Good-for-games ω-Pushdown Automata

Karoliina Lehtinen and Martin Zimmermann

University of Liverpool, Liverpool, United Kingdom
{Lehtinen,Martin.Zimmermann}@liverpool.ac.uk

Abstract. We introduce good-for-games ω-pushdown automata (ω-GFG-PDA). These are automata whose nondeterminism can be resolved based on the run constructed thus far. Good-for-gameness enables automata to be composed with games, trees, and other automata, applications which otherwise require deterministic automata.

Our main results are that ω-GFG-PDA are more expressive than deterministic ω-pushdown automata and that solving infinite games with winning conditions specified by ω-GFG-PDA is EXPTIME-complete. Thus, we have identified a new class of ω-contextfree winning conditions for which solving games is decidable. It follows that the universality problem for ω-GFG-PDA is in EXPTIME as well.

Moreover, we study closure properties of the class of languages recognized by ω-GFG-PDA and decidability of good-for-gameness of ω-pushdown automata and languages.

1 Introduction

Good-for-gameness is the new determinism, and not just for solving games. Good-for-games automata also lend themselves to composition with other automata and trees. These problems have in common that they are traditionally addressed with deterministic automata, which, depending on the exact type used, may be less succinct or even less expressive than nondeterministic ones. Good-for-games automata overcome this restriction by allowing a limited form of nondeterminism that does not interfere with composition.

As an example, consider the setting of infinite-duration two-player zero-sum games of perfect information. In such a game, two players interact to produce a play, an infinite word over some alphabet. A winning condition specifies a partition of the set of plays indicating the winner of each play. Here, we are concerned with games whose winning condition is explicitly given by an automaton recognizing the winning plays for one player.[1] This setting arises, for example, when solving the LTL synthesis problem where the winning condition is specified by an LTL formula, which can be turned into an automaton.

The usual approach to solving a game with a winning condition given by an automaton $\mathcal{A}$ is to obtain an equivalent deterministic automaton $\mathcal{D}$ and then solve an arena-based game with an implicit winning condition given by the acceptance condition of $\mathcal{D}$. The arena simultaneously captures the interaction between the players, which results in a play, and constructs the run of $\mathcal{D}$ on this play. The resulting arena-based game has the same winner as the original game with winning condition recognized by $\mathcal{A}$. For example, if the original winning condition is ω-regular, there is a deterministic parity automaton recognizing it, and the resulting game is a parity game, which can be effectively solved.

However, the correctness of this reduction crucially depends on the on-the-fly construction of the run of $\mathcal{D}$ on the play. For nondeterministic automata, one might be tempted to let the nondeterministic choices be resolved by the player who wins if the automaton accepts. However, an accepting run cannot necessarily be constructed on-the-fly, even if one exists, as the resolution of nondeterministic choices might depend on the whole play rather than on the current finite play prefix. A simple example is a winning condition that allows the player who resolves the nondeterminism to win the original game while her opponent wins in the arena-based game by using her nondeterministic choices against her. This is the case in the parity automaton presented in Figure 1, which accepts all words, but in which nondeterminism must decide whether a word has finitely or infinitely many occurrences of a.

Assume that the opponent has control over the number of a's that appear during a play. This number does not affect the winner of the original game, as the automaton is universal. But if the initial nondeterminism is resolved by moving to q_1, then the opponent wins by enforcing infinitely many a, while if the nondeterminism is resolved by moving to q_2 then he wins by enforcing finitely many a.

[1] This should be contrasted with the classical setting of say parity games, where the winning condition is implicitly encoded by a coloring of the arena specifying the interaction between the players.

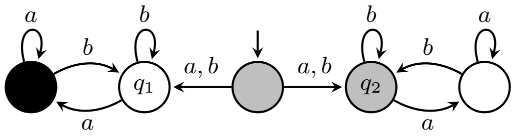

Fig. 1. A universal nondeterministic parity automaton. A run is accepting if it visits the black vertex only finitely often and a white vertex infinitely often.

Good-for-games automata (also known as history-deterministic automata [11]), introduced by Henzinger and Piterman [17], are nondeterministic (or even alternating [12, 4]) automata whose nondeterminism can be resolved based only on the input processed so far. This property implies that the previously described procedure yields the correct winner, even if the automaton is not deterministic.

Since their introduction, Boker, Kupferman, Kuperberg and Skrzypczak have shown that ω-regular good-for-games automata are also suitable for composition with trees [3], which can be seen as the special case of one-player arena-based games, while Boker and Lehtinen have shown that good-for-games automata are suitable for automata composition in the following sense [4]: If an ω-regular good-for-games automaton $\mathcal{B}$ recognizes the set of accepting runs of an alternating ω-regular automaton $\mathcal{A}$, then the composition of $\mathcal{A}$ and $\mathcal{B}$ is an ω-regular automaton that recognizes the same language as $\mathcal{A}$, but with the acceptance condition of $\mathcal{B}$. In other words, good-for-games automata, like deterministic automata, can be used both to simplify the winning conditions of games and the acceptance conditions of automata. Kuperberg and Skrzypczak showed that ω-regular good-for-games co-Büchi automata can be exponentially more succinct than deterministic ones while good-for-games Büchi automata are at most quadratically more succinct [19].

However, since deterministic parity automata, which are trivially good-for-games, express all ω-regular languages, succinctness is the most good-for-gameness can offer in the ω-regular setting. In contrast, deterministic automata models are in general less expressive, not just less succinct, than their nondeterministic counterparts. This is true, for example, for pushdown automata and for various types of quantitative automata. We argue that in such cases, it is worthwhile to investigate good-for-games automata as an alternative to deterministic ones, as they form a potentially larger class of winning conditions for which solving games is decidable.

Indeed, the study of quantitative automata lead to the independent introduction of good-for-games automata by Colcombet. In particular, in the setting of regular cost functions, good-for-games cost automata are as expressive as nondeterministic ones, unlike deterministic ones [11].

So far the case of ω-pushdown automata has not been considered. Here, the increased expressiveness of nondeterministic automata comes at a heavy price: games with winning conditions given by nondeterministic ω-pushdown automata are undecidable [14] while those with winning conditions given by deterministic ones are decidable [24]. Hence, in this work, we introduce and study good-for-games ω-pushdown automata (ω-GFG-PDA) to push the frontier of decidability for games with ω-contextfree winning conditions.

Our contributions Our first results concern expressiveness: We prove that ω-GFG-PDA are strictly more expressive than deterministic ω-pushdown automata (ω-DPDA), but not as expressive as nondeterministic ω-pushdown automata (ω-PDA). So, they do form a new class of ω-contextfree languages. This is in contrast to the ω-regular setting where, for each of the usual acceptance conditions, deterministic and GFG automata recognize exactly the same languages.

Second, we show that ω-GFG-PDA live up to their name: Determining the winner of a game with a winning condition specified by an ω-GFG-PDA is ExpTime-complete, as is for the special case of games with winning conditions specified by ω-DPDA [24]. This has to be contrasted with the undecidability of games with a winning condition specified by an ω-PDA [14]. As a corollary, the universality of ω-GFG-PDA is also in ExpTime, while it is undecidable for ω-PDA.

Third, we compare ω-GFG-PDA with visibly pushdown automata (ω-VPA) [1], a class of ω-PDA with robust closure properties and for which solving games is also decidable [22]. We show that the classes of languages recognized by ω-GFG-PDA and ω-VPA are incomparable with respect to inclusion. See Figure 2 for an overview of our results on the relations between these classes.

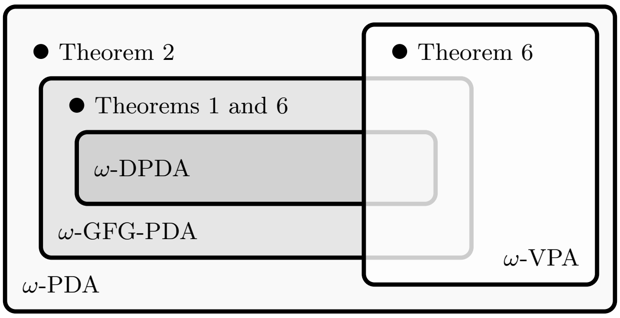

Fig. 2. The classes of ω-languages recognized by the automata considered in this work. Languages separating these classes are shown as black dots.

Fourth, we study the closure properties of ω-GFG-PDA, which are almost nonexistent, and prove that both the problems of deciding whether a given ω-PDA is good-for games, and of deciding whether a given ω-PDA is language equivalent to an ω-GFG-PDA are undecidable. Table 1 sums up our results on closure properties and decidability.

	Closure properties					Decision problems			
	Intersection	Union	Complement	Set difference	Homomorphism	Emptiness	Universality	Solving Games	
ω-DPDA	✗	✗	✓	✗	✗	PTime	PTime	ExpTime	[10, 24]
ω-GFG-PDA	✗	✗	✗	✗	✗	PTime	ExpTime*	ExpTime	here
ω-VPA	✓	✓	✓	✓	✗	PTime	ExpTime	2ExpTime	[1, 22]
ω-PDA	✗	✓	✗	✗	✓	PTime	undecidable	undecidable	[8, 9, 14]

Table 1. Summary of our results (in light gray) and comparison to other classes of contextfree languages. All problems are complete for the respective complexity class unless marked with an asterisks.

2 Preliminaries

An alphabet Σ is a finite nonempty set of letters. The set of finite words over Σ is denoted by Σ^*, the set of nonempty finite words over Σ by Σ^+, and the set of infinite words over Σ by Σ^ω. The empty word is denoted by ε, the length of a finite word v is denoted by $|v|$, and the n-th letter of a finite or infinite word is denoted by $w(n)$ (starting with $n = 0$). An ω-language over Σ is a subset of Σ^ω.

For alphabets Σ_1, Σ_2, we extend functions $f\colon \Sigma_1 \to \Sigma_2^*$ homomorphically to finite and infinite words over Σ_1 via

$$f(w) = f(w(0))f(w(1))f(w(2)) \cdots .$$

For example, if $\Sigma_1 = \Sigma \times \Sigma'$, then $\pi_i(a_1, a_2) \mapsto a_i$ for $(a_1, a_2) \in \Sigma_1$ denotes the projection to the i-th component ($i \in \{1, 2\}$).

An ω-pushdown automaton (ω-PDA for short)

$$\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$$

consists of a finite set Q of states with the initial state $q_I \in Q$, an input alphabet Σ, a stack alphabet Γ, a transition relation Δ to be specified, and a coloring $\Omega\colon \Delta \to \mathbb{N}$. For notational convenience, we define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\perp = \Gamma \cup \{\perp\}$, where $\perp \notin \Gamma$ is a designated stack bottom symbol. Then, the transition relation Δ is a subset of $Q \times \Gamma_\perp \times \Sigma_\varepsilon \times Q \times \Gamma_\perp^{\leq 2}$ that we require to neither write nor delete the stack bottom symbol from the stack: If $(q, \perp, a, q', \gamma) \in \Delta$, then $\gamma \in \perp \cdot (\Gamma \cup \{\varepsilon\})$, and if $(q, X, a, q', \gamma) \in \Delta$

for $X \in \Gamma$, then $\gamma \in \Gamma^{\leq 2}$. Given a transition $\tau = (q, X, a, q', \gamma)$ let $\ell(\tau) = a \in \Sigma_\varepsilon$. We say that τ is an $\ell(\tau)$-transition and that τ is a Σ-transition, if $\ell(\tau) \in \Sigma$. For a finite or infinite sequence ρ over Δ, $\ell(\rho)$ is defined by applying ℓ homomorphically to every transition.

A stack content is a finite word in $\bot\Gamma^*$ (i.e., the top of the stack is at the end) and a configuration $c = (q, \gamma)$ of $\mathcal{P}$ consists of a state $q \in Q$ and a stack content γ. The stack height of c is $\mathrm{sh}(c) = |\gamma| - 1$. The initial configuration is $(q_I, \bot)$.

