Conditionally Optimal Algorithms for Generalized Büchi Games

Krishnendu Chatterjee[1], Wolfgang Dvořák[2], Monika Henzinger[2], and Veronika Loitzenbauer[2]

1 **IST Austria**

2 **University of Vienna, Faculty of Computer Science, Vienna, Austria**

Abstract

Games on graphs provide the appropriate framework to study several central problems in computer science, such as verification and synthesis of reactive systems. One of the most basic objectives for games on graphs is the liveness (or Büchi) objective that given a target set of vertices requires that some vertex in the target set is visited infinitely often. We study generalized Büchi objectives (i.e., conjunction of liveness objectives), and implications between two generalized Büchi objectives (known as GR(1) objectives), that arise in numerous applications in computer-aided verification. We present improved algorithms and conditional super-linear lower bounds based on widely believed assumptions about the complexity of (A1) combinatorial Boolean matrix multiplication and (A2) CNF-SAT. We consider graph games with n vertices, m edges, and generalized Büchi objectives with k conjunctions. First, we present an algorithm with running time $O(k \cdot n^2)$, improving the previously known $O(k \cdot n \cdot m)$ and $O(k^2 \cdot n^2)$ worst-case bounds. Our algorithm is optimal for dense graphs under (A1). Second, we show that the basic algorithm for the problem is optimal for sparse graphs when the target sets have constant size under (A2). Finally, we consider GR(1) objectives, with k_1 conjunctions in the antecedent and k_2 conjunctions in the consequent, and present an $O(k_1 \cdot k_2 \cdot n^{2.5})$-time algorithm, improving the previously known $O(k_1 \cdot k_2 \cdot n \cdot m)$-time algorithm for $m > n^{1.5}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Generalized Büchi objective, GR(1) objective, Conditional lower bounds, Graph games, Graph algorithms, Computer-aided verification

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Games on graphs. Two-player games on graphs, between player 1 and the adversary player 2, are central in many problems in computer science, specially in formal analysis of reactive systems, where vertices of the graph represent states of the system, edges represent transitions, infinite paths of the graph represent behaviors (or non-terminating executions) of the system, and the two players represent the system and the environment, respectively. Games on graphs have been used in many applications related to verification and synthesis of systems, such as, synthesis of systems from specifications and controller-synthesis [30, 54, 55], verification of open systems [8], checking interface compatibility [31], well-formedness of specifications [32], and many others. We will distinguish between results most relevant for *sparse graphs*, where the number of edges m is roughly proportional to the number of vertices n, and *dense graphs* with $m = \Theta(n^2)$. Sparse graphs arise naturally in program verification, as control-flow graphs

© Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger and Veronika Loitzenbauer;
licensed under Creative Commons License CC-BY
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are sparse [57, 28]. Graphs obtained as synchronous product of several components (where each component makes transitions at each step) [45, 23] can lead to dense graphs.

Objectives. Objectives specify the desired set of behaviors of the system. The most basic objective for reactive systems is the *reachability* objective, and the next basic objective is the *Büchi* (*also called liveness or repeated reachability*) objective that was introduced in the seminal work of Büchi [17, 18, 19] for automata over infinite words. Büchi objectives are specified with a target set T and the objective specifies the set of infinite paths in the graph that visit some vertex in the target set infinitely often. Since for reactive systems there are multiple requirements, a very central objective to study for games on graphs is the conjunction of Büchi objectives, which is known as generalized Büchi objective. Finally, currently a very popular class of objectives to specify behaviors for reactive systems is called the GR(1) (generalized reactivity (1)) objectives [53]. A GR(1) objective is an implication between two generalized Büchi objectives.

We present a brief discussion about the significance of the objectives we consider, for a detailed discussion see [26]. The conjunction of liveness objectives is required to specify progress conditions of mutual exclusion protocols, and deterministic Büchi automata can express many important properties of linear-time temporal logic (LTL) (the de-facto logic to specify properties of reactive systems) [47, 46, 9, 44]. The analysis of reactive systems with such objectives naturally gives rise to graph games with generalized Büchi objectives. Finally, graph games with GR(1) objectives have been used in many applications, such as the industrial example of synthesis of AMBA AHB protocol [14, 36] as well as in robotics applications [35, 21].

Basic problem and conditional lower bounds. In this work we consider games on graphs with generalized Büchi and GR(1) objectives, and the basic algorithmic problem is to compute the *winning set*, i.e., the set of starting vertices where player 1 can ensure the objective irrespective of the way player 2 plays; the way player 1 achieves that is called her *winning strategy*. These are core algorithmic problems in verification and synthesis. For the problems we consider, while polynomial-time algorithms are known, there are no super-linear lower bounds. Since for polynomial-time algorithms unconditional super-linear lower bounds are extremely rare in the whole of computer science, we consider *conditional lower bounds*, which assume that for some well-studied problem the known algorithms are optimal up to some lower-order factors. In this work we consider two such well-studied assumptions: (A1) there is no combinatorial[1] algorithm with running time of $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$ to multiply two $n \times n$ Boolean matrices; or (A2) for all $\varepsilon > 0$ there exists a k such that there is no algorithm for the k-CNF-SAT problem that runs in $O(2^{(1-\varepsilon)\cdot n} \cdot \mathrm{poly}(m))$ time, where n is the number of variables and m the number of clauses. These two assumptions have been used to establish lower bounds for several well-studied problems, such as dynamic graph algorithms [3, 5], measuring the similarity of strings [4, 15, 16, 10, 2], context-free grammar parsing [49, 1], and verifying first-order graph properties [52, 61].

Our results. We consider games on graphs with n vertices, m edges, generalized Büchi objectives with k conjunctions, and target sets of size $b_1, b_2, \ldots, b_k$, and GR(1) objectives with k_1 conjunctions in the assumptions and k_2 conjunctions in the guarantee. Our results are as follows.

- *Generalized Büchi objectives.* The classical algorithm for generalized Büchi objectives requires $O(k \cdot \min_{1\le i\le k} b_i \cdot m)$ time. Further there exists an $O(k^2 \cdot n^2)$-time algorithm

[1] Combinatorial here means avoiding fast matrix multiplication [48], see also the discussion in [38].

via a reduction to Büchi games [13, 26].

1. *Dense graphs.* Since $\min_{1 \le i \le k} b_i = O(n)$ and $m = O(n^2)$, the classical algorithm has a worst-case running time of $O(k \cdot n^3)$. First, we present an algorithm with worst-case running time $O(k \cdot n^2)$, which is an improvement for instances with $\min_{1 \le i \le k} b_i \cdot m = \omega(n^2)$. Second, for dense graphs with $m = \Theta(n^2)$ and $k = \Theta(n^c)$ for any $0 < c \le 1$ our algorithm is optimal under (A1); i.e., improving our algorithm for dense graphs would imply a faster (sub-cubic) combinatorial Boolean matrix multiplication algorithm.
2. *Sparse graphs.* We show that for $k = \Theta(n^c)$ for any $0 < c \le 1$, for target sets of constant size, and sparse graphs with $m = \Theta(n^{1+o(1)})$ the basic algorithm is optimal under (A2). In fact, our conditional lower bound under (A2) holds even when each target set is a singleton. Quite strikingly, our result implies that improving the basic algorithm for sparse graphs even with singleton sets would require a major breakthrough in overcoming the exponential barrier for SAT.

In summary, for games on graphs, we present an improved algorithm for generalized Büchi objectives for dense graphs that is optimal under (A1); and show that under (A2) the basic algorithm is optimal for sparse graphs and constant size target sets.
The conditional lower bound for dense graphs means in particular that for unrestricted inputs the dependence of the runtime on n cannot be improved, whereas the bound for sparse graphs makes the same statement for the dependence on m. Moreover, as the graphs in the reductions for our lower bounds can be made acyclic by deleting a single vertex, our lower bounds also apply to a broad range of digraph parameters. For instance let w be the DAG-width [12] of a graph, then there is no $O(f(w) \cdot n^{3-\epsilon})$-time algorithm under (A1) and no $O(f(w) \cdot m^{2-\epsilon})$-time algorithm under (A2).

- *GR(1) objectives.* We present an algorithm for games on graphs with GR(1) objectives that has $O(k_1 \cdot k_2 \cdot n^{2.5})$ running time and improves the previously known $O(k_1 \cdot k_2 \cdot n \cdot m)$-time algorithm [43], for $m > n^{1.5}$. Note that since generalized Büchi objectives are special cases of GR(1) objectives, our conditional lower bounds for generalized Büchi objectives apply to GR(1) objectives as well but are not tight.

All our algorithms can easily be modified to also return the corresponding winning strategies for both players within the same time bounds.

Implications. We discuss the implications of our results.

1. *Comparison with related models.* We compare our results for game graphs to the special case of standard graphs (i.e., games on graphs with only player 1) and the related model of Markov decision processes (MDPs) (with only player 1 and stochastic transitions). First note that for reachability objectives, linear-time algorithms exist for game graphs [11, 39], whereas for MDPs[2] the best-known algorithm has running time $O(\min(n^2, m^{1.5}))$ [29, 26]. For MDPs with reachability objectives, a linear or even $O(m \log n)$ time algorithm is a major open problem, i.e., there exist problems that seem harder for MDPs than for game graphs. Our conditional lower bound results show that under assumptions (A1) and (A2) the algorithmic problem for generalized Büchi objectives is strictly harder for games on graphs as compared to standard graphs and MDPs. More concretely, for $k = \Theta(n)$, (a) for dense graphs ($m = \Theta(n^2)$) and $\min_{1 \le i \le k} b_i = \Omega(\log n)$, our lower bound for games on graphs under (A2) is $\Omega(n^{3-o(1)})$, whereas both the graph and the MDP problems can be solved in $O(n^2)$ time [25, 26]; and (b) for sparse graphs ($m = \Theta(n^{1+o(1)})$) with

[2] For MDPs the winning set refers to the almost-sure winning set that requires that the objective is satisfied with probability 1.

$\min_{1 \le i \le k} b_i = O(1)$, our lower bound for games on graphs under (A1) is $\Omega(m^{2-o(1)})$, whereas the graph problem can be solved in $O(m)$ time and the MDP problem in $O(m^{1.5})$ time [7, 24]; respectively.

2. *Relation to SAT.* We present an algorithm for game graphs with generalized Büchi objectives and show that improving the algorithm would imply a better algorithm for SAT, and thereby establish an interesting algorithmic connection for classical objectives in game graphs and the SAT problem.

Due to the lack of space, some technical details are omitted, but can be found in the attached appendix.

2 Preliminaries

2.1 Basic definitions for Games on Graphs

Game graphs. A *game graph* $\mathcal{G} = ((V, E), (V_1, V_2))$ is a directed graph $G = (V, E)$ with n vertices V and m edges E and a partition of V into *player 1 vertices* V_1 and *player 2 vertices* V_2. Given such a game graph $\mathcal{G}$, we denote with $\overline{\mathcal{G}}$ the game graph where the player 1 and player 2 vertices of $\mathcal{G}$ are interchanged, i.e, $\overline{\mathcal{G}} = ((V, E), (V_2, V_1))$. We use p to denote a player and $\bar{p}$ to denote its opponent. For a vertex $u \in V$, we write $Out(u) = \{v \in V \mid (u, v) \in E\}$ for the set of successor vertices of u and $In(u) = \{v \in V \mid (v, u) \in E\}$ for the set of predecessor vertices of u. If necessary, we refer to the successor vertices in a specific graph by using, e.g., $Out(G, u)$. We denote by $Outdeg(u) = |Out(u)|$ the number of outgoing edges from u, and by $Indeg(u) = |In(u)|$ the number of incoming edges. We assume for technical convenience $Outdeg(u) \ge 1$ for all $u \in V$.

Plays and strategies. A *play* on a game graph is an infinite sequence $\omega = \langle v_0, v_1, v_2, \ldots \rangle$ of vertices such that $(v_\ell, v_{\ell+1}) \in E$ for all $\ell \ge 0$. The set of all plays is denoted by Ω. Given a finite prefix $\omega \in V^* \cdot V_p$ of a play that ends at a player p vertex v, a *strategy* $\sigma : V^* \cdot V_p \to V$ of player p is a function that chooses a successor vertex $\sigma(\omega)$ among the vertices of $Out(v)$. We denote by Σ and Π the set of all strategies for player 1 and player 2 respectively. The play $\omega(v, \sigma, \pi)$ is uniquely defined by a start vertex v, a player 1 strategy $\sigma \in \Sigma$, and a player 2 strategy $\pi \in \Pi$ as follows: $v_0 = v$ and for all $j \ge 0$, if $v_j \in V_1$, then $v_{j+1} = \sigma(\langle v_1, \ldots, v_j \rangle)$, and if $v_j \in V_2$, then $v_{j+1} = \pi(\langle v_1, \ldots, v_j \rangle)$.

Objectives. An objective ψ is a set of plays that is winning for a player. We consider zero-sum games where for a player-1 objective ψ the complementary objective $\Omega \setminus \psi$ is winning for player 2. In this work we consider only *prefix independent objectives*, for which the set of desired plays is determined by the set of vertices $\mathrm{Inf}(\omega)$ that occur *infinitely often* in a play ω. Given a target set $T \subseteq V$, a play ω belongs to the *Büchi objective* Büchi (T) iff $\mathrm{Inf}(\omega) \cap T \neq \emptyset$. For the complementary *co-Büchi objective* we have $\omega \in$ coBüchi (T) iff $\mathrm{Inf}(\omega) \cap T = \emptyset$. A *generalized* (*or conjunctive*) *Büchi objective* is specified by a set of k target sets T_ℓ for $1 \le \ell \le k$ and is satisfied for a play ω iff $\mathrm{Inf}(\omega) \cap T_\ell \neq \emptyset$ for *all* $1 \le \ell \le k$. Its complementary objective is the *disjunctive co-Büchi objective* that is satisfied iff $\mathrm{Inf}(\omega) \cap T_\ell = \emptyset$ for *one of* the k target sets. A *generalized reactivity-1* (GR(1)) objective is specified by two generalized Büchi objectives, $\bigwedge_{t=1}^{k_1}$ Büchi (L_t) and $\bigwedge_{\ell=1}^{k_2}$ Büchi (U_ℓ), and is satisfied if whenever the first generalized Büchi objective holds, then also the second generalized Büchi objective holds; in other words, either $\bigvee_{t=1}^{k_1}$ coBüchi (L_t) holds, or $\bigwedge_{\ell=1}^{k_2}$ Büchi (U_ℓ) holds.

All the games in this paper will be given by a game graph $\mathcal{G}$ and an objective ψ for player 1 (player 2 has the complementary objective $\Omega \setminus \psi$).

Winning strategies and sets. A strategy σ is winning for player p at a start vertex v if the resulting play is winning for player p irrespective of the strategy of his opponent, player $\bar{p}$, i.e., $\omega(v, \sigma, \pi) \in \psi$ for all π. A vertex v belongs to the *winning set* W_p of player p if player p has a winning strategy from v. Every vertex is winning for exactly one of the two players [50]. When required for explicit reference of a specific game graph $\mathcal{G}$ and objective ψ, we use $W_p(\mathcal{G}, \psi)$ to refer to the winning sets.

Closed sets and attractors. A set $U \subseteq V$ is *p-closed* (in $\mathcal{G}$) if for all p-vertices u in U we have $Out(u) \subseteq U$ and for all $\bar{p}$-vertices v in U there exists a vertex $w \in Out(v) \cap U$. Note that player $\bar{p}$ can ensure that a play that currently ends in a p-closed set never leaves the p-closed set against any strategy of player p by choosing an edge (v, w) with $w \in Out(v) \cap U$ whenever the current vertex v is in $U \cap V_{\bar{p}}$ [62]. Given a game graph $\mathcal{G}$ and a p-closed set U, we denote by $\mathcal{G}[U]$ the game graph induced by the set of vertices U. Note that given that in $\mathcal{G}$ each vertex has at least one outgoing edge, the same property holds for $\mathcal{G}[U]$. We further use the shortcut $\mathcal{G} \setminus X$ to denote $\mathcal{G}[V \setminus X]$.

In a game graph $\mathcal{G}$, a *p-attractor* $Attr_p(\mathcal{G}, U)$ of a set $U \subseteq V$ is the set of vertices from which player p has a strategy to reach U against all strategies of player $\bar{p}$ [62]. We have that $U \subseteq Attr_p(\mathcal{G}, U)$. A p-attractor can be constructed inductively as follows: Let $R_0 = U$; and for all $j \geq 0$ let $R_{j+1} = R_j \cup \{v \in V_p \mid Out(v) \cap R_j \neq \emptyset\} \cup \{v \in V_{\bar{p}} \mid Out(v) \subseteq R_j\}$. Then $Attr_p(\mathcal{G}, U) = \bigcup_{j \geq 0} R_j$. The computation of attractors can be done in linear time [11, 39].

