct, we need to show that for each word w and every RRDAG $\mathcal{G}^{\Delta}_{w}$ it holds that $\mathcal{G}^{\Delta}_{w}$ is not accepting wrt Inf(1) iff there is an accepting Fin-run of $\mathbb{S}^{\mathsf{inf}}$ over $\mathcal{G}^{\Delta}_{w}$. We begin with the proof of the statement from left to right. Assume that $\mathcal{G}^{\Delta}_{w}$ is not accepting wrt Inf(1). There is either no run of $\mathcal{G}^{\Delta}_{w}$ on w at all or all runs do not satisfy the formula. If there is no run of $\mathcal{G}^{\Delta}_{w}$ on w, then there is a sequence $(M_0, M_1, \ldots)$ where $M_0 = I$ and $M_{j+1} = \Delta(M_j, a)$ for all $j \geq 0$ such that there is some $i \geq 0$ such that $M_l = \emptyset$ for all $l \geq i$. The predicate $\mathsf{EmptyBreak}(M_l)$ is true for all $l \geq i$, so it holds infinitely often, and there therefore exists an accepting run of $\mathbb{S}^{\mathsf{inf}}$ over $\mathcal{G}^{\Delta}_{w}$. Now assume that there is a run of $\mathcal{G}^{\Delta}_{w}$ on w. Then, no matter from which point there are no transitions from 0, the condition Inf(1) does not hold for the particular run. With every transition $(f, i) \to (f', O', i')$ we sample all currently reachable states and then check that all runs from these states contain transitions from 1 only finitely often by modified Schewe's rank-based algorithm. The O-component is emptied infinitely often and so there is an accepting run of $\mathbb{S}^{\mathsf{inf}}$ over $\mathcal{G}^{\Delta}_{w}$.

Now we prove the equivalence in the opposite direction. Assume that there is an accepting run of $\mathbb{S}^{\mathsf{inf}}$ over $\mathcal{G}^{\Delta}_{w}$. There is therefore a run where the EmptyBreak predicate

is true infinitely often. The first possible option is that $\mathsf{EmptyBreak}(P)$ is true infinitely many times. That can happen only if there is no run on w and $\mathcal{G}_w^\Delta$ is finite. If there is no such run, the formula is not satisfied and $\mathcal{G}_w^\Delta$ is not accepting. The second option is that $\mathsf{EmptyBreak}((f, O, i))$ is true infinitely many times. That means that the formula $\mathrm{Inf}(1)$ does not hold for any run, no matter when the run stops containing transitions from c. The formula is therefore not satisfied in any run and $\mathcal{G}_w^\Delta$ is not accepting. □

The following lemma shows that using our approach, handling the $\mathrm{Fin}(c)$ condition is "*for free*," i.e., the asymptotical complexity stays the same as for the optimal algorithm for BA complementation from [38].

Lemma 17. $|\mathsf{FinCompl}(\mathbb{S}^{\mathsf{inf}}, \mathcal{A})| \in O(\mathit{tight}(n+1))$.

Proof. It suffices to count the number of macrostates of the form (S, P, f, O, i). Consider a macrostate (S, P, f, O, i). We uniquely encode the macrostate as (h, i) where $h\colon Q \to \{-3, \ldots, 2n-1\}$ is defined as follows:

$$h(q) = \begin{cases} -1 & \text{if } q \in O, \\ -2 & \text{if } q \in Q \setminus S, \\ -3 & \text{if } q \in S \setminus P, \text{ and} \\ f(q) & \text{otherwise.} \end{cases} \tag{2}$$

For a fixed i we compute the number of such encodings h. First we divide all encodings into groups according to the set $\mathrm{img}(h) \cap \{-3, -2, -1\}$ (8 groups at most) and we will show for each of the groups how we can "*shuffle*" the ranks in h to obtain the bound $O(\mathit{tight}(n))$ for each of the groups. We will denote each of the groups by g_M with $M \subseteq \{-3, -2, -1\}$.

$g_\emptyset$: from the definition, f is tight so $|g_\emptyset| = O(\mathit{tight}(n))$

$g_{\{-1\}}$: since there is at least one state q with $h(q) = -1$, this means that $q \in O$ so q has an even rank. As a consequence, at least one of the positive odd ranks of h will not be taken, so we can infer that $h\colon Q \to \{-1, \ldots, 2n-3\}$. We can therefore uniquely map h to a mapping h' by incrementing all ranks of h by two, so $h'\colon Q \to \{1, \ldots, 2n-1\}$. But then $h' \in \mathcal{T}(n)$, so $|g_{\{-1\}}| \in O(\mathit{tight}(n))$.