A transition $\tau = (q, X, a, q', \gamma') \in \Delta$ is enabled in a configuration c if $c = (q, \gamma X)$ for some $\gamma \in \Gamma^*$. In this case, we write $(q, \gamma X) \xrightarrow{\tau} (q', \gamma\gamma')$. A run of $\mathcal{P}$ is an infinite sequence $\rho = c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ of configurations and transitions with c_0 being the initial configuration and $c_n \xrightarrow{\tau_n} c_{n+1}$ for every n. Finite run prefixes are defined analogously and are required to end in a configuration. The infinite run ρ is a run of $\mathcal{P}$ on $w \in \Sigma^\omega$, if $w = \ell(\rho)$ (this implies that ρ contains infinitely many Σ-transitions). We say that ρ is accepting if $\limsup_{n\to\infty} \Omega(\tau_n)$ is even, i.e., if the maximal color labeling infinitely many transitions is even. The language $L(\mathcal{P})$ recognized by $\mathcal{P}$ contains all $w \in \Sigma^\omega$ such that $\mathcal{P}$ has an accepting run on w.

Remark 1. Let $c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ be a run of $\mathcal{P}$. Then, the sequence $c_0c_1c_2\cdots$ of configurations is uniquely determined by the sequence $\tau_0\tau_1\tau_2\cdots$ of transitions. Hence, whenever convenient, we treat a sequence of transitions as a run if it indeed induces one (not every such sequence does induce a run, e.g., if a transition τ_n is not enabled in c_n).

We say that an ω-PDA $\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $X \in \Gamma$, and every $a \in \Sigma_\varepsilon$, there is at most one transition of the form $(q, X, a, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $X \in \Gamma$, if there is a transition $(q, X, \varepsilon, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is a transition $(q, X, a, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

As expected, a deterministic ω-pushdown automaton (ω-DPDA) has at most one run on every ω-word.

The class of ω-languages recognized by ω-PDA is denoted by ω-CFL and the class of ω-languages recognized by ω-DPDA by ω-DCFL. Cohen and Gold showed that ω-DCFL is a strict subset of ω-CFL [10, Theorem 3.2].[2]

Example 1. The ω-PDA $\mathcal{P}$ depicted in Figure 3 recognizes the ω-language

$$\{ac^nd^n\#^\omega \mid n \geq 1\} \cup \{bc^nd^{2n}\#^\omega \mid n \geq 1\}.$$

Note that while $\mathcal{P}$ is nondeterministic, $L(\mathcal{P})$ is in ω-DCFL.

[2] Formally, Cohen and Gold considered automata with state-based Muller acceptance while we consider, for technical convenience, automata with transition-based parity acceptance. However, using *latest appearance records* (see, e.g., [16]) shows that both definitions are equivalent.

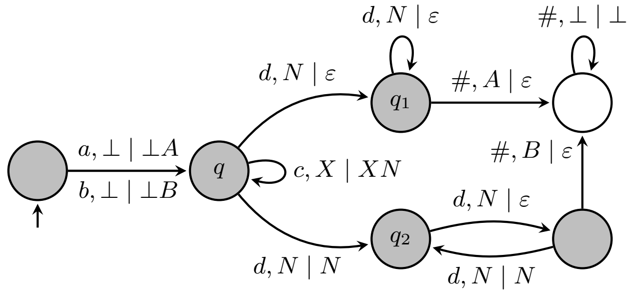

Fig. 3. The ω-PDA $\mathcal{P}$ from Example 1. The self-loop at the white state has color 2 while all other transitions have color 1, and X represents an arbitrary stack symbol.

3 Good-for-games Pushdown Automata

Here, we introduce good-for-games ω-pushdown automata (ω-GFG-PDA for short), nondeterministic ω-pushdown automata whose nondeterminism can be resolved based on the run prefix constructed thus far and on the next input letter to be processed, but independently of the continuation of the input beyond the next letter.

As an example, consider the ω-PDA $\mathcal{P}$ from Example 1. It is nondeterministic, but knowing whether the first transition of the run processed an a or a b allows to resolve the nondeterminism in a configuration of the form $(q, \gamma N)$: in the former case, take the transition to state q_1, in the latter case the transition to state q_2. Afterwards, there are no nondeterministic choices to make and the resulting run is accepting whenever the input is in the language. This automaton is therefore good-for-games.

Formally, we say that an ω-PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ is good-for-games, if there is a (nondeterminism) resolver for $\mathcal{P}$, a function $r \colon \Delta^* \times \Sigma \to \Delta$ such that for every $w \in L(\mathcal{P})$ the sequence $\tau_0\tau_1\tau_2 \cdots \in \Delta^\omega$ defined by

$$\tau_n = r(\tau_0 \cdots \tau_{n-1}, w(|\ell(\tau_0 \cdots \tau_{n-1})|))$$

induces an accepting run. Note that the prefix processed so far can be recovered from r's input, i.e., it is $\ell(\tau_0 \cdots \tau_{n-1})$. However, the converse is not true due to the existence of ε-transitions. This is the reason the run prefix and not the input prefix is the input for the resolver.

Each deterministic automaton is trivially good-for-games. We denote the class of ω-languages recognized by ω-GFG-PDA by ω-GFG-CFL. By definition, the three classes of languages we consider form a hierarchy.

Proposition 1. *ω-DCFL $\subseteq$ ω-GFG-CFL $\subseteq$ ω-CFL.*

We show that both inclusions are strict. In particular, ω-GFG-PDA are more expressive than ω-DPDA.

Let $I = \{\mathtt{0}, \mathtt{+}, \mathtt{-}\}$ and define the energy level $EL(v) \in \mathbb{Z}$ of finite words v over I inductively as $EL(\varepsilon) = 0$, and $EL(v\mathtt{0}) = EL(v)$, $EL(v\mathtt{+}) = EL(v) + 1$, as well as $EL(v\mathtt{-}) = EL(v) - 1$. We say that a word $w \in I^\omega$ is *safe* if $EL(w(0) \cdots w(n)) \geq 0$ for every $n \geq 0$.

Remark 2. Let $w \in I^\omega$.

1. w has a safe suffix if and only if there is an $s \in \mathbb{N}$ such that $EL(w(0) \cdots w(n)) \geq -s$ for all n.
2. If w is safe then there is an $n > 0$ such that $w(n)w(n+1)w(n+2) \cdots$ is safe as well.

We fix $\Sigma = I \times I$ and define L_{ss} to be the language containing all $w \in \Sigma^\omega$ such that $\pi_i(w)$ has a safe suffix, for some $i \in \{1, 2\}$.

Lemma 1. *$L_{ss} \in$ ω-GFG-CFL.*

Proof. Consider the ω-PDA $\mathcal{P}$ with two states 1 and 2 signifying whether a potential safe suffix is being tracked in the first or second component of the input, and a single stack symbol X used to track the energy level of such a suffix. The initial state is arbitrary; we fix it to be 1.

The automaton can at any moment nondeterministically change its state from 1 to 2 and vice versa without changing the stack content (while processing an input letter that is just ignored). When not changing its state, say while staying in state i, $\mathcal{P}$ deterministically processes the next input letter $\binom{a_1}{a_2}$. If $a_i = \mathtt{0}$ then the stack is left unchanged and if $a_i = \mathtt{+}$ then an X is pushed onto the stack. If $a_i = \mathtt{-}$ and the stack is nonempty, then the topmost X is popped from the stack. The stack is left unchanged if $a_i = \mathtt{-}$ and the stack is empty. Note that the only nondeterministic choice is to change the state, i.e., if the state is not changed then the automaton has exactly one transition that can process the next input letter. Furthermore, $\mathcal{P}$ has no ε-transitions, i.e., each transition processes an input letter.

A transition has color 1 if it either changes the state, or if it processes a $\mathtt{-}$ in the i-th component while in state i with an empty stack. All other transitions have color 0. Hence, a run is accepting if and only if it uses transitions of the former two types only finitely often. First, we show that $L_{\mathrm{ss}} = L(\mathcal{P})$, then that $\mathcal{P}$ is good-for-games.

Let $w \in L_{\mathrm{ss}}$, i.e., there is an $i \in \{1, 2\}$ such that $\pi_i(w) = a(0)a(1)a(2) \cdots$ has a safe suffix, say $a(n)a(n+1)a(n+2) \cdots$. Due to Remark 2.2, we assume w.l.o.g. $n > 0$. Consider the unique run $\rho =$

$c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ of $\mathcal{P}$ on w that immediately switches to state i (if necessary) and otherwise always executes the unique enabled transition that processes the next input letter without changing state. We have $\ell(\tau_0\cdots\tau_{j-1}) = w(0)\cdots w(j-1)$ due to the absence of ε-transitions.

An induction shows the invariant $\mathrm{sh}(c_{n+j}) = \mathrm{sh}(c_n) + EL(a(n)\cdots a(n+j-1))$ for every $j \geq 0$. Here we use the assumption $n > 0$, which ensures that no transitions changing the state are used to process the safe suffix. Hence, the state is equal to i at all but possibly the first configuration and the invariant implies that no $\texttt{-}$ is processed in component i by a transition τ_{n+j} while the stack is empty. This implies that the run is accepting, i.e., $w \in L(\mathcal{P})$.

Conversely, let $w \in L(\mathcal{P})$. Then, there is an accepting run $c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ of $\mathcal{P}$ on w. Again, as $\mathcal{P}$ has no ε-transitions, $\ell(\tau_0\cdots\tau_{j-1}) = w(0)\cdots w(j-1)$. Now, let τ_{n-1} be the last transition with color 1 (pick $\tau_{n-1} = \tau_0$ if there is no such transition) and let $c_n = (i, \bot X^s)$. We define $a(n)a(n+1)a(n+2)\cdots$ to be the suffix of $\pi_i(w) = a(0)a(1)a(2)\cdots$ starting at position n.

By the choice of n, after τ_{n-1} no transition changes the state (it is always equal to i) and there is no $j \geq 0$ such that $a(n+j) = \texttt{-}$ and $\mathrm{sh}(c_{n+j}) = 0$. Thus, an induction shows that $EL(a(n)\cdots a(n+j)) = \mathrm{sh}(c_{n+j}) - s$. Hence, the energy level of the prefixes of $a(n)a(n+1)a(n+2)\cdots$ is bounded from below by $-s$. Thus, Remark 2.1 implies that $a(n)a(n+1)a(n+2)\cdots$ has a safe suffix, i.e., $w \in L_{\mathrm{ss}}$.

It remains to show that $\mathcal{P}$ is good-for-games.[3] Intuitively, we construct a resolver r that searches for a safe suffix in the component that has the longest suffix that can still be extended to an infinite safe word (preferring the first component in case of a tie). Fix $v = \binom{a_1(0)}{a_2(0)}\cdots\binom{a_1(n)}{a_2(n)} \in \Sigma^+$ and let S_i contain those $j \leq n$ such that $a_i(j)\cdots a_i(n)$ is safe (which is defined as expected). Further, let

$$i(v) = \begin{cases} 1 & \text{if } \min S_1 \leq \min S_2, \\ 2 & \text{otherwise,} \end{cases}$$

where we use $\min\emptyset = n+1$. We define $r(\tau_0\cdots\tau_{n-1}, a)$ inductively to always ensure that its target state is equal to $i(\ell(\tau_0\cdots\tau_{n-1})a)$ and, if this does not require a state change, then the unique transition processing the $i(\ell(\tau_0\cdots\tau_{n-1})a)$-th component of a is returned.

Now, let $w \in L_{\mathrm{ss}}$, i.e., there is an $i \in \{1,2\}$ such that $\pi_i(w)$ has a safe suffix. Then, r produces a run that tracks the safe suffix that starts as early as possible (again favoring the first component in case of a tie). As in the argument above one can show that this run is accepting, as it switches states only finitely often and does not process a $\texttt{-}$ while the stack is empty while tracking the safe suffix. Hence, r has the desired properties and $\mathcal{P}$ is good-for-games.