Dominions. A set of vertices $D \subseteq V$ is a player-p *dominion* if $D \neq \emptyset$ and player p has a winning strategy from every vertex in D that also ensures only vertices in D are visited. The notion of dominions was introduced by [42]. Note that a player-p dominion is also a $\bar{p}$-closed set and the p-attractor of a player-p dominion is again a player-p dominion.

▶ **Lemma 1.** *The following assertions hold for game graphs $\mathcal{G}$ where each vertex has at least one outgoing edge. The assertions referring to winning sets hold for graph games with prefix independent objectives. Let $X \subseteq V$.*

1. *The set $V \setminus Attr_p(\mathcal{G}, X)$ is p-closed on $\mathcal{G}$ [62, Lemma 4].*
2. *Let X be p-closed on $\mathcal{G}$. Then $W_{\bar{p}}(\mathcal{G}[X]) \subseteq W_{\bar{p}}(\mathcal{G})$ [42, Lemma 4.4].*
3. *Let X be a subset of the winning set $W_p(\mathcal{G})$ of player p and let A be its p-attractor $Attr_p(\mathcal{G}, X)$. Then the winning set $W_p(\mathcal{G})$ of the player p is the union of A and the winning set $W_p(\mathcal{G}[V \setminus A])$, and the winning set $W_{\bar{p}}(\mathcal{G})$ of the opponent $\bar{p}$ is equal to $W_{\bar{p}}(\mathcal{G}[V \setminus A])$ [42, Lemma 4.5].*

2.2 Conjectured Lower Bounds

While classical complexity results are based on complexity-theoretical assumptions about relationships between complexity classes, e.g., P ≠ NP, polynomial lower bounds are often based on widely believed, conjectured lower bounds about well studied algorithmic problems. We next discuss the popular conjectures that will be the basis for our lower bounds.

First, we consider conjectures on Boolean matrix multiplication [58, 3] and triangle detection [3] in graphs, which build the basis for our lower bounds on dense graphs. A triangle in a graph is a triple x, y, z of vertices such that $(x, y), (y, z), (z, x) \in E$.

▶ **Conjecture 2** (Combinatorial Boolean Matrix Multiplication Conjecture (BMM))**.** *There is no $O(n^{3-\varepsilon})$ time combinatorial algorithm for computing the Boolean product of two $n \times n$ matrices for any $\varepsilon > 0$.*

▶ **Conjecture 3** (Strong Triangle Conjecture (STC))**.** *There is no $O(n^{3-\varepsilon})$ time combinatorial algorithm that can detect whether a graph contains a triangle for any $\varepsilon > 0$.*

BMM is equivalent to STC [58]. A weaker assumption, without the restriction to combinatorial algorithms, is that detecting a triangle in a graph takes super-linear time.

Second, we consider the Strong Exponential Time Hypothesis [40, 20] and the Orthogonal Vectors Conjecture [6], the former dealing with satisfiability in propositional logic and the latter with the *Orthogonal Vectors Problem.*

The Orthogonal Vectors Problem (OV). Given two sets S_1, S_2 of d-bit vectors with $|S_i| \leq N$ and $d \in \Theta(\log N)$, are there $u \in S_1$ and $v \in S_2$ such that $\sum_{i=1}^{d} u_i \cdot v_i = 0$?

▶ **Conjecture 4** (Strong Exponential Time Hypothesis (SETH))**.** *For each $\varepsilon > 0$ there is a k such that k-CNF-SAT on n variables and m clauses cannot be solved in time $O(2^{(1-\varepsilon)n} \operatorname{poly}(m))$.*

▶ **Conjecture 5** (Orthogonal Vectors Conjecture (OVC))**.** *There is no $O(N^{2-\varepsilon})$ time algorithm for the Orthogonal Vectors Problem for any $\varepsilon > 0$.*

SETH implies OVC [59], i.e., whenever a problem is hard assuming OVC, it is also hard when assuming SETH. Hence, it is preferable to use OVC for proving lower bounds. Finally, to the best of our knowledge, no such relations between the former two conjectures and the latter two conjectures are known.

▶ Remark. The conjectures that no *polynomial* improvements over the best known running times are possible do not exclude improvements by sub-polynomial factors such as poly-logarithmic factors or factors of, e.g., $2^{\sqrt{\log n}}$ as in [60].

3 Algorithms for Generalized Büchi Games

For generalized Büchi games we first present the basic algorithm that follows from the results of [33, 51, 62]. The basic algorithm runs in time $O(knm)$. We then improve it to an $O(k \cdot n^2)$-time algorithm by exploiting ideas from the $O(n^2)$-time algorithm for Büchi games in [25]. The basic algorithm is fast for instances where one Büchi set, say T_1, is small, i.e., the algorithm runs in time $O(k \cdot b_1 \cdot m)$ time, where $b_1 = |T_1|$. Generalized Büchi games can also be solved via a reduction to Büchi games [13], which yields an $O(k^2 n^2)$ time algorithm when combined with the $O(n^2)$-time Büchi algorithm [25].

Our algorithms iteratively identify sets of vertices that are winning for player 2, i.e., player-2 dominions, and remove them from the graph. We denote the sets in the jth-iteration with superscript j, in particular $\mathcal{G}^1 = \mathcal{G}$, where $\mathcal{G}$ is the input game graph, G^j is the graph of $\mathcal{G}^j$, V^j is the vertex set of G^j, and $T_\ell^j = V^j \cap T_\ell$. We also use $\{T_\ell^j\}$ to denote the list of Büchi sets $(T_1^j, T_2^j, \ldots, T_k^j)$, in particular when updating all the sets in a uniform way.

Basic Algorithm. For each set U that is closed for player 1 we have that from each vertex $u \in U$ player 2 has a strategy to ensure that the play never leaves U [62]. Thus, if there is a Büchi set T_ℓ with $T_\ell \cap U = \emptyset$, then the set U is a player-2 dominion. Moreover, if U is a player-2 dominion, also the attractor $Attr_2(\mathcal{G}, U)$ of U is a player-2 dominion. The basic algorithm proceeds as follows. It iteratively computes vertex sets S^j closed for player 1 that do not intersect with one of the Büchi sets. If such a player-2 dominion S^j is found, then all vertices of $Attr_2(\mathcal{G}^j, S^j)$ are marked as winning for player 2 and removed from the game graph; the remaining game graph is denoted by $\mathcal{G}^{j+1}$. To find a player-2 dominion S^j, for each $1 \leq \ell \leq k$ the attractor $Y_\ell^j = Attr_1(\mathcal{G}^j, T_\ell^j)$ of the Büchi set T_ℓ^j is determined. If for some ℓ the complement of Y_ℓ^j is not empty, then we assign $S^j = V^j \setminus Y_\ell^j$ for the smallest such ℓ. The algorithm terminates if in some iteration j for each $1 \leq \ell \leq k$ the attractor Y_ℓ^j contains all vertices of V^j. In this case the set V^j is returned as the winning set of player 1. The winning strategy of player 1 from these vertices is then a combination of the attractor strategies to the sets T_ℓ^j.

Algorithm GenBuchiGame: Algorithm for Generalized Büchi Games

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$ and objective $\bigwedge_{1 \le \ell \le k}$ Büchi (T_ℓ)
Output: Winning set of player 1

1 $\mathcal{G}^1 \leftarrow \mathcal{G}$; $\{T_\ell^1\} \leftarrow \{T_\ell\}$; $j \leftarrow 0$
2 **repeat**
3 $\quad j \leftarrow j + 1$
4 $\quad$ **for** $i \leftarrow 1$ **to** $\lceil \log_2 n \rceil$ **do**
5 $\quad\quad$ construct G_i^j
6 $\quad\quad Z_i^j \leftarrow \{v \in V_2 \mid \mathit{Outdeg}(G_i^j, v) = 0\} \cup \{v \in V_1 \mid \mathit{Outdeg}(G_i^j, v) > 2^i\}$
7 $\quad\quad$ **for** $1 \le \ell \le k$ **do**
8 $\quad\quad\quad Y_{\ell,i}^j \leftarrow \mathit{Attr}_1(G_i^j, T_\ell^j \cup Z_i^j)$
9 $\quad\quad\quad S^j \leftarrow V^j \setminus Y_{\ell,i}^j$
10 $\quad\quad\quad$ **if** $S^j \neq \emptyset$ **then** player 2 dominion found, continue with line 11
11 $\quad D^j \leftarrow \mathit{Attr}_2(G^j, S^j)$
12 $\quad \mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$; $\{T_\ell^{j+1}\} \leftarrow \{T_\ell^j \setminus D^j\}$
13 **until** $D^j = \emptyset$
14 **return** V^j

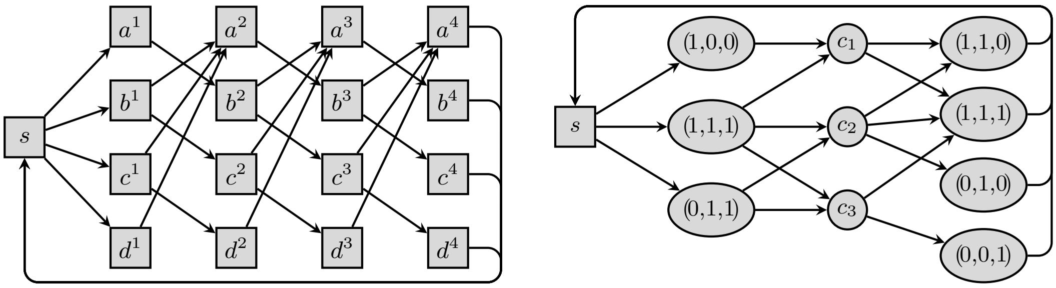
▶ **Theorem 6.** *The basic algorithm for generalized Büchi games computes the winning set for player 1 in $O(k \cdot \min_{1 \le \ell \le k} b_\ell \cdot m)$ time, where $b_\ell = |T_\ell|$, and thus also in $O(knm)$ time.*

Our Improved Algorithm. The $O(k \cdot n^2)$-time Algorithm GenBuchiGame for generalized Büchi games combines the basic algorithm described above with the method used for the $O(n^2)$-time Büchi game algorithm [26], called *hierarchical graph decomposition* [37]. The hierarchical graph decomposition defines for a directed graph $G = (V, E)$ and integers $1 \le i \le \lceil \log_2 n \rceil$ the graphs $G_i = (V, E_i)$. Assume the incoming edges of each vertex in G are given in some fixed order in which first the edges from vertices of V_2 and then the edges from vertices of V_1 are listed. The set of edges E_i contains all the outgoing edges of each $v \in V$ with $\mathit{Outdeg}(G, v) \le 2^i$ and the first 2^i incoming edges of each vertex. Note that $G = G_{\lceil \log_2 n \rceil}$ and $|E_i| \in O(n \cdot 2^i)$. The runtime analysis uses that we can identify small player-2 dominions (i.e., player-1 closed sets that do not intersect one of the target sets) that contain $O(2^i)$ vertices by only looking at G_i. The algorithm first searches for such a set S^j in G_i for $i = 1$ and each target set and then increases i until the search is successful. In this way the time spent for the search is proportional to $k \cdot n$ times the number of vertices in the found dominion, which yields a total runtime bound of $O(k \cdot n^2)$. To obtain the $O(k \cdot n^2)$ running time bound, it is crucial to put the loop over the different Büchi sets as the innermost part of the algorithm. Given a game graph $\mathcal{G} = (G, (V_1, V_2))$, we denote by $\mathcal{G}_i$ the game graph where G was replaced by G_i from the hierarchical graph decomposition, i.e., $\mathcal{G}_i = (G_i, (V_1, V_2))$.

Properties of hierarchical graph decomposition. The following lemma identifies two essential properties of the hierarchical graph decomposition. The first is crucial for correctness: When searching in $\mathcal{G}_i$ for a player-1 closed set that does not contain one of the target sets, we can ensure that such a set is also closed for player 1 in $\mathcal{G}$ by excluding certain vertices that are missing outgoing edges in $\mathcal{G}_i$ from the search. The second is crucial for the runtime: Whenever the basic algorithm would remove (i.e., identify as winning for player 2) a set with at most 2^i vertices, then we can identify this set also by searching in $\mathcal{G}_i$ instead of $\mathcal{G}$.

▶ **Lemma 7.** *Let $\mathcal{G} = (G = (V, E), (V_1, V_2))$ be a game graph and $\{G_i\}$ its hierarchical graph decomposition. For $1 \le i \le \lceil \log_2 n \rceil$ let Z_i be the set consisting of the player 2 vertices that have no outgoing edge in $\mathcal{G}_i$ and the player 1 vertices with $> 2^i$ outgoing edges in $\mathcal{G}$.*

1. *If a set $S \subseteq V \setminus Z_i$ is closed for player 1 in $\mathcal{G}_i$, then S is closed for player 1 in $\mathcal{G}$.*



(a) Reduction 10 applied to $G = (\{a, b, c, d\}, \{(a, b), (b, a), (b, c), (c, a), (c, d), (d, a)\})$.

(b) Reduction 12 applied to $S_1 = \{(1, 0, 0), (1, 1, 1), (0, 1, 1)\}$ and $S_2 = \{(1, 1, 0), (1, 1, 1), (0, 1, 0), (0, 0, 1)\}$.

Figure 1 Illustration of Reductions 10 and 12.

2. *If a set $S \subseteq V$ is closed for player 1 in $\mathcal{G}$ and $|Attr_2(\mathcal{G}, S)| \leq 2^i$, then (i) $\mathcal{G}_i[S] = \mathcal{G}[S]$, (ii) the set S is in $V \setminus Z_i$, and (iii) S is closed for player 1 in $\mathcal{G}_i$.*

With the above lemma we can show that whenever a player-2 dominion is found in $\mathcal{G}_i$ but not in $\mathcal{G}_{i-1}$, then at least $\Omega(2^i)$ vertices are removed from the maintained game graph. Together with a runtime bound of $O(k \cdot 2^i \cdot n)$ for the search, this yields a total runtime of $O(k \cdot n)$ per vertex, i.e., time $O(k \cdot n^2)$ in total.

▶ **Theorem 8.** *Algorithm* GenBuchiGame *computes the winning set of player 1 in a generalized Büchi game in $O(k \cdot n^2)$ time.*

4 Conditional Lower bounds for Generalized Büchi Games

In this section we present two conditional lower bounds, one for dense graphs ($m = \Theta(n^2)$) based on STC & BMM, and one for sparse graphs ($m = \Theta(n^{1+o(1)})$) based on OVC & SETH.

▶ **Theorem 9.** *There is no combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$-time algorithm (for any $\epsilon > 0$) for generalized Büchi games under Conjecture 3 (i.e., unless STC & BMM fail).*

The result can be obtained from a reduction from triangle detection to disjunctive co-Büchi objectives on graphs in [22], and we present the reduction in terms of game graphs below and illustrate it on an example in Figure 1a.

▶ **Reduction 10.** *Given a graph $G = (V, E)$ (for triangle detection), we build a game graph $\mathcal{G}' = (G = (V', E'), (V_1, V_2))$ (for generalized Büchi objectives) as follows. As vertices V' we have four copies V^1, V^2, V^3, V^4 of V and a vertex s. A vertex $v^i \in V^i, i \in \{1, 2, 3\}$ has an edge to a vertex $u^{i+1} \in V^{i+1}$ iff $(v, u) \in E$. Moreover, s has an edge to all vertices of V^1 and all vertices of V^4 have an edge to s. All the vertices are owned by player 2, i.e., $V_1 = \emptyset$ and $V_2 = V$. Finally, we consider the generalized Büchi objective $\bigwedge_{v \in V}$ Büchi(T_v), with $T_v = (V^1 \setminus \{v^1\}) \cup (V^4 \setminus \{v^4\})$.*

We have that there is a triangle in the graph G if and only if the vertex s is winning for player 2 in the generalized Büchi game on $\mathcal{G}'$. Notice that the sets T_v in the above reduction are of linear size but can be reduced to logarithmic size using a construction from [22]. Next we present an $\Omega(m^{2-o(1)})$ lower bound for generalized Büchi objectives.

