Alternating Automata and a Temporal Fixpoint Calculus for Visibly Pushdown Languages

Laura Bozzelli

Università di Napoli Federico II , Via Cintia, 80126 - Napoli, Italy

Abstract. We investigate various classes of alternating automata for visibly pushdown languages (*VPL*) over infinite words. First, we show that alternating visibly pushdown automata (*AVPA*) are exactly as expressive as their nondeterministic counterpart (*NVPA*) but basic decision problems for *AVPA* are 2EXPTIME-complete. Due to this high complexity, we introduce a new class of alternating automata called *alternating jump automata* (*AJA*). *AJA* extend classical alternating *finite-state* automata over infinite words by also allowing *non-local* moves. A non-local forward move leads a copy of the automaton from a call input position to the matching-return position. We also allow local and non-local backward moves. We show that one-way *AJA* and two-way *AJA* have the same expressiveness and capture exactly the class of *VPL*. Moreover, boolean operations for *AJA* are easy and basic decision problems such as emptiness, universality, and pushdown model-checking for parity two-way *AJA* are EXPTIME-complete. Finally, we consider a linear-time fixpoint calculus which subsumes the full linear-time μ-calculus (with both forward and backward modalities) and the logic CARET and captures exactly the class of *VPL*. We show that formulas of this logic can be *linearly* translated into parity two-way *AJA*, and vice versa. As a consequence satisfiability and pushdown model checking for this logic are EXPTIME-complete.

1 Introduction

An active field of research is model-checking of pushdown systems. These represent an infinite-state formalism suitable to model the control flow of recursive sequential programs. The model checking problem of pushdown systems against regular properties is decidable and it has been intensively studied in recent years leading to efficient verification algorithms and tools (see for example [16,7,6,9]).

For context-free properties, the pushdown model checking problem is in general undecidable. However, algorithmic solutions have been proposed for checking interesting classes of context-free properties [9,10,8,3,4,1]. In particular, the linear temporal logic CARET, a context-free extension of *LTL*, has been recently introduced [3] which preserves decidability of pushdown model checking. CARET formulas are interpreted on infinite words over an alphabet (called *pushdown alphabet*) which is partitioned into three disjoint sets of calls, returns, and internal symbols. A call denotes invocation of a procedure (i.e. a push stack-operation) and the *matching* return (if any) along a given word denotes the exit from this procedure (corresponding to a pop stack-operation). CARET extends *LTL* by also allowing non-regular versions of the standard *LTL* temporal modalities: the *abstract modalities* can specify non-regular context-free properties which require matching of calls and returns such as correctness of procedures with respect to pre and post conditions, while the (backward) *caller modalities* are useful to express a variety of security properties that require inspection of the call-stack [9,10,8]. In [4], the class of *nondeterministic visibly pushdown automata* (*NVPA*) is proposed as an automata theoretic generalization of CaRet. *NVPA* are pushdown automata which push onto the stack only when a call is read, pops the stack only at returns, and do not use the stack on reading internal symbols. Hence, the input controls the kind of stack operations which can be performed. The resulting class of languages (*visibly pushdown languages* or *VPL*, for short) includes strictly the class of regular languages and that defined by CaRet and is robust like the class of regular languages. In particular, *VPL* are closed under all boolean operations and problems such as universality and inclusion that are undecidable for context-free languages are Exptime-complete for *VPL*.

Our contribution. We further investigate the class of *VPL*. We study various classes of alternating automata for *VPL* and derive interesting connections between a special class of two-way alternating finite-state automata and a fixpoint calculus for *VPL* with both forward and backward modalities. Note that the introduction of backward modalities in fixpoint logics can pose some difficulty in developing decision procedures for such logics since the interaction of backward modalities with the other modalities can be quite subtle. For example, while the standard modal μ-calculus has the finite-model property, this does not hold for the modal μ-calculus extended with backward modalities [15].

Alternating automata [12], i.e. automata featuring nondeterministic as well as universal choices, are interesting for many aspects[1] For example, boolean operations, in particular complementation, are easy [12]. Moreover, alternating *finite-state* automata over words or trees have been founded to be particularly useful to derive optimal decision procedures for various regular temporal logics.

In this paper, first, we consider the alternating version of visibly pushdown automata. While unrestricted alternating pushdown automata on infinite words are more expressive than their nondeterministic counterpart (in particular, emptiness is undecidable), *alternating visibly pushdown automata* (*AVPA*) are exactly as expressive as *NVPA*: any parity *AVPA* $\mathcal{P}$ can be translated into an equivalent Büchi *NVPA* whose size is *doubly* exponential in the size of $\mathcal{P}$. This double-exponential blowup cannot be avoided. In fact we show that emptiness for parity or Büchi *AVPA* is 2Exptime-complete (recall that emptiness for parity *NVPA* is in Ptime [4]). As a consequence the pushdown model checking problem against *AVPA*-specifications is 2Exptime-complete.

Due to the high complexity of basic decision problems for *AVPA*, we introduce a new class of alternating automata called *alternating jump automata* (*AJA*). *AJA* operate on infinite words over a pushdown alphabet, are closed under boolean operations, and extend classical alternating *finite-state* automata by also allowing *non-local* moves. A non-local forward move leads a copy of the automaton from a call position to the matching-return position. We also allow local and non-local backward moves: by performing a non-local backward move, a copy of the automaton jumps from the current input position to the most recent unmatched call position. We show that one-way *AJA*

[1] The notion of alternating automaton over words or trees as defined in [12] applies to all known classes of nondeterministic automata such as pushdown automata or Turing machines.

and two-way *AJA* have the same expressiveness and capture exactly the class of *VPL*. Given a Büchi *NVPA* $\mathcal{P}$, one can construct an equivalent one-way Büchi *AJA* whose size is quadratic in the size of $\mathcal{P}$. Moreover, any parity two-way *AJA* $\mathcal{A}$ can be translated into an equivalent Büchi *NVPA* whose size is *singly* exponential in the size of $\mathcal{A}$. Some ideas in the proposed translation from two-way *AJA* to *NVPA* are taken from standard constructions for two-way finite-state automata [14,15]. However, due to the presence of both (local and non-local) forward and backward moves in two-way *AJA*, we have to face new non-trivial questions which require a more sophisticated approach.

Finally, we consider a fixpoint calculus, called *VP-μTL*, which subsumes CaRet [3] and captures exactly the class of *VPL*. *VP-μTL* extends the full linear-time μ-calculus (with both forward and backward modalities) introduced in [14] by also allowing non-local forward and backward modalities corresponding to the abstract-next and caller modalities of CaRet. We show that each *VP-μTL* sentence can be *linearly* translated into an equivalent parity two-way *AJA*, and vice versa. As a consequence satisfiability of *VP-μTL* is Exptime-complete and the pushdown model-checking problem against *VP-μTL* is Exptime-complete (and Ptime-complete in the size of the pushdown system), hence it is no more costly than that for weaker logics such as CaRet. Note that the backward modalities in *VP-μTL* do not add any expressive power since future *VP-μTL* formulas correspond exactly to one-way *AJA*. However, many interesting properties which require for example inspection of the call-stack are much easier to express using past operators.

Due to the lack of space, for the omitted details we refer the interested reader to a forthcoming extended version of this paper.

Related work. As mentioned above, the class of *NVPA* over infinite words has been studied in [4]. In [4], it is also given a logical MSO-characterization of *VPL* and a characterization in terms of regular tree languages. Games on pushdown graphs against visibly pushdown winning conditions are decidable and have been studied in [11]. In [5], the results given in [4] are reformulated in terms of nondeterministic finite-state automata (*NFA*) over *nested words*. A nested word is an infinite word augmented with a binary relation over the set of positions which encodes the implicit nesting structure of calls and returns. An *NFA* over nested words behaves like an *NFA* over ordinary words with the difference that at a return, the next state depends on both the current state and the state at the matching call. In [2], the notion of nested word is extended to trees in order to allow the automata-theoretic specification of a class of branching-time context-free properties. In particular, the authors introduce one-way alternating automata over nested trees (*AP-NTA*): *AP-NTA* are strictly more expressive than their non-deterministic counterpart and while their emptiness is undecidable, the related pushdown model checking problem is instead Exptime-complete. Finally, an extension of the modal μ-calculus on nested trees as expressive as *AP-NTA* has been studied in [2,1]. When interpreted on infinite words over a pushdown alphabet, this logic corresponds exactly to future *VP-μTL*. As for the modal μ-calculus, the pushdown model checking problem for this new logic is Exptime-complete (even for a fixed formula). Satisfiability is instead undecidable.