$g_{\{-2,-1\}}$: via the same reasoning as for $g_{\{-1\}}$ we get that $|g_{\{-2,-1\}}| \in O(\mathit{tight}(n))$.

$g_{\{-2\}}$: the reasoning is similar to the one for $g_{\{-1\}}$, with the exception that now, we know that there is a state $q \in Q \setminus S$, which is, according to the definition of a ranking, assigned the rank 0. This means that one positive odd rank of h is, again, not taken, so we increment all non-negative ranks of h by two and map all states in $Q \setminus S$ to 1, obtaining a tight ranking $h' \in \mathcal{T}(n)$. Therefore, $|g_{\{-2\}}| \in O(\mathit{tight}(n))$.

$g_{\{-3\}}$: the reasoning is, again, similar to the one for $g_{\{-1\}}$, with the exception that now, we know that there is a state $q \in S \setminus P$ such that its rank is, according to the definition 0. Therefore, we increment all non-negative ranks of h by two and map the states in $S \setminus P$ to 1, obtaining a tight ranking $h' \in \mathcal{T}(n)$; therefore, $|g_{\{-3\}}| \in O(\mathit{tight}(n))$.

$g_{\{-3,-2\}}$, $g_{\{-3,-1\}}$: similarly as for $g_{\{-2\}}$, we increment all non-negative ranks of h by two and set $h'(q) = 0$ if $h(q) = -3$ and $h'(q) = 1$ if $h(q) = -2$ (resp. if $h(q) = -1$). Then $h' \in \mathcal{T}(n)$ and so $|g_{\{-3,-2\}}| = O(\mathit{tight}(n))$ and $|g_{\{-3,-1\}}| \in O(\mathit{tight}(n))$.

$g_{\{-3,-2,-1\}}$: in this case, we know that there is at least one state $q_1 \in O$ and at least one state $q_2 \in Q \setminus S$. Therefore, there will be at least two odd positions not taken in h, so we can infer that $h\colon \{-3, \ldots, 2n-5\}$. We create h' by incrementing all ranks in h by *four*; in this way, we obtain a tight ranking $h'\colon Q \to \{0, \ldots, 2n-1\}$, so $|g_{\{-3,-2,-1\}}| \in O(tight(n))$.

Since the size of all groups is bounded by $O(tight(n))$, for a fixed i, the total number of these encodings is still $O(tight(n))$. When we sum the encodings for all possible i's, we obtain that the number is bounded by $O(tight(n+1))$, since $O(n \cdot tight(n)) = O(tight(n+1))$ [38]. □

The modular construction instantiated with $\mathbb{S}^{\mathsf{inf}}$ gives us a procedure for complementing Rabin automata with a single pair. To get a procedure for general Rabin automata, we construct a complement automaton for each Rabin pair, make a product of these automata, and obtain a GBA accepting the complement of the original automaton. The complexity reasoning is straightforward and is summarized by the following corollary.

Corollary 18. *Let $\mathcal{A}$ be a Rabin automaton with k Rabin pairs. Then we can construct a GBA accepting the complement of the language of $\mathcal{A}$ with $O(tight(n+1)^k) = O(n^k(0.76n)^{nk})$ states and k colours.*

Proof. $O(tight(n+1)^k) = O((n \cdot tight(n))^k) = O((n(0.76n)^n)^k) = O(n^k(0.76n)^{nk})$ □

To the best of our knowledge, the state complexity of our procedure is better than the complexity of other approaches for complementing Rabin automata (even if we require the output to be a BA and not a GBA—the BA would have $O(k \cdot tight(n+1)^k) = O(kn^k(0.76n)^{nk})$ states). In particular, it is better than the complexity $O(k \cdot 3^n \cdot (2n+1)^{nk})$ of [27]. Comparing the two techniques, the main difference is that our modular approach allows us to use tight rankings (and the optimal construction of, e.g., Schewe [38]), which are a significant factor in decreasing the size of the complement (both in theory and in practice). On the other hand, [27] does not use tight rankings (their run DAG ranking procedure does not allow it since ranks can change arbitrarily when a Fin state is encountered), however, it performs the complementation for the k Rabin pairs at once and avoids performing the product. The complexity of our approach is better; combining the two approaches to get an even better complexity is future work.