After having shown that L_{ss} is in ω-GFG-CFL, we show that it is not in ω-DCFL, thereby separating ω-DCFL and ω-GFG-CFL.

Lemma 2. *$L_{ss} \notin \omega$-DCFL.*

Proof. We assume towards a contradiction that there is an ω-DPDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ recognizing L_{ss}. Define $x_1 = \binom{\texttt{+}}{\texttt{0}}\binom{\texttt{+}}{\texttt{-}}$ and $x_2 = \binom{\texttt{0}}{\texttt{+}}\binom{\texttt{-}}{\texttt{+}}$, i.e., in x_i the energy level in component i is increased by two while it is decreased by one in the other component. Every infix of length at least 3 of a word built by concatenating copies of the x_i has a strictly positive energy level in one component (note that the infix may start or end within an x_i).

Define
$$w = x_1\,(x_2)^3\,(x_1)^7\,(x_2)^{15}\,(x_1)^{31}\,(x_2)^{63}\ \cdots.$$
An induction shows $EL(\pi_1(x_1(x_2)^3\cdots(x_2)^{2^{2j}-1})) = -j$ for every $j > 1$ and $EL(\pi_2(x_1(x_2)^3\cdots(x_1)^{2^{2j-1}-1})) = -j$ for every $j > 0$. Hence, due to Remark 2.1, $w \notin L_{\mathrm{ss}}$.

Now, let $\rho = c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ be the unique run of $\mathcal{P}$ on w, which is rejecting. This run exists, as each prefix of w can be extended to a word that is in the language of $\mathcal{P}$. Indeed, if there is no run on w, because either there is no enabled transition that processes the next input letter or because it ends in an infinite tail of ε-transitions, then some word in $L(\mathcal{P})$ has no accepting run of w, which is a contradiction.

A step of ρ is a position n such that $\mathrm{sh}(c_n) \leq \mathrm{sh}(c_{n+j})$ for all $j \geq 0$. Every infinite run has infinitely many steps. Hence, we can find two steps $s < s'$ satisfying the following properties:

[3] Although the language we use here is different, this argument is similar to the one used by Kuperberg and Skrzypczak [19] to show that good-for-games co-Büchi automata are exponentially more succinct than deterministic ones.

1. There is a state $q \in Q$ and a stack symbol $X \in \Gamma_\bot$ such that c_s and $c_{s'}$ have the form $(q, \gamma X)$ for some γ, i.e., both configurations have the same state and topmost stack symbol.
2. The maximal color labeling the sequence $\tau_s \cdots \tau_{s'-1}$ of transitions leading from c_s to $c_{s'}$ is odd.
3. This sequence $\tau_s \cdots \tau_{s'-1}$ processes an infix v of w with $EL(\pi_i(v)) > 0$, for some $i \in \{1, 2\}$.

Consider the sequence $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})^\omega$ of transitions. Due to the first property, it induces a run ρ' of $\mathcal{P}$, which is rejecting due to the second property. Finally, due to the third property, ρ' processes a word with suffix v^ω. Such a word has a safe suffix in component i, as $EL(\pi_i(v)) > 0$. Hence, we have constructed a word in L_{ss} such that the unique run of $\mathcal{P}$ on w is rejecting, obtaining the desired contradiction to $L(\mathcal{P}) = L_{\mathrm{ss}}$.

Our main result of this section is now a direct consequence of the previous two lemmata: ω-GFG-PDA are more expressive than ω-DPDA.

Theorem 1. *ω-DCFL $\subsetneq$ ω-GFG-CFL.*

The next obvious question is whether every (nondeterministic) ω-contextfree language is good-for-games. Not unexpectedly, this is not the case. The intuitive reason is that good-for-games automata allow to resolve nondeterminism based on the history of a run, but still cannot resolve nondeterminism based on the continuation of the input. Considering a language that requires nondeterministic choices about the continuation of the input yields the desired separation. To this end, we adapt the classical proof that

$$\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$$

is not recognizable by a DPDA over finite words to our setting.

Theorem 2. *ω-GFG-CFL $\subsetneq$ ω-CFL.*

Proof. Define

$$L = \{(a\#)^n (b\#)^n \#^\omega \mid n \geq 1\} \cup \{(a\#)^n (b\#)^{2n} \#^\omega \mid n \geq 1\},$$

which is in ω-CFL, as an ω-PDA can process a prefix $(a\#)^n$ by storing n in unary on the stack and then nondeterministically guess and verify whether the remaining suffix is $(b\#)^n \#^\omega$ (by popping a symbol from the stack for every b) or whether it is $(b\#)^{2n} \#^\omega$ (by popping a symbol from the stack for every other b), similarly to the automaton from Example 1

We claim that L is not in ω-GFG-CFL. We assume towards a contradiction that there is an ω-GFG-PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ with $L(\mathcal{P}) = L$, say with resolver r. In what follows, we will reach a contradiction by constructing an ω-PDA that recognizes the language

$$L_{abc} = \{(a\#)^n (b\#)^n (c\#)^n \#^\omega \mid n \geq 1\},$$

which is not in ω-CFL. This follows from $\{a^n b^n c^n \mid n \geq 1\}$ not being contextfree and from the closure properties of ω-CFL [9] and of the contextfree languages [18].

First, we note that the language

$$C = \{\gamma q \in \bot\Gamma^* Q \mid \mathcal{P} \text{ accepts } \#^\omega \text{ when starting in } (q, \gamma)\}$$

is regular. This can be shown by first noting that

$$\begin{aligned} C_0 = \{\gamma X q \in \bot\Gamma^* Q \mid \mathcal{P} &\text{ accepts } \#^\omega \text{ when starting in} \\ &(q, \gamma X) \text{ with a run that only visits configurations} \\ &\text{of stack height greater or equal to } \mathrm{sh}(q, \gamma X)\} \end{aligned}$$

is a finite union of languages $\Gamma^* X q \cap \bot\Gamma^* q$ for some $X \in \Gamma_\bot$ and some $q \in Q$, and therefore regular. Now, C is equal to

$$\begin{aligned} \{\gamma q \in \bot\Gamma^* Q \mid &\text{ there is a run infix } \rho \text{ with } \ell(\rho) \in \#^* \\ &\text{leading from } (q, \gamma) \text{ to } C_0\}. \end{aligned}$$

An application of standard saturation techniques [6][4] (applied to the restriction of $\mathcal{P}$ to transitions labeled by $\#$ or ε) shows that the latter set is regular, as the target set C_0 is regular.

Using a deterministic finite automaton $\mathcal{A} = (Q', \Gamma_\perp \cup Q, q'_I, \delta, F)$ recognizing C we construct an ω-PDA $\mathcal{P}'$ as follows: We extend the stack alphabet Γ of $\mathcal{P}$ to $\Gamma \times Q'$ and define the transition relation of $\mathcal{P}'$ so that it simulates a run of $\mathcal{P}$ and keeps track of the state of $\mathcal{A}$ reached by processing the stack content, i.e., if $\mathcal{P}'$ reaches a stack content

$$\perp(X_1, q_1) \cdots (X_s, q_s)$$

then we have $q_j = \delta^*(q'_I, \perp X_0 \cdots X_j)$ for every $0 \leq j \leq s$.[5]

Additionally, the states of $\mathcal{P}'$ have a Boolean flag that is changed if $\mathcal{P}'$ reaches a configuration of the form (q, γ) with $\gamma q \in C$ for the first time. As the stack symbols of $\mathcal{P}'$ encode the run of $\mathcal{A}$ on the stack content, this can be checked easily. From there on, $\mathcal{P}'$ continues to simulate a run of $\mathcal{P}$, but now every transition labeled by a b in $\mathcal{P}$ is labeled by a c.

Finally, we define the acceptance condition of $\mathcal{P}'$ such that it only accepts if it switches the flag, afterwards continues the simulation of a run of $\mathcal{P}$ that is accepting, and processes at least one c after the switch. Note that this requires adding a second Boolean flag to the states to check whether a c has been processed. We claim that $\mathcal{P}'$ recognizes the language L_{abc}.

To this end, let

$$w = (a\#)^n(b\#)^n(c\#)^n\#^\omega \in L_{abc}$$

and define

$$w_1 = (a\#)^n(b\#)^n\#^\omega \qquad \text{and} \qquad w_2 = (a\#)^n(b\#)^{2n}\#^\omega,$$

which are both in L. Thus, let ρ_1 and ρ_2 be the accepting runs of $\mathcal{P}$ on w_1 and w_2 induced by the resolver r. A prefix ρ'_1 of ρ_1 processing $w'_1 = (a\#)^n(b\#)^{n-1}b$ is also a prefix of ρ_2 processing w'_1 (note that we have removed the last $\#$, as the resolver inducing the runs has access to the next letter to be processed). As ρ_1 is an accepting run of $\mathcal{P}$ on $w_1 = w'_1\#^\omega$, the last configuration of ρ'_1 is in C.

Hence, $\mathcal{P}'$ can simulate the run prefix ρ'_1 processing w'_1, switch the first flag and then continue to simulate the suffix of ρ'_2 obtained by removing ρ'_1, which processes $\#(c\#)^n\#^\omega$. This run is accepting, i.e., we have $w = w'_1\#(c\#)^n\#^\omega \in L(\mathcal{P}')$.

Now, let $w \in L(\mathcal{P}')$, i.e., there is an accepting run ρ' of $\mathcal{P}'$ on w. By construction of $\mathcal{P}'$, we can split ρ' into a finite prefix ρ'_p before the first flag is switched and the corresponding infinite suffix ρ'_s starting with the switch. Again, by construction, ρ'_p is the simulation of a run ρ_p of $\mathcal{P}$ that processes the same input and ends in a configuration in C, and no prefix of ρ_p ends in C. Hence, we can conclude that both ρ_p and ρ'_p process $(a\#)^n(b\#)^{n-1}b$ for some $n > 0$, as these are the minimal words leading to a configuration from which $\#^\omega$ can be accepted.

Now, consider the suffix ρ'_s, which processes at least one c. It also simulates a run suffix ρ_s of $\mathcal{P}$ and $\ell(\rho_s)$ is obtained from $\ell(\rho'_s)$ by replacing each c by a b. Furthermore, ρ_s starts in the last configuration of ρ_p and satisfies the acceptance condition, as ρ'_s satisfies the acceptance condition. Hence, ρ_s processes $\#(b\#)^n\#^\omega$, as the concatenation of ρ_p and ρ_s is an accepting run. Altogether, ρ'_p processes $(a\#)^n(b\#)^{n-1}b$ and ρ'_s processes $\#(c\#)^n\#^\omega$, i.e., $w = (a\#)^n(b\#)^n(c\#)^n\#^\omega$, which is in L_{abc}.

We conclude that $L(\mathcal{P}') = L_{abc}$, which contradicts $L_{abc} \notin \omega$-CFL. Therefore, L separates ω-GFG-CFL and ω-CFL.

The proof of Theorem 6 is based on another language separating ω-GFG-CFL and ω-CFL. A third such language is the following, based on palindromes.

Let $h\colon \{0,1,\#\}^* \to \{0,1\}^*$ be the homomorphism induced by $h(0) = 0$, $h(1) = 1$, and $h(\#) = \varepsilon$. Define

$$P = \{v\#^\omega \mid h(v) = xx^R \text{ for some } x \in \{0,1\}^*\},$$

where x^R denotes the reversal of x. The proof that P indeed separates ω-CFL and ω-GFG-CFL is left as an exercise for the enthusiastic reader and can be found in the appendix.

[4] Also, see the survey of Carayol and Hague [7] for more details.

[5] Note that the simplest way to implement this is to replace each transition that swaps the topmost stack symbol from X to X' by two transitions, the first popping X from the stack, and the second pushing X' onto the stack (using a fresh state reached between the new transitions).