2 Preliminaries

Labelled trees. Let $\mathbb{N}$ be the set of natural numbers. A tree T is a prefix closed subset of $\mathbb{N}^*$. Elements of T are called *nodes* and the empty word ε is the root of T. For $x \in T$, a child of x in T is a T-node of the form $x \cdot i$ with $i \in \mathbb{N}$. A *path* of T is a maximal sequence $x_0 x_1 \ldots$ of nodes s.t. $x_0 = \varepsilon$ and for each i, x_{i+1} is a child of x_i. For a set A, an A-labelled tree is a pair $r = \langle T, V \rangle$, where T is a tree and $V : T \rightarrow A$ maps each T-node to an element in A. For $x \in T$, *the subtree of* r *rooted at* x is the A-labelled tree $\langle T_x, V_x \rangle$, where $T_x = \{y \in \mathbb{N}^* \mid x \cdot y \in T\}$ and $V_x(y) = V(x \cdot y)$ for each $y \in T_x$.

Positive boolean formulas and regular acceptance conditions. Throughout this paper, we consider various classes of automata over infinite words equipped with parity or Büchi acceptance conditions over the finite set of (control) states. Formally, for a *finite* set Q, a parity condition over Q is a mapping $\Omega : Q \rightarrow \mathbb{N}$ assigning to each element in Q an integer (called *priority*). The *index* of Ω is the cardinality of the set $\{\Omega(q) \mid q \in Q\}$. A Büchi condition over Q is a subset F of Q. For an infinite sequence $\pi = q_0, q_1 \ldots$ over Q, we say that π satisfies the parity condition Ω if the smallest priority of the elements in Q that occur infinitely often along π is *even*. We say that π satisfies the Büchi condition F if there is some $q \in F$ that occurs infinitely often along π.

For a finite set X, $\mathcal{B}^+(X)$ denotes the set of positive boolean formulas over X built from elements in X using $\vee$ and $\wedge$ (we also allow the formulas $\mathtt{true}$ and $\mathtt{false}$). A subset Y of X *satisfies* $\theta \in \mathcal{B}^+(X)$ iff the truth assignment that assigns $\mathtt{true}$ to the elements in Y and $\mathtt{false}$ to the elements of $X \setminus Y$ satisfies θ. The set Y *exactly satisfies* θ if Y satisfies θ and every proper subset of Y does not satisfy θ. The dual $\widetilde{\theta}$ of formula θ is obtained from θ by exchanging $\vee$ with $\wedge$ and $\mathtt{true}$ with $\mathtt{false}$.

Visibly pushdown languages. A *pushdown alphabet* Σ is an alphabet which is partitioned in three disjoint finite alphabets Σ_c, Σ_r, and Σ_{int}, where Σ_c is a finite set of *calls*, Σ_r is a finite set of *returns*, and Σ_{int} is a finite set of *internal actions*.

A *Büchi nondeterministic visibly pushdown automaton* (Büchi *NVPA*) [4] on infinite words over a pushdown alphabet $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$ is a tuple $\mathcal{P} = \langle Q, q_0, \Gamma, \Delta, F \rangle$, where Q is a finite set of (control) states, $q_0 \in Q$ is the initial state, Γ is the finite stack alphabet, $\Delta \subseteq (Q \times \Sigma_c \times Q \times \Gamma) \cup (Q \times \Sigma_r \times (\Gamma \cup \{\bot\}) \times Q) \cup (Q \times \Sigma_{int} \times Q)$ is the transition relation (where $\bot \notin \Gamma$ is the special *stack bottom symbol*), and $F \subseteq Q$ is a Büchi condition over Q. A transition of the form $(q, a, q', B) \in Q \times \Sigma_c \times Q \times \Gamma$ is a push transition, where on reading the call a the symbol $B \neq \bot$ is pushed onto the stack and the control changes from q to q'. A transition of the form $(q, a, B, q') \in Q \times \Sigma_r \times (\Gamma \cup \{\bot\}) \times Q$ is a pop transition, where on reading the return a, B is popped from the stack and the control goes from q to q'. Finally, on reading an internal action a, $\mathcal{P}$ can choose only transitions of the form (q, a, q') which do not use the stack. Thus, $\mathcal{P}$ pushes onto the stack only on reading a call, pops the stack only at returns, and does not use the stack on internal actions. Hence, the input controls the kind of operations permissible on the stack, and thus the stack depth at every position [4].

A configuration of $\mathcal{P}$ is a pair (q, β), where $q \in Q$ and $\beta \in \Gamma^* \cdot \{\bot\}$ is a stack content. For $w \in \Sigma^\omega$, $w(i)$ denotes the i-th symbol of w. A run of $\mathcal{P}$ over w is an infinite sequence of configurations $r = (q'_0, \beta_0)(q'_1, \beta_1) \ldots$ such that $\beta_0 = \bot$, q'_0 is the initial state, and for each $i \geq 0$: [**push**] if $w(i)$ is a call, then $\exists B \in \Gamma$ such that $\beta_{i+1} = B \cdot \beta_i$ and $(q'_i, w(i), q'_{i+1}, B) \in \Delta$; [**pop**] if $w(i)$ is a return, then $\exists B \in \Gamma$ s.t. $(q'_i, w(i), B, q'_{i+1}) \in \Delta$ and *either* $\beta_i = \beta_{i+1} = B = \bot$, or $B \neq \bot$ and $\beta_i = B \cdot \beta_{i+1}$; [**internal**] if $w(i)$ is an internal action, $(q'_i, w(i), q'_{i+1}) \in \Delta$ and $\beta_i = \beta_{i+1}$. The run is accepting iff its projection over Q satisfies the Büchi condition F. The language of $\mathcal{P}$, $\mathcal{L}(\mathcal{P})$, is the set of $w \in \Sigma^\omega$ s.t. there is an accepting run of $\mathcal{P}$ over w. A language $\mathcal{L}$ over Σ is a *visibly pushdown language* (*VPL*) if there is a Büchi *NVPA* $\mathcal{P}$ s.t. $\mathcal{L}(\mathcal{P}) = \mathcal{L}$.

In order to model formal verification problems of pushdown systems M using finite specifications (such as *NVPA*) denoting *VPL* languages, we choose a suitable pushdown alphabet $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_{int}$, and associate a symbol in Σ with each transition of M with the restriction that push transitions are mapped to Σ_c, pop transitions are mapped to Σ_r, and transitions that do not use the stack are mapped to Σ_{int}. Note that M equipped with such a labelling is a Büchi *NVPA* where all the states are accepting. The specification S describes another *VPL* $\mathcal{L}(S)$ over the same alphabet, and M is correct iff $\mathcal{L}(M) \subseteq \mathcal{L}(S)$. Given a class $\mathcal{C}$ of specifications describing *VPL* over Σ, the *pushdown model checking problem against* $\mathcal{C}$*-specifications* is to decide, given a pushdown system M over Σ and a specification S in the class $\mathcal{C}$, whether $\mathcal{L}(M) \subseteq \mathcal{L}(S)$.

3 Alternating Visibly Pushdown Automata

In this section we study the class of *alternating visibly pushdown automata* (*AVPA*). We show that any parity *AVPA* $\mathcal{P}$ can be translated into an equivalent Büchi *NVPA* whose size is doubly exponential in the size of $\mathcal{P}$. This double-exponential blowup cannot be avoided. In fact we show that emptiness for this class of automata is 2EXPTIME-complete (recall that emptiness for parity or Büchi *NVPA* is in PTIME [4]).