The complexity of our approach is also better than the complexity of a procedure that would first transform the input Rabin automaton into a BA with $m = nk$ states and run the optimal BA complementation with complexity $O(m(0.76m)^m) = O(nk(0.76nk)^{nk})$ [38], as shown by the following lemmas.

Lemma 19. $O(n^k(0.76n)^{nk}) \subset O(k \cdot 3^n \cdot (2n+1)^{nk})$

Proof. $n^k(0.76n)^{nk} = (\sqrt[k]{n} \cdot 0.76n)^{nk}$. The global maximum of the function $\sqrt[k]{n}$ is less than 1.5, so $(\sqrt[k]{n} \cdot 0.76n)^{nk} < (1.14n)^{nk} < (2n+1)^{nk}$ for $n \geq 1$. □

Lemma 20. $O(n^k(0.76n)^{nk}) \subset O(nk(0.76nk)^{nk})$

Proof. Similar reasoning as in the proof of Lemma 19. □

5.3 Parity Automata

Since the parity condition is a special case of the Rabin condition [15], we can easily give an upper bound on the complementation of parity automata.

Lemma 21. *For a parity automaton $\mathcal{A}$ with index k, there is a GBA for the complement of $\mathcal{L}(\mathcal{A})$ with $\frac{k}{2}$ colours and $O(tight(n+1)^{\frac{k}{2}}) = O(n^{\frac{k}{2}}(0.76n)^{\frac{nk}{2}})$ states.*

Proof. The min-odd parity acceptance condition is of the form $\mathsf{Acc} = \mathrm{Fin}(0) \wedge (\mathrm{Inf}(1) \vee (\mathrm{Fin}(2) \wedge (\mathrm{Inf}(3) \vee (\mathrm{Fin}(4) \wedge \ldots))))$. If we transform the acceptance condition into the DNF, we obtain $\mathsf{Acc}' = (\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)) \vee (\mathrm{Fin}(0+2) \wedge \mathrm{Inf}(1+3)) \vee (\mathrm{Fin}(0+2+4) \wedge \mathrm{Inf}(1+3+5)) \vee \ldots$ which is a Rabin acceptance condition with $\frac{k}{2}$ Rabin pairs. In the condition, e.g., $0+2$ denotes *union* of colours 0 and 2, obtained by changing all occurrences of 0 and 2 in $\mathcal{A}$'s colouring function p to the new colour $0+2$. Note that we can use a new colour for each union of colours and we obtain the same number of colours as in Acc. According to Corollary 18, the parity automaton $\mathcal{A}$ can be complemented into a GBA with $O(tight(n+1)^{\frac{k}{2}})$ states. □

We note that the complexity obtained by our general procedure is worse than the best one we are aware of, which is $2^{O(n \log n)}$ [7].

5.4 Generalized Rabin Automata

Recall that the generalized Rabin pair is of the form $\mathrm{Fin}(0) \wedge \bigwedge_{j=1}^{n} \mathrm{Inf}(j)$. We can now easily combine the procedure for (standard) Rabin automata from the previous section and the procedure for Inf-TELA from Section 3.2 to construct the subprocedure $\mathbb{S}^{\wedge\mathsf{inf}}$ for $\bigwedge_{j=1}^{n} \mathrm{Inf}(j)$. The set of macrostates will be

$$\mathcal{M}^{\wedge\mathsf{inf}} = 2^{Q} \cup (\mathcal{T} \times 2^{Q} \times \{0, 2, \ldots, 2n-2\} \times \mathsf{LM}) \cup (\mathcal{T} \times \{0, 2, \ldots, 2n-2\} \times \mathsf{LM})$$

Details are given in [20]. Similarly to Sections 3.2 and 5.2, one can then obtain the following bound on the size of the complement.

