Algorithms for Büchi Games*

Krishnendu Chatterjee[1], Thomas A. Henzinger[1,2], and Nir Piterman[2]

[1] University of California, Berkeley, USA
[2] EPFL, Switzerland
c_krish@eecs.berkeley.edu, {tah,Nir.Piterman}@epfl.ch

Abstract. The classical algorithm for solving Büchi games requires time $O(n \cdot m)$ for game graphs with n states and m edges. For game graphs with constant outdegree, the best known algorithm has running time $O(n^2/\log n)$. We present two new algorithms for Büchi games. First, we give an algorithm that performs at most $O(m)$ more work than the classical algorithm, but runs in time $O(n)$ on infinitely many graphs of constant outdegree on which the classical algorithm requires time $O(n^2)$. Second, we give an algorithm with running time $O(n \cdot m \cdot \log \delta(n)/\log n)$, where $1 \leq \delta(n) \leq n$ is the outdegree of the game graph. Note that this algorithm performs asymptotically better than the classical algorithm if $\delta(n) = O(\log n)$.

1 Introduction

The algorithmic complexity of solving Büchi games is one of the more intriguing questions in graph algorithms. The input of the Büchi game problem consists of a directed graph whose states S are partitioned into player 1 states and player 2 states, and a set $B \subseteq S$ of Büchi states. In each player 1 state, the first player chooses an outgoing edge, and each player 2 state, the second player chooses an outgoing edge. The question is if, from a given start state, player 1 has a strategy to visit a state in B infinitely often.

The classical algorithm for solving Büchi games has time complexity $O(n \cdot m)$, where n is the number of states and m is the number of edges. The classical algorithm proceeds in a seemingly naive fashion: first it computes the set W_1 of states from which player 1 has a strategy to visit B once, which requires time $O(m)$; then it computes the set of states $\overline{W}_1$ such that player 2 has a strategy to visit $S \setminus W_1$ once. The set $\overline{W}_1$ is removed from the game graph and the algorithm iterates over the reduced game graph unless $\overline{W}_1$ is empty for the current game graph. The algorithm converges in at most n iterations, is the answer to the Büchi game problem. This algorithm seemingly performs unnecessarily repetitive work, yet no asymptotically faster algorithm is known.

In [1], we gave a subquadratic algorithm for the special case of graphs with constant outdegree. In this special case $m = O(n)$, and thus the classical algorithm performs in time $O(n^2)$. We gave an algorithm with running time $O(n^2/\log n)$. While the classical algorithm computes each set W_1 by backward search of the graph, our algorithm alternated backward searches with bounded amounts of forward searches.

In this paper, we present two new algorithms for Büchi games. While they fall short of the ultimate goal of a better than $O(n \cdot m)$ algorithm, they present progress in that direction. First, in Section 3 we give an algorithm that performs at most $O(m)$ more work than the classical algorithm. However, there exist families of game graphs of outdegree 2 on which our algorithm performs in time $O(n)$, while the classical algorithm requires time $\Omega(n^2)$. Also there exist families of game graphs of $O(n \log n)$ states with outdegree at most 2 on which both the classical and the algorithm of [1] requires $O(n^2 \log n)$ time, where as our algorithm requires $O(n \log n)$ time. However, there

* This research was supported in part by the AFOSR MURI grant F49620-00-1-0327 and the NSF ITR grant CCR-0225610 and the SNSF under the Indo-Swiss Joint Research Programme.

exist game graphs where our algorithm requires $O(n \cdot m)$ time in the worst case. Our algorithm performs several backward searches, but instead of backward search from B it performs backward search from a subset of $S \setminus B$ that are candidates to be in $S \setminus W_1$.

Second, in Section 4 we give an algorithm with running time $O(n \cdot m \cdot \log \delta(n)/ \log n)$, where $1 \leq \delta(n) \leq n$ is the outdegree of the graph. If $\delta(n) = O(1)$, then this algorithm performs asymptotically like the algorithm of [1]; indeed, the new algorithm is based on the algorithm from [1]. The forward search of the algorithm of [1] was very naive; we develop a more refined forward search which saves repetitive work of the forward search of the algorithm of [1] and thus obtain the generalization of the algorithm of [1] to general game graphs. If $\delta(n) = O(\log n)$, then our algorithm performs better than the classical algorithm; it is to our knowledge the first algorithm that improves on the classical algorithm in this case.

2 Definitions

We consider turn-based deterministic games played by two-players with Büchi and complementary coBüchi objectives for the players, respectively. We define game graphs, plays, strategies, objectives and notion of winning below.

Game graphs. A *game graph* $G = ((S, E), (S_1, S_2))$ consists of a directed graph (S, E) with a finite state space S and a set E of edges, and a partition (S_1, S_2) of the state space S into two sets. The states in S_1 are player 1 states, and the states in S_2 are player 2 states. For a state $s \in S$, we write $E(s) = \{t \in S \mid (s, t) \in E\}$ for the set of successor states of s. We assume that every state has at least one out-going edge, i.e., $E(s)$ is non-empty for all states $s \in S$.

Plays. A game is played by two players: player 1 and player 2, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial state, and then they take moves indefinitely in the following way. If the token is on a state in S_1, then player 1 moves the token along one of the edges going out of the state. If the token is on a state in S_2, then player 2 does likewise. The result is an infinite path in the game graph; we refer to such infinite paths as plays. Formally, a *play* is an infinite sequence $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that $(s_k, s_{k+1}) \in E$ for all $k \geq 0$. We write Ω for the set of all plays.