As *NVPA*, an *AVPA* $\mathcal{P}$ pushes onto (resp., pops) the stack only when it reads a call (resp., a return), and does not use the stack on internal actions. However, at any instant $\mathcal{P}$ can choose nondeterministically to split in *many* copies, each of them moving to the next input symbol. Formally, a parity *AVPA* over a pushdown alphabet Σ is a tuple $\mathcal{P} = \langle Q, q_0, \Gamma, \delta, \Omega \rangle$, where Q, q_0, and Γ are defined as for *NVPA*, Ω is a parity condition over Q, and $\delta : Q \times \Sigma \times (\Gamma \cup \{\bot\}) \to \mathcal{B}^+(Q) \cup \mathcal{B}^+(Q \times \Gamma)$ is the transition function satisfying: (i) $\delta(q, a, B) \in \mathcal{B}^+(Q)$ if a is *not* a call, (ii) $\delta(q, a, B) \in \mathcal{B}^+(Q \times \Gamma)$ if a is a call, and (3) $\delta(q, a, B) = \delta(q, a, B')$ if a is *not* a return.

Given a word $w \in \Sigma^* \cup \Sigma^\omega$, a state q, and stack content $\beta \in \Gamma^* \cdot \{\bot\}$, a (q, β)-*run of* $\mathcal{P}$ *over* w is a $Q \times \Gamma^* \cdot \{\bot\}$-labelled tree $r = \langle T, V \rangle$, where each node x of T labelled by (q', β') describes a copy of $\mathcal{P}$ in state q' and stack content β' reading the symbol $w(|x|)$. Moreover, we require that $V(\varepsilon) = (q, \beta)$ and for each $x \in T$ with $V(x) = (q', B \cdot \beta')$, there is a set $H = \{p_0, \ldots, p_m\}$ exactly satisfying $\delta(q', w(|x|), B)$ (note that $H \subseteq Q \times \Gamma$ if $w(|x|)$ is a call, and $H \subseteq Q$ otherwise) such that x has children $x \cdot 0, \ldots, x \cdot m$ and for all $0 \leq i \leq m$, the following holds: [**Push**] If $w(|x|)$ is a call, $p_i = (q_i, B_i)$ and $V(x \cdot i) = (q_i, B_i \cdot B \cdot \beta')$; [**Pop**] If $w(|x|)$ is a return, $V(x \cdot i) = (p_i, \beta')$ if $B \neq \top$, and $V(x \cdot i) = (p_i, \bot)$ otherwise; [**Internal**] If $w(|x|)$ is an internal action, $V(x \cdot i) = (p_i, B \cdot \beta')$. The run $r = \langle T, V \rangle$ is *memoryless* if for all $x_1, x_2 \in T$ such that $|x_1| = |x_2|$ and $V(x_1) = V(x_2)$, the subtrees of r rooted at x_1 and x_2 coincide (i.e., fixed a position along w, the behaviour of $\mathcal{P}$ depends only on the current state and stack content, and is independent on the past choices). If w is infinite, the run r is *accepting* if for each infinite path $\pi = x_0 x_1 \dots$ of r, the projection of $V(x_0)V(x_1)\dots$ over Q satisfies the parity condition Ω. The ω-language $\mathcal{L}(\mathcal{P})$ of $\mathcal{P}$ is the set of $w \in \Sigma^\omega$ such that there is an accepting $(q_0, \bot)$-run of $\mathcal{P}$ over w.

Remark 1. For $w \in \Sigma^\omega$, we can associate in a standard way [12] to $\mathcal{P}$ and w an infinite-state parity game, where player 0 plays for acceptance, while player 1 plays for rejection. Winning strategies of player 0 correspond to accepting runs of $\mathcal{P}$ over w. Since the existence of a winning strategy in parity games implies the existence of a memoryless one, we can restrict ourselves to consider only memoryless runs of $\mathcal{P}$. Moreover, by [12] the *dual automaton* (*AVPA*) $\widetilde{\mathcal{P}} = \langle Q, q_0, \Gamma, \widetilde{\delta}, \widetilde{\Omega} \rangle$ of $\mathcal{P}$, where $\widetilde{\delta}(q, a, B)$ is the dual of $\delta(q, a, B)$ and $\widetilde{\Omega}(q) = \Omega(q) + 1$ for all $q \in Q$, accepts the complement of $\mathcal{L}(\mathcal{P})$.

From parity *AVPA* to Büchi *NVPA*. Fix a parity *AVPA* $\mathcal{P} = \langle Q, q_0, \Gamma, \delta, \Omega \rangle$ over Σ. Let n be the index of Ω and $[n] = \{0, \dots, n\}$. W.l.o.g. assume that for each $q \in Q$, $0 \leq \Omega(q) \leq n$ and $\delta(q, a, B) \notin \{\texttt{true}, \texttt{false}\}$. We will construct a Büchi *NVPA* accepting $\mathcal{L}(\mathcal{P})$. Our approach is a generalization of the technique of "summaries" used in [4] to show that *NVPA* are closed under complementation.

A finite word $w \in \Sigma^*$ is *well-matched* if inductively or (1) $w = \epsilon$, or (2) $w = aw'$, $a \in \Sigma_{int}$ and w' is well-matched, or (3) $w = a_c w' a_r w''$, $a_c \in \Sigma_c$, $a_r \in \Sigma_r$, and w' and w'' are well-matched. The set L_{mwm} of *minimally well-matched words* is the set of words of the form $a_c w a_r$ where $a_c \in \Sigma_c$, $a_r \in \Sigma_r$, and w is well-matched.

Let $P \subseteq Q \times [n]$. For a *finite* run $r = \langle T, V \rangle$ of $\mathcal{P}$ over $w \in \Sigma^*$, r is *consistent with* P if for each $(q, i) \in Q \times [n]$: $(q, i) \in P$ iff there is a path $\pi = x_0 \dots x_k$ (note that $k = |w|$) of r with $V(x_j) = (q_j, \beta_j)$ for all $0 \leq j \leq k$ such that $q_k = q$ and $i = min\{\Omega(q_j)\}_{0 \leq j \leq k}$.

Let $\mathcal{H} = Q \times 2^{Q \times [n]}$. A *summary* for a well-matched word $w \in \Sigma^*$ is a nonempty set $\mathcal{S} \subseteq \mathcal{H}$ such that for all $(q, P) \in \mathcal{S}$, there is a memoryless (q, β)-run of $\mathcal{P}$ over w for some stack content β which is consistent with P. Intuitively, each $(q, P) \in \mathcal{S}$ keeps track of the meaningful information associated with some (q, β)-run r of $\mathcal{P}$ over w; in particular, for each path π of r, P keeps track of the smallest priority of the states visited by π and of the last state along π (since w is well-matched, the stack content associated with such a state is still β and the initial stack content β is not modified along the run r). Given a memoryless run r of $\mathcal{A}$ over an *infinite* word w', the notion of summary is used to capture the meaningful information of the portions of r associated with well-matched subwords w of w'. Note that for such a subword w, fixed a state q, there can be different portions of r associated with w corresponding to finite (q, β)-runs of $\mathcal{P}$ over w whose initial stack contents are distinct. Since the local choices of $\mathcal{P}$ depend also on the stack content, a summary must keep track for each state q of distinct sets $P_1, \dots, P_n \subseteq Q \times [n]$, which are consistent with different finite (q, β)-runs over w.

For $w \in \Sigma^\omega$, there is a unique factorization $w_1 w_2 \dots$ of w such that for all $i \geq 1$, $w_i \in \Sigma \cup L_{mwm}$ and if w_i is a call, then there is no $j > i$ such that w_j is a return. Let $\widehat{\Sigma}$ be the pushdown alphabet given by $\Sigma \cup (2^{\mathcal{H}} \setminus \{\emptyset\})$, where symbols in $2^{\mathcal{H}} \setminus \{\emptyset\}$ are internal actions. A *pseudo-word* $\widehat{w}$ is an infinite word over $\widehat{\Sigma}$ s.t. if $\widehat{w}(i)$ is a call, then there is no $j > i$ such that $\widehat{w}(j)$ is a return. Given $w \in \Sigma^\omega$, a *pseudo-word of*

w is obtained by replacing each factor $w_i \in L_{mwm}$ in the factorization of w with a summary of w_i.