Lemma 22. *Let $\mathcal{A}$ be a generalized Rabin automaton with one generalized Rabin pair with ℓ Infs. Then, there exists a BA accepting the complement of $\mathcal{A}$ with $O(\ell^n tight(n+1)) = O(n\ell^n(0.76n)^n)$ states.*

Theorem 23. *Let $\mathcal{A}$ be a generalized Rabin automaton with k generalized Rabin pairs, each with at most ℓ Infs. Then, there exists a GBA with k colours and $O(\ell^{nk} tight(n+1)^k) = O(n^k(0.76\ell n)^{nk})$ states accepting $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.*

There is not much work on the complementation of generalized Rabin automata or general TELAs (we are only aware of the upper bound $2^{2^{O(n)}}$ from [37])). One could approach the complementation by translation of the generalized Rabin automaton into a GBA using the technique from [22]. The technique first performs Fin-removal, i.e., it makes k copies of $\mathcal{A}$, each with the corresponding Fin-transitions removed, obtaining a GBA with $n(k+1)$ states and ℓ colours (one can share colours across the independent copies). After that, we could use our GBA complementation algorithm from Section 3, which would give us a BA with $O(n(k+1)(0.76\ell n(k+1))^{n(k+1)})$ states, which is worse.

Lemma 24. $O(n^k(0.76\ell n)^{nk}) \subset O(n(k+1)(0.76\ell n(k+1))^{n(k+1)})$

Proof (Idea). Let us observe the behaviour of the fraction with a simplified right-hand side: $\frac{nk(0.76\ell nk)^{nk}}{n^k(0.76\ell n)^{nk}} = \frac{nk^{nk+1}}{n^k}$. There are two options:

(i) $n \geq k$: in this case, $k^{nk} \gg n^k$ and the claim holds.
(ii) $k \geq n$: in this case, $k^k \gg n^k$ and the claim holds. □

5.5 General TELAs

For complementation of general TELAs, we use the fact that any TELA can be converted into a generalized Rabin automaton with the same structure by modifying the acceptance condition into the DNF form (and not touching the structure of the automaton). For a TELA with k colours, the DNF will have at most 2^k clauses (i.e., generalized Rabin pairs), each one with at most k literals.

Theorem 25. *Let $\mathcal{A}$ be a TELA with k colours. Then, there exists a GBA with 2^k colours and $O(k^{n2^k}\, tight(n+1)^{2^k}) = O(n^{2^k}(0.76nk)^{n2^k})$ states accepting $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.*

Proof. By substituting to Theorem 23. □

6 Related Work

Lower bounds for complementation of classes of ω-automata using the full automata technique were established in [45] (improving the previous $\Omega(n!)$ lower bound of Michel [33]). The technique was later generalized to improve the lower bound of Rabin automata complementation [8]. A double exponential lower bound for complementation of general Emerson-Lei automata was given in [37]. See the survey in [4] for more details.

Simultaneously to establishing the lower bound, there emerged algorithms for determinizing and complementing various classes of ω-automata. The optimal determinization approach for GBAs introduced in [39] yields a deterministic Rabin automaton with the number of states bounded by $(1.47nk)^n$ for large k and $2^n - 1$ Rabin pairs. In [13], the Miyano-Hayashi construction [34] is used within Büchi determinization. Rank-based complementation of GBAs was proposed in [28]. Furthermore, there are approaches for semideterminization-based complementation of GBAs [3] with double exponential complexity. Regarding other acceptance conditions, determinization of parity automata based on root history trees was proposed in [40]. A rank-based complementation of Streett and Rabin automata was introduced in [27] and later improved by tree structures in [7]. Tight determinization of Streett automata was presented in [43]. A tight complementation technique for parity automata based on flattened nested history trees was then proposed in [41]. A lot of effort has been put into complementation of Büchi automata leading to algorithms roughly divided into several groups: Ramsey-based [5,6,42], rank-based [16,19,18,44,26,38], determinization-based [36,35,30], slice-based [23], and others [1,17,31]. There are specialized more efficient algorithms for subclasses of BAs, such as inherently-weak [34], deterministic [29], semideterministic [2], elevator [19,17], or unambiguous [32,12] BAs.

Acknowledgments

We thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work has been supported by the Czech Ministry of Education, Youth and Sports ERC.CZ project LL1908, the Czech Science Foundation project 25-18318S, and the FIT BUT internal project FIT-S-23-8151.