▶ **Theorem 11.** *There is no $O(m^{2-\epsilon})$ or $O(\min_{1 \leq i \leq k} b_i \cdot (k \cdot m)^{1-\epsilon})$-time algorithm (for any $\epsilon > 0$) for generalized Büchi games under Conjecture 5 (i.e., unless OVC & SETH fail).*

The above theorem is by a linear time reduction from OV provided below (cf. Figure 1b).

▶ **Reduction 12.** *Given two sets S_1, S_2 of d-dimensional vectors, we build the following game graph. The vertices V of the graph G are given by a start vertex s, vertices S_1 and S_2 representing the sets of vectors, and vertices $\mathcal{C} = \{c_i \mid 1 \leq i \leq d\}$ representing the coordinates. The edges E of G are defined as follows: the start vertex s has an edge to every vertex of S_1 and every vertex of S_2 has an edge to s; further for each $x \in S_1$ there is an edge to $c_i \in \mathcal{C}$ iff $x_i = 1$ and for each $y \in S_2$ there is an edge from $c_i \in \mathcal{C}$ iff $y_i = 1$. The set of vertices V is partitioned into player 1 vertices $V_1 = S_1 \cup S_2 \cup \mathcal{C}$ and player 2 vertices $V_2 = \{s\}$. Finally, the generalized Büchi objective is given by $\bigwedge_{v \in S_2}$ Büchi(T_v) with $T_v = \{v\}$.*

▶ **Lemma 13.** *Given two sets S_1, S_2 of d-dimensional vectors and the corresponding graph game $\mathcal{G}$ given by Reduction 12 with $T_v = \{v\}$ for $v \in S_2$, (1) there exist orthogonal vectors $x \in S_1$ and $y \in S_2$ if and only if (2) $s \notin W_1(\mathcal{G}, \bigwedge_{v \in S_2}$ Büchi$(T_v))$.*

Proof. W.l.o.g. we assume that the 1-vector, i.e., the vector with all coordinates being 1, is contained in S_2 (adding the 1-vector does not change the result of the OV instance), which guarantees that each vertex $c \in \mathcal{C}$ in the construction below has at least 1 outgoing edge. Then a play in the game graph $\mathcal{G}$ proceeds as follows. Starting from s, player 2 chooses a vertex $x \in S_1$; then player 1 first picks a vertex $c \in \mathcal{C}$ and then a vertex $y \in S_2$; then the play goes back to s (at each $y \in S_2$ player 1 has only this choice), starting another cycle of the play. (1)⇒(2): Assume there are orthogonal vectors $x \in S_1$ and $y \in S_2$. Now player 2 can satisfy coBüchi (T_y) by simply going to x whenever the play is in s. Then player 1 can choose some adjacent $c \in \mathcal{C}$ and then some adjacent vertex in S_2, but as x and y are orthogonal, this c is not connected to y. Thus the play will never visit y. (2)⇒(1): By the fact that $W_1 = V \setminus W_2$ [50] we have that (2) is equivalent to $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2}$ Büchi $(T_v))$. Assume $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2}$ Büchi $(T_v))$ and consider a corresponding strategy for player 2 that satisfies $\bigvee_{v \in S_2}$ coBüchi (T_v). Notice that the graph is such that player 2 has to visit at least one of the vertices v in S_1 infinitely often. Moreover, for such a vertex v then player 1 can visit all vertices $v' \in S_2$ that correspond to non-orthogonal vectors infinitely often. That is, if v has no orthogonal vector, player 1 can satisfy all the Büchi constraints, a contradiction to our assumption that $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2}$ Büchi $(T_v))$. Thus there must be a vector $x \in S_1$ such that there exists a vector $y \in S_2$ that is orthogonal to x. ◀

Let $N = \max(|S_1|, |S_2|)$. The number of vertices in the game graph, constructed by Reduction 12, is $O(N)$, the number of edges m is $O(N \log N)$ (recall that $d \in O(\log N)$), we have $k \in \Theta(N)$ target sets, each of size 1, and the construction can be performed in $O(N \log N)$ time. Thus, if we would have an $O(m^{2-\epsilon})$ or $O(\min_{1 \leq i \leq k} b_i \cdot (k \cdot m)^{1-\epsilon})$ time algorithm for any $\epsilon > 0$, we would immediately get an $O(N^{2-\epsilon})$ algorithm for OV, which contradicts OVC (and thus SETH).

▶ Remark. Notice that the lower bounds apply to instances with $k \in \Theta(n^c)$ for arbitrary $0 < c \leq 1$, although the reductions produce graphs with $k \in \Theta(n)$. This is because of the specific type of the constructed instances, for which each $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for $k \in \Theta(n^c)$ also implies an $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for $k \in \Theta(n)$.

5 Generalized Reactivity-1 Games

GR(1) games deal with an objective of the form $\bigwedge_{t=1}^{k_1}$ Büchi $(L_t) \to \bigwedge_{\ell=1}^{k_2}$ Büchi (U_ℓ) and can be solved in $O(k_1 k_2 \cdot m \cdot n)$ time [43] with an extension of the progress measure algorithm of [41] and in $O((k_1 k_2 \cdot n)^{2.5})$ time by combining the reduction to one-pair Streett objectives by [13] with the algorithm of [27]. In this section we develop an $O(k_1 k_2 \cdot n^{2.5})$-time algorithm by modifying the algorithm of [43] to compute dominions. We further use our $O(k \cdot n^2)$-time algorithm for generalized Büchi games with $k = k_1$ as a subroutine.

We first describe a basic, direct algorithm for GR(1) games that is based on repeatedly identifying player-2 dominions in generalized Büchi games. We then show how the progress measure algorithm of [43] can be modified to identify player-2 dominions in generalized Büchi games with k_1 Büchi objectives in time proportional to $k_1 \cdot m$ times the size of the dominion. In the $O(k_1 k_2 \cdot n^{2.5})$-time algorithm we use the modified progress measure algorithm in combination with the hierarchical graph decomposition of [26, 27] to identify dominions that contain up to $\sqrt{n}$ vertices and our $O(k_1 \cdot n^2)$-time algorithm for generalized Büchi games to identify dominions with more than $\sqrt{n}$ vertices. Each time we search for a dominion we might have to consider k_2 different subgraphs.

We denote the sets in the jth-iteration of our algorithms with superscript j, in particular $\mathcal{G}^1 = \mathcal{G}$, where $\mathcal{G}$ is the input game graph, G^j is the graph of $\mathcal{G}^j$, V^j is the vertex set of G^j, $V_1^j = V_1 \cap V^j$, $V_2^j = V_2 \cap V^j$, $L_t^j = L_t \cap V^j$, and $U_\ell^j = U_\ell \cap V^j$.

Basic Algorithm. Similar to generalized Büchi games, the basic algorithm for GR(1) games identifies a player-2 dominion S^j, removes the dominion and its player-2 attractor D^j from the graph, and recurses on the remaining game graph $\mathcal{G}^{j+1} = \mathcal{G}^j \setminus D^j$. If no player-2 dominion is found, the remaining set of vertices V^j is returned as the winning set of player 1. Given the set S^j is indeed a player-2 dominion, the correctness of this approach follows from Lemma 1(3). A player-2 dominion in $\mathcal{G}^j$ is identified as follows: For each $1 \le \ell \le k_2$ first the player-1 attractor Y_ℓ^j of U_ℓ^j is temporarily removed from the graph. Then a generalized Büchi game with target sets $L_1^j, \ldots, L_{k_1}^j$ is solved on $\overline{\mathcal{G}^j \setminus Y_\ell^j}$. The generalized Büchi player in this game corresponds to player 2 in the GR(1) game and his winning set to a player-2 dominion in the GR(1) game. Note that $V^j \setminus Y_\ell^j$ is player-1 closed and does not contain U_ℓ^j. Thus if in the game induced by the vertices of $V^j \setminus Y_\ell^j$ player 2 can win w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1} \text{Büchi}(L_t^j)$, then these vertices form a player-2 dominion in the GR(1) game. Further, we can show that when a player-2 dominion in the GR(1) games on $\mathcal{G}^j$ exists, then for one of the sets U_ℓ^j the winning set of the generalized Büchi game on $\overline{\mathcal{G}^j \setminus Y_\ell^j}$ is non-empty; otherwise we can construct a winning strategy of player 1 for the GR(1) game on $\mathcal{G}^j$. Note that this algorithm computes a player-2 dominion $O(k_2 \cdot n)$ often using our $O(k_1 \cdot n^2)$-time generalized Büchi Algorithm GenBuchiGame.

▶ **Theorem 14.** *The basic algorithm for GR(1) games computes the winning set for player 1 in $O(k_1 \cdot k_2 \cdot n^3)$ time.*

Improved Algorithm. The overall structure of our $O(k_1 k_2 \cdot n^{2.5})$-time algorithm for GR(1) games (see Algorithm GR(1)Game) is the same as for the basic algorithm: We search for a player-2 dominion S^j and if one is found, then its player-2 attractor D^j is determined and removed from the current game graph $\mathcal{G}^j$ (with $\mathcal{G}^1 = \mathcal{G}$) to create the game graph for the next iteration, $\mathcal{G}^{j+1}$. If no player-2 dominion exists, then the remaining vertices are returned as the winning set of player 1. The difference to the basic algorithm lies in the way player-2 dominions are searched. Two different procedures are used for this purpose: First we search for "small" dominions with the subroutine `kGenBüchiDominion`. If no small dominions exist, then we search for player-2 dominions as in the basic algorithm. The guarantee that we find a "large" dominion allows us to bound the number of times the second case can happen.

Progress Measure Algorithm. In the Procedure `kGenBüchiDominion` we use a subroutine that finds in a generalized Büchi game all dominions of the generalized Büchi player that have size at most h (where h is an input parameter). This subroutine is based on a so-called *progress*

Algorithm GR(1)Game: Algorithm for GR(1) Games

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$, Obj. $\bigwedge_{t=1}^{k_1}$ Büchi $(L_t) \rightarrow \bigwedge_{\ell=1}^{k_2}$ Büchi (U_ℓ)
Output: Winning set of player 1

$\mathcal{G}^1 \leftarrow \mathcal{G}$; $\{U_\ell^1\} \leftarrow \{U_\ell\}$; $\{L_t^1\} \leftarrow \{L_t\}$
$j \leftarrow 0$
repeat
  $j \leftarrow j + 1$
  $S^j \leftarrow$ `kGenBüchiDominion`$(\mathcal{G}^j, \{U_\ell^j\}, \{L_t^j\}, \sqrt{n})$
  if $S^j = \emptyset$ **then**
    for $1 \leq \ell \leq k_2$ **do**
      $Y_\ell^j \leftarrow Attr_1(\mathcal{G}^j, U_\ell^j)$
      $S^j \leftarrow$ `GenBüchiGame`$(\overline{\mathcal{G}^j \setminus Y_\ell^j}, \bigwedge_{\ell=1}^{k_1}$ Büchi $\left(L_t^j \setminus Y_\ell^j\right))$
      if $S^j \neq \emptyset$ **then break**
  $D^j \leftarrow Attr_2(\mathcal{G}^j, S^j)$
  $\mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$; $\{U_\ell^{j+1}\} \leftarrow \{U_\ell^j \setminus D^j\}$; $\{L_t^{j+1}\} \leftarrow \{L_t^j \setminus D^j\}$
until $D^j = \emptyset$
return V^j

Procedure `kGenBüchiDominion`$(\mathcal{G}^j, \{U_\ell^j\}, \{L_t^j\}, h_{max})$
  for $i \leftarrow 1$ **to** $\lceil \log_2(h_{max}) \rceil$ **do**
    construct G_i^j
    $Z_i^j \leftarrow \{v \in V_2 \mid Outdeg(G_i^j, v) = 0\} \cup \{v \in V_1 \mid Outdeg(G_i^j, v) > 2^i\}$
    for $1 \leq \ell \leq k_2$ **do**
      $Y_{i,\ell}^j \leftarrow Attr_1(\mathcal{G}_i^j, U_\ell^j \cup Z_i^j)$
      $X_{i,\ell}^j \leftarrow$ `GenBüchiProgressMeasure`$(\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}, \bigwedge_{\ell=1}^{k_1}$ Büchi $\left(L_t^j \setminus Y_{i,\ell}^j\right), 2^i)$
      if $X_{i,\ell}^j \neq \emptyset$ **then return** $X_{i,\ell}^j$
  return $\emptyset$

measure for generalized Büchi objectives which is a special instance of the progress measure for GR(1) objectives presented in [43, Section 3.1], which itself is based on [41]. We modify the progress measure to efficiently identify dominions of size at most h (instead of computing the whole winning set) by restricting the range of allowed values for the progress measure functions similar to [56]. Finally, we give an $O(k \cdot h \cdot m)$-time algorithm for computing the progress measure functions based on [34, 43] (details are provided in the appendix).

▶ **Theorem 15.** *For a game graph $\mathcal{G}$ and objective $\psi = \bigwedge_{1 \leq \ell \leq k}$ Büchi(T_ℓ), there is an $O(k \cdot h \cdot m)$ time procedure* `GenBüchiProgressMeasure`$(\mathcal{G}, \psi, h)$ *that either returns a player-1 dominion or the empty set, and, if there is at least one player-1 dominion of size $\leq h$ then returns a player-1 dominion containing all player-1 dominions of size $\leq h$.*

Procedure `kGenBüchiDominion`*.* The procedure `kGenBüchiDominion` searches for player-2 dominions in the GR(1) game, and returns some dominion if there exists a dominion D with $|Attr_2(\mathcal{G}, D)| \leq h_{\max}$. To this end we again consider generalized Büchi games with target sets $L_1^j, \ldots, L_{k_1}^j$, where the generalized Büchi player corresponds to player 2 in the GR(1) game. We use the same hierarchical graph decomposition as for Algorithm GenBuchiGame: Let the incoming edges of each vertex be ordered such that the edges from vertices of V_2 come first; for a given game graph $\mathcal{G}^j$ the graph G_i^j contains all vertices of $\mathcal{G}^j$, for each vertex its first 2^i incoming edges, and for each vertex with outdegree at most 2^i all its outgoing edges. The set Z_i^j contains all vertices of V_1 with outdegree larger than 2^i and all vertices of V_2 that have no outgoing edge in G_i^j. We start with $i = 1$ and increase i by one as long as no dominion was found. For a given i we perform the following operations for each $1 \leq \ell \leq k_2$: First the player 1 attractor $Y^j_{i,\ell}$ of $U^j_\ell \cup Z^j_i$ is determined. Then we search for player-1 dominions on $\overline{\mathcal{G}^j_i \setminus Y^j_{i,\ell}}$ w.r.t. the objective $\bigwedge_{t=1}^{k_1}$ Büchi$\,(L_t)$ with the generalized Büchi progress measure algorithm and parameter $h = 2^i$, i.e., by Theorem 15 the progress measure algorithm returns all generalized Büchi dominions in $\overline{\mathcal{G}^j_i \setminus Y^j_{i,\ell}}$ of size at most h.

The following lemma shows how the properties of the hierarchical graph decomposition extend to GR(1) games. The first part is crucial for correctness: Every non-empty set found by the progress measure algorithm on $\overline{\mathcal{G}^j_i \setminus Y^j_{i,\ell}}$ for some i and ℓ is indeed a player-2 dominion in the GR(1) game. The second part is crucial for the runtime argument: Whenever the basic algorithm for GR(1) games would identify a player-2 dominion D with $|Attr_2(\mathcal{G}, D)| \leq 2^i$, then D is also a generalized Büchi dominion in $\overline{\mathcal{G}^j_i \setminus Y^j_{i,\ell}}$ for some ℓ.

▶ **Lemma 16.** *Let the notation be as in Algorithm* GR(1)Game.

1. *Every $X^j_{i,\ell} \neq \emptyset$ is a player-2 dominion in the GR(1) game on $\mathcal{G}^j$ with $X^j_{i,\ell} \cap U^j_\ell = \emptyset$.*
2. *If for player 2 there exists in $\mathcal{G}^j$ a dominion D w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi(L^j_t) such that $D \cap U^j_\ell = \emptyset$ for some $1 \leq \ell \leq k_2$ and $|Attr_2(\mathcal{G}^j, D)| \leq 2^i$, then D is a dominion w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi$(L^j_t \setminus Y^j_{i,\ell})$ in $\mathcal{G}^j_i \setminus Y^j_{i,\ell}$.*

From this we can draw the following two corollaries: (1) When we had to go up to i^* in the graph decomposition to find a dominion, then its attractor has size at least 2^{i^*-1} and (2) when `kGenBüchiDominion` returns an empty set, then all player-2 dominions in the current game graph have more than $h_{\max} = \sqrt{n}$ vertices. In the second case either no player-2 dominion exists or the subsequent call to `GenBüchiGame` returns one with more than $\sqrt{n}$ vertices, which can happen at most $O(\sqrt{n})$ times. Together with (1), this means we can (a) charge the time spent in `kGenBüchiDominion` to the vertices in the dominion identified in this iteration of the repeat-until loop (except for the last iteration) and (b) bound the number of calls to `GenBüchiGame` with $O(\sqrt{n})$.