4 Good-for-games Pushdown Automata are Indeed Good for Games

In this section, we show that the winner of infinite-duration games with ω-GFG-CFL winning conditions can be effectively determined. This result is best phrased in terms of Gale-Stewart games, abstract games without an arena [15], as we are interested in the influence of the winning condition on the decidability of solving games.[6]

Formally, a Gale-Stewart game $\mathcal{G}(L)$ is given by an ω-language $L \subseteq (\Sigma_1 \times \Sigma_2)^\omega$. It is played between Player 1 and Player 2 in rounds $n = 0, 1, 2, \ldots$: In each round, first Player 1 picks a letter $a_1(n) \in \Sigma_1$, then Player 2 picks a letter $a_2(n) \in \Sigma_2$. After ω rounds, the players have constructed an outcome

$$w = \binom{a_1(0)}{a_2(0)}\binom{a_1(1)}{a_2(1)}\binom{a_1(2)}{a_2(2)}\cdots$$

which is winning for Player 2 if it is in L. A strategy for Player 2 in $\mathcal{G}(L)$ is a mapping $\sigma\colon \Sigma_1^* \to \Sigma_2$. The outcome w is consistent with σ, if $a_2(n) = \sigma(a_1(0)\cdots a_1(n))$ for all n. A strategy σ for Player 2 is winning if every outcome that is consistent with σ is in L. Player 2 wins $\mathcal{G}(L)$ if she has a winning strategy for $\mathcal{G}(L)$.

Proposition 2 ([14, 24]).

1. *The following problem is undecidable: Given an ω-PDA $\mathcal{P}$, does Player 2 win $\mathcal{G}(L(\mathcal{P}))$?*
2. *The following problem is* ExpTime*-complete: Given an ω-DPDA $\mathcal{P}$, does Player 2 win $\mathcal{G}(L(\mathcal{P}))$?*

Walukiewicz's decidability result [24] is formulated for parity games on configuration graphs of pushdown automata. However, a Gale-Stewart game with ω-DCFL winning condition can be reduced in polynomial time to a parity game on a configuration graph of a pushdown machine. This construction crucially depends on the determinism of the automaton recognizing the winning condition, as witnessed by the undecidability result for winning conditions recognized by (possibly nondeterministic) ω-PDA.

Our main result shows that decidability extends to games given by ω-GFG-PDA, i.e., not all types of nondeterminism lead to undecidability.

Theorem 3. *The following problem is* ExpTime*-complete: Given an ω-GFG-PDA $\mathcal{P}$, does Player 2 win $\mathcal{G}(L(\mathcal{P}))$?*

Proof. Given $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ with $\Sigma = \Sigma_1 \times \Sigma_2$ we construct an ω-DPDA $\mathcal{P}'$ such that Player 2 wins $\mathcal{G}(L(\mathcal{P}))$ if and only if she wins $\mathcal{G}(L(\mathcal{P}'))$. This yields ExpTime membership, as determining the winner of Gale-Stewart games with ω-DCFL winning conditions is in ExpTime (see Proposition 2.2) and the size of $\mathcal{P}'$ is polynomial in the size of $\mathcal{P}$. The matching lower bound is immediate, as determining the winner of games with ω-DCFL winning conditions is already ExpTime-hard (again, see Proposition 2.2).

Intuitively, we construct an ω-DPDA $\mathcal{P}'$ that processes simultaneously both an input of $\mathcal{P}$ and a run of $\mathcal{P}$, and checks whether the run is indeed an accepting run of $\mathcal{P}$ on the input. In the corresponding Gale-Stewart game with winning condition $L(\mathcal{P}')$, Player 2 has to both choose a letter in Σ_2 and transitions of $\mathcal{P}$. In other words, we have moved the nondeterminism of $\mathcal{P}$ into Player 2's moves.

Formally, we construct $\mathcal{P}'$ such that it recognizes all ω-words $m_0 m_1 m_2 \cdots$ over $\Sigma_1 \times (\Sigma_2 \cup \Delta)$ where each *block* m_j is of the form

$$\binom{a_1(j)}{a_2(j)}\binom{b_{j,0}}{\tau_{j,0}}\cdots\binom{b_{j,n_j-1}}{\tau_{j,n_j-1}}\binom{b_{j,n_j}}{\tau_{j,n_j}}$$

for some $n_j \geq 0$ satisfying the following conditions:

1. The transitions $\tau_{j,0}, \ldots, \tau_{j,n_j-1}$ are ε-transitions, and τ_{j,n_j} is an $\binom{a_1(j)}{a_2(j)}$-transition.
2. The sequence $\tau_{0,0}\cdots\tau_{0,n_0}\tau_{1,0}\cdots\tau_{1,n_1}\tau_{2,0}\cdots\tau_{2,n_2}\cdots$ of transitions induces an accepting run of $\mathcal{P}$ on the ω-word $\binom{a_1(0)}{a_2(0)}\binom{a_1(1)}{a_2(1)}\binom{a_1(2)}{a_2(2)}\cdots$. Note that all the $b_{j,j'}$ picked by Player 1 are ignored while Player 2 constructs the run, only the letters $a_1(j)$ are relevant.

[6] Games in finite arenas can easily be encoded as Gale-Stewart games.

If w is of that form then the decomposition into blocks is unique.

An ω-DPDA $\mathcal{P}'$ recognizing this language can easily be constructed in polynomial time from $\mathcal{P}$. To this end, $\mathcal{P}'$ deterministically simulates the transitions given in a block on the letter from $\Sigma_1 \times \Sigma_2$ at the beginning of the block. If a transition is not applicable, then the run terminates and is therefore rejecting. Some standard constructions are necessary to ensure that the input has the right format; in particular, we need to adapt the coloring to rule out that from some point onwards only ε-transitions appear in the input. It remains to show that Player 2 wins $\mathcal{G}(L(\mathcal{P}))$ if and only if she wins $\mathcal{G}(L(\mathcal{P}'))$.

First, let Player 2 win $\mathcal{G}(L(\mathcal{P}))$, say with a winning strategy σ. Furthermore, as $\mathcal{P}$ is good-for-games, there is a resolver $r\colon \Delta^* \times \Sigma \to \Delta$. We define a winning strategy σ' for Player 2 in $\mathcal{G}(L(\mathcal{P}'))$.

Intuitively, the strategy σ' alternates between simulating a move of σ and then uses r to construct a sequence of transitions that processes the letter determined by the move. The sequence starts with a finite number of ε-transitions followed by one transition processing the letter.

More formally, define σ' inductively starting with $\sigma'(a) = \sigma(a)$ for $a \in \Sigma_1$. Now, let $v = a_1(0) \cdots a_1(n) \in \Sigma_1^*$ with $n > 0$ be an input such that

$$a_2(j) = \sigma'(a_1(0) \cdots a_1(j))$$

is already defined for every $j < n$. To define $\sigma'(v)$ we consider two cases.

If $a_2(n-1) \in \Sigma_2 \cup \Delta$ is a non-ε-transition, then we define $\sigma'(v) = \sigma(v')$, where v' is obtained from v by removing the letters at positions j with $a_2(j) \in \Delta$. This simulates the next move of σ, as the transition $a_2(n-1)$ has processed the last letter. In the other case (i.e., if $a_2(n-1)$ is either a letter in Σ_2 or an ε-transition) define $\sigma'(v) = r(\rho, \binom{a_1(j')}{a_2(j')})$ where $\rho \in \Delta^*$ is obtained from $a_2(0) \cdots a_2(n-1)$ by removing the letters at positions j with $a_2(j) \in \Sigma_2$ and where $j' < n$ is maximal with $a_2(j') \in \Sigma_2$. This move continues the construction of a run infix that processes the last letter from $\Sigma_1 \times \Sigma_2$, which appears at position j'.

Now, let $w' \in (\Sigma_1 \times (\Sigma_2 \cup \{\Delta\}))^\omega$ be consistent with σ'. An induction shows that w' is a sequence of blocks that encodes an outcome w over $\Sigma_1 \times \Sigma_2$ that is consistent with σ, and a run ρ of $\mathcal{P}$ on w induced by r. As σ is a winning strategy, w is in $L(\mathcal{P})$, which implies that ρ is accepting. Hence, w' is in $L(\mathcal{P}')$, i.e., σ' is indeed winning for Player 2 in $\mathcal{G}(L(\mathcal{P}'))$.

Conversely, let Player 2 win $\mathcal{G}(L(\mathcal{P}'))$, say with winning strategy σ'. We define a winning strategy σ for Player 2 in $\mathcal{G}(L(\mathcal{P}))$. Intuitively, to define σ, we simulate a play in $\mathcal{G}(L(\mathcal{P}))$ by a play in $\mathcal{G}(L(\mathcal{P}'))$, and copy the choice of letters made by σ' while ignoring the moves building the run of $\mathcal{P}$.

Fix some $b \in \Sigma_1$. We inductively define for every input $v \in \Sigma_1^+$ for σ an input $v' \in \Sigma_1^+$ for σ' and then define $\sigma(v) = \sigma'(v')$. We begin by defining $a' = a$ for every $a \in \Sigma_1$. Now, assume we have defined v' for some v. Let n be minimal such that $\sigma'(v'b^n) \in \Sigma_2 \cup \Delta$ is a non-ε-transition. Then, we define $(va)' = v'b^n a$. Intuitively, we extend v' by irrelevant inputs until σ' completes the run infix processing the last letter.

Let $w \in (\Sigma_1 \times \Sigma_2)^\omega$ be consistent with σ. An induction shows that there is an $w' \in (\Sigma_1 \times (\Sigma_2 \cup \Delta))^\omega$ that is consistent with σ' that encodes w and an accepting run of $\mathcal{P}$ on w. Hence, $w \in L(\mathcal{P})$, i.e., σ is indeed winning for Player 2 in $\mathcal{G}(L(\mathcal{P}))$.

As universality of $L \subseteq \Sigma^\omega$ is equivalent to Player 2 winning $\mathcal{G}(\{\binom{w}{\#^\omega} \mid w \in L\})$ and as $\{\binom{w}{\#^\omega} \mid w \in L\}$ is in ω-GFG-CFL if L is in ω-GFG-CFL, we obtain the following corollary of our main theorem.

Corollary 1. *The following problem is in* ExpTime*: Given an ω-GFG-PDA $\mathcal{P}$, is $L(\mathcal{P})$ universal.*

This contrasts with the universality problem for ω-PDA, which is undecidable. Unfortunately, we will prove in Section 7 that it is undecidable to determine whether a given ω-PDA is good-for-games.

Emptiness of ω-GFG-PDA is also decidable, as it is decidable for ω-PDA. And although equivalence is decidable for ω-DPDA with weak acceptance conditions [21] (that is, each strongly connected component is either rejecting or accepting), it is already undecidable for ω-DPDA with Büchi acceptance conditions [5]. Whether co-Büchi or weaker acceptance conditions make this problem decidable for ω-GFG-PDA is, to the best of our knowledge, open.

Also, let us mention that one can apply the reduction presented in the proof of Theorem 3 also if $\mathcal{P}$ is not known to be good-for-games. If Player 2 wins $\mathcal{G}(L(\mathcal{P}'))$, then she wins $\mathcal{G}(L(\mathcal{P}))$ as well. However, if Player 2 does not win $\mathcal{G}(L(\mathcal{P}'))$, then she might or might not win $\mathcal{G}(L(\mathcal{P}))$, i.e., the reduction is sound, but not complete, if $\mathcal{P}$ is not good-for-games. The same holds true for Corollary 1.