Let us consider the *AVPA* $\widehat{\mathcal{P}} = \langle Q \times \{1, \ldots, n\}, (q_0, \Omega(q_0)), \Gamma, \widehat{\delta}, \widehat{\Omega}\rangle$ over $\widehat{\Sigma}$ where (1) for all $a \in \Sigma$, $\widehat{\delta}((q,i), a, B)$ is obtained from $\delta(q, a, B)$ by replacing each $q' \in Q$ with $(q', \Omega(q'))$, (2) for all $\mathcal{S} \in 2^{\mathcal{H}} \setminus \{\emptyset\}$, $\widehat{\delta}((q,i), \mathcal{S}, B) = \bigvee_{(q,P)\in\mathcal{S}} \bigwedge_{(p,h)\in P} (p,h)$, and (3) $\widehat{\Omega}((q,i)) = i$. For a word $\widehat{w} \in \widehat{\Sigma}^\omega$, $\widehat{\mathcal{P}}$ simulates $\mathcal{P}$ over the Σ-letters of $\widehat{w}$, and on letters $\mathcal{S} \in 2^{\mathcal{H}} \setminus \{\emptyset\}$ it updates the current state q by splitting in n copies in states $(q_1, i_1), \ldots, (q_n, i_n)$, respectively, such that $(q, \{(q_1, i_1), \ldots, (q_n, i_n)\}) \in \mathcal{S}$. By construction for every $w \in \Sigma^\omega$, $\mathcal{P}$ accepts w iff there is a pseudo-word of w that is accepted by $\widehat{\mathcal{P}}$. Note that for a $((q_0, \Omega(q_0)), \bot)$-run of $\widehat{\mathcal{P}}$ over a pseudo-word, whenever a return occurs, the current stack content is empty. Hence, we can construct a parity alternating *finite-state* automaton $\mathcal{A}$ that simulates $\widehat{\mathcal{P}}$ over pseudo-words and accepts only pseudo-words (on reading a return, $\mathcal{A}$ simulates $\widehat{\mathcal{P}}$ when the stack is empty). By [13,15] one can construct a nondeterministic Büchi finite-state automaton $\mathcal{A}_{PW}$ with a number of states exponential in $|Q|$ that accepts $\mathcal{L}(\mathcal{A})$. Thus, we obtain the following:

Proposition 1. *One can construct a nondeterministic Büchi finite-state automaton $\mathcal{A}_{PW}$ over $\widehat{\Sigma}$ whose number of states is exponential in the number of states of $\mathcal{P}$ such that for each $w \in \Sigma^\omega$: $w \in \mathcal{L}(\mathcal{P})$ iff there is a pseudo-word of w that is accepted by $\mathcal{A}_{PW}$.*

The next step consists of computing the summary information associated with minimally well-matched words.

Proposition 2. *We can build in* doubly exponential *time a* NVPA $\mathcal{P}_S$ *on* finite *words over Σ with a special stack symbol '$-$' and set of states containing $2^{\mathcal{H}}$ such that for each $w \in \Sigma^*$, $\mathcal{S} \in 2^{\mathcal{H}} \setminus \{\emptyset\}$, and stack content $\beta \in \{-\}^*.\bot$: there is an accepting $(\mathcal{S}, \beta)$-run of $\mathcal{P}_S$ over w* iff *$w \in L_{mwm}$ and $\mathcal{S}$ is a summary of w. $\mathcal{P}_S$ has a unique accepting state $q_{acc} \notin 2^{\mathcal{H}}$ (with no outgoing transitions).*

Now, we are ready to construct a Büchi *NVPA* $\mathcal{P}_N$ accepting exactly $\mathcal{L}(\mathcal{P})$. Let $\mathcal{A}_{PW}$ be the Büchi nondeterministic finite-state automaton of Proposition 1 with states Q_{PW}, and let $\mathcal{P}_S$ be the *NVPA* of Proposition 2 with states $Q_S \supseteq 2^{\mathcal{H}}$, accepting state $q_{acc} \notin 2^{\mathcal{H}}$, and special stack symbol '$-$'. Given a word $w \in \Sigma^\omega$, $\mathcal{P}_N$ guesses a pseudo word $\widehat{w}$ of w and checks that it is in $\mathcal{L}(\mathcal{A}_{PW})$. The set of states of $\mathcal{P}_N$ is $Q_{PW} \times Q_S$. At any step, $\mathcal{P}_N$ either simulates $\mathcal{A}_{PW}$ on the first component of the state (and the other one remains constant with value q_{acc}) pushing onto the stack only the symbol '$-$', or simulates $\mathcal{P}_S$ on the second component (and the other one remains constant). Whenever a call a_c occurs and $\mathcal{P}_N$ is in state (q^1_{PW}, q_{acc}), $\mathcal{P}_N$ can guess that a_c is the first letter of a factor $w_m \in L_{mwm}$ of w and there is a summary $\mathcal{S} \in 2^{\mathcal{H}}$ of w_m by simulating a push move of $\mathcal{P}_S$ with source state $\mathcal{S}$, input symbol a_c, and target q_S, and moving to state (q^2_{PW}, q_S) s.t. $q^1_{PW} \xrightarrow{\mathcal{S}} q^2_{PW}$ is a transition of $\mathcal{A}_{PW}$ (by Proposition 2, $q_S \neq q_{acc}$). From (q^2_{PW}, q_S), $\mathcal{P}_N$ simulates $\mathcal{P}_S$. If the guess was correct, then $\mathcal{P}_N$ can reach a state of the form (q^2_{PW}, q_{acc}), and the whole procedure is repeated. Otherwise, $\mathcal{P}_N$ will continue to simulate $\mathcal{P}_S$ visiting only states of the form (q, q') with $q' \neq q_{acc}$. Therefore, the accepting states of $\mathcal{P}_N$ are of the form (q, q_{acc}) where q is an accepting state of $\mathcal{A}_{PW}$. Thus, we obtain the following:

Theorem 1. *Given a parity* AVPA $\mathcal{P}$*, one can construct in* doubly exponential *time a Büchi* NVPA $\mathcal{P}_N$ *with size doubly exponential in the size of* $\mathcal{P}$ *s.t.* $\mathcal{L}(\mathcal{P}_N) = \mathcal{L}(\mathcal{P})$.

***Decision problems for (parity)* AVPA.** First, we consider emptiness and universality. Note that these problems are equivalent from the complexity point of view since the dual automaton of a *AVPA* $\mathcal{P}$ has size linear in the size of $\mathcal{P}$. Since emptiness of Büchi *NVPA* is in PTIME, by Theorem 1, emptiness of *AVPA* is in 2EXPTIME. We can show that the problem is also 2EXPTIME-hard (also for Büchi *AVPA*) by a reduction from the word problem for EXPSPACE-bounded Turing Machines. For the pushdown model checking problem, given a pushdown system M and an *AVPA* $\mathcal{P}$, checking whether $\mathcal{L}(M) \subseteq \mathcal{L}(\mathcal{P})$ reduces to checking emptiness of $\mathcal{L}(M) \cap \mathcal{L}(\widetilde{\mathcal{P}}_N)$, where $\mathcal{P}_N$ is the *NVPA* equivalent to the dual of $\mathcal{P}$. By [4] and Theorem 1 this check can be done in time polynomial in the size of M and doubly exponential in the size of $\mathcal{P}$. The problem is at least as hard as universality of *AVPA*. Thus, we obtain the following:

Theorem 2. *Emptiness and universality of* (*parity or Büchi*) AVPA *are* 2EXPTIME-*complete. Moreover, the pushdown model checking problem against* AVPA *specifications is* 2EXPTIME-*complete (and polynomial in the size of the pushdown system).*

4 Alternating Jump Finite-State Automata

In this section we introduce the class of *alternating jump* (*finite-state*) *automata* (*AJA*) operating on infinite words over a pushdown alphabet. *AJA* extend standard alternating finite-state automata by also allowing non-local moves: when the current input symbol is a call a_c and the *matching return* a_r of a_c along the input word exists, a copy of the automaton can move (jump) to the return a_r. We also allow local and non-local backward moves (details are given below). We show that one-way and two-way *AJA* have the same expressiveness and capture exactly the class of visibly pushdown languages. The main result is an algorithm for translating a given parity two-way *AJA* (*2-AJA*) $\mathcal{A}$ into an equivalent Büchi *NVPA* whose size is *singly* exponential in the size of $\mathcal{A}$. We also study some decision problems for the considered class of automata.