Strategies. A strategy for a player is a recipe that specifies how to extend plays. Formally, a *strategy* σ for player 1 is a function σ: $S^* \cdot S_1 \to S$ that, given a finite sequence of states (representing the history of the play so far) which ends in a player 1 state, chooses the next state. The strategy must choose only available successors, i.e., for all $w \in S^*$ and $s \in S_1$ we have $\sigma(w \cdot s) \in E(s)$. The strategies for player 2 are defined analogously. We write Σ and Π for the sets of all strategies for player 1 and player 2, respectively. An important special class of strategies are *memoryless* strategies. The memoryless strategies do not depend on the history of a play, but only on the current state. Each memoryless strategy for player 1 can be specified as a function σ: $S_1 \to S$ such that $\sigma(s) \in E(s)$ for all $s \in S_1$, and analogously for memoryless player 2 strategies. Given a starting state $s \in S$, a strategy $\sigma \in \Sigma$ for player 1, and a strategy $\pi \in \Pi$ for player 2, there is a unique play, denoted $\omega(s, \sigma, \pi) = \langle s_0, s_1, s_2, \ldots \rangle$, which is defined as follows: $s_0 = s$ and for all $k \geq 0$, if $s_k \in S_1$, then $\sigma(s_0, s_1, \ldots, s_k) = s_{k+1}$, and if $s_k \in S_2$, then $\pi(s_0, s_1, \ldots, s_k) = s_{k+1}$.

Büchi and coBüchi objectives. We consider game graphs with a Büchi objective for player 1 and the complementary coBüchi objective for player 2. For a play $\omega = \langle s_0, s_1, s_2, \ldots \rangle \in \Omega$, we define $\mathrm{Inf}(\omega) = \{s \in S \mid s_k = s \text{ for infinitely many } k \geq 0\}$ to be the set of states that occur infinitely often in ω. We also define reachability and safety objectives as they will be useful in the analysis of the algorithms.

1. *Reachability and safety objectives.* Given a set $T \subseteq S$ of states, the reachability objective $\mathrm{Reach}(T)$ requires that some state in T be visited, and dually, the safety objective $\mathrm{Safe}(F)$

requires that only states in F be visited. Formally, the sets of winning plays are $\mathrm{Reach}(T) = \{\langle s_0, s_1, s_2, \ldots \rangle \in \Omega \mid \exists k \geq 0.\ s_k \in T\}$ and $\mathrm{Safe}(F) = \{\langle s_0, s_1, s_2, \ldots \rangle \in \Omega \mid \forall k \geq 0.\ s_k \in F\}$. The reachability and safety objectives are dual in the sense that $\mathrm{Reach}(T) = \Omega \setminus \mathrm{Safe}(S \setminus T)$.
2. *Büchi and co-Büchi objectives.* Given a set $B \subseteq S$ of states, the Büchi objective $\mathrm{Buchi}(B)$ requires that some state in B be visited infinitely often, and dually, the co-Büchi objective $\mathrm{coBuchi}(C)$ requires that only states in C be visited infinitely often. Thus, the sets of winning plays are $\mathrm{Buchi}(B) = \{\omega \in \Omega \mid \mathrm{Inf}(\omega) \cap B \neq \emptyset\}$ and $\mathrm{coBuchi}(C) = \{\omega \in \Omega \mid \mathrm{Inf}(\omega) \subseteq C\}$. The Büchi and coBüchi objectives are dual in the sense that $\mathrm{Buchi}(B) = \Omega \setminus \mathrm{coBuchi}(S \setminus B)$.

Winning strategies and sets. Given an objective $\Phi \subseteq \Omega$ for player 1, a strategy $\sigma \in \Sigma$ is a *winning strategy* for player 1 from a state s if for all player 2 strategies $\pi \in \Pi$ the play $\omega(s, \sigma, \pi)$ is winning, i.e., $\omega(s, \sigma, \pi) \in \Phi$. The winning strategies for player 2 are defined analogously. A state $s \in S$ is winning for player 1 with respect to the objective Φ if player 1 has a winning strategy from s. Formally, the set of *winning states* for player 1 with respect to the objective Φ is $W_1(\Phi) = \{s \in S \mid \exists \sigma \in \Sigma.\ \forall \pi \in \Pi.\ \omega(s, \sigma, \pi) \in \Phi\}$. Analogously, the set of winning states for player 2 with respect to an objective $\Psi \subseteq \Omega$ is $W_2(\Psi) = \{s \in S \mid \exists \pi \in \Pi.\ \forall \sigma \in \Sigma.\ \omega(s, \sigma, \pi) \in \Psi\}$. We say that there exists a memoryless winning strategy for player 1 with respect to the objective Φ if there exists such a strategy from all states in $W_1(\Phi)$; and similarly for player 2.

Theorem 1 (Classical memoryless determinacy). *The following assertions hold.*

1. *For all game graphs $G = ((S, E), (S_1, S_2))$, all Büchi objectives Φ for player 1, and the complementary coBüchi objective $\Psi = \Omega \setminus \Phi$ for player 2, we have $W_1(\Phi) = S \setminus W_2(\Psi)$.*
2. *For all game graphs and all Büchi objectives Φ for player 1 and the complementary coBüchi objective Ψ for player 2, there exists a memoryless winning strategy for both players.*

Observe that for Büchi objective Φ and the coBüchi objective $\Psi = \Omega \setminus \Phi$ by definition we have $S \setminus W_2(\Psi) = \{s \in S \mid \forall \pi \in \Pi.\ \exists \sigma \in \Sigma.\ \omega(s, \sigma, \pi) \in \Phi\}$. Theorem 1 states that $S \setminus W_2(\Psi) = \{s \in S \mid \exists \sigma \in \Sigma.\ \forall \pi \in \Pi.\ \omega(s, \sigma, \pi) \in \Phi\}$, i.e., the order of the universal and the existential quantifiers can be exchanged.

3 Iterative Algorithms for Büchi Games

In this section we present the classical iterative algorithm for Büchi games and an alternative iterative algorithm, i.e., algorithms to compute the winning sets in Büchi games. The running time of the alternative algorithm is never more than the running time the classical algorithm by an additive factor of $O(m)$, where m is the number of edges in the game graph. We also present a family of game graphs with Büchi objectives where the classical algorithm requires quadratic time and the alternative algorithm works in linear time. We start with the notion of *closed sets* and *attractors* which are key notions for the analysis of the algorithm.