While we only consider the realizability problem here, i.e., the problem of determining whether Player 2 wins the game, our proof of Theorem 3 can be extended to the synthesis problem, i.e., the problem of computing a winning strategy for Player 2, if she wins the game. Such a strategy can be finitely represented by a deterministic pushdown automaton with output reading finite sequences over Σ_1 and outputting a single letter from Σ_2. These are efficiently computable for Gale-Stewart games with ω-DCFL winning conditions [24]. Hence, one can compute a winning strategy for Player 2 in $\mathcal{G}(L(\mathcal{P}'))$ and then apply the transformation described in the second part of the proof of Theorem 3, which is implementable by deterministic pushdown transducer.

5 Closure Properties

In Section 3, we have shown that ω-GFG-CFL is a new subclass of ω-contextfree languages. Here, we study the closure properties of this class, which differ considerably from those of ω-DCFL and ω-CFL.

We say that a class $\mathcal{L}$ of ω-languages is closed under (ε-free) homomorphisms if $\{f(w) \mid w \in L\}$ is in $\mathcal{L}$ for every $L \in \mathcal{L}$ and every $f\colon \Sigma \to (\Sigma')^+$. Here, we disallow ε in the image of f to ensure that $f(w)$ is infinite.

Theorem 4. *ω-GFG-CFL is not closed under union, intersection, complementation, set difference, nor homomorphism.*

Proof. **Union:** As ω-DCFL contains $L_1 = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\}$ and $L_2 = \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$ whose union is not in ω-GFG-CFL (Theorem 2), Proposition 1 implies that ω-GFG-CFL is not closed under union.

Intersection: As ω-DCFL contains $L_1 = \{a^nb^na^*b^\omega \mid n \geq 1\}$ and $L_2 = \{a^*b^na^nb^\omega \mid n \geq 1\}$ whose intersection $L_1 \cap L_2 = \{a^nb^na^nb^\omega \mid n \geq 1\}$ is not even in ω-CFL [9, Proposition 1.3], Proposition 1 implies that ω-GFG-CFL is not closed under intersection.

Complementation: We show that the complement of the language $L_{\mathrm{ss}} \in \omega$-GFG-CFL used in Section 3 to separate ω-DCFL and ω-GFG-CFL is not even in ω-CFL. A word w is not in L_{ss} if $\pi_i(w)$ does not have a safe suffix for both $i \in \{1, 2\}$, which due to Remark 2 is equivalent to

$$\liminf_{n\to\infty} EL(\pi_i(w(0)\cdots w(n))) = -\infty$$

for both i.

Towards a contradiction, assume there is an ω-PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ with $L(\mathcal{P}) = \Sigma^\omega \setminus L_{\mathrm{ss}}$. As in the proof of Lemma 2, define $x_1 = \binom{+}{0}\binom{+}{-}$ and $x_2 = \binom{0}{+}\binom{-}{+}$. Recall that every infix of length at least 3 of a word built by concatenating copies of the x_i has a strictly positive energy level in one component.

Again, we define

$$w = x_1\,(x_2)^3\,(x_1)^7\,(x_2)^{15}\,(x_1)^{31}\,(x_2)^{63}\cdots$$

which satisfies $EL(\pi_1(x_1(x_2)^3\cdots(x_2)^{2^{2j}-1})) = -j$ for every $j > 1$ and $EL(\pi_2(x_1(x_2)^3\cdots(x_1)^{2^{2j-1}-1})) = -j$ for every $j > 0$. Hence, $w \in \Sigma^\omega \setminus L_{\mathrm{ss}}$, i.e., there is an accepting run $\rho = c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ of $\mathcal{P}$ on w.

Recall that a step of ρ is a position n such that $\mathrm{sh}(c_n) \leq \mathrm{sh}(c_{n+j})$ for all $j \geq 0$. Every infinite run has infinitely many steps. Hence, we can find two steps $s < s'$ satisfying the following properties:

1. There is a state $q \in Q$ and a stack symbol $X \in \Gamma_\perp$ such that c_s and $c_{s'}$ have the form $(q, \gamma X)$ for some γ, i.e., both configurations have the same state and topmost stack symbol.
2. The maximal color labeling the sequence $\tau_s\cdots\tau_{s'-1}$ of transitions leading from c_s to $c_{s'}$ is even.
3. This sequence $\tau_s\cdots\tau_{s'-1}$ processes an infix v of w with $EL(\pi_i(v)) > 0$, for some $i \in \{1, 2\}$.

Consider the sequence $\tau_0\cdots\tau_{s-1}(\tau_s\cdots\tau_{s'-1})^\omega$ of transitions. Due to the first property, it induces a run ρ' of $\mathcal{P}$, which is accepting due to the second property. Finally, due to the third property, ρ' processes a word with suffix v^ω. Such a word has a safe suffix in component i, as $EL(\pi_i(v)) > 0$.

Hence, we have constructed a word w in L_{ss} such that there is an accepting run of $\mathcal{P}$ on w, i.e., we have derived a contradiction to $L(\mathcal{P}) = \Sigma^\omega \setminus L_{\mathrm{ss}}$. As we have picked $\mathcal{P}$ arbitrarily, we have shown that $\Sigma^\omega \setminus L_{\mathrm{ss}}$ is not in ω-CFL. Thus, due to Proposition 1 and Lemma 1, ω-GFG-CFL is not closed under complementation.

Set difference: As Σ^ω is in ω-DCFL $\subseteq$ ω-GFG-CFL for every alphabet Σ, ω-GFG-CFL cannot be closed under set difference, as complementation is set difference with Σ^ω.

Homomorphism: As ω-DCFL contains

$$L = \left\{ \left[\binom{a}{1}\binom{\#}{1} \right]^n \left[\binom{b}{1}\binom{\#}{1} \right]^n \binom{\#}{1}^\omega \,\middle|\, n \geq 1 \right\} \cup \left\{ \left[\binom{a}{2}\binom{\#}{2} \right]^n \left[\binom{b}{2}\binom{\#}{2} \right]^{2n} \binom{\#}{2}^\omega \,\middle|\, n \geq 1 \right\}$$

whose projection (which is a homomorphism)

$$\pi_1(L) = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\} \cup \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$$

is not in ω-GFG-CFL (Theorem 2), Proposition 1 implies that ω-GFG-CFL is not closed under homomorphisms.

As we only used languages in ω-DCFL $\subseteq$ ω-GFG-CFL to witness the failure of closure under intersection, union, and set difference, we obtain the following corollary.

Corollary 2. *ω-GFG-CFL is not closed under union, intersection, and set difference with languages in ω-DCFL.*

Finally, using standard arguments one can show that closure under these operations with ω-regular languages holds, as it does for ω-DCFL and ω-CFL.

Theorem 5. *If $L \in \omega$-GFG-CFL and R is ω-regular, then $L \cap R$, $L \cup R$, and $L \setminus R$ are in ω-GFG-CFL as well.*

Proof. Let $L = L(\mathcal{P})$ for some ω-GFG-PDA and R be ω-regular, i.e., $R = L(\mathcal{A})$ for some deterministic parity automaton $\mathcal{A}$ (see, e.g., [16] for definitions). Furthermore, let $\mathcal{P} \times \mathcal{A}$ be the product automaton of these two automata, which is again an ω-PDA that simulates a run of $\mathcal{P}$ and the unique run of $\mathcal{A}$ simultaneously.

Using a detour via the Muller acceptance condition and the LAR construction (see [16]), one can turn $\mathcal{P} \times \mathcal{A}$ into automata $(\mathcal{P} \times \mathcal{A})_\cap$, $(\mathcal{P} \times \mathcal{A})_\cup$, and $(\mathcal{P} \times \mathcal{A})_\setminus$ such that the following holds true:

- A run of $(\mathcal{P} \times \mathcal{A})_\cap$ is accepting if the simulated run of $\mathcal{P}$ and the simulated run of $\mathcal{A}$ are accepting.
- A run of $(\mathcal{P} \times \mathcal{A})_\cup$ is accepting if either the simulated run of $\mathcal{P}$ or the simulated run of $\mathcal{A}$ is accepting.
- A run of $(\mathcal{P} \times \mathcal{A})_\setminus$ is accepting if the simulated run of $\mathcal{P}$ is accepting and the simulated run of $\mathcal{A}$ is not accepting.

All three automata can be shown to be good-for-games, as only the nondeterminism of $\mathcal{P}$ has to be resolved: A resolver for $\mathcal{P}$ can easily be turned into one for the three automata that just ignores the additional inputs stemming from taking the product (note that this crucially depends on $\mathcal{A}$ and the LAR memory being deterministic).

Note that $R \setminus L$ is not necessarily in ω-GFG-CFL (not even in ω-CFL) if R is ω-regular and L is in ω-GFG-CFL.

6 Comparison to Visibly Pushdown Languages

In this section, we compare ω-GFG-CFL to another important subclass of ω-CFL, the class of visibly pushdown languages [1], for which solving games is decidable as well [22].

Visibly pushdown automata are defined with respect to a partition $\widetilde{\Sigma} = (\Sigma_c, \Sigma_r, \Sigma_s)$ of the input alphabet and have to satisfy the following conditions:

- A letter $a \in \Sigma_c$ is only processed by transitions of the form (q, X, a, q', XY) with $X \in \Gamma_\bot$, i.e., some stack symbol Y is pushed onto the stack.

– A letter $a \in \Sigma_r$ is only processed by transitions of the form $(q, X, a, q', \varepsilon)$ with $X \neq \bot$ or $(q, \bot, a, q', \bot)$, i.e., the topmost stack symbol is removed, or if the stack is empty, it is left unchanged.
– A letter $a \in \Sigma_s$ is only processed by transitions of the form (q, X, a, q', X) with $X \in \Gamma_\bot$, i.e., the stack is left unchanged.
– There are no ε-transitions.

Intuitively, the stack height of the last configuration of a run processing some $v \in (\Sigma_c \cup \Sigma_r \cup \Sigma_s)^*$ only depends on v.

A language $L \subseteq \Sigma^\omega$ is in ω-VPL if there is a partition $\widetilde{\Sigma}$ of Σ such that there is a nondeterministic ω-visibly pushdown automaton $\mathcal{P}$ recognizing L with respect to $\widetilde{\Sigma}$.

Theorem 6. *ω-GFG-CFL and ω-VPL are incomparable with respect to inclusion.*

Proof. The language $L_{\mathrm{ss}} \in \omega$-GFG-CFL used in Section 3 to separate ω-GFG-CFL and ω-DCFL is not in ω-VPL, as ω-VPL $\subseteq$ ω-CFL is closed under complementation [1] while the complement of L_{ss} is not in ω-CFL (Theorem 4). Thus, ω-GFG-CFL is not included in ω-VPL.

For the other non-inclusion, let $\Sigma = \{\texttt{+}, \texttt{-}\}$ and define the value[7] $\mathrm{val}(v) \in \mathbb{N}$ of a finite word $v \in \Sigma^*$ inductively as $\mathrm{val}(\varepsilon) = 0$ as well as $\mathrm{val}(v\texttt{+}) = \mathrm{val}(v) + 1$ and $\mathrm{val}(v\texttt{-}) = \max\{0, \mathrm{val}(v) - 1\}$. We show that

$$L_{\mathrm{repbdd}} = \{w \in \Sigma^\omega \mid \text{there is an } s \in \mathbb{N} \text{ such that } \mathrm{val}(w(0)\cdots w(n)) = s \text{ for infinitely many } n\},$$

which is in ω-VPL [1], is not in ω-GFG-CFL.

To this end, fix an ω-GFG-PDA $\mathcal{P}$ with resolver $r \colon \Delta^* \times \Sigma \to \Delta$. We will show that it does not recognize L_{repbdd}. We assume without loss of generality that all ε-transitions have color 0 while all Σ-transitions have a nonzero color. This can be achieved by adding a component to $\mathcal{P}$'s states that accumulates the maximal color seen along a sequence of ε-transitions until a Σ-transition is used. As a consequence, a run of $\mathcal{P}$ on some infinite input satisfies the acceptance condition if and only if the sequence of Σ-transitions satisfies the acceptance condition, i.e., the colors of ε-transitions are irrelevant and will be ignored in the following.