Fix a pushdown alphabet Σ. Given an infinite word $w \in \Sigma^\omega$, we consider four different notions of successor for a position $i \in \mathbb{N}$ along w:

– The *forward local successor of* i *along* w, written $succ(\downarrow, w, i)$, is $i + 1$.
– The *backward local successor of* i *along* w, written $succ(\uparrow, w, i)$, is $i - 1$ if $i > 0$, and it is undefined otherwise (in this case we set $succ(\uparrow, w, i) = \top$).
– The *abstract successor of* i *along* w [3], written $succ(\downarrow_{\mathsf{a}}, w, i)$, is the forward local successor if i is not a call position. If instead $w(i)$ is a call, $succ(\downarrow_{\mathsf{a}}, w, i)$ points to the *matching return position* of i (if any), i.e.: if there is $j > i$ such that $w(j)$ is a return and $w(i + 1) \dots w(j - 1)$ is well-matched, then $succ(\downarrow_{\mathsf{a}}, w, i) = j$ (note that j is uniquely determined), otherwise $succ(\downarrow_{\mathsf{a}}, w, i) = \top$.
– The *caller of* i *along* w [3], written $succ(\uparrow^{\mathsf{c}}, w, i)$, points to the last unmatched call of the prefix of w until position i. Formally, if there is $j < i$ such that $w(j)$ is a call and $w(j + 1) \dots w(h)$ is well-matched (where $h = i - 1$ if i is a call position, and $h = i$ otherwise), then $succ(\uparrow^{\mathsf{c}}, w, i) = j$ (note that j is uniquely determined), otherwise the caller is undefined and we set $succ(\uparrow^{\mathsf{c}}, w, i) = \top$.

Let $DIR = \{\downarrow, \downarrow_{\mathsf{a}}, \uparrow, \uparrow^{\mathsf{c}}\}$. A *parity 2-AJA* over Σ is a tuple $\mathcal{A} = \langle Q, q_0, \delta, \Omega \rangle$, where Q, q_0, and Ω are defined as for parity *AVPA* and $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(DIR \times Q \times Q)$ is the transition function. Intuitively, a target of a move of $\mathcal{A}$ is encoded by a triple $(dir, q, q') \in DIR \times Q \times Q$, meaning that a copy of $\mathcal{A}$ moves to the dir-successor of the current input position i in state q if such a successor is defined, and to position $i+1$ in state q' otherwise. Note that the q'-component of the triple above is irrelevant if $dir = \downarrow$ (we give it only to have a uniform notation). A *1-AJA* is a *2-AJA* whose transition function δ satisfies $\delta(q, a) \in \mathcal{B}^+(\{\downarrow, \downarrow_{\mathsf{a}}\} \times Q \times Q)$ for each $(q, a) \in Q \times \Sigma$.

A q-run of $\mathcal{A}$ over an infinite word $w \in \Sigma^\omega$ is a $\mathbb{N} \times Q$-labelled tree $r = \langle T, V \rangle$, where a node $x \in T$ labelled by (i, q') describes a copy of $\mathcal{A}$ that is in q' and reads the i-th input symbol. Moreover, we require that $r(\varepsilon) = (0, q)$ and for all $x \in T$ with $r(x) = (i, q')$, there is a set $H = \{(dir_0, q'_0, q''_0), \ldots, (dir_m, q'_m, q''_m)\} \subseteq DIR \times Q \times Q$ exactly satisfying $\delta(q', w(i))$ such that the children of x are $x \cdot 0, \ldots, x \cdot m$, and for each $0 \leq h \leq m$: $V(x \cdot h) = (i+1, q''_h)$ if $succ(dir_h, w, i) = \top$, and $V(x \cdot h) = (succ(dir_h, w, i), q'_h)$ otherwise. The run r is *memoryless* if for all nodes $x, y \in T$ such that $V(x) = V(y)$, the (labelled) subtrees rooted at x and y coincide. The q-run r is *accepting* if for each infinite path $x_0 x_1 \ldots$, the projection over Q of $V(x_0)V(x_1)\ldots$ satisfies the parity condition Ω. The ω-language of $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$, is the set of words $w \in \Sigma^\omega$ such that there is an accepting q_0-run r of $\mathcal{A}$ over w. A q_0-run is called simply run.

As for *AVPA*, we can give a standard game-theoretic interpretation of acceptance in *AJA*. In particular, by [12] the *dual automaton* $\widetilde{\mathcal{A}} = \langle Q, q_0, \widetilde{\delta}, \widetilde{\Omega} \rangle$ of an *AJA* $\mathcal{A}$, where for all $(q, a) \in Q \times \Sigma$, $\widetilde{\Omega}(q) = \Omega(q) + 1$ and $\widetilde{\delta}(q, a)$ is the dual of $\delta(q, a)$, accepts the complement of $\mathcal{L}(\mathcal{A})$. Moreover, the existence of an accepting run of $\mathcal{A}$ over w implies the existence of a memoryless one. Since *AJA* are clearly closed under intersection and union, we obtain the following result.

Proposition 3. 2-AJA *and* 1-AJA *are closed under boolean operations. Moreover, given a* 2-AJA $\mathcal{A}$*,* $w \in \mathcal{L}(\mathcal{A})$ *iff there is an accepting memoryless run of* $\mathcal{A}$ *over* w*.*

4.1 Relation Between Nondeterministic Visibly Pushdown Automata and *AJA*

In this subsection we present translations forth and back between *NVPA* and *2-AJA*.

***From parity* 2-AJA *to Büchi* NVPA.** Fix a parity *2-AJA* $\mathcal{A} = \langle Q, q_0, \delta, \Omega \rangle$ over Σ. Let n be the index of Ω and $[n] = \{0, \ldots, n\}$. Without loss of generality we can assume that for each $q \in Q$, $0 \leq \Omega(q) \leq n$ and $\delta(q, a) \notin \{\mathtt{true}, \mathtt{false}\}$.

By Proposition 3, we can restrict ourselves to consider *memoryless* runs of $\mathcal{A}$. For a word $w \in \Sigma^\omega$, we represent memoryless runs of $\mathcal{A}$ over w as follows. For a set $H \subseteq Q \times DIR \times Q \times Q$, let $Dom(H) = \{q \in Q \mid (q, dir, q', q'') \in H\}$. A *strategy of* $\mathcal{A}$ *over* w is a mapping $\mathsf{St} : \mathbb{N} \rightarrow 2^{Q \times DIR \times Q \times Q}$ satisfying: (i) $q_0 \in Dom(\mathsf{St}(0))$, (ii) for each $i \in \mathbb{N}$ and $q \in Dom(\mathsf{St}(i))$, the set $\{(dir, q', q'') \mid (q, dir, q', q'') \in \mathsf{St}(i)\}$ exactly satisfies $\delta(q, w(i))$, and (iii) for each $(q, dir, q', q'') \in \mathsf{St}(i)$, $q' \in Dom(\mathsf{St}(h))$ if $succ(dir, w, i) = h \neq \top$, and $q'' \in Dom(\mathsf{St}(i+1))$ otherwise. Intuitively, St is an infinite word over $2^{Q \times DIR \times Q \times Q}$ encoding a memoryless run r of $\mathcal{A}$ over w. In particular, $Dom(\mathsf{St}(i))$ is the set of states in which the automaton is (along r) when $w(i)$ is read, and for $q \in Dom(\mathsf{St}(i))$, the set $\{(dir, q', q'') \mid (q, dir, q', q'') \in \mathsf{St}(i)\}$ is the set of choices made by $\mathcal{A}$ on reading $w(i)$ in state q.