Closed sets. A set $U \subseteq S$ of states is a *closed set* for player 1 if the following two conditions hold: (a) for all states $u \in (U \cap S_1)$, we have $E(u) \subseteq U$, i.e., all successors of player 1 states in U are again in U; and (b) for all $u \in (U \cap S_2)$, we have $E(u) \cap U \neq \emptyset$, i.e., every player 2 state in U has a successor in U. The closed sets for player 2 are defined analogously. Every closed set U for player ℓ, for $\ell \in \{1, 2\}$, induces a sub-game graph, denoted $G \upharpoonright U$.

Proposition 1. *Consider a game graph G, and a closed set U for player 1. Then the following assertions hold.*

1. *There exists a memoryless strategy π for player 2 such that for all strategies σ for player 1 and for all states $s \in U$ we have $\omega(s, \sigma, \pi) \in \mathrm{Safe}(U)$.*

2. For all $T \subseteq S \setminus U$, we have $W_1(\text{Reach}(T)) \cap U = \emptyset$.

Attractors. Given a game graph G, a set $U \subseteq S$ of states, and a player $\ell \in \{1,2\}$, the set $Attr_\ell(U,G)$ contains the states from which player ℓ has a strategy to reach a state in U against all strategies of the other player; that is, $Attr_\ell(U,G) = W_\ell(\text{Reach}(U))$. The set $Attr_1(U,G)$ can be computed inductively as follows: let $R_0 = U$; let

$$R_{i+1} = R_i \cup \{s \in S_1 \mid E(s) \cap R_i \neq \emptyset\} \cup \{s \in S_2 \mid E(s) \subseteq R_i\} \qquad \text{for all } i \geq 0;$$

then $Attr_1(U,G) = \bigcup_{i \geq 0} R_i$. The inductive computation of $Attr_2(U,G)$ is analogous. For all states $s \in Attr_1(U,G)$, define $rank(s) = i$ if $s \in R_i \setminus R_{i-1}$, that is, $rank(s)$ denotes the least $i \geq 0$ such that s is included in R_i. Define a memoryless strategy $\sigma \in \Sigma$ for player 1 as follows: for each state $s \in (Attr_1(U,G) \cap S_1)$ with $rank(s) = i$, choose a successor $\sigma(s) \in (R_{i-1} \cap E(s))$ (such a successor exists by the inductive definition). It follows that for all states $s \in Attr_1(U,G)$ and all strategies $\pi \in \Pi$ for player 2, the play $\omega(s,\sigma,\pi)$ reaches U in at most $|Attr_1(U,G)|$ transitions.

Proposition 2. *For all game graphs G, all players $\ell \in \{1,2\}$, and all sets $U \subseteq S$ of states, the set $S \setminus Attr_\ell(U,G)$ is a closed set for player ℓ.*

3.1 Classical algorithm for Büchi games

In this subsection we present the classical algorithm for Büchi games. We start with an informal description of the algorithm.

Informal description of classical algorithm. The *classical algorithm* (Algorithm 1) works as follows. We describe an iteration i of the algorithm: the set of states at iteration i is denoted as S^i, the game graph as G_i and the set of Büchi states $B \cap S^i$ as B_i. At iteration i, the algorithm first finds the set of states R_i from which player 1 has a strategy to reach the set B_i, i.e., computes $Attr_1(B_i, G_i)$. The rest of the states $Tr_i = S^i \setminus R_i$ is a closed subset for player 1, and $Tr_i \cap B_i = \emptyset$. The set Tr_i is identified as winning for player 2. Then the set of states W_{i+1}, from which player 2 has a strategy to reach the set Tr_i, i.e., $Attr_2(Tr_i, G_i)$ is computed. The set W_{i+1} is identified as a subset of the winning set for player 2 and it is removed from the state set. The algorithm then iterates on the reduced game graph. Observe that at every iteration the set of states removed is an attractor set and by Proposition 2 the reduced game graph (the complement of an attractor) is a closed set and hence a game graph. In every iteration it performs a *backward* search from the current Büchi states to find the set of states which can reach the Büchi set. Each iteration takes $O(m)$ time and the algorithm runs for at most $O(n)$ iterations, where m and n denote the number of edges and states in the game graph, respectively. The algorithm is formally described as Algorithm 1. The correctness of the algorithm easily follows from the results in [2, 3].

Theorem 2 (Correctness and running time). *Given a game graph $G = ((S,E),(S_1,S_2))$ and $B \subseteq S$ the following assertions hold:*

1. *$W = W_2(\text{coBuchi}(S \setminus B))$ and $S \setminus W = W_1(\text{Buchi}(B))$, where W is the output of Algorithm 1;*
2. *the running time of Algorithm 1 is $O(b \cdot m)$ where $b = |B|$ and $m = |E|$.*

Remark 1. Observe that the size of the set of Büchi states B can be $O(n)$, where $n = |S|$ is the number of states, i.e., b in Theorem 2 can be $O(n)$. Hence the worst case running time of the classical algorithm can be $O(n \cdot m)$, where $n = |S|$ and $m = |E|$.

Algorithm 1 Classical algorithm for Büchi Games

Input : A 2-player game graph $G = ((S, E), (S_1, S_2)$ and $B \subseteq S$.
Output: $W \subseteq S$.
1. $G_0 := G$; $S^0 := S$; 2. $W_0 := \emptyset$; 3. $i := 0$
4. **repeat**
    4.1 $W_{i+1} := \mathsf{AvoidSetClassical}(G_i, B \cap S^i)$
    4.2 $S^{i+1} := S^i \setminus W_{i+1}$; $G_{i+1} = G \upharpoonright S^{i+1}$; 4.3 $i := i + 1$;
  until $W_i = \emptyset$
5. **return** $W := \bigcup_{k=1}^{i} W_k$.

Procedure AvoidSetClassical
Input: Game graph G_i and $B_i \subseteq S^i$.
Output: set $W_{i+1} \subseteq S^i$.
1. $R_i := Attr_1(B_i, G_i)$; 2. $Tr_i := S^i \setminus R_i$; 3. $W_{i+1} := Attr_2(Tr_i, G_i)$

3.2 Alternative algorithm for Büchi games

We now present a new alternative iterative algorithm for Büchi games. The algorithm differs from the classical algorithm in its computation of the set Tr_i (computed in step 2 of procudure AvoidSetClassical) at every iteration. Recall that the set Tr_i is a player 1 closed set with empty intersection with the set of Büchi states. The alternative algorithm at every iteration identifies the set Tr_i in an alternative way. We first present an informal description of the algorithm.