▶ **Theorem 17.** *Algorithm* GR(1)Game *computes the winning set of player 1 in a GR(1) game in $O(k_1 \cdot k_2 \cdot n^{2.5})$ time.*

6 Conclusion

In this work we present improved algorithms for generalized Büchi and GR(1) objectives, and conditional lower bounds for generalized Büchi objectives. The existing upper bounds and our conditional lower bounds are tight for (a) for dense graphs, and (b) sparse graphs with constant size target sets. Two interesting open questions are as follows: (1) For sparse graphs with $\theta(n)$ many target sets of size $\theta(n)$ the upper bounds are cubic, whereas the conditional lower bound is quadratic, and closing the gap is an interesting open question. (2) For GR(1) objectives we obtain the conditional lower bounds from generalized Büchi objectives, which are not tight in this case; whether better (conditional) lower bounds can be established also remains open.

Acknowledgements. K. C., M. H., and W. D. are partially supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-003. K. C. is partially supported by the Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE) and an ERC Start grant (279307: Graph Games). For W. D., M. H., and V. L. the research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *FOCS*, pages 98–117, 2015.
2. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results For LCS and other Sequence Similarity Measures. In *FOCS*, pages 59–78, 2015.
3. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443, 2014.
4. Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP 2014, Proceedings, Part I*, pages 39–51, 2014.
5. Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50, 2015.
6. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, pages 377–391, 2016.
7. Rajeev Alur and Thomas A. Henzinger. Computer-aided verification, 2004. Unpublished, available at `http://www.cis.upenn.edu/group/cis673/`.
8. Rajeev Alur, Thomas. A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.
9. Rajeev Alur and Salvatore La Torre. Deterministic generators and games for ltl fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.
10. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *STOC*, pages 51–58, 2015.
11. Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, pages 241–259, 1980.
12. Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. Dag-width and parity games. In *STACS*, pages 524–536, 2006.
13. Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Robustness in the presence of liveness. In *CAV*, pages 410–424, 2010.
14. Roderick Bloem, Stefan J. Galler, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Martin Weiglhofer. Interactive presentation: Automatic hardware synthesis from specifications: a case study. In *DATE*, pages 1188–1193, 2007.
15. Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *FOCS*, pages 661–670, 2014.
16. Karl Bringmann and Marvin Künnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *FOCS*, pages 79–97, 2015.
17. J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
18. J. Richard Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Proceedings of the First International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–11. Stanford University Press, 1962.
19. J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the AMS*, 138:295–311, 1969.
20. Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *IWPEC*, pages 75–85, 2009.
21. Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Qualitative analysis of pomdps with temporal logic specifications for robotics applications. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 325–330, 2015.
22. Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger, and Veronika Loitzenbauer. Model and objective separation with conditional lower bounds: Disjunction is harder than conjunction. In *LICS*, 2016. To appear, available at `http://arxiv.org/abs/1602.02670`.

23 Krishnendu Chatterjee, Amir Kafshdar Goharshady, Rasmus Ibsen-Jensen, and Andreas Pavlogiannis. Algorithms for algebraic path properties in concurrent systems of constant treewidth components. In *POPL*, pages 733–747, 2016.

24 Krishnendu Chatterjee and Monika Henzinger. Faster and Dynamic Algorithms For Maximal End-Component Decomposition And Related Graph Problems In Probabilistic Verification. In *SODA*, pages 1318–1336, 2011.

25 Krishnendu Chatterjee and Monika Henzinger. An $O(n^2)$ Time Algorithm for Alternating Büchi Games. In *SODA*, pages 1386–1399, 2012.

26 Krishnendu Chatterjee and Monika Henzinger. Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-component Decomposition. *Journal of the ACM*, 61(3):15, 2014.

27 Krishnendu Chatterjee, Monika Henzinger, and Veronika Loitzenbauer. Improved Algorithms for One-Pair and k-Pair Streett Objectives. In *LICS*, pages 269–280, 2015.

28 Krishnendu Chatterjee, Rasmus Ibsen-Jensen, Andreas Pavlogiannis, and Prateesh Goyal. Faster algorithms for algebraic path properties in recursive state machines with constant treewidth. In *POPL*. ACM, 2015.

29 Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Simple stochastic parity games. In *CSL*, pages 100–113, 2003.

30 Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.

31 Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *FSE'01*, pages 109–120. ACM Press, 2001.

32 David L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits*. The MIT Press, 1989.

33 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991.

34 Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for büchi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005.

35 Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 2020–2025, 2005.

36 Yashdeep Godhal, Krishnendu Chatterjee, and Thomas A. Henzinger. Synthesis of AMBA AHB from formal specification: A case study. *Journal of Software Tools Technology Transfer*, 2011.

37 Monika Henzinger, Valerie King, and Tandy Warnow. Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Evolutionary Biology. *Algorithmica*, 24(1):1–13, 1999.

38 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30, 2015.

39 Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, pages 384–406, 1981.

40 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

41 Marcin Jurdziński. Small Progress Measures for Solving Parity Games. In *STACS*, pages 290–301, 2000.

42 Marcin Jurdziński, Mike Paterson, and Uri Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.

43 Sudeep Juvekar and Nir Piterman. Minimizing generalized büchi automata. In *CAV*, pages 45–58, 2006.

44 Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Deterministic Ω automata vis-a-vis deterministic buchi automata. In *ISAAC*, pages 378–386, 1994.

45 Wouter Kuijper and Jaco van de Pol. Computing weakest strategies for safety games of imperfect information. In *TACAS*, pages 92–106, 2009.

46 Orna Kupferman and Moshe Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *LICS*, pages 81–92, 1998.

47 Orna Kupferman and Moshe Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.

48 François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *ISSAC*, pages 296–303, 2014.

49 Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, January 2002.

50 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

51 Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

52 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, pages 1065–1075, 2010.

53 Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, LNCS 3855, Springer, pages 364–380, 2006.

54 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

55 P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

56 Sven Schewe. Solving Parity Games in Big Steps. In *FSTTCS*, pages 449–460, 2007.

57 Mikkel Thorup. All Structured Programs Have Small Tree Width and Good Register Allocation. *Information and Computation*, 142(2):159 – 181, 1998.

58 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.

59 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. Announced at ICALP'04.

60 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673, 2014.

61 Ryan Williams. Faster decision of first-order graph properties. In *CSL-LICS*, pages 80:1–80:6, 2014.

62 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.

A Omitted Details for Algorithms for Generalized Büchi Games

A.1 Reduction to Büchi Games

Another way to implement generalized Büchi games is by a reduction to Büchi games as follows (see also [13]). Make k copies V^ℓ of the vertices of the original game graph and draw an edge (v^j, u^j) if (v, u) is an edge in the original graph and $v \notin T_j$, and an edge $(v^j, u^{j \oplus 1})$ if (v, u) is an edge in the original graph and $v \in T_j$ (where $j \oplus 1 = j+1$ for $j < k$ and $k \oplus 1 = 1$). Finally, pick the Büchi set T_ℓ of minimal size and make its copy T_ℓ^ℓ in V^ℓ the target set for the Büchi game. This reduction results an $O(k \cdot min_{1 \le \ell \le k} b_\ell \cdot m)$ time procedure when combined with the basic algorithm for Büchi ($b_\ell = |T_\ell|$) and an $O(k^2 n^2)$ time procedure when combined with the $O(n^2)$ time algorithm for Büchi [25].

A.2 Omitted Details for Basic Algorithm

The basic algorithm for generalized Büchi games is given in Algorithm GenBuchiGameBasic. The proof of Theorem 6 consists of the following three propositions.

Algorithm GenBuchiGameBasic: Algorithm for Generalized Büchi Objective in Game Graphs

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$ and objective $\bigwedge_{1 \le \ell \le k}$ Büchi (T_ℓ)
Output: Winning set of player 1

$\mathcal{G}^1 \leftarrow \mathcal{G}$
$\{T_\ell^1\} \leftarrow \{T_\ell\}$
$j \leftarrow 0$
repeat
    $j \leftarrow j + 1$
    for $1 \le \ell \le k$ **do**
        $Y_\ell^j \leftarrow Attr_1(T_\ell^j, \mathcal{G}^j)$
        $S^j \leftarrow V^j \setminus Y_\ell^j$
        if $S^j \neq \emptyset$ **then break**
    $D^j \leftarrow Attr_2(S^j, \mathcal{G}^j)$
    $\mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$
    $\{T_\ell^{j+1}\} \leftarrow \{T_\ell^j \setminus D^j\}$
until $D^j = \emptyset$
return V^j

▶ **Proposition 18** (Runtime Algorithm GenBuchiGameBasic)**.** *The basic algorithm for generalized Büchi games terminates in $O(k \cdot b_1 \cdot m)$ time, where $b_1 = |T_1|$, and thus also in $O(knm)$ time .*

Proof. In each iteration of the repeat-until loop at most $k + 1$ attractor computations are performed, which can each be done in $O(m)$ time. We next argue that the repeat-until loop terminates after at most $2b_1 + 2$ iterations. We use that (a) a player-2 edge from $Y_\ell^j = Attr_1(\mathcal{G}^j, T_\ell^j)$ to $V^j \setminus Y_\ell^j$ has to originate from a vertex of T_ℓ^j and (b) if a player-1 attractor contains a vertex, then it contains also the player-1 attractor of this vertex. In each iteration we have one of the following situations:

1. $S^j = \emptyset$: The algorithm terminates.
2. $Attr_1(\mathcal{G}^j, T_1^j) = V^j$ and $Attr_1(\mathcal{G}^j, T_\ell^j) \neq V^j$ for some $\ell > 1$: We have that $T_1^j \not\subseteq Attr_1(\mathcal{G}^j, T_\ell^j)$ as $T_1^j \subseteq Attr_1(\mathcal{G}^j, T_\ell^j)$ would imply that also $Attr_1(\mathcal{G}^j, T_1^j) = V^j \subseteq Attr_1(\mathcal{G}^j, T_\ell^j)$ which is in contradiction to the assumption. Thus we obtain $|T_1^{j+1}| < |T_1^j|$.

3. $Attr_1(\mathcal{G}^j, T_1^j) \neq V^j$ and $D^j \cap T_1^j \neq \emptyset$: We immediately get $|T_1^{j+1}| < |T_1^j|$.
4. $Attr_1(\mathcal{G}^j, T_1^j) \neq V^j$ and $D^j \cap T_1^j = \emptyset$: In this case $D^j = Attr_2(\mathcal{G}^j, S^j) = S^j$ and thus in the next iteration we have either situation (1) or (2). Notice that, for each vertex $v \in Attr_1(\mathcal{G}^j, T_1^j)$ player 1 has a strategy to reach T_1^j and thus for v to be in D^j the set D^j has to contain at least one vertex of T_1^j.

By the above we have that T_1^j is decreased in at least every second iteration of the loop. For $T_1^j = \emptyset$ we have $Attr_1(\mathcal{G}^j, T_1^j) = \emptyset$ and thus $V^{j+1} = \emptyset$, which terminates the algorithm in the next iteration. Thus we have that each iteration takes time $O(km)$ and there are $O(b_1)$ iterations. ◀

As we can always rearrange the Büchi sets such that $b_1 = \min_{1 \leq \ell \leq k} b_\ell$, this gives an $O(k \cdot \min_{1 \leq \ell \leq k} b_\ell \cdot m)$ algorithm for generalized Büchi games.

For the final game graph $\mathcal{G}^j$ we have that all vertices are in all the player-1 attractors of Büchi sets T_ℓ. Thus player 1 can win the game by following one attractor strategy until the corresponding Büchi set is reached and then switching to the attractor strategy of the next Büchi set.

▶ **Proposition 19** (Soundness Algorithm GenBuchiGameBasic). *Player 1 has a winning strategy from each vertex in the set returned by the algorithm.*

Proof. Let the j^*-th iteration be the last iteration of the algorithm. We have $S^{j^*} = \emptyset$. Thus each vertex of V^{j^*} is contained in $Attr_1(T_\ell^{j^*}, \mathcal{G}^{j^*})$ for each $1 \leq \ell \leq k$. Additionally, either $V^{j^*} = \emptyset$ or $T_\ell^{j^*} \neq \emptyset$ for all $1 \leq \ell \leq k$. Further we have that V^{j^*} is closed for player 2 as only player 2 attractors were removed from V to obtain V^{j^*} (i.e., we apply Lemma 1(1) inductively). Hence player 1 has the following winning strategy (with memory) on the vertices of V^{j^*}: On the vertices of $V^{j^*} \setminus \bigcap_{\ell=1}^k T_\ell^{j^*}$ first follow the attractor strategy for $T_1^{j^*}$ until a vertex of $T_1^{j^*}$ is reached, then the attractor strategy for $T_2^{j^*}$ until a vertex of $T_2^{j^*}$ is reached and so on until the set $T_k^{j^*}$ is reached; then restart with $T_1^{j^*}$. On the vertices of $\bigcap_{\ell=1}^k T_\ell^{j^*} \cap V_1$ player 1 can pick any outgoing edge whose endpoint is in V^{j^*}. Since V^{j^*} is closed for player 2 and $T_\ell^{j^*} \neq \emptyset$ for all $1 \leq \ell \leq k$, this strategy exists, never leaves the set V^{j^*}, and satisfies the generalized Büchi objective. ◀

For completeness we use that each 1-closed set that avoids one Büchi set is winning for player 2 and that, by Lemma 1(3), we can remove such sets from the game graph.

▶ **Proposition 20** (Completeness Algorithm GenBuchiGameBasic). *Let V^{j^*} be the set returned by the algorithm. Player 2 has a winning strategy from each vertex in $V \setminus V^{j^*}$.*

Proof. By Lemma 1(3) it is sufficient to show that, in each iteration j, player 2 has a winning strategy in $\mathcal{G}^j$ from each vertex of S^j. Let ℓ be such that $S^j = V^j \setminus Attr_1(\mathcal{G}^j, T_\ell^j)$. By Lemma 1(1) S^j is closed for player 1 in $\mathcal{G}^j$, that is, player 2 has a strategy that keeps the play within $\mathcal{G}^j[S_j]$ against any strategy of player 1. Since $S^j \cap T_\ell^j = \emptyset$, this strategy is winning for player 2 (i.e., satisfies coBüchi $\left(T_\ell^j\right)$ and thus the disjunctive co-Büchi objective). ◀

A.3 Omitted Details for Improved Algorithm

▶ **Lemma 7** (restated). *Let $\mathcal{G} = (G = (V, E), (V_1, V_2))$ be a game graph and $\{G_i\}$ its hierarchical graph decomposition. For $1 \leq i \leq \lceil \log_2 n \rceil$ let Z_i be the set consisting of the player 2 vertices that have no outgoing edge in $\mathcal{G}_i$ and the player 1 vertices with $> 2^i$ outgoing edges in $\mathcal{G}$.*

1. *If a set $S \subseteq V \setminus Z_i$ is closed for player 1 in $\mathcal{G}_i$, then S is closed for player 1 in $\mathcal{G}$.*
2. *If a set $S \subseteq V$ is closed for player 1 in $\mathcal{G}$ and $|Attr_2(S, \mathcal{G})| \leq 2^i$, then (i) $\mathcal{G}_i[S] = \mathcal{G}[S]$, (ii) the set S is in $V \setminus Z_i$, and (iii) S is closed for player 1 in $\mathcal{G}_i$.*

Proof. 1. By $S \subseteq V \setminus Z_i$ we have for all $v \in S \cap V_1$ that $Out(G, v) = Out(G_i, v)$. Thus if $Out(G_i, v) \subseteq S$, then also $Out(G, v) \subseteq S$. The claim then follows from $E_i \subseteq E$.