References

1. Joël D. Allred and Ulrich Ultes-Nitsche. A simple and optimal complementation algorithm for Büchi automata. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 46–55. ACM, 2018. doi:10.1145/3209108.3209138.
2. Frantisek Blahoudek, Matthias Heizmann, Sven Schewe, Jan Strejcek, and Ming-Hsien Tsai. Complementing semi-deterministic Büchi automata. In Marsha Chechik and Jean-François Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 770–787. Springer, 2016. doi:10.1007/978-3-662-49674-9_49.
3. František Blahoudek, Alexandre Duret-Lutz, and Jan Strejček. Seminator 2 can complement generalized Büchi automata via improved semi-determinization. In *Proceedings of the 32nd International Conference on Computer-Aided Verification (CAV'20)*, volume 12225 of *Lecture Notes in Computer Science*, pages 15–27. Springer, July 2020. doi:10.1007/978-3-030-53291-8_2.
4. Udi Boker. Why these automata types? In *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 143–163. EasyChair, 2018.
5. Stefan Breuers, Christof Löding, and Jörg Olschewski. Improved Ramsey-based Büchi complementation. In *Proc. of FOSSACS'12*, pages 150–164. Springer, 2012.
6. J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Proc. of International Congress on Logic, Method, and Philosophy of Science 1960*. Stanford Univ. Press, Stanford, 1962.
7. Yang Cai and Ting Zhang. Tight upper bounds for Streett and parity complementation. In Marc Bezem, editor, *Computer Science Logic, 25th International Workshop / 20th Annual Conference of the EACSL, CSL 2011, September 12-15, 2011, Bergen, Norway, Proceedings*, volume 12 of *LIPIcs*, pages 112–128. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.CSL.2011.112.
8. Yang Cai, Ting Zhang, and Haifeng Luo. An improved lower bound for the complementation of Rabin automata. In *Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science*, LICS '09, page 167–176, USA, 2009. IEEE Computer Society. doi:10.1109/LICS.2009.13.
9. Yu-Fang Chen, Vojtech Havlena, and Ondrej Lengál. Simulations in rank-based Büchi automata complementation. In Anthony Widjaja Lin, editor, *Programming Languages and Systems - 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*, volume 11893 of *Lecture Notes in Computer Science*, pages 447–467. Springer, 2019. doi:10.1007/978-3-030-34175-6_23.
10. Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *Principles of Security and Trust - Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8414 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.
11. E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.*, 8(3):275–306, 1987. doi:10.1016/0167-6423(87)90036-0.

12. Weizhi Feng, Yong Li, Andrea Turrini, Moshe Y. Vardi, and Lijun Zhang. On the power of finite ambiguity in Büchi complementation. *Inf. Comput.*, 292:105032, 2023. URL: https://doi.org/10.1016/j.ic.2023.105032, doi:10.1016/J.IC.2023.105032.
13. Dana Fisman and Yoad Lustig. A modular approach for Büchi determinization. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 368–382. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2015.368, doi:10.4230/LIPICS.CONCUR.2015.368.
14. Ehud Friedgut, Orna Kupferman, and Moshe Y. Vardi. Büchi complementation made tighter. *Int. J. Found. Comput. Sci.*, 17(4):851–868, 2006. doi:10.1142/S0129054106004145.
15. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-36387-4.
16. Vojtech Havlena and Ondrej Lengál. Reducing (to) the ranks: Efficient rank-based Büchi automata complementation. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 2:1–2:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2021.2, doi:10.4230/LIPICS.CONCUR.2021.2.
17. Vojtěch Havlena, Ondřej Lengál, Yong Li, Barbora Šmahlíková, and Andrea Turrini. Modular mix-and-match complementation of Büchi automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France*, Lecture Notes in Computer Science. Springer, 2023.
18. Vojtěch Havlena, Ondřej Lengál, and Barbora Šmahlíková. Complementing Büchi automata with ranker. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 188–201. Springer, 2022. doi:10.1007/978-3-031-13188-2_10.
19. Vojtěch Havlena, Ondřej Lengál, and Barbora Šmahlíková. Sky is not the limit: Tighter rank bounds for elevator automata in Büchi automata complementation. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part II*, volume 13244 of *Lecture Notes in Computer Science*, pages 118–136. Springer, 2022. doi:10.1007/978-3-030-99527-0_7.
20. Vojtěch Havlena, Ondřej Lengál, and Barbora Šmahlíková. Complementation of Emerson-Lei automata (technical report). *CoRR*, abs/2410.11644, 2024. URL: https://arxiv.org/abs/2410.11644, arXiv:2410.11644.
21. Philipp Hieronymi, Dun Ma, Reed Oei, Luke Schaeffer, Christian Schulz, and Jeffrey O. Shallit. Decidability for Sturmian words. *Log. Methods Comput. Sci.*, 20(3), 2024. URL: https://doi.org/10.46298/lmcs-20(3:12)2024, doi:10.46298/LMCS-20(3:12)2024.
22. Tobias John, Simon Jantsch, Christel Baier, and Sascha Klüppelholz. From Emerson-Lei automata to deterministic, limit-deterministic or good-for-MDP automata. *Innov. Syst. Softw. Eng.*, 18(3):385–403, 2022. doi:10.1007/s11334-022-00445-7.
23. Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proc. of ICALP'08*, pages 724–735. Springer, 2008.
24. Hrishikesh Karmarkar and Supratik Chakraborty. On minimal odd rankings for Büchi complementation. In Zhiming Liu and Anders P. Ravn, editors, *Automated Technology for*