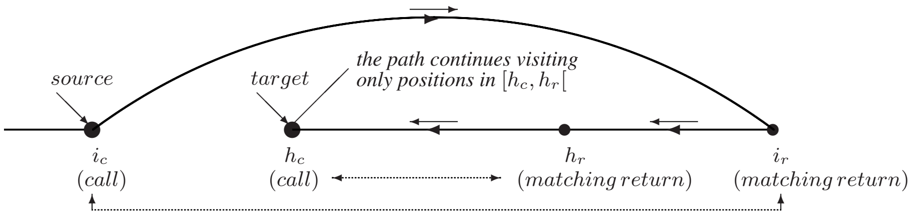

Fig. 1. Structure of a zig-zag prefix

A j-path γ of St is a sequence of pairs in $\mathbb{N} \times Q$ of the form $\gamma = (i_1, q_1)(i_2, q_2)\ldots$ where $i_1 = j$ and for each $1 \leq h < |\gamma|$: $\exists(q_h, dir, q'_h, q''_h) \in \mathsf{St}(i_h)$ such that *either* $i_{h+1} = succ(dir, w, i_h)$ and $q_{h+1} = q'_h$, *or* $succ(dir, w, i_h) = \top$, $i_{h+1} = i_h + 1$ and $q_{h+1} = q''_h$. The path γ is *forward* (resp., *backward*) if $i_{h+1} > i_h$ (resp., $i_{h+1} < i_h$) for any h. For a finite path $\gamma = (i_1, q_1) \ldots (i_k, q_k)$, the *index of* γ is $min\{\Omega(q_l) \mid 1 \leq l \leq k\}$ if $k > 1$, and $n + 1$ otherwise. In case $i_k = i_1$, we say that γ is *closed*. For a finite path $\gamma_1 = (i_1, q_1), \ldots, (i_k, q_k)$ and a path $\gamma_2 = (i_k, q_k)(i_{k+1}, q_{k+1})\ldots$, let $\gamma_1 \circ \gamma_2$ be the path $\gamma_1(i_{k+1}, q_{k+1})\ldots$. A path γ is *positive* (resp., *negative*) if γ is of the form $\gamma = \gamma_1 \circ \gamma_2 \circ \ldots$, where each γ_i is either a closed path, or a forward (resp., backward) path. A *i-cycle* of St is an infinite path γ that can be decomposed in the form $\gamma = \gamma_1 \circ \gamma_2 \circ \ldots$, where each γ_h is a closed i-path.

The strategy St is *accepting* if for each infinite path γ starting from $(0, q_0)$, the projection of γ over Q satisfies the parity condition Ω of $\mathcal{A}$. Clearly, there is an accepting memoryless run of $\mathcal{A}$ over w iff there is an accepting strategy of $\mathcal{A}$ over w.

Now, we have to face the problem that the infinite paths γ of St can use backward moves. If γ is *positive*, then the idea is to collapse the closed subpaths (in the decomposition of γ) in a unique move and to keep track of the associated meaningful information by finite local auxiliary structures. However, there can be infinite paths that are not positive. Fortunately, also for this class of paths, it is possible to individuate a decomposition that is convenient for our purposes. This decomposition is formalized as follows. A *zig-zag i-path* is an infinite i-path γ inductively defined as follows:

- (**Positive prefix**) $\gamma = \gamma_p \circ \gamma_s$, where γ_p is a finite *non-empty* positive i-path leading to position $h \geq i$, and γ_s is a zig-zag h-path visiting only positions in $[h, \infty[$.
- (**Zig-zag prefix**) $w(i)$ is a call and $\gamma = \gamma_{i,i_r} \circ \gamma_{i_r,h_r} \circ \gamma_{h_r,h_c} \circ \gamma_s$, where γ_{i,i_r} is a path from i to the matching return position i_r, γ_{i_r,h_r} is a *negative* path from i_r to the return position $h_r \in]i, i_r[$, γ_{h_r,h_c} is a path from h_r to the matching-call position $h_c \in]i, i_r[$, and γ_s is a zig-zag h_c-path visiting only positions in $[h_c, h_r[$.
- (**terminal Zig-zag**): $\gamma = \gamma_p \circ \gamma_s$, where γ_s is a h-cycle, and *either* γ_p is empty and $h = i$, *or* $\gamma_p = \gamma_{i,i_r} \circ \gamma_{i_r,h}$, where: γ_{i,i_r} is a path from the call position i to the matching return position i_r, and $\gamma_{i_r,h}$ is a negative path from i_r to $h \in]i, i_r[$.

The structure of a zig-zag prefix is illustrated in Figure 1. Note that an infinite positive path is a special case of zig-zag path. The following holds.

Proposition 4. *An infinite i-path visiting only positions in $[i, \infty[$ is a* zig-zag i-path.

Our next goal is to keep track locally for each position i of the meaningful information associated with closed i-paths. In case $w(i)$ is a return with matching-call $w(i_c)$, we also need to keep track in a finite way of the i_c-paths (resp., i-paths) leading to i (resp., i_c). Thus, we give the following definition. A *path summary of* $\mathcal{A}$ is a triple of mappings $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ where $\Delta : \mathbb{N} \to 2^{Q\times[n+1]\times Q}$ and $\Delta_{\mathsf{a}}, \Delta_{\mathsf{c}} : \mathbb{N} \times \{\mathsf{down}, \mathsf{up}\} \to 2^{Q\times[n+1]\times Q}$. Such a triple is a *path summary of the strategy* St if it satisfies some *closure conditions*. Since there are many conditions, we do not formalize them here. We only give their general form, to show (as we will see) that they can be checked by an *NVPA* (using the stack): each condition has the form $check(i, h) \implies \Phi(i, h)$, where (1) $i, h \in \mathbb{N}$ are implicitly universally quantified, (3) $check(i, h) := succ(dir, w, i) = h \mid succ(dir, w, i) = \top \wedge h = i+1$ (where $dir \in DIR$). (2) $\Phi(i, h)$ is a boolean formula over propositions of the form $(q, dir, q', q'') \in \mathsf{St}(j)$ or $(q, in, q') \in \mathcal{F}(j)$, where $j \in \{i, h\}$, and $\mathcal{F}(j) \in \{\Delta(j), \Delta_{\mathsf{a}}(j, \mathsf{down}), \Delta_{\mathsf{a}}(j, \mathsf{up}), \Delta_{\mathsf{c}}(j, \mathsf{down}), \Delta_{\mathsf{c}}(j, \mathsf{up})\}$.

The *intended meaning* for a path summary $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ of strategy St is as follows:

– $(q, in, q') \in \Delta(i)$ **if** there is a closed i-path of St of index in from (i, q) to (i, q').
– $(q, in, q') \in \Delta_{\mathsf{a}}(i, \mathsf{up})$ (resp., $(q, in, q') \in \Delta_{\mathsf{a}}(i, \mathsf{down})$) **if** $i = succ(\downarrow_{\mathsf{a}}, w, h)$, $w(h)$ is a call, and there is a path of index in from (i, q) to (h, q') (resp., from (h, q) to (i, q')).
– $(q, in, q') \in \Delta_{\mathsf{c}}(i, \mathsf{up})$ (resp., $(q, in, q') \in \Delta_{\mathsf{c}}(i, \mathsf{down})$) **if** $succ(\uparrow^{\mathsf{c}}, w, i) = h$ and there is a path of of index in from (i, q) to (h, q') (resp., from (h, q) to (i, q')).