Informal description of alternative algorithm. We describe an iteration i of Algorithm 2. We denote by $C = S \setminus B$ the set of coBüchi states. We denote the set of states at iteration i by S^i, the game graph as G_i, the set of Büchi states $B \cap S^i$ as B_i, and the set of coBüchi states as $C \cap S^i$ as C^i. The algorithm proceeds as follows: first it computes the set of player 1 states in C^i with all successors in C^i and the set of player 2 states in C^i with a successor in C^i. Then the player 2 attractor to the union of the above two sets is computed and let this set be Z_i. The states of Z_i such that player 1 has a strategy to leave Z_i is not a part of the player 1 closed set, and the remaining states of Z_i is a player 1 closed set with empty intersection with B_i, and this is identified as the set similar to the set Tr_i of Algorithm 1. The details of the algorithms is as follows. In step 4.2 the set of player 1 states C_1^i is computed where for all $s \in C_1^i$, all successors of s in G_i is in C^i; and in step 4.3 the set of player 2 states C_2^i is computed where for all $s \in C_2^i$, there is a successor of s in C^i. Then the set X_i is computed as the set of states such that player 2 has a strategy in G_i to reach $C_1^i \cup C_2^i$ against all player 1 strategies, i.e., $X_i = Attr_2(C_1^i \cup C_2^i, G_i)$, and the set Z_i is obtained as $Z_i = X_i \cap C^i$. The set D_i denotes the set of states such that player 1 can escape Z_i in one step, i.e., either a player 1 state with an edge out of Z_i or a player 2 state with all edges out of Z_i. The set L_i denotes the set of states where player 1 has a strategy in X_i to reach D_i against all player 2 strategies, i.e., $L_i = Attr_1(D_i, G_i \upharpoonright X_i)$. Observe that the set X_i is not always a proper sub-game, however, for all states in $s \in X_i \setminus D_i$ we have $E(s) \cap X_i \neq \emptyset$, and hence for the purpose of the computation of the attractor of D_i we can consider $G_i \upharpoonright X_i$ as a sub-game. The set $\widehat{Tr}_i = Z_i \setminus L_i$ is identified as winning for player 2. Then the set of states W_{i+1}, from which player 2 has a strategy to reach the set $\widehat{Tr}_i$, i.e., $Attr_2(\widehat{Tr}_i, G_i)$ is computed. The set W_{i+1} is identified as a subset of the winning set for player 2 and it is removed from the state set. The algorithm then iterates on the reduced game graph. The algorithm is described formally as Algorithm 2.

Correctness arguments. The main argument to prove the correctness of Algorithm 2 is as follows: we will show that, given that the game graph G_i are same at iteration i of Algorithm 1 and Algorithm 2, set Tr_i computed in step 2 of the iteration of classical algorithm and the set

Algorithm 2 Alternative algorithm for Büchi Games

Input : A 2-player game graph $G = ((S, E), (S_1, S_2)$ and $B \subseteq S$.
Output: $W \subseteq S$.
1. $G_0 := G$; $S^0 := S$; $C = S \setminus B$; 2. $W_0 := \emptyset$; 3. $i := 0$
4. **repeat**
 4.1 $C^i := C \cap S^i$;
 4.2 $C_1^i := \{s \in S_1 \cap C^i \mid E(s) \cap S^i \subseteq C^i\}$;
 4.3 $C_2^i := \{s \in S_2 \cap C^i \mid E(s) \cap C^i \neq \emptyset\}$;
 4.4 $X_i := Attr_2(C_1^i \cup C_2^i, G_i)$;
 4.5 $Z_i := X_i \cap C^i$;
 4.6 $D_i := \{s \in S_1 \cap Z_i \mid E(s) \cap S^i \cap (S^i \setminus Z_i) \neq \emptyset\}$
  $\cup\{s \in S_2 \cap Z_i \mid E(s) \cap S^i \subseteq (S^i \setminus Z_i)\} \cup (X_i \setminus Z_i)$.
 4.7 $L_i = Attr_1(D_i, G_i \upharpoonright X_i)$;
 4.8 $\widehat{Tr}_i = Z_i \setminus L_i$;
 4.9 $W_{i+1} := Attr_2(\widehat{Tr}_i, G_i)$;
 4.10 $S^{i+1} := S^i \setminus W_{i+1}$; $G_{i+1} = G \upharpoonright S^{i+1}$; $i := i + 1$;
until $W_i = \emptyset$
5. **return** $W := \bigcup_{k=1}^{i} W_k$.

$\widehat{Tr}_i$ computed in step 4.8 of the alternative algorithm coincide. Once we prove this result the correctness of the alternative algorithm follows easily. We prove this result in several steps. The following proposition states that Tr_i is the largest player 1 closed subset of C^i and it follows easily from the properties of attractors and Proposition 1.

Proposition 3. *Let G_i be the graph at iteration i if Algorithm 1 and let $U_i \subseteq C^i$ such that U_i is player 1 closed, then $U_i \subseteq Tr_i$.*

By Proposition 3 to prove our desired claim it suffices to show that $Tr_i \subseteq \widehat{Tr}_i$, and $\widehat{Tr}_i$ is a player 1 closed subset of C^i (this would imply $\widehat{Tr}_i \subseteq Tr_i$).