Verification and Analysis, 7th International Symposium, ATVA 2009, Macao, China, October 14-16, 2009. Proceedings, volume 5799 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2009. doi:10.1007/978-3-642-04761-9_18.

25. Yonit Kesten and Amir Pnueli. A complete proof systems for QPTL. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pages 2–12. IEEE Computer Society, 1995. doi:10.1109/LICS.1995.523239.
26. Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001. doi:10.1145/377978.377993.
27. Orna Kupferman and Moshe Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, volume 3440 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2005. doi:10.1007/978-3-540-31980-1_14.
28. Orna Kupferman and Moshe Y. Vardi. From complementation to certification. *Theor. Comput. Sci.*, 345(1):83–100, 2005. doi:10.1016/j.tcs.2005.07.021.
29. Robert P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *J. Comput. Syst. Sci.*, 35(1):59–71, 1987. doi:10.1016/0022-0000(87)90036-5.
30. Yong Li, Andrea Turrini, Weizhi Feng, Moshe Y. Vardi, and Lijun Zhang. Divide-and-conquer determinization of Büchi automata based on SCC decomposition. In Sharon Shoham and Yakir Vizel, editors, *Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II*, volume 13372 of *Lecture Notes in Computer Science*, pages 152–173. Springer, 2022. doi:10.1007/978-3-031-13188-2_8.
31. Yong Li, Andrea Turrini, Lijun Zhang, and Sven Schewe. Learning to complement Büchi automata. In *Proc. of VMCAI'18*, pages 313–335. Springer, 2018.
32. Yong Li, Moshe Y. Vardi, and Lijun Zhang. On the power of unambiguity in Büchi complementation. In Jean-Francois Raskin and Davide Bresolin, editors, Proceedings 11th International Symposium on *Games, Automata, Logics, and Formal Verification,* Brussels, Belgium, September 21-22, 2020, volume 326 of *Electronic Proceedings in Theoretical Computer Science*, pages 182–198. Open Publishing Association, 2020. doi:10.4204/EPTCS.326.12.
33. Max Michel. Complementation is more difficult with automata on infinite words. *CNET, Paris*, 15, 1988.
34. Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. In Bruno Courcelle, editor, *CAAP'84, 9th Colloquium on Trees in Algebra and Programming, Bordeaux, France, March 5-7, 1984, Proceedings*, pages 195–210. Cambridge University Press, 1984.
35. Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. of LICS'06*, pages 255–264. IEEE, 2006.
36. Shmuel Safra. On the complexity of ω-automata. In *Proc. of FOCS'88*, pages 319–327. IEEE, 1988.
37. Shmuel Safra and Moshe Y. Vardi. On ω-automata and temporal logic (preliminary report). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 127–137. ACM, 1989. doi:10.1145/73007.73019.
38. Sven Schewe. Büchi complementation made tight. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages

661–672. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1854.

39. Sven Schewe and Thomas Varghese. Tight bounds for the determinisation and complementation of generalised Büchi automata. In Supratik Chakraborty and Madhavan Mukund, editors, *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, volume 7561 of *Lecture Notes in Computer Science*, pages 42–56. Springer, 2012. doi:10.1007/978-3-642-33386-6_5.
40. Sven Schewe and Thomas Varghese. Determinising parity automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 486–498. Springer, 2014. doi:10.1007/978-3-662-44522-8_41.
41. Sven Schewe and Thomas Varghese. Tight bounds for complementing parity automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 499–510. Springer, 2014. doi:10.1007/978-3-662-44522-8_42.
42. A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.
43. Cong Tian, Wensheng Wang, and Zhenhua Duan. Making streett determinization tight. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 859–872, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3373718.3394757.
44. Moshe Y. Vardi. The Büchi complementation saga. In *Proc. of STACS'07*, pages 12–22. Springer, 2007.
45. Qiqi Yan. Lower bounds for complementation of ω-automata via the full automata technique. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 589–600, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.