By using the path summary $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ of St, we define a convenient representation for zig-zag paths which can be simulated by *NVPA*. A forward (resp., backward) move of St and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ is a pair $(i_1, q_1, in_1) \to (i_2, q_2, in_2)$ of triples in $\mathbb{N} \times Q \times [n+1]$ such that $i_2 > i_1$ (resp., $i_1 > i_2$) and: $\exists (q_1, in, q) \in \Delta(i_1)$ such that $(i_1, q)(i_2, q_2)$ is a path of St and $in_1 = min\{in, \Omega(q)\}$. A *downward path* ρ *of* St *and* $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ is a sequence $\rho = (i_1, q_1, in_1)(i_2, q_2, in_2)\ (i_3, q_3, in_3)\ldots$ inductively defined as follows:

– (**Forward move**) $(i_1, q_1, in_1) \to (i_2, q_2, in_2)$ is a forward move and (i_2, q_2, in_2) $(i_3, q_3, in_3)\ldots$ is a downward path visiting only positions in $[i_2, \infty[$.
– (**Zig-zag move**) $i_1 < i_2$, $w(i_1)$ and $w(i_2)$ are calls with matching-return positions h_1^r and h_2^r, there is a path $(h_1^r, q_1^r, in_1^r)\ldots(h_2^r, q_2^r, in_2^r)$ using only backward moves s.t. $(q_1, in, q_1^r) \in \Delta_{\mathsf{a}}(h_1^r, \mathsf{down})$ and $(q_2^r, in', q_2) \in \Delta_{\mathsf{a}}(h_2^r, \mathsf{up})$ for some $in, in' \in [n]$, and $\rho' = (i_2, q_2, in_2)\ (i_3, q_3, in_3)\ldots$ is a downward path visiting only positions in $[i_2, h_r^2[$.
– (**Terminal move**) $\rho = \rho'(i, q, in)(i, q', in')$, where $(q, in, q') \in \Delta(i)$, $(q', in', q') \in \Delta(i)$, and $in' \in [n]$. Moreover, either ρ' is empty or $\rho' = (i_c, q_c, in_c)$, i_c is a call position with matching-return i_r, $i \in]i_c, i_r[$, and there are $(q_c, in_c, q_r) \in \Delta_{\mathsf{a}}(i_r, \mathsf{down})$ and a path $(i_r, q_r, in_r)\ldots(i, q, in)$ using only backward moves.

Note that a downward path is either infinite (and uses only forward moves) or is finite and ends with a terminal move (corresponding to a cycle of St). Let Ω' be the parity condition over $Q \times [n+1]$ defined as $\Omega'((q, in)) = in$. A downward path ρ is *accepting* if either (i) ρ is infinite and its projection over $Q \times [n+1]$ satisfies Ω' or (ii) ρ is finite and leads to a triple (i, q, in) such that in is *even*. The path summary $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ of strategy St is *accepting* if each downward path starting from a triple of the form $(0, q_0, in)$ is accepting. The following holds.

Proposition 5. *For each $w \in \Sigma^\omega$, $w \in \mathcal{L}(\mathcal{A})$* iff *$\mathcal{A}$ has a strategy* St *over w and an accepting path summary of* St*.*

Let $\Sigma_{ext} = \Sigma \times (2^{Q\times DIR\times Q\times Q}) \times (2^{Q\times[n+1]\times Q})^5$, where the partition in calls, returns, and internal actions is induced by Σ. For $w \in \Sigma^\omega$, $\mathsf{St} \in (2^{Q\times DIR\times Q\times Q})^\omega$, and a path summary $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ of $\mathcal{A}$, the infinite word over Σ_{ext} associated with w, St, and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$, written $(w, \mathsf{St}, (\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}}))$, is defined in the obvious way.

Theorem 3. *Given a parity* 2-AJA *$\mathcal{A}$, one can construct in* singly exponential *time a Büchi* NVPA *$\mathcal{P}_{\mathcal{A}}$ with size exponential in the size of $\mathcal{A}$ such that $\mathcal{L}(\mathcal{P}_{\mathcal{A}}) = \mathcal{L}(\mathcal{A})$.*

Proof. We first build a Büchi *NVPA* $\mathcal{P}_{\mathcal{A}}^{ext}$ over Σ_{ext} that accepts $(w, \mathsf{St}, (\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}}))$ iff St is a strategy of $\mathcal{A}$ over w and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ is an *accepting* path summary of St. The desired automaton $\mathcal{P}_{\mathcal{A}}$ is obtained by projecting out the St and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ components of the input word. $\mathcal{P}_{\mathcal{A}}^{ext}$ is the intersection of two Büchi *NVPA* $\mathcal{P}_1^{ext}$ and $\mathcal{P}_2^{ext}$.

$\mathcal{P}_1^{ext}$ checks that St is a strategy of $\mathcal{A}$ over w and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ is a path summary of St. These checks can be easily done as follows. $\mathcal{P}_1^{ext}$ keeps track by its finite control of the caller (if any) of the current input symbol and of the previous input symbol. On reading a call c_{ext}, $\mathcal{P}_1^{ext}$ pushes onto the stack the current caller and the call c_{ext}, and moves to the next input symbol updating the caller information to c_{ext}. Thus, in case the call c_{ext} returns (and the stack is popped), $\mathcal{P}_1^{ext}$ can check that the constraints between c_{ext} and the matching return are satisfied. Moreover, on reading c_{ext}, $\mathcal{P}_1^{ext}$ *either* (i) guesses that the matching return of c_{ext} does *not* exist and pushes onto the stack *also* the symbol 'NO' (the guess is correct iff 'NO' will be never popped from the stack), *or* (ii) guesses that the matching return of c_{ext} exists, pushes onto the stack also the symbol 'YES' and moves to a non-accepting state (the guess is correct iff 'YES' is eventually popped). In the second case, before 'YES' is eventually popped, whenever a call c_{new} occurs, the behaviour of $\mathcal{P}_1^{ext}$ is deterministic since the matching return of c_{new} must exists if the guess was correct (in this phase the symbols 'NO' and 'YES' are not used).

$\mathcal{P}_2^{ext}$ checks that $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ is accepting. First we build a parity *NVPA* $\mathcal{P}_C$ that accepts $(w, \mathsf{St}, (\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}}))$ iff there is a downward path of St and $(\Delta, \Delta_{\mathsf{a}}, \Delta_{\mathsf{c}})$ that is *not* accepting. Essentially, $\mathcal{P}_C$ guesses such a path ρ and checks that it is not accepting. The computation of $\mathcal{P}_C$ is subdivided in phases. At the beginning of any phase, $\mathcal{P}_C$ keeps track by its finite control of the projection (q, in) of the current triple of the simulated path ρ, and chooses to start the simulation of a forward move, or of a zig-zag move, or of a terminal move. We describe the simulation of a zig-zag move (the other cases are simpler) with source (q, in, i) where i is a call position. Then, $\mathcal{P}_C$ guesses two triples $(q, in', q_r), (q_r^2, in'', q_c^2) \in Q \times [n] \times Q$. The first one must be in $\Delta_{\mathsf{a}}(h_r, \mathsf{down})$ where h_r is the matching return position of i, and the second one must be in $\Delta_{\mathsf{a}}(h_r^2, \mathsf{up})$, where $h_r^2 < h_r$ and h_r^2 is the matching return of a call representing the target of the zig-zag move. To check this $\mathcal{P}_C$ pushes onto the stack the triple (q, in', q_r) (when (q, in', q_r) is popped, $\mathcal{P}_C$ can check that the constraint on it is satisfied) and moves to the next input symbol in state (q_r^2, in'', q_c^2). $\mathcal{P}_C$ must also check that there is a finite path using only backward moves from the return position h_r in state q_r to the return h_r^2 in state q_r^2. This part is discussed below. If the simulated move is the first zig-zag move along ρ, $\mathcal{P}_C$ pushes onto the stack also a special symbol '*'. Note that after the first zig-zag move, for each call visited by the remaining portion of ρ, the matching return exists. Thus, $\mathcal{P}_C$

must check the existence of the matching return only for the call associated with the first zig-zag move (this reduces to check that '*' is eventually popped).