Lemma 1. *$Tr_i \subseteq C_1^i \cup C_2^i$.*

Proof. Observe that Tr_i is a player 1 closed of C^i. Hence for all states $s \in Tr_i$ the following assertions hold: (a) if $s \in S_1$, then $E(s) \cap S^i \subseteq Tr_i \subseteq C^i$; and (b) if $s \in S_2$, then $E(s) \cap Tr_i \neq \emptyset$, and hence $E(s) \cap C^i \neq \emptyset$. Hence if $s \in Tr_i$, then $s \in C_1^i \cup C_2^i$. ∎

Lemma 2. *$Tr_i \subseteq Z_i$, where Z_i is the set computed at step 4.5 of Algorithm 2.*

Proof. By Lemma 1 we have $Tr_i \subseteq C_1^i \cup C_2^i$ and hence we have $Tr_i \subseteq Attr_2(C_1^i \cup C_2^i, G_i)$. Since $Tr_i \subseteq C^i$, we have $Tr_i \subseteq X_i \cap C^i = Z_i$. ∎

Lemma 3. *$Tr_i \subseteq Z_i \setminus L_i$, where Z_i and L_i are the sets computed at step 4.5 and 4.7 of Algorithm 2, respectively.*

Proof. By Lemma 2 we have $Tr_i \subseteq Z_i$ and hence we have $S^i \setminus Z_i \subseteq S^i \setminus Tr_i$. Observe that from states in D_i player 1 can force the game to reach $(S^i \setminus Z_i)$. Similarly, in the sub-game X_i with D_i as target set, player 1 can force the game to reach D_i from L_i and then force the game to reach $S^i \setminus Z_i$. Since $Tr_i \subseteq Z_i \subseteq X_i$ and Tr_i is a player 1 closed set, player 2 can keep the game in Tr_i forever (by Proposition 1). Hence we must have $Tr_i \cap L_i = \emptyset$. Hence we have $Tr_i \subseteq Z_i$ and $Tr_i \cap L_i = \emptyset$. Thus we obtain $Tr_i \subseteq Z_i \setminus L_i$. ∎

Lemma 4. *$Z_i \setminus L_i \subseteq C^i$ and $Z_i \setminus L_i$ is a player 1 closed set.*

Proof. Since $X_i = Attr_2(C_1^i \cup C_2^i, G_i)$, it follows that for all states $s \in X_i$ we have (a) if $s \in S_1 \cap X_i$, then $E(s) \cap S^i \subseteq X_i$, and (b) if $s \in S_2 \cap X_i$, then we have $E(s) \cap X_i \neq \emptyset$. Since $L_i = Attr_1(D_i, G_i \upharpoonright X_i)$, for all states $s \in (Z_i \setminus L_i)$ the following assertions hold: (a) if $s \in S_1$, then $E(s) \cap S^i \subseteq (Z_i \setminus L_i)$, as otherwise s would have been in L_i; and (b) if $s \in S_2$, then $E(s) \cap (Z_i \setminus L_i) \neq \emptyset$, as otherwise s would have been in L_i. Hence it follows that $(Z_i \setminus L_i)$ is player 1 closed, and since $Z_i \subseteq C^i$, the desired result follows. ∎

Lemma 5. *$Tr_i = \widehat{Tr}_i$.*

Proof. By Lemma 3 we have $Tr_i \subseteq \widehat{Tr}_i$. By Lemma 4 we have $\widehat{Tr}_i$ is a player 1 closed subset of C^i. Then, by Proposition 3 we have $\widehat{Tr}_i \subseteq Tr_i$. This proves the desired result. ∎

The correctness of Algorithm 2 easily follows from Lemma 5 and induction.

Theorem 3 (Correctness of Algorithm 2). *Given a game graph $G = ((S, E), (S_1, S_2))$ and $B \subseteq S$ we have $W = W_2(\text{coBuchi}(S \setminus B))$ and $S \setminus W = W_1(\text{Buchi}(B))$, where W is the output of Algorithm 2.*

Work analysis of Algorithm 2. We first analyze the work for the computation of step 4.2 and step 4.3 over all iterations of Algorithm 2. It easily follows that $C_2^i = C_2^0 \cap S^i$, this follows since if a state $s \in S_2$ has an edge to W_{i-1}, then s would have been in W_{i-1} itself. The total work in the computation of the sets C_1^i overall iterations is $O(m)$: this is achieved as follows. For every state $s \in S_1$ we keep a counter for the number of edges to the set $S \setminus C = B$, and once an edge to $S \setminus C$ is removed from the graph the counter for the respective state is decremented by 1. Once the counter for a state reaches 0 it is included in C_1^i. Hence the total work for step 4.2 and step 4.3 overall iterations is $O(m)$. We now argue that the excess work of Algorithm 2 as compared to the classical algorithm is at most $O(m)$. The total work of step 4.2 and step 4.3 is already bounded by $O(m)$. The rest of the argument is as follows: the classical algorithm for the computation of Tr_i never works on the edges in Tr_i, i.e., on edges in $E \cap (Tr_i \times Tr_i)$. At iteration i Algorithm 2 does excess work as compared to classical algorithm on the edges in Tr_i and does only constant amount of work on this edges. However, edges in Tr_i are removed at every iteration i, and hence the excess work of Algorithm 2 as compared to the classical algorithm is $O(m)$.

Theorem 4 (Running time of Algorithm 2). *Let $\mathsf{RT}_1(G, B)$ denote the running time of Algorithm 1 on a game graph G with $B \subseteq S$ and $\mathsf{RT}_2(G, B)$ denote the running time of Algorithm 2. Then we have $\mathsf{RT}_2(G, B) \leq \mathsf{RT}_1(G, B) + O(m)$, where $m = |E|$.*

We now present an example of a family of game graphs where the classical algorithm requires quadratic time, whereas Algorithm 2 works in linear time.

Example 1. The family of game graphs is constructed as follows. Given $n \geq 1$ we consider a game graph consisting of n gadgets $\mathcal{H}(0), \mathcal{H}(1), \ldots \mathcal{H}(n)$ as follows. The gadget $\mathcal{H}(i)$ consits of a player 1 state t_i (shown as a $\bigcirc$-state in Fig 1) and a player 2 state w_i (shown as a $\diamond$ state in Fig 1). The set of edges is as follows:

1. For $0 \leq i < n$ we have $E(w_i) = \{t_i, t_{i+1}\}$ and $E(w_n) = \{t_n\}$.
2. For $0 < i \leq n$ we have $E(t_i) = \{t_i, w_{i-1}\}$ and $E(t_0) = \{t_0\}$.