2. Since S is closed for player 1 and $|S| \leq 2^i$, (a) the set S does not contain vertices $v \in V_1$ with $Outdeg(G, v) > 2^i$. Further for every vertex of S also the vertices in V_2 from which it has incoming edges are contained in $Attr_2(S, \mathcal{G})$. Thus by $|Attr_2(S, \mathcal{G})| \leq 2^i$ no vertex of S has more than 2^i incoming edges from vertices of V_2. Hence, by the ordering of incoming edges in the construction of G_i, we obtain (b) for the vertices of S all incoming edges from vertices of V_2 are contained in E_i. Combining (a), i.e., $Out(G, v) = Out(G_i, v)$ for $v \in S \cap V_1$, and (b), i.e., $(u, w) \in E_i$ for $u \in V_2$ and $w \in S$, we have (i) $\mathcal{G}_i[S] = \mathcal{G}[S]$. Since S is closed for player 1 in G, every vertex $u \in S \cap V_2$ has an outgoing edge to another vertex $w \in S$ in G. Thus in particular these edges (u, w) are contained in E_i and hence every vertex $u \in S \cap V_2$ has an outgoing edge to another vertex $w \in S$ in G_i. It follows that (ii) $S \cap Z_i = \emptyset$, and (iii) S is closed for player 1 in $\mathcal{G}_i$ (by (1)). ◀

▶ **Corollary 21.** *If in Algorithm* GenBuchiGame *for some ℓ, j, and $i > 1$ we have that $S^j = V^j \setminus Attr_1(T^j_\ell \cup Z^j_i, \mathcal{G}^j_i)$ is not empty but for $i - 1$ the set $V^j \setminus Attr_1(T^j_\ell \cup Z^j_{i-1}, \mathcal{G}^j_{i-1})$ is empty, then $|Attr_2(S^j, \mathcal{G}^j)| > 2^{i-1}$.*

Proof. By Lemma 1(1) S^j is closed for player 1 in $\mathcal{G}^j_i$ and by Lemma 7(1) also in $\mathcal{G}^j$. Assume by contradiction that $|Attr_2(S^j, \mathcal{G}^j)| \leq 2^{i-1}$. Then by Lemma 7(2) we have that $S^j \subseteq V^j \setminus Z^j_{i-1}$ and S^j is closed for player 1 in $\mathcal{G}^j_{i-1}$. Since this means that in $\mathcal{G}^j_{i-1}$ player 1 has a strategy to keep a play within S^j against any strategy of player 2 and S^j does not contain a vertex of Z^j_{i-1} or T^j_ℓ, the set S^j does not intersect with $Attr_1(T^j_\ell \cup Z^j_{i-1}, \mathcal{G}^j_{i-1})$, a contradiction to $V^j \setminus Attr_1(T^j_\ell \cup Z^j_{i-1}, \mathcal{G}^j_{i-1})$ being empty. ◀

Theorem 8 is shown by the following three propositions.

▶ **Proposition 22** (Soundness Algorithm GenBuchiGame). *Player 1 has a winning strategy from each vertex in the set returned by the algorithm.*

Proof. When the algorithm terminates we have $i = \lceil \log_2 n \rceil$ and $S^j = \emptyset$. Since for $i = \lceil \log_2 n \rceil$ we have $G^j_i = G^j$ and $Z^j_i = \emptyset$, the winning strategy of player 1 can be constructed in the same way as for the set returned by Algorithm GenBuchiGameBasic (cf. Proof of Proposition 19). ◀

▶ **Proposition 23** (Completeness Algorithm GenBuchiGame). *Let V^{j^*} be the set returned by the algorithm. Player 2 has a winning strategy from each vertex in $V \setminus V^{j^*}$.*

Proof. By Lemma 1(3) it is sufficient to show that, in each iteration j, player 2 has a winning strategy in G^j from each vertex of S^j. For a fixed j with $S^j \neq \emptyset$, let i and ℓ be such that $S^j = V^j \setminus Attr_1(\mathcal{G}^j_i, T^j_\ell \cup Z^j_i)$. By Lemma 1(1) S^j is closed for player 1 in $\mathcal{G}^j_i$ and by Lemma 7(1) also in $\mathcal{G}^j$. That is, player 2 has a strategy that keeps the play within $\mathcal{G}^j[S_j]$ against any strategy of player 1. Since $S^j \cap T^j_\ell = \emptyset$, this strategy is winning for player 2 (i.e., satisfies the disjunctive co-Büchi objective). ◀

▶ **Proposition 24** (Runtime Algorithm GenBuchiGame). *The algorithm can be implemented to terminate in $O(k \cdot n^2)$ time.*

Proof. To efficiently construct the graphs G_i^j and the vertex sets Z_i^j we maintain (sorted) lists of the incoming and the outgoing edges of each vertex. These lists can be updated whenever an obsolete entry is encountered in the construction of G_i^j; as each entry is removed at most once, maintaining this data structures takes total time $O(m)$. For a given iteration j of the outer repeat-until and the ith iteration of the inner repeat-until loop we have that the graph G_i^j contains $O(2^i \cdot n)$ edges and both G_i^j and the set Z_i^j can be constructed from the maintained lists in time $O(2^i \cdot n)$. Further the k attractor computations in the for-loop can be done in time $O(k \cdot 2^i \cdot n)$, thus for any j the ith iteration of the inner repeat-until loop takes time $O(k \cdot 2^i \cdot n)$. The time spent in the iterations up to the ith iteration forms a geometric series and can thus also be bounded by $O(k \cdot 2^i \cdot n)$. When a non-empty set S^j is found in the jth iteration of the outer repeat-until and in the ith iteration of the inner repeat-until loop, then by Corollary 21 we have $|Attr_2(S^j, \mathcal{G}^j)| > 2^{i-1}$. The vertices in $Attr_2(S^j, \mathcal{G}^j)$ are then removed from G^j to obtain G^{j+1} and are not considered further by the algorithm. Thus we can charge the time of $O(k \cdot 2^i \cdot n)$ to identify S^j to the vertices in $Attr_2(S^j, \mathcal{G}^j)$, which yields a bound on the total time spent in the inner repeat-until loop, whenever $S^j \neq \emptyset$, of $O(k \cdot n^2)$. The total time for computing the attractors $Attr_2(S^j, \mathcal{G}^j)$ can be bounded by $O(m)$. Finally the time for the last iteration of the while loop, when $S^j = \emptyset$ and $i = \lceil \log_2 n \rceil$, can be bounded by $O(k \cdot 2^{\lceil \log_2 n \rceil} \cdot n) = O(k \cdot n^2)$. ◀

▶ Remark (Winning Strategies). Algorithm GenBuchiGame can be modified to additionally return winning strategies for both players within the same time bound: For player 2 a winning strategy for the dominion D^j that is identified in iteration j of the algorithm can be constructed by combining his strategy to stay within the player-1 closed set S^j with his attractor strategy to the set S^j. For player 1 we can obtain a winning strategy in the last iteration of the algorithm by combining her attractor strategies to the sets T_ℓ for $1 \leq \ell \leq k$ (cf. proof of Proposition 19).

B Omitted Details for Conditional Lower Bounds for Generalized Büchi Games

The following lemma shows the correctness of the reduction from triangle detection.

▶ **Lemma 25.** *Let G' be the graph given by Reduction 10 for a graph G and let $T_v = (V^1 \setminus \{v^1\}) \cup (V^4 \setminus \{v^4\})$. Then the following statements are equivalent.*

1. *G has a triangle.*
2. *$s \notin W_1(G', \bigwedge_{v \in V} Büchi(T_v))$.*
3. *The winning set $W_1(G', \bigwedge_{v \in V} Büchi(T_v))$ is empty.*

Proof. (1)⇒(2): Assume that G has a triangle with vertices a, b, c and let a^i,b^i,c^i be the copies of a, b, c in V^i. Now a strategy for player 2 in G' to satisfy coBüchi (T_a) is as follows: When in s, go to a^1; when in a^1, go to b^2; when in b^2, go to c^3; when in c^3, go to a^4; and when in a^4, go to s. As a, b, c form a triangle, all the edges required by the above strategy exist. When player 1 starts in s and follows the above strategy, then he plays an infinite path that only uses vertices s, a^1, b^2, c^3, a^4 and thus satisfies coBüchi (T_a).

(2)⇒(1): Assume that there is a memoryless winning strategy for player 2 starting in s and satisfying coBüchi (T_a). Starting from s, this strategy has to go to a^1, as all other successors of s are contained in T_a and thus would violate the coBüchi (T_a) objective. Then the play continues on some vertex $b^2 \in V^2$ and $c^3 \in V^3$ and then, again by the coBüchi constraint, has to enter a^4. Now by construction of G' we know that there must be edges $(a,b),(b,c),(c,a)$ in the original graph G, i.e. there is a triangle in G.

(2)⇔(3): Notice that when removing s from G' we get an acyclic graph and thus each infinite path has to contain s infinitely often. Thus, if the winning set is non-empty, there is a cycle winning for some vertex and then this cycle is also winning for s. For the converse direction we have that if s is in the winning set, then the winning set is non-empty. ◀

The size and the construction time of the graph G', given in Reduction 10, is linear in the size of the original graph G and we have $k = \Theta(n)$ target sets. Thus if we would have a combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$ algorithm for generalized Büchi games, we would immediately get a combinatorial $O(n^{3-\epsilon})$ algorithm for triangle detection, which contradicts STC (and thus BMM).

Notice, that the sets T_v in the above reduction are of linear size but can be reduced to logarithmic size by modifying the graph constructed in Reduction 10 as follows. Remove all edges incident to s and replace them by two complete binary trees. The first tree with s as root and the vertices V^1 as leaves is directed towards the leaves, the second tree with root s and leaves V^4 is directed towards s. Now for each pair v^1, v^4 one can select one vertex of each non-root level of the trees to be in the set T_v such that the only safe path for player 2 starting in s has to use v^1 and each safe path for player 2 to s must pass v^4 (see also [22]).

▶ Remark (Size of k). Notice that the lower bounds apply to instances with $k \in \Theta(n^c)$ for arbitrary $0 < c \leq 1$, although the reductions produce graphs with $k \in \Theta(n)$. The instances constructed by the reductions have the property that whenever player 2 has a winning strategy, he also has a winning strategy for a specific co-Büchi set T_v. Now instead of solving the instance with $\Theta(n)$ many target sets, one can simply consider $O(n^{1-c})$ many instances with $\Theta(n^c)$ target sets and obtain the winning set for player 2 in the original instance by the union of the player 2 winning sets of the new instances. Finally, towards a contradiction, assume there would be an $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for $k \in \Theta(n^c)$, then together we the above observation we would get an $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for the original instance.

▶ Remark (Digraph Parameters). In both reductions the constructed graph becomes acyclic when deleting vertex s. Thus, our lower bounds also apply for a broad range of digraph parameters. For instance let w be the DAG-width [12] of a graph, then there is no $O(f(w) \cdot (k \cdot n^2)^{1-\epsilon})$-time algorithm (under BMM) and no $O(f(w) \cdot (km)^{1-\epsilon})$-time algorithm (under SETH).

C Omitted Details for Generalized Reactivity-1 Games

C.1 Omitted Details for Basic Algorithm

The basic algorithm for GR(1) games is described in Algorithm GR(1)GameBasic.

To prove Theorem 14, we first show that the dominions we compute via the generalized Büchi games are indeed player-2 dominions for the GR(1) game.

▶ **Lemma 26.** *Given a game with game graph $\mathcal{G}$ and GR(1) objective $\bigwedge_{t=1}^{k_1}$ Büchi$(L_t) \rightarrow \bigwedge_{\ell=1}^{k_2}$ Büchi(U_ℓ). Each player-1 dominion D of the game graph $\overline{\mathcal{G}}$ with generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi(L_t), i.e., each 2-closed set $D \subseteq W_1\left(\overline{\mathcal{G}}, \bigwedge_{t=1}^{k_1} \text{Büchi}(L_t)\right)$, for which there is an index $1 \leq \ell \leq k_2$ with $D \cap U_\ell = \emptyset$, is a player-2 dominion of $\mathcal{G}$ with the original GR(1) objective.*

Proof. By definition of a dominion, in $\overline{\mathcal{G}}$ player 1 has a strategy that visits all sets L_t infinitely often and only visits vertices in D. But then for some ℓ the set Büchi set U_ℓ is not visited at all and thus in $\mathcal{G}$ the strategy is player 2 winning for the GR(1) objective. ◀

Next we show that each player-2 dominion contains a sub-dominion that does not intersect with one of the sets U_ℓ, and thus can be computed via generalized Büchi.

▶ **Lemma 27.** *Given a game with game graph $\mathcal{G}$ and GR(1) objective $\bigwedge_{t=1}^{k_1}$ Büchi$(L_t) \to \bigwedge_{\ell=1}^{k_2}$ Büchi(U_ℓ). Each player-2 dominion D has a subset $D' \subseteq D$ that is a player-2 dominion with $D' \cap U_\ell = \emptyset$ for some $1 \le \ell \le k_2$.*

Proof. First, note that D is closed for player-1 and by definition of a dominion then $D = W_2(\mathcal{G}[D], \bigwedge_{t=1}^{k_1} \text{Büchi}(L_t) \to \bigwedge_{\ell=1}^{k_2} \text{Büchi}(U_\ell))$. Moreover, as each 1-closed set in $\mathcal{G}[D]$ is also 1-closed in $\mathcal{G}$, a set $D' \subseteq D$ is a player-2 dominion of $\mathcal{G}$ iff it is player-2 dominion of $\mathcal{G}[D]$.

Towards a contradiction we will assume that there does not exist such a player-2 dominion D' in $\mathcal{G}[D]$. Then player-1 has a winning strategy for the game graphs $\mathcal{G}^\ell[D] = \mathcal{G}[D] \setminus Attr_1(U_\ell, \mathcal{G}[D])$ with the GR(1) objective, for all $1 \le \ell \le k_2$. As $U_\ell \cap \mathcal{G}^\ell[D] = \emptyset$ the same strategy is also winning for the disjunctive co-Büchi objective $\bigvee_{t=1}^{k_1} \text{coBüchi}(L_t)$. Now consider the following strategy for player 1 in $\mathcal{G}[D]$. The winning strategy of player 1 is constructed from his winning strategies for the game graphs $\mathcal{G}^\ell[D]$ and the attractor strategies for $Attr_1(U_\ell, \mathcal{G}[D])$ for $1 \le j \le k_2$ as follows. Player 1 maintains a counter $c \in \{1, \dots, k_2\}$ that is initialized to 1. As long as the current vertex in the play is contained in $\mathcal{G}^c[D]$, player 1 plays his winning strategy for $\mathcal{G}^c[D]$. If a vertex of $Attr_1(U_c, \mathcal{G}[D])$ is reached, player 1 follows the corresponding attractor strategy until U_c is reached. Then player 1 increases the counter by one or sets the counter to 1 if its value was k_2 and continues playing the above strategy for the new value c. In each play one of two cases must happen:

- Case 1: After some prefix of the play for some counter value c the set $Attr_1(U_c, \mathcal{G}[D])$ is never reached. Then the play satisfies the disjunctive co-Büchi objective $\bigvee_{t=1}^{k_1} \text{coBüchi}(L_t)$ and thus the GR(1) objective.
- Case 2: For all $c \in \{1, \dots k_2\}$ the set U_c is reached infinitely often. Then the play satisfies the generalized Büchi objective $\bigwedge_{\ell=1}^{k_2} \text{Büchi}(U_\ell)$ and thus the GR(1) objective.

Hence, we have shown that $D \subseteq W_1$, a contradiction. ◀

Next we show that whenever Algorithm GR(1)GameBasic removes vertices from the game graph these vertices are indeed winning for player 2. This is due to Lemma 26, stating that these sets are dominions in the current game graph and Lemma 1, stating that all player-2 dominions of the current game graph $\mathcal{G}^j$ are also winning for player 2 in the original game graph $\mathcal{G}$.