Given a run prefix $\rho \in \Delta^*$ and a letter $a \in \Sigma$, let $\mathrm{ext}(\rho, a)$ be the unique extension of ρ induced by r when processing a. Formally, we define

$$\mathrm{ext}(\rho, a) = \begin{cases} \rho \cdot r(\rho, a) & \text{if } \ell(r(\rho, a)) = a, \\ \mathrm{ext}(\rho \cdot r(\rho, a), a) & \text{if } \ell(r(\rho, a)) = \varepsilon. \end{cases}$$

Note that $\mathrm{ext}(\rho, a)$ is a finite extension of ρ, if ρ is a run prefix that is consistent with r. This follows from the fact that every prefix in Σ^* can be extended to an ω-word that is in $L(\mathcal{P})$.

We now build an ω-word w, letter by letter, based on how r resolves nondeterminism on the prefix built so far. The intuition is that whenever the run built by r sees an even color after a prefix v, the value of prefixes extending v remains above $\mathrm{val}(v)$ until a larger odd color is seen, and then returns to $\mathrm{val}(v)$ (unless an even higher even color is seen in the meantime). The result will be that either w is in L_{repbdd} but rejected by $\mathcal{P}$, or w is not in L_{repbdd} but accepted by $\mathcal{P}$.

Formally, we inductively define an infinite sequence of sequences $\rho_n \in \Delta^*$, all ending in a Σ-transition. To start, we define $\rho_0 = \mathrm{ext}(\mathrm{ext}(\varepsilon, \texttt{+}), \texttt{+})$. To define ρ_{n+1} we have to consider several cases.

First, assume ρ_n ends with a $\texttt{+}$-transition. Let c be the last and c' be the second-to-last nonzero color appearing in ρ_n (this is well-defined as ρ_0 contains two colors). If c is odd and $c \geq c'$, then we define $\rho_{n+1} = \mathrm{ext}(\rho_n, \texttt{-})$ (Case 1), otherwise, $\rho_{n+1} = \mathrm{ext}(\rho_n, \texttt{+})$ (Case 2).

Now, assume ρ_n ends with a $\texttt{-}$-transition. Let $\mathrm{sff}(\rho_n)$ be the suffix of ρ_n starting with the last $\texttt{+}$-transition (this is well-defined, as ρ_0 contains $\texttt{+}$-transitions). Furthermore, let $\mathrm{prf}(\rho_n)$ be the prefix of ρ_n ending with the last transition having an even color that is at least as large as the maximal color labeling a transition in $\mathrm{sff}(\rho_n)$. See Figure 4 for an illustration: $\mathrm{prf}(\rho_n)$ is the prefix ending in the last transition that has an even color c that is at least as large as the colors $c_0, \ldots, c_j$, the colors occurring in $\mathrm{sff}(\rho_n)$. If there is no transition with such a color, then define $\mathrm{prf}(\rho_n) = \varepsilon$. Note that $\mathrm{sff}(\rho_n)$ and $\mathrm{prf}(\rho_n)$ might

[7] Alur and Madhusudan used the term *stack height* instead of *value*, but this is misleading here, since our automata are not necessarily visibly, i.e., the stack height of a run prefix on v might differ from $\mathrm{val}(v)$.

overlap and that $\mathrm{prf}(\rho_n) = \rho_n$ is possible if the last transition of ρ_n has an even color that is maximal among those in $\mathrm{sff}(\rho_n)$. If

$$\mathrm{val}(\ell(\rho_n)) > \mathrm{val}(\ell(\mathrm{prf}(\rho_n))) + 1,$$

then we define $\rho_{n+1} = \mathrm{ext}(\rho_n, \texttt{-})$ (Case 3), otherwise, $\rho_{n+1} = \mathrm{ext}(\rho_n, \texttt{+})$ (Case 4). Note that the even-numbered cases extend by a +, the odd-numbered cases by a -.

$$\rho_n\colon\ \overbrace{\longrightarrow\ \longrightarrow\ \cdots\ \longrightarrow\ \xrightarrow[c]{}}^{\mathrm{prf}(\rho_n)}\ \longrightarrow\ \overbrace{\xrightarrow[c_0]{\texttt{+}}\ \xrightarrow[c_1]{\texttt{-}}\ \cdots\ \xrightarrow[c_j]{\texttt{-}}}^{\mathrm{sff}(\rho_n)}$$

Fig. 4. Illustration of the definition of ρ_{n+1} in the case where ρ_n ends with a -‑transition. Each arrow depicts a transition of ρ_n, its color is depicted below the arrow, the letter it processes above (some transitions in $\mathrm{sff}(\rho_n)$ may be ε-transitions, but only the first one is a +-transition).

Now, let $\rho \in \Delta^\omega$ be the unique infinite sequence such that each ρ_n is a prefix of ρ, and let $w = \ell(\rho)$, which is an infinite word. Then, ρ induces a run of $\mathcal{P}$ on w that is consistent with r.

First, assume w is of the form $v\texttt{+}^\omega$. Then, from some point onwards, we only use Case 2 to extend ρ_n to ρ_{n+1}, i.e., if the last color of ρ_n is odd, then the second-to-last color is strictly larger. This implies that the maximal color occurring infinitely often in ρ is even. Thus, the run ρ is accepting although w is not in L_{repbdd}.

Now, assume w is of the form $v\texttt{-}^\omega$, i.e., $\mathrm{val}(\ell(\rho_n)) = 0$ for almost all n. Then, from some point onwards, we only use Case 3 to extend ρ_n to ρ_{n+1}, i.e., we have

$$\mathrm{val}(\ell(\rho_n)) > \mathrm{val}(\ell(\mathrm{prf}(\rho_n))) + 1$$

for almost all n. Combining both equations yields a contradiction, as $\mathrm{val}(\ell(\mathrm{prf}(\rho_n)))+1$ is positive. Thus, w cannot have the form $v\texttt{-}^\omega$.

As a last case, assume w contains infinitely many + and infinitely many -. First, we study the case where the maximal color occurring infinitely often in ρ, call it c, is odd. Let n_0 be such that the suffix of ρ obtained from removing ρ_{n_0} only contains colors that occur infinitely often and ρ_{n_0} contains at least one c that is not followed by a larger even color. Furthermore, let ρ^* be the longest prefix of ρ ending in a transition with an even color that is larger than c (which by construction is a prefix of ρ_{n_0}). We define $\rho^* = \varepsilon$ if there is no such color. Let $b = \mathrm{val}(\rho^*)$. We claim that there are infinitely many n such that $\mathrm{val}(\ell(\rho_n)) \le b + 1$. Then, ρ is rejecting while w is in L_{repbdd} due to the pigeon-hole principle.

To this end, let $n' > n_0$ such that $\rho_{n'}$ ends with a transition τ of color c. As there are infinitely many such n', it suffices to show that there is a $n \ge n'$ such that $\mathrm{val}(\ell(\rho_n)) \le b + 1$. First, assume $\ell(\tau) = \texttt{+}$. By construction, the last nonzero color occurring before the final c is not larger than c. Hence, we have $\rho_{n'+1} = \mathrm{ext}(\rho_{n'}, \texttt{-})$ due to Case 1. Now, we either already have $\mathrm{val}(\rho_{n'+1}) \le b + 1$ or we apply Case 3 repeatedly until we have produced some ρ_n with $\mathrm{val}(\rho_n) = b + 1$. The reason why Case 3 is always applicable is that the suffix $\mathrm{sff}(\rho_{n''})$ for $n' + 1 \le n'' \le n$ always contains the last transition of $\rho_{n'}$, with color c. This in turn implies that the prefix $\mathrm{prf}(\rho_{n''})$ is equal to ρ^*.

The case for $\ell(\tau) = \texttt{-}$ is similar: either we already have $\mathrm{val}(\rho_{n'}) \le b+1$ or we apply Case 3 repeatedly until we have produced some ρ_n with $\mathrm{val}(\rho_n) = b + 1$.

Finally, we consider the case where c, the maximal color occurring infinitely often in ρ, is even. We show that there are infinitely many n such that $\mathrm{val}(\ell(\rho_{n'})) > \mathrm{val}(\ell(\rho_n))$ for every $n' > n$. This implies that for every s there are only finitely many n such that $\mathrm{val}(\ell(\rho_n)) = s$. As the prefixes of w are of the form $\ell(\rho_n)$ for $n \in \mathbb{N}$, we obtain that w is not in L_{repbdd}, even though the run of $\mathcal{P}$ on w induced by ρ is accepting.

Let n_0 be such that the suffix of ρ obtained from removing ρ_{n_0} only contains colors that occur infinitely often and ρ_{n_0} contains a suffix ρ_s starting with a transition of color c such that ρ_s does not contain a larger color than c, but does contain a +-transition. By definition, the resulting suffix of ρ has infinitely many transitions of color c, all of which mark the end of some ρ_n with $n \ge n_0$. We show that each such n has the desired property.

By the choice of n_0, the last transition of ρ_{n+1} is a +-transition, no matter whether the last transition of ρ_n is a +-transition or a −-transition. If it is a +-transition, then ρ_{n+1} is obtained by applying Case 2 to ρ_n, as the color of the last transition of ρ_n is even. On the other hand, if the last transition of ρ_n is a −-transition, then ρ_{n+1} is obtained by applying Case 4 to ρ_n, as the prefix $\mathrm{prf}(\rho_n)$ is equal to ρ_n by the fact that ρ_{n_0} contains an occurrence of a c that is not followed by a larger color, but by a +-transition.

Now, assume towards a contradiction that there is some $n' > n$ such that $\mathrm{val}(\ell(\rho_{n'})) = \mathrm{val}(\ell(\rho_n))$. Pick n' minimal with this property. Then, due to minimality, the last transition of $\rho_{n'}$ and the last transition of $\rho_{n'-1}$ are both −-transitions. Hence, $\rho_{n'} = \mathrm{ext}(\rho_{n'-1}, \texttt{-})$ due to Case 3.

Now, consider the prefix $\mathrm{prf}(\rho_{n'-1})$ in the application of Case 3 to $\rho_{n'-1}$. It is either equal to ρ_n (as its color is even and at least as large as all colors that may be appear in the suffix $\mathrm{sff}(\rho_{n'-1})$ of $\rho_{n'-1}$), or it is some extension of ρ_n. In both cases, we have $\mathrm{val}(\ell(\rho_n)) \leq \mathrm{val}(\ell(\mathrm{prf}(\rho_{n'-1})))$ (in the latter due to the minimality of n'). Hence,

$$\begin{aligned} \mathrm{val}(\ell(\rho_n)) &\leq \mathrm{val}(\ell(\mathrm{prf}(\rho_{n'} - 1))) < \mathrm{val}(\ell(\rho_{n'-1})) - 1 \\ &= \mathrm{val}(\ell(\rho_{n'})) = \mathrm{val}(\ell(\rho_n)), \end{aligned}$$

which yields the desired contradiction. Here, the strict inequality follows from the definition of Case 3.

To conclude, in either of the cases we considered, the run ρ is accepting, but $w \notin L_{\mathrm{repbdd}}$, or the run ρ induced by the resolver r is rejecting, but $w \in L_{\mathrm{repbdd}}$. Hence, either $\mathcal{P}$ does not recognize L_{repbdd} or r is not a resolver for $\mathcal{P}$, i.e., L_{repbdd} is not in ω-GFG-CFL.

7 Deciding Good-for-gameness

In this section, we show that deciding good-for-gameness is, unfortunately, undecidable, both for automata and for languages. These results contrast with good-for-gameness being decidable for parity automata [19], even in polynomial time for Büchi and co-Büchi automata [19, 2], and every ω-regular language being good-for-games, as deterministic parity automata recognize all ω-regular languages.