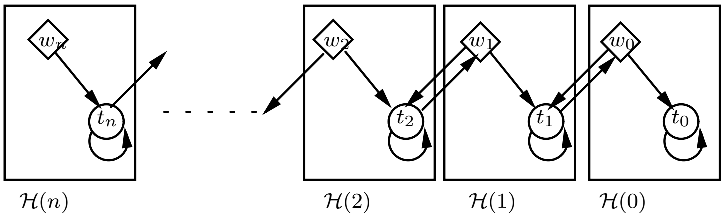

Fig. 1.

The set of Büchi states is $B = \{w_i \mid 0 \leq i \leq n\}$. Given $n \geq 1$, the game graph constructed has $2n$ states and $4n - 2$ edges, i.e., we have $O(n)$ states and $O(n)$ edges.

The sets Tr_i and W_{i+1} obtained at iteration i, for $i \geq 0$, for the classical algorithm are as follows: $Tr_i = \{t_i\}$ and $W_{i+1} = \{t_i, w_i\}$. At iteration i the classical algorithm works on edges of gadgets $\mathcal{H}(\ell)$, for $\ell \geq i+1$. Hence the total work of the classical algorithm is at least $\sum_{i=1}^{n-1}(n-i) = O(n^2)$, i.e., the classical algorithm requires time quadratic in the size of the game graph.

We now analyze the work of Algorithm 2. The sets of Algorithm 2 of iteration i, for $i \geq 0$, is as follows: (a) $C_1^i = \{t_i\}$ and $C_2^i = \emptyset$; (b) $X_i = \{t_i, w_i\}$ and $Z_i = \{t_i\}$; and (c) $D_i = \{t_i\}$, $L_i = \{w_i\}$ and $\widehat{Tr}_i = \{t_i\}$. The work of Algorithm 2 is constant for every iteration i: since the computation of steps 4.4 to steps 4.9 only works on edges in $\mathcal{H}(i)$ and $\mathcal{H}(i+1)$. Hence the total work of Algorithm 2 is $O(n)$, i.e., the alternative algorithm works in linear time.

Also if in every gadget $\mathcal{H}(i)$ the self-loop at state t_i is replaced by a cycle of $2\log(n)$ states, then both the classical algorithm and the algorithm of [1] requires $O(n^2 \log(n))$ time, whereas Algorithm 2 requires $O(n \log(n))$ time. ∎

Dovetailing Algorithm 1 and Algorithm 2. We already proved that the set Tr_i and $\widehat{Tr}_i$ of Algorithm 1 and Algorithm 2 coincide. Algorithm 1 never works on the edges in Tr_i and is favorable when Tr_i is large and Algorithm 2 is favorable when Tr_i is small. Hence in every iteration both the algorithms can be run in a dovetailing fashion and obtaining Tr_i by the algorithm that completes first. The computation of the sets C_1^{i+1} and C_2^{i+1} can be computed during the computation of the set W_{i+1}.

4 Improved Algorithm for Büchi Games

In this section we present the improved algorithm for Büchi games. The algorithm is a generalization of the improved algorithm of [1] to general game graphs, as compared to the algorithm of [1] which works only for binary game graphs (game graphs with every state having out-degree at most 2). We will use the following notations in this section.

Notation. For a set U and a game graph G we denote by $\text{source}(U, G) = \{s \in S \mid E(s) \cap U \neq \emptyset\}$ is the set of states with edges that enter U. Given a game graph G we denote by $\delta(G) = \max\{|E(s)| \mid s \in S\}$ the maximum out-degree of a state in G.

Informal description of the new algorithm We observe that in step 1 of every iteration i of the classical algorithm an $O(m)$ *backward* alternating search is performed to compute the set R_i, where m is the number of edges. The key idea of our *improved algorithm* (Algorithm 3) is to perform a cheap *forward* exploration of edges in order to discover subsets of the winning set for player 2. Let U be the set of sources of edges entering the winning set of player 2 discovered in

Algorithm 3 Improved Algorithm for Büchi Games

Input : A 2-player game graph $G = ((S, E), (S_1, S_2)$ and $B \subseteq S$.
Output: $W \subseteq S$.
1. $G_0 := G$; $S^0 := S$; $C = S \setminus B$; 2. $W_0 := \emptyset$; 3. $i := 0$
4. **repeat**
   4.1 **if** $(|\text{source}(W_i, G)| \geq \frac{m}{\log(n)})$
      4.1.1 $W_{i+1} := \mathsf{AvoidSetClassical}(G_i, B \cap S^i)$
      4.1.2 **go to** step 4.3.
   4.2 **else**
      4.2.1 **add** a state $\widehat{s}$ and an edge fro $\widehat{s}$ to a state in $s \in \text{source}(W_i, G)$
      4.2.2 Find the reachable subgraph $R_{\widehat{s}}$ by a BFS for $\left(2 \cdot \frac{m}{\log n}\right)$ steps
      4.2.3 Let $F_{\widehat{s}}$ denote the set of states in the frontier of the BFS
      4.2.4 $T_{\widehat{s}} := \{s \in S_2 \cap F_{\widehat{s}} \mid E(s) \cap S^i \cap R_{\widehat{s}} = \emptyset\} \cup (S_1 \cap F_{\widehat{s}})$
      4.2.5 $A_{\widehat{s}} := \mathit{Attr}_1((R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}, G_i \upharpoonright R_{\widehat{s}})$
      4.2.6 $\mathit{Tr}_i := (R_{\widehat{s}} \setminus A_{\widehat{s}})$
      4.2.7 **if** $(\mathit{Tr}_i \neq \emptyset)$
         4.2.7.1 **then** $W_{i+1} := \mathit{Attr}_2(\mathit{Tr}_i, G_i)$
      4.2.8 **else**
         4.2.8.1 $W_{i+1} := \mathsf{AvoidSetClassical}(G_i, B \cap S^i)$
      4.2.9 **remove** the state $\widehat{s}$;