▶ **Proposition 28** (Completeness Algorithm GR(1)GameBasic)**.** *Let V^{j^*} be the set of vertices returned by Algorithm* GR(1)GameBasic*. Each vertex in $V \setminus V^{j^*}$ is winning for player 2.*

Proof. By Lemma 1(3) it is sufficient to show that in each iteration j with $S^j \neq \emptyset$ player 2 has a winning strategy from the vertices in S^j in $\mathcal{G}^j$. Let j be such that $S^j = W_1\left(\overline{\mathcal{G}^j \setminus Y_\ell^j}, \bigwedge_{t=1}^{k_1} \text{Büchi}\left(L_t^j\right)\right)$. We first show that S^j is also a player-1 dominion for the generalized Büchi game on $\overline{\mathcal{G}^j}$ (i.e., a player-2 dominion on $\mathcal{G}^j$). By Lemma 1(1) the set $V^j \setminus Y_\ell^j$ is 1-closed in $\mathcal{G}^j$, i.e., it is 2-closed in $\overline{\mathcal{G}^j}$. Thus each dominion of $\overline{\mathcal{G}^j \setminus Y_\ell^j}$ is also 2-closed in $\overline{\mathcal{G}^j}$ and hence a dominion in $\overline{\mathcal{G}^j}$ (see also Lemma 1(2)). Now as S^j does not contain any vertices of U_ℓ it is a player-2 dominion in $\mathcal{G}$ with the GR(1) objective. Finally, from the above and Lemma 1 we have that also $Attr_2(\mathcal{G}^j, S^j)$ is a player-2 dominion in $\mathcal{G}$ with the GR(1) objective. ◀

Algorithm GR(1)GameBasic: Basic Algorithm for GR(1)

Input : Game graph $\mathcal{G}$, Obj. $\bigwedge_{t=1}^{k_1} \text{Büchi}\,(L_t) \to \bigwedge_{\ell=1}^{k_2} \text{Büchi}\,(U_\ell)$
Output: Winning set of player 1

$\mathcal{G}^1 \leftarrow \mathcal{G}$
$\{U_\ell^1\} \leftarrow \{U_\ell\}$; $\{L_t^1\} \leftarrow \{L_t\}$
$j \leftarrow 0$
repeat
  $j \leftarrow j + 1$
  for $1 \leq \ell \leq k_2$ **do**
    $Y_\ell^j \leftarrow Attr_1(U_\ell^j, \mathcal{G}^j)$
    $S^j \leftarrow W_1 \left(\overline{\mathcal{G}^j \setminus Y_\ell^j}, \bigwedge_{t=1}^{k_1} \text{Büchi}\left(L_t^j \setminus Y_\ell^j\right)\right)$
    if $S^j \neq \emptyset$ **then break**
  $D^j \leftarrow Attr_2(S^j, \mathcal{G}^j)$
  $\mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$
  $\{U_\ell^{j+1}\} \leftarrow \{U_\ell^j \setminus D^j\}$; $\{L_t^{j+1}\} \leftarrow \{L_t^j \setminus D^j\}$
until $D^j = \emptyset$
return V^j

For the final game graph $\mathcal{G}^{j^*}$ we can build a winning strategy for player 1 by combining his winning strategies for the disjunctive objective in the subgraphs $\overline{\mathcal{G}_\ell^{j^*}}$ and the attractor strategies for $Attr_1(U_\ell, \mathcal{G}^{j^*})$.

▶ **Proposition 29** (Soundness Algorithm GR(1)GameBasic). *Let V^{j^*} be the set of vertices returned by Algorithm* GR(1)GameBasic*. Each vertex in V^{j^*} is winning for player 1.*

Proof. First note that V^{j^*} is closed for player 2 by Lemma 1(1). Thus as long as player 1 plays a strategy that stays within V^{j^*}, a play that reaches V^{j^*} will never leave V^{j^*}. The following strategy for player 1 for the vertices of $V^{j^*}1$ satisfies this condition. The winning strategy of player 1 is constructed from the winning strategies of player 2, i.e., the disjunctive co-Büchi player, in the generalized Büchi games with game graphs $\overline{\mathcal{G}_\ell^{j^*}} = \overline{\mathcal{G}^{j^*} \setminus Y_\ell^{j^*}}$ and objectives $\bigwedge_{t=1}^{k_1} \text{Büchi}\left(L_t^{j^*}\right)$ and the attractor strategies for $Attr_1(U_\ell, \mathcal{G}^{j^*})$ for $1 \leq j \leq k_2$. Player 1 maintains a counter $c \in \{1, \dots, k_2\}$ that is initialized to 1 and proceeds as follows. (1) As long as the current vertex in the play is contained in $G_c^{j^*} = G^{j^*} \setminus Y_c^{j^*}$, player 1 plays his winning strategy for the disjunctive co-Büchi objective on $\mathcal{G}_c^{j^*}$. (2) If a vertex of $Attr_1(U_c, \mathcal{G}^{j^*})$ is reached, player 1 follows the corresponding attractor strategy until U_c is reached. Then player 1 increases the counter by one or sets the counter to 1 if its value was k_2 and continues with (1). As the play stays within W_1, one of two cases must happen: Case 1: After some prefix of the play for some counter value c the set $Attr_1(U_c, \mathcal{G}^{j^*})$ is never reached. Then the play satisfies the disjunctive co-Büchi objective $\bigvee_{t=1}^{k_1} \text{coBüchi}\,(L_t)$ and thus the GR(1) objective. Case 2: For all $c \in \{1, \dots k_2\}$ the set U_c is reached infinitely often. Then the play satisfies the generalized Büchi objective $\bigwedge_{\ell=1}^{k_2} \text{Büchi}\,(U_\ell)$ and thus the GR(1) objective. ◀

Finally the runtime of Algorithm GR(1)GameBasic is composed of the number of iterations of the nested loops and the runtime for the generalized Büchi algorithm.

▶ **Proposition 30** (Runtime Algorithm GR(1)GameBasic). *Algorithm* GR(1)GameBasic *runs in $O(k_2 \cdot n \cdot B)$ where B is the runtime bound for the used ConjBüchi algorithm, i.e., if we use Algorithm* GenBuchiGame *to solve ConjBüchi, the bound is $O(k_1 \cdot k_2 \cdot n^3)$.*

Proof. As in each iteration of the outer loop, except the last one, at least one vertex is removed from the maintained graph, there are only $O(n)$ iterations. In the inner loop we have k_2 iterations, each with a call to the generalized Büchi game algorithm. Thus, in total we have a running time of $O(k_2 \cdot n \cdot B)$. ◀

C.2 Progress Measure Algorithm for Finding Small Dominions

In this section we prove Theorem 15. We first restate the progress measure of [43] in our notation and simplified to generalized Büchi, then adapt it to not compute the winning sets but dominions of a given size, and finally give an efficient algorithm to compute the progress measure.

Recall the iterative construction of an attractor $Attr_p(\mathcal{G}, U)$, with $R_0 = U$ and $R_{j+1} = R_j \cup \{v \in V_p \mid Out(v) \cap R_j \neq \emptyset\} \cup \{v \in V_{\bar{p}} \mid Out(v) \subseteq R_j\}$ for all $j \geq 0$. The *p-rank* of a vertex v w.r.t. a set U is given by $rank_p(\mathcal{G}, v, U) = \min\{j \mid v \in R_j\}$ if $v \in Attr_p(\mathcal{G}, U)$ and is ∞ otherwise.

The progress measure of [43] is defined as follows. Let $\bigwedge_{\ell=1}^{k}$ Büchi(T_i) be a generalized Büchi objective. For each $1 \leq \ell \leq k$ we define a value m_ℓ to be $m_\ell = |V \setminus T_\ell|$ and a function $\rho_\ell : V \to \{0, 1, \ldots, m_\ell, \infty\}$. The intuitive meaning of a value $\rho_\ell(v)$ is the number of moves player 1 needs, when starting in v, to reach a vertex of $T_\ell \cap W_1$, i.e., $\rho_\ell(v)$ will equal to the rank $rank_p(\mathcal{G}, v, T_\ell \cap W_1)$. As there are only m_ℓ many vertices which are not in T_ℓ, one can either reach them within m_ℓ steps or cannot reach them at all.

The actual value $\rho_\ell(v)$ is defined in a recursive fashion via the values of the successor vertices of v. That is, for $v \notin T_\ell$ we define $\rho_\ell(v)$ by the values $\rho_\ell(w)$ for $(v, w) \in E$. Otherwise, if $v \in T_\ell$, then we already reached T_ℓ and we only have to check whether v is in the winning set. That is, whether v can reach a vertex of the next target set $T_{\ell \oplus 1}$ that is also in the winning set W_1. Hence, for $v \in T_\ell$ we define $\rho_\ell(v)$ by the values $\rho_{\ell \oplus 1}(w)$ for $(v, w) \in E$, where $\ell \oplus 1 = \ell + 1$ if $\ell < k$ and $k \oplus 1 = 1$. For $v \in V$ one considers all the successor vertices and their values and then picks the minimum if $v \in V_1$ or the maximum if $v \in V_2$. In both cases $\rho_\ell(v)$ is set to this value increased by 1 if $v \notin T_\ell$. If $v \in T_\ell$, the value is set to ∞ if the minimum (resp. maximum) over the successors is ∞ and to 0 otherwise. This procedure is formalized via two functions. First, $best_\ell(v)$ returns the value of the best neighbor for the player owning v.

$$best_\ell(v) = \begin{cases} \min_{(v,w) \in E} \rho_{\ell \oplus 1}(w) & \text{if } v \in V_1 \wedge v \in T_\ell \\ \min_{(v,w) \in E} \rho_\ell(w) & \text{if } v \in V_1 \wedge v \notin T_\ell \\ \max_{(v,w) \in E} \rho_{\ell \oplus 1}(w) & \text{if } v \in V_2 \wedge v \in T_\ell \\ \max_{(v,w) \in E} \rho_\ell(w) & \text{if } v \in V_2 \wedge v \notin T_\ell \end{cases}$$

Second, the function $incr_v^\ell$ formalizes the incremental step described above. To this end for each set $\{0, 1, \ldots, m_\ell, \infty\}$ we define the unary ++ operator as $x{+}{+} = x + 1$ for $x < m_\ell$ and $x{+}{+} = \infty$ otherwise.

$$incr_v^\ell(x) = \begin{cases} 0 & \text{if } v \in T_\ell \wedge x \neq \infty \\ x{+}{+} & \textit{otherwise} \end{cases}$$

The functions $\rho_\ell(.)$ are now defined as the least fixed-point of the operation that updates all $\rho_\ell(v)$ to $\max(\rho_\ell(v), incr_v^\ell(best_\ell(v)))$. The least fixed-point can be computed via the lifting algorithm [41], that starts with all the $\rho_\ell(.)$ initialized as the zero functions and iteratively updates $\rho_\ell(v)$ to $incr_v^\ell(best_\ell(v))$, for all $v \in V$, until the least fixed-point is reached.

Now given the progress measure, we can decide the generalized Büchi game by the following theorem. Intuitively, player 1 can win starting from a vertex with $\rho_1(v) < \infty$ by keeping a counter ℓ that is initialized to 1, choosing the outgoing edge to $best_\ell(v)$ whenever at a vertex of V_1, and increasing the counter with $\oplus 1$ when a vertex of T_ℓ is reached.

▶ **Theorem 31.** *[43, Thm. 1] Player 1 as a winning strategy starting in a vertex v iff $\rho_1(v) < \infty$.*

As our goal is to compute small dominions, say of size h, instead of the whole winning set, we have to modify the above progress measure as follows. In the definition of the functions ρ_ℓ we redefine the value m_ℓ to be $\min\{h-1, |V \setminus T_\ell|\}$ instead of $|V \setminus T_\ell|$. The intuition behind this is that if the dominion contains at most h vertices, then from each vertex in the dominion we can reach each set T_ℓ within $h-1$ steps and we do not care about vertices with a larger distance.

With Algorithm GenBuchiProgressMeasure we give an $O(k \cdot h \cdot m)$-time realization of the lifting algorithm for computing the functions ρ_ℓ. It is a corrected version of the lifting algorithm in [43, Section 3.1], tailored to generalized Büchi objectives and dominion computation, and exploits ideas from the lifting algorithm in [34]. The main idea is to consider a value $\rho_\ell(v)$ for a pair (v, ℓ) only h times and each time increase the value but only doing computations in the order of the degree of v. To this end we maintain a list of pairs (v, ℓ) for which $\rho_\ell(v)$ must be increased because of some update on v's neighbors, values $B_\ell(v)$ storing the value of $best_\ell(v)$ from the last time we updated $\rho_\ell(v)$, and for $v \in V_1$ a counter $C_\ell(v)$ which stores the number of successors $w \in Out(v)$ with $\rho_\ell(w) = B_\ell(v)$. Moreover, in order to initialize $C_\ell(v)$ when $B_\ell(v)$ is updated, we use the function $cnt_\ell(v)$ counting the number of successor vertices that have minimal ρ_j. Notice that for $v \in T_\ell$ we only distinguish whether $\rho_{\ell\oplus 1}(v)$ is finite or not.

$$cnt_\ell(v) = \begin{cases} |\{w \in Out(v) \mid \rho_{\ell\oplus 1}(w) < \infty\}| & \text{if } v \in T_\ell \\ |\{w \in Out(v) \mid \rho_\ell(w) = best_\ell(v)\}| & \text{if } v \notin T_\ell \end{cases}$$

Whenever the algorithm considers a pair (v, ℓ), it first computes $best_\ell(v)$, $cnt_\ell(v)$ in $O(Outdeg(v))$, stores these values in $B_\ell(v)$ and $C_\ell(v)$, and updates $\rho_\ell(v)$ to $incr_v^\ell(best_\ell(v))$. It then identifies the pairs (w, ℓ), $(w, \ell \ominus 1)$ that are affected by the change of the value $\rho_\ell(v)$ and adds them to the set L, in $O(Indeg(v))$.

▶ Remark. While for the progress measure in [43] $\rho_\ell(v) \neq \infty$ is equivalent to $\rho_{\ell'}(v) \neq \infty$ for all $1 \leq \ell' \leq k$, this does not hold in general for our modified progress measure ρ. Thus we consider the set $\{v \in V \mid \rho_\ell(v) \neq \infty \text{ for some } \ell\}$ as a player-1 dominion and not just the set $\{v \in V \mid \rho_1(v) \neq \infty\}$.

The correctness of Algorithm GenBuchiProgressMeasure is by the following invariants that are maintained during the whole algorithm. These invariants show that (a) the data structures L, B_ℓ, and C_ℓ are maintained correctly, and (b) the values $\rho_\ell(v)$ are bounded from above by (i) $incr_v^\ell(best_\ell(v))$ and (ii) by the rank $rank_1(\mathcal{G}, v, T_\ell \cap D)$ if v is in a dominion D of size $\leq h$.

▶ **Invariant 32.** *The while loop in Algorithm* GenBuchiProgressMeasure *has the following loop invariants.*

1. *For all $v \in V$ and all $1 \leq \ell \leq k$ we have $\rho_\ell(v) \leq incr_v^\ell(best_\ell(v))$.*
2. *For all $v \in V$ and all $1 \leq \ell \leq k$ we have that if $\rho_\ell(v) \neq 0$ or $v \in T_\ell$, then $\rho_\ell(v) = incr_v^\ell(B_\ell(v))$.*

Algorithm GenBuchiProgressMeasure: Lifting Algorithm for Generalized Büchi

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$, objective $\bigwedge_{1 \le \ell \le k}$ Büchi (T_ℓ), integer h
Output: player 1 dominion / winning set for player 1 if $h = n$

foreach $v \in V$, $1 \le l \le k$ **do**
  $B_\ell(v) \leftarrow 0$
  if $v \in V_1$ **then** $C_\ell(v) \leftarrow Outdeg(v)$
  $\rho_\ell(v) \leftarrow 0$
$L \leftarrow \{(v, \ell) \mid v \in V, 1 \le \ell \le k, v \notin T_\ell\}$
while $L \ne \emptyset$ **do**
  pick some $(v, \ell) \in L$ and remove it from L
  $t \leftarrow \rho_\ell(v)$
  $B_\ell(v) \leftarrow best_\ell(v)$
  if $v \in V_1$ **then** $C_\ell(v) \leftarrow cnt_\ell(v)$
  $\rho_\ell(v) \leftarrow incr^\ell_v(best_\ell(v))$
  foreach $w \in In(v) \setminus T_\ell$ *with* $(w, \ell) \notin L$, $\rho_\ell(w) < \infty$ **do**
    if $w \in V_1, t = B_\ell(w)$ **then**
      $C_\ell(w) \leftarrow C_\ell(w) - 1$
      if $C_\ell(w) = 0$ **then** $L \leftarrow L \cup \{(w, \ell)\}$
    else if $w \in V_2, \rho_\ell(v) > B_\ell(w)$ **then** $L \leftarrow L \cup \{(w, \ell)\}$
  if $\rho_\ell(v) = \infty$ **then**
    foreach $w \in In(v) \cap T_{\ell \ominus 1}$ *with* $(w, \ell \ominus 1) \notin L$, , $\rho_{\ell \ominus 1}(w) < \infty$ **do**
      if $w \in V_1$, **then**
        $C_{\ell \ominus 1}(w) \leftarrow C_{\ell \ominus 1}(w) - 1$
        if $C_{\ell \ominus 1}(w) = 0$ **then** $L \leftarrow L \cup \{(w, \ell \ominus 1)\}$
      else if $w \in V_2$ **then** $L \leftarrow L \cup \{(w, \ell \ominus 1)\}$
return $\{v \in V \mid \rho_\ell(v) \ne \infty$ for some $\ell\}$

3. *For* $v \in V_1$ *we have* $C_\ell(v) = \begin{cases} |\{w \in Out(v) \mid \rho_{\ell \oplus 1}(w) < \infty\}| & \text{if } v \in T_\ell\,, \\ |\{w \in Out(v) \mid \rho_\ell(w) = B_\ell(v)\}| & \text{if } v \notin T_\ell, \rho_\ell(v) < \infty\,. \end{cases}$
4. *The set* L *consists exactly of the pairs* (v, ℓ) *with* $\rho_\ell(v) < incr^\ell_v(best_\ell(v))$.
5. *For all* $v \in V$ *and all* $1 \le \ell \le k$ *we have* $\rho_\ell(v) \le rank_1(\mathcal{G}, v, T_\ell \cap D) < h$ *for each player-1 dominion* D *with* $|D| \le h$.