To show these undecidability results, we introduce some additional notation. If $\mathcal{P}$ is an ω-PDA and q one of its states, then we write $L(\mathcal{P}, q)$ for the language accepted by the automaton obtained from $\mathcal{P}$ by replacing its initial state with q.

Theorem 7. *The following problems are undecidable:*

1. *Given an ω-PDA $\mathcal{P}$, is $\mathcal{P}$ good-for-games?*
2. *Given an ω-PDA $\mathcal{P}$, is $L(\mathcal{P}) \in \omega$-GFG-CFL?*

Proof. Both proofs proceed by reduction from an undecidable problem for PDA over finite words (see, e.g., [18]). Such an automaton has the same structure as an ω-PDA, but the coloring Ω is replaced by a set F of accepting states. A finite run is accepting, if it ends in an accepting state.

1.) We reduce the inclusion problem for DPDA over finite words to deciding whether an ω-PDA is good-for-games. Since the inclusion problem is undecidable [23], so is deciding whether an ω-PDA is good-for-games.

Given DPDA $\mathcal{D}_1$ and $\mathcal{D}_2$ over Σ^*, we first define infinitary versions of $\mathcal{D}_1$ and $\mathcal{D}_2$: Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be ω-DPDA over $(\Sigma \cup \{\#\})^\omega$, where $\# \notin \Sigma$, that are identical to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, except with additional #-transitions from states that were accepting in $\mathcal{D}_1$ and $\mathcal{D}_2$ to accepting sinks; all other states are made rejecting. Then $L(\mathcal{P}_i)$ consists of words of the form $v\#w$ where $v \in L(\mathcal{P}_i)$ and $w \in (\Sigma \cup \{\#\})^\omega$. We have $L(\mathcal{P}_1) \subseteq L(\mathcal{P}_2)$ if and only if $L(\mathcal{D}_1) \subseteq L(\mathcal{D}_2)$.

Consider $\mathcal{P}$, an ω-PDA over $\Sigma \cup \{\#, \$\}$ built as follows: A fresh initial state q_I has \$-transitions to fresh states q_1 and q_2; from q_1 there is an ε-transition to the initial state of $\mathcal{P}_1$, and from q_2 there is a \$-transition to an accepting sink q_s and an ε-transition to the initial state of $\mathcal{P}_2$. None of these transitions manipulate the stack. See Figure 5. We show that $\mathcal{P}$ is good-for-games if and only if $L(\mathcal{D}_1) \subseteq L(\mathcal{D}_2)$.

If $L(\mathcal{D}_1) \subseteq L(\mathcal{D}_2)$, then $L(\mathcal{P}_1) \subseteq L(\mathcal{P}_2)$ and $L(\mathcal{P}, q_1) \subseteq L(\mathcal{P}, q_2)$. Let r be defined such that

- $r(\varepsilon, \$) = (q_I, \bot, \$, q_2, \bot)$,
- $r((q_I, \bot, \$, q_2, \bot), \$) = (q_2, \bot, \$, q_s, \bot)$, and
- $r((q_I, \bot, \$, q_2, \bot), a) = (q_2, \bot, \$, q_I^2, \bot)$ where $a \neq \$$ and q_I^2 denotes the initial state of $\mathcal{P}_2$,

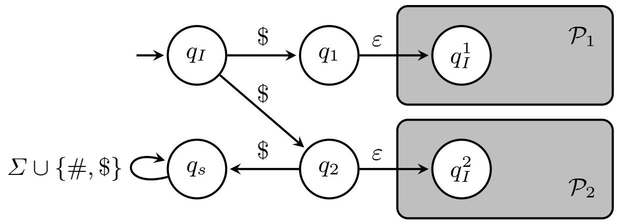

Fig. 5. The construction of $\mathcal{P}$ from $\mathcal{P}_1$ and $\mathcal{P}_2$. All new transitions do not manipulate the stack, i.e., if $\mathcal{P}_i$ is entered via the ε-transition, then the initial configuration of $\mathcal{P}_i$ is reached.

i.e., r produces a run prefix that reaches either the accepting sink q_s or the initial state q_I^2 of $\mathcal{P}_2$, which is deterministic. Hence, in both cases, there are no further nontrivial choices to make.

We claim that r is a resolver. Let $w \in L(\mathcal{P})$, i.e., either $w = \$w'$ with

$$w' \in L(\mathcal{P}, q_1) \cup L(\mathcal{P}, q_2) \subseteq L(\mathcal{P}, q_2),$$

i.e., the second letter of w is not equal to \$, or $w = \$\$w'$ with $w' \in (\Sigma \cup \{\#, \$\})^\omega$. In both cases, the run of $\mathcal{P}$ on w induced by r is accepting.

Conversely, if there is a word $v \in L(\mathcal{D}_1) \setminus L(\mathcal{D}_2)$, there is no resolver for $\mathcal{P}$. Indeed, if a resolver r chooses the transition $(q_I, \bot, \$, q_1, \bot)$ to q_1 to begin with, it is not a resolver since $\$^\omega \in L(\mathcal{P})$, but $\$^\omega \notin L(\mathcal{P}, q_1)$. Finally, if r chooses the transition $(q_I, \bot, \$, q_2, \bot)$ to q_2 to begin with, then r is not a resolver either since $\$v\#^\omega \in L(\mathcal{P})$, but $v\#^\omega \notin L(\mathcal{P}, q_2)$.

2.) We proceed by a reduction from the universality problem for PDA over finite words, which is undecidable [18]. Namely, given a PDA $\mathcal{F}$ over Σ we build an ω-PDA $\mathcal{P}$ over $\Sigma_\# \times \{a, b, \#\}$ with $\Sigma_\# = \Sigma \cup \{\#\}$ such that $L(\mathcal{P}) \in \omega$-GFG-CFL if and only if $L(\mathcal{F})$ is universal.

First, note that

$$L_1 = \{v\#w | v \in L(\mathcal{F}) \text{ and } w \in \Sigma_\#^\omega\} \cup \Sigma^\omega$$

is universal if and only if $L(\mathcal{F})$ is universal. Furthermore, $\mathcal{F}$ can easily be turned into an ω-PDA recognizing L_1. So, we can construct from $\mathcal{F}$ an ω-PDA $\mathcal{P}$ recognizing the language

$$L(\mathcal{P}) = \{w \in (\Sigma_\# \times \{a, b, \#\})^\omega \mid \pi_1(w) \in L_1 \text{ or } \pi_2(w) \in L_2\},$$

where L_2 is the ω-language from the proof of Theorem 2, which is not in ω-GFG-CFL but in ω-CFL. We claim that $\mathcal{P}$ has the desired property. Trivially, if $L(\mathcal{F})$ is universal, then so is $L(\mathcal{P})$, which implies that $L(\mathcal{P})$ is in ω-GFG-CFL.

Now assume that $L(\mathcal{F})$ is not universal which is witnessed by some word $v\#^\omega \notin L_1$, i.e, such that $v \notin L(\mathcal{F})$. Towards a contradiction, assume that there is an ω-GFG-PDA $\mathcal{R}$ with $L(\mathcal{R}) = L(\mathcal{P})$, say with resolver r. We turn $\mathcal{R}$ into another ω-GFG-PDA $\mathcal{R}'$ recognizing the language

$$\left\{ w \in \{a, b, \#\}^\omega \;\middle|\; \binom{v\#^\omega}{w} \in L(\mathcal{R}) \right\},$$

which yields the desired contradiction, as this language is equal to L_2, which is not in ω-GFG-CFL.

To this end, we equip $\mathcal{R}$ with a counter ranging over the positions of $v\#$ to simulate a run of $\mathcal{R}$ on input $\binom{v\#^\omega}{w}$ when given the input w. As the counter behaves deterministically, no new nondeterminism is introduced when constructing $\mathcal{R}'$ from $\mathcal{R}$. Hence, r can be turned into a resolver for $\mathcal{R}'$.

8 Conclusion

We have introduced good-for-games ω-pushdown automata and proved that they recognize a novel class of ω-contextfree languages for which solving games is decidable. Furthermore, we have studied (the mostly nonexistent) closure properties of the new class, proven that it is incomparable to ω-visibly pushdown languages, and that deciding good-for-gameness is undecidable for ω-pushdown automata and ω-contextfree languages.

For ω-regular automata, there are many equivalent ways to defining good-for-gameness: via resolvers as done here, via a game characterization [17], and via composition with games, with trees [3], or with other automata [4]. All these characterizations can be lifted to ω-GFG-PDA, but this is beyond the scope of this paper.

We hope this paper is the catalyst for an in-depth study of good-for-games automata in settings where nondeterministic automata are more expressive than deterministic ones. But even for the setting of ω-contextfree languages considered here, we open many interesting directions for further research. Let us conclude by listing a few:

- Is the universality problem for ω-GFG-PDA ExpTime-complete? Note that this problem is a promise problem with an undecidable promise, e.g., we only consider good-for-games ω-PDA as input.
- Does every ω-GFG-PDA have a resolver that is implementable by a deterministic pushdown automaton with output? If yes, can such a resolver be effectively computed? Here, we conjecture that pumping arguments similar to those presented in the proof of Theorem 2 show that there is no such resolver for the ω-GFG-PDA from the proof of Lemma 1 recognizing L_{ss}, i.e., in general the answer is no.
- Can ω-GFG-CFL be characterized by some extension of Monadic Second-order Logic? Such characterizations haven been exhibited for contextfree languages of finite [20] and infinite words [13] as well as for the class of visibly pushdown languages [1]. However, there is, to the best of our knowledge, no characterization of the deterministic contextfree languages, neither for finite nor infinite words.
- Is it worthwhile to extend the concept of good-for-gameness to pushdown automata over finite words and to visibly pushdown automata over infinite or finite words? Note that deterministic and nondeterministic visibly pushdown automata recognize the same class of languages[8], i.e., good-for-games visibly pushdown automata cannot be more expressive, just more succinct.
- Another interesting direction, proposed by one of the reviewers, is to further restrict the model of ω-GFG-PDA, either by considering weaker classes of resolvers, e.g., finite-state implementable ones, or by considering weaker classes of nondeterministic ω-PDA, e.g., unambiguous ones. Note that some of our results from Section 7 directly carry over to these restricted models, but our expressiveness results from Section 3 do not.

Acknowledgements We thank Sven Schewe and Patrick Totzke for fruitful discussions leading to the results presented in Section 7.

This work is part of a project that receives funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 892704.

References

1. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Babai, L. (ed.) STOC 2004. pp. 202–211. ACM (2004), https://doi.org/10.1145/1007352.1007390
2. Bagnol, M., Kuperberg, D.: Büchi Good-for-Games Automata Are Efficiently Recognizable. In: Ganguly, S., Pandya, P. (eds.) FSTTCS 2018. LIPIcs, vol. 122, pp. 16:1–16:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2018), http://drops.dagstuhl.de/opus/volltexte/2018/9915
3. Boker, U., Kuperberg, D., Kupferman, O., Michał Skrzypczak: Nondeterminism in the presence of a diverse or unknown future. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M.Z., Peleg, D. (eds.) ICALP 2013 (Part II). LNCS, vol. 7966, pp. 89–100. Springer (2013), https://doi.org/10.1007/978-3-642-39212-2_11
4. Boker, U., Lehtinen, K.: Good for games automata: From nondeterminism to alternation. In: Fokkink, W., van Glabbeek, R. (eds.) CONCUR 2019. LIPIcs, vol. 140, pp. 19:1–19:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2019), http://drops.dagstuhl.de/opus/volltexte/2019/10921
5. Böhm, S., Göller, S., Halfon, S., Hofman, P.: On Büchi One-Counter Automata. In: Vollmer, H., Vallée, B. (eds.) STACS 2017. LIPIcs, vol. 66, pp. 14:1–14:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2017), http://drops.dagstuhl.de/opus/volltexte/2017/7019
6. Büchi, J.R.: Regular canonical systems. Archiv für mathematische Logik und Grundlagenforschung 6(3), 91–111 (Apr 1964), https://doi.org/10.1007/BF01969548
7. Carayol, A., Hague, M.: Saturation algorithms for model-checking pushdown systems. In: Ésik, Z., Fülöp, Z. (eds.) AFL 2014. EPTCS, vol. 151, pp. 1–24 (2014), https://doi.org/10.4204/EPTCS.151.1

[8] Using stair-deterministic automata [22] in the case of infinite words.