   4.3 $S^{i+1} := S^i \setminus W_{i+1}$; $G_{i+1} = G \upharpoonright S^{i+1}$; $i := i + 1$;
   until $W_i = \emptyset$
5. **return** $W := \bigcup_{k=1}^{i} W_k$.

the previous iteration. The states in set U are new candidates to be included in the winning set of player 2. The cheap forward exploration of edges is performed when the size of the set U is small. Formally, if $|U| \geq \left(\frac{m}{\log(n)}\right)$, then an iteration of the classical algorithm is executed (step 4.1), i.e., the backward search is performed. Otherwise, we perform the cheap forward search as follows: we add an auxiliary state $\widehat{s}$ with an edge to every state in U. From the state $\widehat{s}$ a BFS is performed for $\left(2 \cdot \frac{m}{\log n}\right)$ steps in step 4.2.2 of Algorithm 3. In steps 4.2.3—4.2.7 we check if the explored subgraph contains a closed set for player 1 in which player 2 has a winning strategy. If no such set is detected then one iteration of the classical algorithm is executed. The key for an improved bound of our algorithm is the observation that if step 4.2.7 fails to identify a non-empty winning subset for player 2, then the set discovered by the following iteration of the classical algorithm has at least $\left(\frac{\log n}{\log(\delta(G))}\right)$ states. A formal presentation of the algorithm is given as Algorithm 3.

Theorem 5 (Correctness of Algorithm 3). *Given a game graph $G = ((S, E), (S_1, S_2))$ and $B \subseteq S$ we have $W = W_2(\text{coBuchi}(S \setminus B))$ and $S \setminus W = W_1(\text{Buchi}(B))$, where W is the output of Algorithm 3.*

Proof. We prove by induction that W_i computed in any iteration of the improved algorithm satisfies $W_i \subseteq W_2(\text{coBuchi}(S \setminus B))$.
Base case: $W_0 = \emptyset \subseteq W_2(\text{coBuchi}(S \setminus B))$.
Inductive case: We argue that $W_i \subseteq W_2(\text{coBuchi}(S \setminus B))$ implies that $W_{i+1} \subseteq W_2(\text{coBuchi}(S \setminus B))$. The case when step 4.1.1 gets executed, or step 4.2.7 fails and step 4.2.8 gets executed, then the correctness follows from the correcntess of the iteration of the classical algorithm. We focus on the case when step 4.2.7 gets executed, i.e., a non-empty set Tr_i is discovered as $(R_{\widehat{s}} \setminus A_{\widehat{s}})$. For state $s \in S_1 \cap (R_{\widehat{s}} \setminus F_{\widehat{s}})$, we have $E(s) \cap S^i \subseteq R_{\widehat{s}}$. It follows from step 4.2.4 that $F_{\widehat{s}} \cap S_1 \subseteq T_{\widehat{s}}$. Let

$Z = (R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}$. Hence the following two conditions hold:

$$(1).\ F_{\widehat{s}} \cap S_1 \subseteq T_{\widehat{s}} \subseteq Attr_1(Z, R_{\widehat{s}}) \subseteq A_{\widehat{s}}; \qquad (2).\ R_{\widehat{s}} \cap B \cap S^i \subseteq Attr_1(Z, R_{\widehat{s}}) \subseteq A_{\widehat{s}}.$$

By property of attractor we have the following property for $(R_{\widehat{s}} \setminus A_{\widehat{s}})$; for all states $s \in (R_{\widehat{s}} \setminus A_{\widehat{s}})$ the following assertions hold: (a) if $s \in S_1$, then $E(s) \cap S^i \subseteq (R_{\widehat{s}} \setminus A_{\widehat{s}})$, and (b) if $s \in S_2$, then $E(s) \cap S^i \cap (R_{\widehat{s}} \setminus A_{\widehat{s}}) \neq \emptyset$.

Hence $(R_{\widehat{s}} \setminus A_{\widehat{s}})$ is a player 1 closed set in G_i and $(R_{\widehat{s}} \setminus A_{\widehat{s}}) \cap B = \emptyset$. It follows that $(R_{\widehat{s}} \setminus A_{\widehat{s}}) \subseteq W_2(\text{coBuchi}(S \setminus B))$. Hence it follows that $W_{i+1} \subseteq W_2(\text{coBuchi}(S \setminus B))$. The correctness of Algorithm 3 follows. ∎

Work analysis of algorithm 3. We now focus on the work analysis of Algorithm 3. Let us denote by ℓ the depth of the search of the BFS at step 4.2.2 of Algorithm 3. Since source$(W_i, G) \leq \left(\frac{m}{\log(n)}\right)$ and the BFS proceeds for $\left(2 \cdot \frac{m}{\log(n)}\right)$ steps, the BFS explores at least $\frac{m}{\log(n)}$ edges of the game graph G_i. Hence must have $\delta(G)^\ell \geq \left(\frac{m}{\log(n)}\right)$. Thus we obtain that $\ell \geq O\left(\frac{\log(m)}{\log(\delta(G))}\right) = O\left(\frac{\log(n)}{\log(\delta(G))}\right)$. In the following lemma we denote by ℓ the depth of the BFS search at step 4.2.2 of Algorithm 3.