By the following lemmata we prove the above loop invariants to be valid.

▶ **Lemma 33.** *After each iteration of the while loop in Algorithm* GenBuchiProgressMeasure *we have* $\rho_\ell(v) \le incr^\ell_v(best_\ell(v))$, *for all* $v \in V$ *and all* $1 \le \ell \le k$.

Proof. As all $\rho_\ell(v)$ are in initialized to 0 and 0 is the minimum value, the inequalities are all satisfied in the *base case* when the algorithm first enters the the while loop. Now for the *induction step* consider an iteration of the loop and let us assume the invariant is satisfied beforehand. The value $\rho_\ell(v)$ is only changed when the pair (v, ℓ) is processed and then set to $incr^\ell_v(best_\ell(v))$. Thus the invariant is satisfied after these iterations. In all the other iterations with different pairs (v', ℓ') the values $\rho_{\ell'}(v')$ are either unchanged or increased. As $incr^\ell_v(best_\ell(v))$ is monotonic in the values of the neighbors, this can only increase the right side of the inequality and thus this invariant is also satisfied after these iterations. Hence, if the invariant is valid before an iteration of the loop it is also valid afterwards. ◀

▶ **Lemma 34.** *After each iteration of the while loop in Algorithm* GenBuchiProgressMeasure *we have that if* $\rho_\ell(v) \ne 0$ *or* $v \in T_\ell$, *then* $\rho_\ell(v) = incr^\ell_v(B_\ell(v))$, *for all* $v \in V$ *and all* $1 \le \ell \le k$.

Proof. As $\rho_\ell(v)$ is initialized to 0, this is trivially satisfied in the *base case*. Now for the *induction step* consider an iteration of the loop and let us assume the invariant is satisfied beforehand. The values $\rho_\ell(v)$, and $B_\ell(v)$ are only changed when the pair (v, ℓ) is processed and then the invariant is trivially satisfied by the assignments in line 9 and line 11 of the algorithm. ◀

▶ **Lemma 35.** *After each iteration of the while loop in Algorithm* GenBuchiProgressMeasure *for $v \in V_1$ we have*

$$C_\ell(v) = \begin{cases} |\{w \in Out(v) \mid \rho_{\ell\oplus 1}(w) < \infty\}| & \text{if } v \in T_\ell\,, \\ |\{w \in Out(v) \mid \rho_\ell(w) = B_\ell(v)\}| & \text{if } v \notin T_\ell, \rho_\ell(v) < \infty\,. \end{cases}$$

Proof. As *base case* consider the point where the algorithm first enters the the while loop. All $\rho_\ell(v)$ and $B_\ell(v)$ are initialized to 0 and thus in both cases the right side of the invariant is equal to $Outdeg(v)$, which is exactly the value assigned to $C_\ell(v)$.

Now for the *induction step* consider an iteration of the loop and let us assume the invariant is satisfied beforehand. Let $v \in V_1$. In an iteration where (v, ℓ) is processed in line 10 we set $C_\ell(v)$ to $cnt_\ell(v)$ and hence the invariant is satisfied by the definition of $cnt_\ell(v)$. Otherwise the condition for $C_\ell(v)$ is only affected if a $u \in Out(v)$ is processed. We distinguish the two cases where $v \in T_\ell$ and where $v \notin T_\ell$.

- If $v \in T_\ell$ then $C_\ell(v)$ is only affected in iterations where pairs $(u, \ell \oplus 1)$ are considered. If the updated value of $\rho_{\ell\oplus 1}(u)$ is less than ∞ then the set $\{w \in Out(v) \mid \rho_{\ell\oplus 1}(w) < \infty\}$ is unchanged and also $C_\ell(v)$ is not changed by the algorithm, i.e., the invariant is still satisfied. Otherwise if the updated value of $\rho_{\ell\oplus 1}(u)$ is ∞ then u drops out from the set $\{w \in Out(v) \mid \rho_{\ell\oplus 1}(w) < \infty\}$ but also the algorithm decreases $C_\ell(v)$ by one, i.e., again the invariant is satisfied.
- If $v \notin T_\ell$ and $\rho_\ell(v) < \infty$ then $C_\ell(v)$ is only affected in iterations where pairs (u, ℓ) are considered. Let $\rho^o_\ell(u)$ be the value of $\rho_\ell(u)$ before its update. If $\rho^o_\ell(u) > B_\ell(v)$ then $u \notin \{w \in Out(v) \mid \rho_\ell(w) = B_\ell(v)\}$ and thus the set is not affected by the increased $\rho_\ell(u)$. In that case the algorithm does not change $C_\ell(v)$ and thus the invariant is satisfied. Otherwise if $\rho^o_\ell(u) = B_\ell(v)$ then $u \in \{w \in Out(v) \mid \rho_\ell(w) = B_\ell(v)\}$ before the iteration but is dropped during the iteration. In that case the algorithm decreases $C_\ell(v)$ by one and thus the invariant is still satisfied.
  Notice that by (1), and (2) it cannot happen that $\rho^o_\ell(u) < B_\ell(v)$. Assume by contradiction $\rho^o_\ell(u) < B_\ell(v)$. Let $best^o_\ell(v)$ denote the value of $best_\ell(v)$ before the update of $\rho_\ell(u)$. By (1) we have $\rho_\ell(v) \leq incr^\ell_v(best^o_\ell(v))$, by the definition of $best^o_\ell(v)$ and $v \in V_1 \setminus T_\ell$ we have $best^o_\ell(v) \leq \rho^o_\ell(u) < B_\ell(v)$. By (2) we have either $\rho_\ell(v) = incr^\ell_v(B_\ell(v))$ or $\rho_\ell(v) = 0$. In the first case, as $incr^\ell_v(x)$ is strictly increasing for $x < \infty$, we have $incr^\ell_v(best^o_\ell(v)) < incr^\ell_v(B_\ell(v)) = \rho_\ell(v)$ and thus a contradiction to (1)). In the second case the pair (v, ℓ) was not processed yet and we have a contradiction by $B_\ell(v) = 0$. ◀

▶ **Lemma 36.** *After each iteration of the while loop in Algorithm* GenBuchiProgressMeasure *we have that the set L consists exactly of the pairs (v, ℓ) with $\rho_\ell(v) < incr^\ell_v(best_\ell(v))$.*

Proof. The set L is initialized in line 5 with all pairs (v, ℓ) such that $v \notin T_\ell$. For all of these vertices we have $best_\ell(v) = 0$ and thus $incr^\ell_v(best_\ell(v)) = 1$, i.e., $\rho_\ell(v) = 0 < incr^\ell_v(best_\ell(v)) = 1$. Now consider $(v, \ell) \notin L$, i.e., $v \in T_\ell$. As all $\rho_\ell(v) = 0$, we have $incr^\ell_v(best_\ell(v)) = 0$ and thus $\rho_\ell(v) = 0 \not< incr^\ell_v(best_\ell(v)) = 0$. Hence, in the *base case* a pair (v, ℓ) is in L iff $\rho_\ell(v) = 0 < incr^\ell_v(best_\ell(v)) = 1$.

Now for the *induction step* consider an iteration of the loop and let us assume the invariant is satisfied beforehand. For the pair (v, ℓ) processed in the iteration $\rho_\ell(v)$ is set to $incr^\ell_v(best_\ell(v))$ and thus it can be removed from L. Notice that (a) the value of $\rho_\ell(w)$ is only changed when a pair (w, ℓ) processed and (b) $incr^\ell_v(best_\ell(w))$ can only increase when other pairs (v, ℓ) are processed. Thus we have to show that in a iteration where the algorithm processes the pair (v, ℓ) all pairs (v, ℓ') with with $\rho_{\ell'}(w) = incr^\ell_v(best_\ell(w))$ before the iteration and $\rho_{\ell'}(w) < incr^\ell_v(best_\ell(w))$ after the iteration are added to the set L. The only vertices affected by the change of $\rho_\ell(v)$ are those in $In(v)$ which are either (i) not in T_ℓ, or (ii) in $T_{\ell\ominus 1}$. In the former case only ρ_ℓ is affected while in the latter case only $\rho_{\ell\ominus 1}$ is affected. Let $\rho^o_\ell(v)$ and $\rho^n_\ell(v)$ be the values before, respectively after, the update on $\rho_\ell(v)$. Notice that if $w \notin T_\ell$ and $\rho_\ell(w) = 0$, then $(w, \ell) \in L$ by the initialization in line 5. Thus in the following, by (3), we can assume $\rho_\ell(w) = incr^\ell_v(B_\ell(w))$ for all $(w, \ell) \notin L$. We consider the following cases.

- $w \in In(v) \setminus T_\ell$ and $w \in V_1$: Then $incr^\ell_v(best_\ell(w)) > \rho_\ell(w)$ iff all $u \in Out(w)$ have $\rho_\ell(u) > B_\ell(w)$. As $(w, \ell) \notin L$ we know that before the iteration there is at least one $u \in Out(w)$ with $\rho_\ell(u) = B_\ell(w)$. In case $u \neq v$ it will not be changed during the iteration and thus $incr^\ell_v(best_\ell(w)) \not> \rho_\ell(w)$. Hence $incr^\ell_v(best_\ell(w)) > \rho_\ell(w)$ iff v is the only vertex in $Out(w)$ with $\rho^o_\ell(v) = B_\ell(w)$. But then by (3) $C_\ell(v) = 1$ and thus the algorithm will reduce $C_\ell(v)$ to 0 and add (v, ℓ) to the set L.
- $w \in In(v) \setminus T_\ell$ and $w \in V_2$: Then $incr^\ell_v(best_\ell(w)) > \rho_\ell(w)$ iff there is an $u \in Out(w)$ with $\rho_\ell(u) > B_\ell(w)$. If there would be such an $u \in Out(w)$ different from v then by induction hypothesis already $(v, \ell) \in L$. Thus we must have that $\rho^n_\ell(v) > B_\ell(w)$ and thus (w, ℓ) is added to L in line 16 of the algorithm.
- $w \in In(v) \cap T_{\ell\ominus 1}$ and $w \in V_1$: Then $incr^\ell_v(best_\ell(w)) > \rho_{\ell\ominus 1}(w)$ iff all $u \in Out(w)$ have $\rho_\ell(u) = \infty$ and $\rho_{\ell\ominus 1}(w) = 0$. This is the case iff v was the only vertex in $Out(w)$ with $\rho_\ell(v) < \infty$. But then by (3) $C_\ell(v) = 1$ and thus the algorithm will decrement $C_\ell(v)$ to 0 and add $(v, \ell \ominus 1)$ to the set L.
- $w \in In(v) \cap T_{\ell\ominus 1}$ and $w \in V_2$: Then $incr^\ell_v(best_\ell(w)) > \rho_{\ell\ominus 1}(w)$ iff there is an $u \in Out(w)$ with $\rho_\ell(u) = \infty$ and $rho_{\ell\ominus 1}(w)$. If there would be such an $u \in Out(w)$ different from v then by induction hypothesis already $(v, \ell \ominus 1) \in L$ or $\rho_{\ell\ominus 1}(w)$. Thus, we have that $\rho^n_\ell(v) = \infty > rho_{\ell\ominus 1}(w)$ and $incr^\ell_v(\rho^n_\ell(v)) = \infty > \rho_{\ell\ominus 1}(w) = 0$. In that case $(w, \ell \ominus 1)$ is added to L in line 22 of the algorithm. ◀

▶ **Lemma 37.** *For all $v \in V$ and all $1 \leq \ell \leq k$ we have $\rho_\ell(v) \leq rank_1(\mathcal{G}, v, T_\ell \cap D) < h$ for each player-1 dominion D with $|D| \leq h$.*

Proof. As all functions $\rho_\ell(.)$ are initialized as the 0-function, the invariant is satisfied trivially in the . *base case* where the algorithm first enters the the while loop.

Now for the *induction step* consider an iteration of the loop and let us assume all the invariants are satisfied beforehand. First, notice that as $|D| \leq h$, we have $rank_1(\mathcal{G}, v, T_\ell \cap D) < h$ for all $1 \leq \ell \leq k$ and $v \in D$. The value $\rho_\ell(v)$ is only updated in line 11 and there set to $incr^\ell_v(best_\ell(v))$. We distinguish three different cases.

- Assume $v \in V_1$ and $rank_1(\mathcal{G}, v, T_\ell \cap D) = j$ with $1 \leq j < h$ then, by definition of $rank_1$, there is a $w \in D, w \neq v$, with $(v, w) \in E$ and $rank_1(\mathcal{G}, w, T_\ell \cap D) = j - 1$. Now as the invariant is valid before the iteration and $\rho_\ell(w)$ is not changed during the iteration, we have $\rho_\ell(w) \leq j - 1$ and thus $best_\ell(v) \leq j - 1$. Hence, $incr^\ell_v(best_\ell(v)) \leq j$ and the invariant is still satisfied.

- Assume $v \in V_2$ and $rank_1(\mathcal{G}, v, T_\ell \cap D) = j$ with $1 \leq j < h$ then, by definition of $rank_1$, $rank_1(\mathcal{G}, w, T_\ell \cap D) = j - 1$ for each $(v, w) \in E$ (as D is 2-closed we have $w \in D$). Now as the invariant is valid before the iteration and $\rho_\ell(w)$ is not changed during the iteration, we have $\rho_\ell(w) \leq j - 1$ for each $(v, w) \in E$ and thus $best_\ell(v) \leq j - 1$. Hence, $incr^\ell_v(best_\ell(v)) \leq j$ and the invariant is still satisfied.
- Finally, assume $rank_1(\mathcal{G}, v, T_\ell \cap D) = 0$, that is $v \in T_\ell$. By induction hypothesis for all $w \in D$ with $(v, w) \in E$ it holds that $\rho_{\ell \oplus 1}(v) < h$ and thus $best_\ell(v) < h$. Hence, $incr^\ell_v(best_\ell(v)) = 0$ and the loop invariant is still satisfied.

Hence, this loop invariant is maintained during the whole algorithm. ◀

The next lemma gives the ingredients to show that the set $W = \{v \in V \mid \rho_\ell(v) \neq \infty \text{ for some } \ell\}$ is a player-1 dominion by exploiting the fact that the functions ρ_ℓ form a fixed-point of the update operator.

▶ **Lemma 38.** *Let $W = \{v \in V \mid \rho_\ell(v) \neq \infty$ for some $\ell\}$ be the set computed by Algorithm* GenBuchiProgressMeasure*.*

1. *For all $v \in V$: If $\rho_\ell(v) < \infty$, then player 1 has a strategy to reach $\{v' \in T_\ell \mid \rho_\ell(v') = 0\}$ from v by only visiting vertices in W.*
2. *For all $v \in T_\ell$: If $\rho_\ell(v) = 0$, then player 1 has a strategy to reach $\{v' \in T_{\ell \oplus 1} \mid \rho_{\ell \oplus 1}(v') = 0\}$ from v by only visiting vertices in W.*

Proof. Notice that by the Invariants (1) & (4) we have $\rho_\ell(v) = incr^\ell_v(best_\ell(v))$ for all $v \in V$ and all $1 \leq \ell \leq k$, i.e., the functions $\rho_\ell(v)$ are a fixed-point for the $incr^\ell_v(best_\ell(v))$ updates. 1) Consider a vertex $v \in V$ with $\rho_\ell(v) = j$ for $0 < j < h$. We will show by induction in j that then player 1 has a strategy to reach $S = \{v' \in T_\ell \mid \rho_\ell(v') = 0\}$ from v by only visiting vertices in W. For the *base case* we exploit that the functions $\rho_\ell(v)$ are a fixed-point of the $incr^\ell_v(best_\ell(v))$ updates. By the definition of $incr^\ell_v$ we have that $\rho_\ell(v) = 0$ only if $v \in T_\ell$ [3] and thus we already have reached S.