8. Cohen, R.S., Gold, A.Y.: Theory of omega-languages. i. characterizations of omega-context-free languages. J. Comput. Syst. Sci. 15(2), 169–184 (1977), https://doi.org/10.1016/S0022-0000(77)80004-4
9. Cohen, R.S., Gold, A.Y.: Theory of omega-languages. II. A study of various models of omega-type generation and recognition. J. Comput. Syst. Sci. 15(2), 185–208 (1977), https://doi.org/10.1016/S0022-0000(77)80005-6
10. Cohen, R.S., Gold, A.Y.: Omega-computations on deterministic pushdown machines. J. Comput. Syst. Sci. 16(3), 275–300 (1978)
11. Colcombet, T.: The theory of stabilisation monoids and regular cost functions. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) ICALP 2009, (Part II). LNCS, vol. 5556, pp. 139–150. Springer (2009)
12. Colcombet, T.: Fonctions régulières de coût. Habilitation, École Doctorale de Sciences Mathématiques de Paris Centre (2013)
13. Droste, M., Dziadek, S., Kuich, W.: Logic for ω-pushdown automata (2019), to appear in Information and Computation. Available at https://www.informatik.uni-leipzig.de/ dziadek/papers/droste_dziadek_kuich-logic_for_omega_pushdown_automata.pdf
14. Finkel, O.: Topological properties of omega context-free languages. Theor. Comput. Sci. 262(1), 669–697 (2001)
15. Gale, D., Stewart, F.M.: Infinite games with perfect information. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games (AM-28), Volume II, chap. 13, pp. 245–266. Princeton University Press (1953)
16. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games: A Guide to Current Research, LNCS, vol. 2500. Springer (2002)
17. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 395–410. Springer (2006), https://doi.org/10.1007/11874683_26
18. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation, 3rd Edition. Pearson international edition, Addison-Wesley (2007)
19. Kuperberg, D., Michal Skrzypczak: On determinisation of good-for-games automata. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015 (Part II). LNCS, vol. 9135, pp. 299–310. Springer (2015), https://doi.org/10.1007/978-3-662-47666-6_24
20. Lautemann, C., Schwentick, T., Thérien, D.: Logics for context-free languages. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 205–216. Springer (1994)
21. Löding, C.: Simplification problems for deterministic pushdown automata on infinite words. International Journal of Foundations of Computer Science 26(08), 1041–1068 (2015)
22. Löding, C., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 408–420. Springer (2004), https://doi.org/10.1007/978-3-540-30538-5_34
23. Valiant, L.: Decision procedures for families of deterministic pushdown automata. Ph.D. thesis, University of Warwick (1973)
24. Walukiewicz, I.: Pushdown processes: Games and model-checking. Inf. Comput. 164(2), 234–263 (2001)

A Appendix: Another Contextfree Language that is not Good-for-Games

Let $h\colon \{0,1,\#\}^* \to \{0,1\}^*$ be the homomorphism induced by $h(0) = 0$, $h(1) = 1$, and $h(\#) = \varepsilon$. Define

$$P = \{v\#^\omega \mid h(v) = xx^R \text{ for some } x \in \{0,1\}^*\},$$

where x^R denotes the reversal of x. It is straightforward to construct an ω-PDA recognizing P, thereby showing $P \in \omega$-CFL. We show $P \notin \omega$-GFG-CFL.

Towards a contradiction, assume P is recognized by an ω-GFG-PDA $\mathcal{P} = (Q, \Sigma, \Gamma, q_I, \Delta, \Omega)$ with resolver $r\colon \Delta^* \times \Sigma \to \Delta$. Note that r induces a unique run ρ_w for every $w \in P$. Furthermore, if two words $w, w' \in P$ share a prefix, then their runs $\rho_w = c_0\tau_0c_1\tau_1c_2\tau_2\cdots$ and $\rho_{w'} = c'_0\tau'_0c'_1\tau'_1c'_2\tau'_2\cdots$ also share a prefix. More formally, if

$$w(0)\cdots w(n) = w'(0)\cdots w'(n)$$

for some $n \geq 0$, then we have $\tau_0\cdots\tau_k = \tau'_0\cdots\tau'_k$ and thus $c_0\cdots c_{k+1} = c'_0\cdots c'_{k+1}$ for every k such that $\ell(\tau_0\cdots\tau_k) = w(0)\cdots w(n-1)$. Here, the -1 stems from the fact that r resolves nondeterminism based on the next input letter to be processed.

Let $\rho = c_0\tau_0c_1\cdots c_kt_kc_{k+1}$ be an alternating sequence of configurations and transitions, and let $a \in \{0,1,\#\}$. We say that (ρ, a) is r-consistent, if

$$\tau_n = r(\tau_0\cdots\tau_{n-1}, v(|\ell(\tau_0\cdots\tau_{n-1})|))$$

and c_n is the configuration reached by $\mathcal{P}$ after the transitions $\tau_0\cdots\tau_{n-1}$, where $v = \ell(\rho)a$. Now, an r-consistent pair (ρ, a) has Property M if the following holds for every $w \in \{0,1,\#\}^*\#^\omega$: Let $c'_0\tau'_0c'_1\tau'_1c'_2\tau'_2\cdots$ be the unique run of $\mathcal{P}$ on $\ell(\rho)aw$ induced by r. Then, we require $\mathrm{sh}(c_{k'+1}) \geq \mathrm{sh}(c_{k+1})$ for all $k' > k$. Note that we have $\tau'_0\cdots\tau'_k = \tau_0\cdots\tau_k$ and $c_0\cdots c_{k+1} = c'_0\cdots c'_{k+1}$ by definition.

Lifting Property-M to words, we say that $v(0)\cdots v(n) \in \{0,1,\#\}^+$ has property M if there is a run prefix ρ with $\ell(\rho) = v(0)\cdots v(n-1)$ such that $(\rho, v(n))$ is r-consistent and has Property M.

We show that for every $v \in \{0,1,\#\}^+$ there is a $v' \in \{0,1,\#\}^*$ such that vv' has Property M. To this end, assume $v(0)\cdots v(n)$ does not have Property M. Let ρ be a run prefix of $\mathcal{P}$ on $v(0)\cdots v(n-1)$ so that $(\rho, v(n))$ is r-consistent. As v is a prefix of some word in P, such a ρ exists. By definition, (ρ, a) does not have property M, i.e., there is a run induced by r processing a prolongation in $v\{0,1,\#\}^*\#^\omega$ that reaches a stack height strictly smaller than the stack height of the last configuration of ρ. As stack heights are bounded from below, there is a minimal stack height that is assumed by such runs. Let vv' with $v' \in \{0,1,\#\}^*$ be a word processed by such a run prefix ρ' ending with a minimal stack height along all runs considered. Then, $vv'0$ has Property M, as $(\rho', 0)$ has property M, as after ρ' no strictly smaller stack height is reached by runs induced by r.

Now, fix $n = 3|Q|(|\Gamma + 1|) + 1$ and define $v_j = 01^j0$ for every j in the range $1 \leq j \leq n$. As shown above, for every such j, there is a v'_j such that $v_jv'_j$ has Property M.

Now, each $v_jv'_j(v_jv'_j)^R\#^\omega$ is in L, i.e., the unique run

$$c_0^j\tau_0^jc_1^j\tau_1^jc_2^j\tau_2^j\cdots$$

of $\mathcal{P}$ on $v_jv'_j(v_jv'_j)^R\#^\omega$ induced by r is accepting. As each $v_jv'_j$ has Property M, there are prefixes $c_0^j\tau_0^j\cdots c_{k_j}^j\tau_{k_j}^jc_{k_j+1}^j$ of $c_0^j\tau_0^jc_1^j\tau_1^jc_2^j\tau_2^j\cdots$ processing $v_jv'_j$ without its last letter a_j such that $(c_0^j\tau_0^j\cdots c_{k_j}^j\tau_{k_j}^jc_{k_j+1}^j, a_j)$ has Property M.

Now, there are $j_0 \neq j_1$ such that the configurations $c_{k_{j_0}+1}^{j_0}$ and $c_{k_{j_1}+1}^{j_1}$ coincide on their state from Q and their top stack symbol from Γ and such that $a_{j_0} = a_{j_1}$.

Consider the sequence

$$\rho^* = \tau_0^{j_0}\cdots\tau_{k_{j_0}}^{j_0}\tau_{k_{j_1}+1}^{j_1}\tau_{k_{j_1}+2}^{j_1}\tau_{k_{j_1}+3}^{j_1}\cdots.$$

We claim that ρ^* induces an accepting run of $\mathcal{P}$ on $\overline{w} = v_{j_0}v'_{j_0}(v_{j_1}v'_{j_1})^R\#^\omega$. We have

$$v_{j_0}v'_{j_0}(v_{j_1}v'_{j_1})^R = 01^{j_0}0v'_{j_0}(v_{j_1})^R01^{j_1}0,$$

which is not of the form vv^R after removing #'s. Hence, this step completes the proof, as $\overline{w} \notin P$ is accepted by $\mathcal{P}$, yielding the desired contradiction.

The sequence ρ^* satisfies the acceptance condition, as it shares a suffix with the sequence of transitions of the accepting run $c_0^{j_1}\tau_0^{j_1}c_1^{j_1}\tau_1^{j_1}c_2^{j_1}\tau_2^{j_1}\cdots$. Furthermore, we have

$$\ell(\rho^*) = \ell(\tau_0^{j_0}\cdots\tau_{k_{j_0}}^{j_0})\ell(\tau_{k_{j_1}+1}^{j_1}\tau_{k_{j_1}+2}^{j_1}\tau_{k_{j_1}+3}^{j_1}\cdots)$$

where $\ell(\tau_0^{j_0}\cdots\tau_{k_{j_0}}^{j_0})$ is equal to $v_{j_0}v'_{j_0}$ without its last letter a_{j_0}, and where $\ell(\tau_{k_{j_1}+1}^{j_1}\tau_{k_{j_1}+2}^{j_1}\tau_{k_{j_1}+3}^{j_1}\cdots) = a_{j_1}(v_{j_1}v'_{j_1})^R\#^\omega$. The concatenation of these two words is indeed $\overline{w}$, as we have $a_{j_0} = a_{j_1}$ by construction.

Finally, as $(c_0^{j_1}\tau_0^{j_1}\cdots c_{k_{j_1}}^{j_1}\tau_{k_{j_1}}^{j_1}c_{k_{j_1}+1}^{j_1}, a_{j_1})$ has Property M, the run $c_0^{j_1}\tau_0^{j_1}c_1^{j_1}\tau_1^{j_1}c_2^{j_1}\tau_2^{j_1}\cdots$ does after position k_{j_1} not depend on the complete stack content at that position, but only on the state and the top stack symbol of $c_{k_{j_1}+1}$. Hence, appending the run suffix resulting from applying the transitions

$$\tau_{k_{j_1}+1}^{j_1}\tau_{k_{j_1}+2}^{j_1}\tau_{k_{j_1}+3}^{j_1}\cdots$$

after $(c_0^{j_0}\tau_0^{j_0}\cdots c_{k_{j_0}}^{j_0}\tau_{k_{j_0}}^{j_0}c_{k_{j_0}+1}^{j_0}, a_{j_0})$ (recall that $c_{k_{j_0}+1}^{j_0}$ and $c_{k_{j_1}+1}^{j_1}$ have the same state and top stack symbol) yields indeed a run of $\mathcal{P}$ (but not necessarily induced by r).