Lemma 6. *Let $R_{\widehat{s}}$ be the set computed in step 4.2.2 of Algorithm 3 and ℓ be the depth of the BFS in step 4.2.2. Let $W \subseteq R_{\widehat{s}}$ be an player 1 closed set such that $W \cap B = \emptyset$ and $|W| < \ell$. Then $W \subseteq (R_{\widehat{s}} \setminus A_{\widehat{s}})$, and hence W is discovered in step 4.2.7.*

Proof. Given a game graph G and a set of states U we define sequences U_i as follows:

$$U_0 = U; \qquad U_{k+1} = U_k \cup \{s \in S_1 \mid E(s) \cap S_k \neq \emptyset\} \cup \{s \in S_2 \mid E(s) \subseteq S_k\} \quad k \geq 0.$$

By definition $Attr_1(U, G) = \bigcup_k U_k$. We prove by induction that $W \cap A_{\widehat{s}} = \emptyset$. By Step 4.2.5 we have $A_{\widehat{s}} = Attr_1((R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}, G_i \upharpoonright R_{\widehat{s}})$. Let U_i be the sequence of set of states in the attractor computation of $(R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}$ with $U_0 = (R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}$. We show by induction that $U_i \cap W = \emptyset$.
Base case. Given $|W| \leq \ell - 1$, for all states $w \in W$ there is a path from $\widehat{s}$ of length at most $|W| \leq \ell - 1$ to w. It follows that in the BFS from $\widehat{s}$ depth of any state $w \in W$ is less than ℓ. Hence we have $W \cap F_{\widehat{s}} = \emptyset$. Since $T_{\widehat{s}} \subseteq F_{\widehat{s}}$ we have $W \cap T_{\widehat{s}} = \emptyset$. Since $W \cap B = \emptyset$ we have $W \cap (B \cap S^i) = \emptyset$. It follows that $W \cap ((R_{\widehat{s}} \cap B \cap S^i) \cup T_{\widehat{s}}) = \emptyset$. This proves the base case that $U_0 \cap W = \emptyset$.
Inductive case. Given $U_k \cap W = \emptyset$ we show that $U_{k+1} \cap W = \emptyset$. Since W is a player 1 closed set, the following assertions hold: for a state $s \in W \cap S_2$, we have $E(s) \cap W \neq \emptyset$ and for a state $s \in W \cap S_1$, we have $E(s) \subseteq W$. Consider a state $s \in W \cap S_2$, since $U_k \cap W = \emptyset$, then $\exists t \in E(s)$ and $t \notin U_k$ and hence $s \notin U_{k+1}$. Consider a state $s \in W \cap S_1$, since $E(s) \subseteq W$ and $W \cap U_k = \emptyset$, we have $E(s) \cap U_k = \emptyset$. Hence $s \notin U_{k+1}$. Hence $W \cap U_{k+1} = \emptyset$.

It follows that $W \cap A_{\widehat{s}} = \emptyset$. Since $W \subseteq R_{\widehat{s}}$ it follows that $W \subseteq (R_{\widehat{s}} \setminus A_{\widehat{s}})$. The result follows. ∎

Lemma 7. *The total work of step 4.2.2 — step 4.2.6 of Algorithm 3 is $O\left(\frac{n \cdot m}{\log(n)}\right)$ and the total work of step 4.2.7 is $O(m)$.*

Proof. Consider an iteration of Algorithm 3: since step 4.2.2 gets executed for $\left(2 \cdot \frac{m}{\log(n)}\right)$ steps it follows that size of the graph $R_{\widehat{s}}$ is $O\left(\frac{m}{\log(n)}\right)$. It follows that in any iteration the total work of step 4.2.2 — step 4.2.6 is $O\left(\frac{m}{\log(n)}\right)$. Since there can be at most $O(n)$ iterations of the algorithm the result for step 4.2.2 — step 4.2.6 follows.

The edges on which step 4.2.7 works are removed for further iteration when we remove W_{i+1} from the present set of states. Hence in step 4.2.7 no edge is worked for more than once. Thus we obtain that total work of step 4.2.7 of Algorithm 3 is $O(m)$. ∎

Lemma 8. *The total work in step 4.2.8 of Algorithm 3 is* $O\big(n \cdot m \cdot \frac{\log(\delta(G))}{\log(n)}\big)$.

Proof. In Algorithm 3 when step 4.2.8 gets executed let Tr_i be the set of vertices identified by the iteration of the classical algorithm. If $|Tr_i| \leq \ell - 1$, where ℓ is the depth of the BFS search of step 4.2.2, then it follows from Lemma 6 that it would have been identified by step 4.2.7 in of the iteration. Hence every time 4.8 gets executed at least ℓ states are removed from the graph. So step 4.2.8 can be executed at most $O\big(\frac{n}{\ell}\big)$ times, where $\ell \geq O\big(\frac{\log(n)}{\log(\delta(G))}\big)$. The work at every iteration is $O(m)$ and hence the total work of step 4.2.8 of Algorithm 3 is $O\big(n \cdot m \cdot \frac{\log(\delta(G))}{\log(n)}\big)$. ∎

Lemma 9. *The total work in step 4.1 of Algorithm 3 is* $O(m \cdot \log(n))$.

Proof. The condition of step 4.1 ensures that whenever step 4.1 gets executed as least $\big(\frac{m}{\log(n)}\big)$ edges are removed from the graph in the previous iteration. Hence step 4.1 gets executed at most $\log(n)$ times and each iteration takes $O(m)$ work. The result follows. ∎

Lemma 7, Lemma 8 and Lemma 9 yield the following result.

Theorem 6. *The total work of Algorithm 3 on a game graph G with a Büchi objective* Buchi(B)*, where $B \subseteq S$, is* $O\big(n \cdot m \cdot \frac{\log(\delta(G))}{\log(n)}\big)$.

Remark 2. Observe that for a game graph G with $n = |S|$ we have $\delta(G) \leq n$, and hence Algorithm 3 is asymptotically no worse than the classical algorithm. In [1] an improved algorithm is presented for binary game graphs (where every state has out-degree at most 2) with a running time of $O\big(\frac{n^2}{\log(n)}\big)$, for game graphs with n-states. For the special case of binary game graphs the running time of Algorithm 3 matches the bound of the algorithm of [1]. However, there exists game graphs where Algorithm 3 out-performs both the classical algorithm and the algorithm of [1]. For example consider the class of game graphs G with $n = |S|$ and $|E(s)| = O(\log(n))$ for all states, and hence $m = O(n \cdot \log(n))$. The classical algorithm and the algorithm of [1] (after reduction to binary game graphs) in the worst case take $O(n^2 \log(n))$ time, whereas the worst case running time of Algorithm 3 is bounded in $O(n^2 \log(\log(n)))$.

References

1. K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Simple stochastic parity games. In *CSL'03*, volume 2803 of *LNCS*, pages 100–113. Springer, 2003.
2. R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.
3. W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997.