For the *induction step* let us assume the claim holds for all $j' < j$ and consider a vertex v with $\rho_\ell(v) = j$. We distinguish the cases $v \in V_1$ and $v \in V_2$.

- $v \in V_1$: Since ρ is a fixed-point of $incr^\ell_v(best_\ell(v))$, we have that there is at least one w with $(v, w) \in E$ and $\rho_\ell(w) = j - 1$. By the induction hypothesis, player 1 has a strategy to reach S starting from w, and, as player 1 can choose the edge (v, w), also a strategy starting from v.
- $v \in V_2$: Since ρ is a fixed-point of $incr^\ell_v(best_\ell(v))$, we have that $\rho_\ell(w) < j$ for all w with $(v, w) \in E$. By the induction hypothesis player 1 has a strategy to reach S starting from any w with $(v, w) \in E$, and thus also when starting from v.

Moreover, in both cases only the vertex v is added to the path induced by the strategy, which by definition is in W. Hence, in both cases player 1 has a strategy to reach S from v by only visiting vertices in W, which concludes the proof of part 1.

2) Let $S' = \{v' \in T_{\ell \oplus 1} \mid \rho_{\ell \oplus 1}(v') = 0\}$. Again we distinguish whether $v \in V_1$ or $v \in V_2$.

- If $v \in V_1$, then, as the functions ρ_ℓ form a fixed-point, there is at least one w with $(v, w) \in E$ and $\rho_{\ell \oplus 1}(w) < \infty$. Then by (1) player 1 has a strategy to reach S' starting from w, and, as player 1 can choose the edge (v, w), also a strategy starting from v.

---

[3] Recall that we assume that each vertex has at least one outgoing edge.

- If $v \in V_2$, then, as ρ is a fixed-point, $\rho_{\ell \oplus 1}(w) < \infty$ for all w with $(v, w) \in E$. Then by (1) player 1 has a strategy to reach S' starting from any w with $(v, w) \in E$, and thus also when starting from v.

Again, in both cases only the vertex v is added to the path induced by the strategy, which by definition is in W, and thus in both cases player 1 has a strategy to reach S', which concludes the proof of part 2. ◀

We are now prepared to prove the correctness of Algorithm GenBuchiProgressMeasure.

▶ **Proposition 39.** *For the game graph $\mathcal{G}$ and objective $\bigwedge_{1 \le \ell \le k}$ Büchi(T_ℓ), Algorithm* GenBuchiProgressMeasure *either returns a player-1 dominion or the empty set, and, if there is at least one player-1 dominion of size $\le h$ then returns a player-1 dominion containing all player-1 dominions of size $\le h$.*

Proof. We will show that (1) $W = \{v \in V \mid \rho_\ell(v) \ne \infty$ for some $\ell\}$ is a player-1 dominion and that (2) each player-1 dominion of size $\le h$ is contained in W.

1) The following strategy is winning for player 1 and does not leave W. First, for vertices $v \in W \setminus \bigcup_{\ell=1}^{k} T_\ell$ pick some ℓ s.t. $\rho_\ell(v) < \infty$ and play the strategy given by Lemma 38(1) to reach $U_\ell \cap W$. The first time a set U_ℓ is reached, start playing the strategies given by Lemma 38(2) to first reach the set $U_{\ell \oplus 1} \cap W$, then the set $U_{\ell \oplus 2} \cap W$ and so on. This strategy visits all Büchi sets infinitely often and will never leave the set W. That is, W is a player-1 dominion.

2) Consider a player-1 dominion D with $|D| \le h$. Then, we have that $rank_1(\mathcal{G}, v, T_\ell \cap D) \le h - 1$ for all T_ℓ and by Invariant (5) that $\rho_\ell(v) \le h - 1$ for all $v \in D$. That is, each $d \in D$ has $\rho_1(v) < \infty$ and thus $D \subseteq W$. ◀

Finally, let us consider the runtime of Algorithm GenBuchiProgressMeasure.

▶ **Proposition 40.** *Algorithm* GenBuchiProgressMeasure *runs in time $O(k \cdot h \cdot m)$.*

Proof. Notice that the functions $best_\ell(v)$ and $cnt_\ell(v)$ can be computed in time $O(Outdeg(v))$ while $incr_v^\ell(.)$ is in constant time. An iteration of the initial foreach loop takes time $O(Outdeg(v))$ and, as each $v \in V$ is considered k times, the entire foreach loop takes time $O(k \cdot m)$. However, the running time of Algorithm GenBuchiProgressMeasure is dominated by the while loop. Processing a pair $(v, \ell) \in L$ takes time $O(Outdeg(v) + Indeg(v))$. Moreover, whenever (v, ℓ) is processed, the value $\rho_\ell(v)$ is increased by 1 if $v \notin T_\ell$ or by ∞ if $v \in T_\ell$ and thus each pair can be considered at most h times. Hence, for the entire while loop we have a running time of $O\left(h \cdot \sum_{\ell=1}^{k} \sum_{v \in V} (Outdeg(v) + Indeg(v))\right)$ which can be simplified to $O(k \cdot h \cdot m)$. ◀

C.3 Omitted Details for Improved Algorithm for GR(1) Objectives

In this section we prove Theorem 17.

▶ **Lemma 16** (restated). *Let the notation be as in Algorithm* GR(1)Game.

1. *Every $X_{i,\ell}^j \ne \emptyset$ is a player-2 dominion in the GR(1) game on $\mathcal{G}^j$ with $X_{i,\ell}^j \cap U_\ell^j = \emptyset$.*
2. *If for player 2 there exists in $\mathcal{G}^j$ a dominion D w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi(L_t^j) such that $D \cap U_\ell^j = \emptyset$ for some $1 \le \ell \le k_2$ and $|Attr_2(D, \mathcal{G}^j)| \le 2^i$, then D is a dominion w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi$(L_t^j \setminus Y_{i,\ell}^j)$ in $\mathcal{G}_i^j \setminus Y_{i,\ell}^j$.*

Proof. 1. By Theorem 15 the set $X^j_{i,\ell}$ is a player-2 dominion on $\mathcal{G}^j_i \setminus Y^j_{i,\ell}$ w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi $\left(L^j_t \setminus Y^j_{i,\ell}\right)$ of player 2. By Lemma 1(1) $V^j \setminus Y^j_{i,\ell}$ is closed for player 1 on $\mathcal{G}^j_i$. Thus by Lemma 1(2) $X^j_{i,\ell}$ is a player-2 dominion w.r.t. the generalized Büchi objective also in $\mathcal{G}^j_i$. As $X^j_{i,\ell}$ is player 1 closed in $\mathcal{G}^j_i$ and does not intersect with Z^j_i, it is player 1 closed in $\mathcal{G}^j$ by Lemma 7(1). Thus by $E^j_i \subseteq E^j$, the set $X^j_{i,\ell}$ is a player-2 dominion w.r.t. the generalized Büchi objective also in $\mathcal{G}^j$. Since $X^j_{i,\ell}$ does not intersect with U^j_ℓ, it is also a player-2 dominion in the GR(1) game on $\mathcal{G}^j$ (cf. Lemma 26).

2. Since every player-2 dominion is player-1 closed, we have by Lemma 7(2) that (i) $\mathcal{G}^j[D] = \mathcal{G}^j_i[D]$, (ii) D does not intersect with Z^j_i, and (iii) D is player 1 closed in $\mathcal{G}^j_i$. Thus we have that (a) D does not intersect with $Y^j_{i,\ell}$ and (b) player 2 can play the same winning strategy for the vertices in D on G^j_i as on $\mathcal{G}^j$. ◀

▶ **Corollary 41.** *Let j be some iteration of the repeat-until loop in Algorithm* GR(1)GAME *and consider the call to* `kGenBüchiDominion`$(\mathcal{G}^j, \{U^j_\ell\}, \{L^j_t\}, h_{max})$.

1. *If for some $i > 1$ we have $X^j_{i,\ell} \neq \emptyset$ but $X^j_{i-1,\ell} = \emptyset$, then $|Attr_2(X^j_{i,\ell}, \mathcal{G}^j)| > 2^{i-1}$.*
2. *If* `kGenBüchiDominion`$(\mathcal{G}^j, \{U^j_\ell\}, \{L^j_t\}, h_{max})$ *returns the empty set, then for every player-2 dominion D in the GR(1) game we have $|Attr_2(D, \mathcal{G}^j)| > h_{max}$.*

Proof. 1. By Lemma 16(1) $X^j_{i,\ell}$ is a player-2 dominion in the GR(1) game on $\mathcal{G}^j$ with $X^j_{i,\ell} \cap U^j_\ell = \emptyset$ and thus in particular a dominion w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi $\left(L^j_t\right)$ such that $X^j_{i,\ell} \cap U^j_\ell = \emptyset$. Assume by contradiction $|Attr_2(X^j_{i,\ell}, \mathcal{G}^j)| \leq 2^{i-1}$. Then by Lemma 16(2) we have $X^j_{i-1,\ell} \neq \emptyset$, a contradiction.

2. Assume there exists a dominion D with $|Attr_2(D, \mathcal{G}^j)| \leq h_{\max}$. Then by Lemma 27 there is also a dominion $D' \subseteq D$ that meets the criteria of Lemma 16(2). Let i' the minimal value such that $|Attr_2(D', \mathcal{G}^j)| \leq 2^{i'}$, certainly $i' \leq \lceil \log_2(h_{\max}) \rceil$. Now, by Lemma 16(2), we have that D' is a dominion w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1}$ Büchi $\left(L^j_t \setminus Y^j_{i',\ell}\right)$ in $\mathcal{G}^j_{i'} \setminus Y^j_{i',\ell}$. By the correctness of Algorithm GENBUCHIPROGRESSMEASURE, the set $X^j_{i',\ell}$ is a dominion containing D' and thus `kGenBüchiDominion`$(\mathcal{G}^j, (V^j_1, V^j_2)), \{U^j_\ell\}, \{L^j_t\}, h_{\max})$ returns a non-empty set. ◀

Next we show that whenever Algorithm GR(1)GAME removes vertices from the game graph these vertices are indeed winning for player 2. This is due to Lemma 16(1), stating that these sets are dominions in the current game graph and Lemma 1, stating that all player-2 dominions of the current game graph $\mathcal{G}^j$ are also winning for player 2 in the original game graph $\mathcal{G}$.

▶ **Proposition 42** (Completeness of Algorithm GR(1)GAME)**.** *Let V^{j^*} be the set of vertices returned by Algorithm* GR(1)GAME*. Each vertex in $V \setminus V^{j^*}$ is winning for player 2.*

Proof. By Lemma 1(3) it is sufficient to show that in each iteration j with $S^\ell \neq \emptyset$ player 2 has a winning strategy from the vertices in S^ℓ in $\mathcal{G}^\ell$. If a non-empty set S^j is returned by `kGenBüchiDominion`, then S^j is winning for player 2 by Lemma 16(1). For the case where S^j is empty after the call to `kGenBüchiDominion`, the set S^j is determined in the same way as in the basic algorithm for GR(1) games and thus is winning by the correctness of Algorithm GR(1)GAMEBASIC (cf. Proof of Proposition 28). ◀

For the final game graph $\mathcal{G}^{j^*}$ we can build a winning strategy for player 1 in the same way as for Algorithm GR(1)GAMEBASIC. That is, by combining his winning strategies for the disjunctive objective in the subgraphs $\overline{\mathcal{G}^{j^*}_\ell}$ and the attractor strategies for $Attr_1(U_\ell, \mathcal{G}^{j^*})$.

▶ **Proposition 43** (Soundness of Algorithm GR(1)Game). *Let V^{j^*} be the set of vertices returned by Algorithm* GR(1)Game. *Each vertex in V^{j^*} is winning for player 1.*

Proof. When the algorithm terminates we have $S^j = \emptyset$. Thus the winning strategy of player 1 can be constructed in the same way as for the set returned by Algorithm GR(1)GameBasic (cf. Proof of Proposition 29). ◀

Finally, as the runtime of the procedure `kGenBüchiDominion` scales with the size of the smallest player-2 dominion in $\mathcal{G}^j$ and we have only make $O(\sqrt{n})$ many calls to `GenBüchiGame` we obtain a runtime of $O(k_1 \cdot k_2 \cdot n^{2.5})$.

▶ **Proposition 44** (Runtime Algorithm GR(1)Game). *The algorithm can be implemented to terminate in $O(k_1 \cdot k_2 \cdot n^{2.5})$ time.*

Proof. We analyze the *total* runtime over all iterations of the repeat-until loop. The analysis uses that whenever a player-2 dominion D^j is identified, then the vertices in D^j are removed from the maintained game graph. In particular we have that whenever `kGenBüchiDominion` returns an empty set, either at least $h_{\max} = \sqrt{n}$ vertices are removed from the game graph or the algorithm terminates. Thus this case can happen at most $O(n/h_{\max}) = O(\sqrt{n})$ times. In this case `GenBüchiGame` is called k_2 times. By Theorem 8 this takes total time $O(\sqrt{n} \cdot k_2 \cdot k_1 \cdot n^2) = O(k_1 k_2 \cdot n^{2.5})$.

We next bound the total time spent in `kGenBüchiDominion`. To efficiently construct the graphs G_i^j and the vertex sets Z_i^j we maintain (sorted) lists of the incoming and the outgoing edges of each vertex. These lists can be updated whenever an obsolete entry is encountered in the construction of G_i^j; as each entry is removed at most once, maintaining this data structures takes total time $O(m)$. Now consider a fixed iteration i of the outer for-loop in `kGenBüchiDominion`. The graph G_i^j has $O(2^i \cdot n)$ edges and thus, given the above data structure for adjacent edges, the graphs G_i^j and the sets Z_i^j can be constructed in $O(2^i \cdot n)$ time. Further the k_2 attractor computations in the inner for-loop can be done in time $O(k_2 \cdot 2^i \cdot n)$. The runtime of iteration i is dominated by the k_2 calls to `GenBüchiProgressMeasure`. By Theorem 15 the calls to `GenBüchiProgressMeasure`, with parameter h set to 2^i, in iteration i take time $O(k_1 k_2 \cdot n \cdot 2^{2i})$. Let i^* be the iteration at which `kGenBüchiDominion` stops after it is called in the jth iteration of the repeat-until loop. The runtime for this call to `kGenBüchiDominion` from $i = 1$ to i^* forms a geometric series that is bounded by $O(k_1 k_2 \cdot n \cdot 2^{2i^*})$. By Corollary 41 either (1) a dominion D with $|Attr_2(D, \mathcal{G}^j)| > 2^{i^*-1}$ vertices was found by `kGenBüchiDominion` or (2) all dominions in $\mathcal{G}^j$ have more than $h_{\max}$ vertices. Thus either (2a) a dominion D with more than $h_{\max}$ vertices is detected in the subsequent call to `GenBüchiGame` or (2b) there is no dominion in $\mathcal{G}^j$ and j is the last iteration of the algorithm. Case (2b) can happen at most once and its runtime is bounded by $O(k_1 k_2 \cdot n \cdot 2^{2\log(h_{\max})}) = O(k_1 k_2 \cdot n^2)$. In the cases (1) and (2a) more than 2^{i^*-1} vertices are removed from the graph in this iteration, as $h_{\max} > 2^{i^*-1}$. We charge each such vertex $O(k_1 k_2 \cdot n \cdot 2^{i^*}) = O(k_1 k_2 \cdot n \cdot h_{\max})$ time. Hence the total runtime for these cases is $O(k_1 k_2 \cdot n^2 \cdot h_{\max}) = O(k_1 k_2 \cdot n^{2.5})$. ◀

▶ Remark (Winning Strategies). Algorithm GR(1)Game can be modified to additionally return winning strategies for both players. Procedure `GenBüchiProgressMeasure`$(\mathcal{G}, \psi, h)$ can be modified to return a winning strategy within the returned dominion (cf. proof of Proposition 39). Procedure `GenBüchiGame` can be modified to return winning strategies for both player in the generalized Büchi game (cf. remark at the end of Appendix A). Thus for player 2 a winning strategy for the dominion D^j that is identified in iteration j of the algorithm can be constructed by combining his winning strategy in the generalized Büchi game in which S^j was identified with his attractor strategy to the set S^j. For player 1 we can obtain a winning strategy in the final iteration of the algorithm by combining for $1 \leq \ell \leq k_2$ her attractor strategies to the sets U_ℓ with her winning strategies in the generalized Büchi games for each of the game graphs $\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}$ (cf. proofs of Proposition 29 and 43).