$\mathcal{P}_C$ will remain in state (q_r^2, in'', q_c^2) (pushing some special symbol onto the stack whenever a call occurs) until[2] on reading a call c_{ext}, $\mathcal{P}_C$ guesses that c_{ext} is the matching call of position h_r^2. Thus, $\mathcal{P}_C$ pushes onto the stack the tuple $((q_r^2, in'', q_c^2), START)$ (recall that from this point, in state q_c^2, the path ρ can visit only positions in $[i_2, h_r^2[$, where i_2 is the position of c_{ext}). When the tuple is popped, $\mathcal{P}_C$ checks that the constraint on (q_r^2, in'', q_c^2) is satisfied and starts to simulate in reversed order backward moves starting from state q_r^2, until (on reading a return) a triple (q, in', q_r) is popped from the stack. The zig-zag move have been correctly simulated iff the current state of the guessed backward path is exactly q_r. The size of $\mathcal{P}_C$ is quadratic in the number of states of $\mathcal{A}$. The Büchi *NVPA* $\mathcal{P}_2^{ext}$ is obtained by complementating $\mathcal{P}_C$. By [4] the size of $\mathcal{P}_2^{ext}$ is exponential in the number of states of $\mathcal{A}$. This concludes the proof. $\square$

***From Büchi* NVPA *to parity* 2-AJA.** For the converse translation, we can show that Büchi *1-AJA* are sufficient to capture the class of *VPL*.

Theorem 4. *Given a Büchi* NVPA $\mathcal{P}$*, we can build in polynomial time an equivalent Büchi* 1-AJA $\mathcal{A}_\mathcal{P}$ *whose number of states is quadratic in the number of states of* $\mathcal{P}$.

4.2 Decision Problems for Alternating Jump Automata

First, we consider emptiness and universality of (parity) *1-AJA* and *2-AJA*. Since the dual automaton of a *1-AJA* (resp., *2-AJA*) $\mathcal{A}$ has size linear in the size of $\mathcal{A}$ and accepts the complement of $\mathcal{L}(\mathcal{A})$, emptiness and universality of *1-AJA* and *2-AJA* are equivalent problems from the complexity point of view. For Büchi *NVPA*, emptiness is in PTIME and universality is EXPTIME-complete [4]. Thus, by Theorems 3 and 4, it follows that emptiness and universality of *1-AJA* and *2-AJA* are EXPTIME-complete.

For the pushdown model checking problem, given a pushdown system M and a *1-AJA* (resp., *2-AJA*) $\mathcal{A}$, checking whether $\mathcal{L}(M) \subseteq \mathcal{L}(\mathcal{A})$ reduces to checking emptiness of $\mathcal{L}(M) \cap \mathcal{L}(\widetilde{\mathcal{P}}_\mathcal{A})$, where $\widetilde{\mathcal{P}}_\mathcal{A}$ is the *NVPA* equivalent to the dual of $\mathcal{A}$ (Theorem 3). This check can be done in time polynomial in the size of M and exponential in the size of $\mathcal{A}$. The problem is at least as hard as universality of *1-AJA* (resp., *2-AJA*), hence:

Theorem 5. *Emptiness and universality of* 1-AJA *and* 2-AJA *are* EXPTIME-*complete. Moreover, the pushdown model checking problem against* 1-AJA *(resp.,* 2-AJA*) specifications is* EXPTIME-*complete (and polynomial in the size of the pushdown system).*

5 Visibly Pushdown Linear-Time μ-Calculus (*VP-μTL*)

In this section we consider a linear-time fixpoint calculus, called *VP-μTL*, which subsumes CARET [3] and captures exactly the class of visibly pushdown languages.

VP-μTL extends the full linear-time μ-calculus (with both forward and backward modalities) introduced in [14] by allowing non-local forward and backward modalities:

[2] If, in the meanwhile, a triple (q, in', q_r) is popped from the stack, $\mathcal{P}_C$ rejects the input.

the *abstract next modality* allows to associate each call with its matching return (if any), and the (backward) *caller modality* allows to associate each position with its caller (if any). Thus, *VP-μTL* formulas are built from atomic actions over a pushdown alphabet Σ using boolean connectives, the standard next temporal operator $\bigcirc$ (here, denoted by $\bigcirc^{\downarrow}$), the standard backward version $\bigcirc^{\uparrow}$ of $\bigcirc$, the abstract version $\bigcirc^{\downarrow_{\mathsf{a}}}$ of $\bigcirc$, the caller version $\bigcirc^{\uparrow^{\mathsf{c}}}$ of $\bigcirc$, as well as the least (μ) and greatest (ν) fixpoint operators.

W.l.o.g. we assume that *VP-μTL* formulas are written in positive normal form (negation only applied to atomic actions). Let $Var = \{X, Y, \ldots\}$ be a finite set of variables and Σ be a pushdown alphabet. *VP-μTL* formulas φ over Σ and Var are defined as:

$$\varphi ::= a \mid \neg a \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc^{dir}\varphi \mid \neg\bigcirc^{dir}\texttt{true} \mid \mu X.\varphi \mid \nu X.\varphi$$

where $a \in \Sigma$, $X \in Var$, and $dir \in \{\downarrow, \downarrow_{\mathsf{a}}, \uparrow, \uparrow^{\mathsf{c}}\}$. *VP-$\mu$TL* formulas φ are interpreted over words $w \in \Sigma^{\omega}$. Given a *valuation* $\mathcal{V} : Var \rightarrow 2^{\mathbb{N}}$ assigning a subset of $\mathbb{N}$ to each variable, the set of positions along w *satisfying* φ *under valuation* $\mathcal{V}$, written $\|\varphi\|^{w}_{\mathcal{V}}$, is defined as (we omit the clauses for atoms and boolean operators, which are standard):

$$\begin{array}{lcl}
\|X\|^{w}_{\mathcal{V}} & = & \mathcal{V}(X) \\
\|\bigcirc^{dir}\varphi\|^{w}_{\mathcal{V}} & = & \{i \in \mathbb{N} \mid succ(dir, w, i) \neq \top \text{ and } succ(dir, w, i) \in \|\varphi\|^{w}_{\mathcal{V}}\} \\
\|\neg\bigcirc^{dir}\texttt{true}\|^{w}_{\mathcal{V}} & = & \{i \in \mathbb{N} \mid succ(dir, w, i) = \top\} \\
\|\mu X.\varphi\|^{w}_{\mathcal{V}} & = & \bigcap\{M \subseteq \mathbb{N} \mid \|\varphi\|^{w}_{\mathcal{V}[X\mapsto M]} \subseteq M\} \\
\|\nu X.\varphi\|^{w}_{\mathcal{V}} & = & \bigcup\{M \subseteq \mathbb{N} \mid \|\varphi\|^{w}_{\mathcal{V}[X\mapsto M]} \supseteq M\}
\end{array}$$

where $\mathcal{V}[X \mapsto M]$ maps X to M and behaves like $\mathcal{V}$ on the other variables. If φ does not contain free variables (φ is a *sentence*), $\|\varphi\|^{w}_{\mathcal{V}}$ does not depend on the valuation $\mathcal{V}$, and we write $\|\varphi\|^{w}$. The set of models of a sentence φ is $\mathcal{L}(\varphi) = \{w \in \Sigma^{\omega} \mid 0 \in \|\varphi\|^{w}\}$.

We can show that parity *2-AJA* are exactly as expressive as *VP-μTL* sentences.

Theorem 6. *Given a* VP-μTL *sentence φ, one can construct in* linear *time a parity* 2-AJA $\mathcal{A}_{\varphi}$ *such that* $\mathcal{L}(\mathcal{A}_{\varphi}) = \mathcal{L}(\varphi)$. *Vice versa, give a parity* 2-AJA $\mathcal{A}$, *one can construct in* linear *time a* VP-μTL *sentence* $\varphi_{\mathcal{A}}$ *such that* $\mathcal{L}(\varphi_{\mathcal{A}}) = \mathcal{L}(\mathcal{A})$.

By Theorems 6 and 5, we obtain the following result.

Corollary 1. *The satisfiability problem of* VP-μTL *is* EXPTIME-*complete. Moreover, the pushdown model checking problem against* VP-μTL *is* EXPTIME-*complete (and polynomial in the size of the pushdown system).*

6 Conclusion

We have investigated various classes of alternating automata over infinite structured words which capture exactly the class of visibly pushdown languages (*VPL*) [4]. First, we have shown that basic decision problems for alternating visibly pushdown automata (*AVPA*) are 2EXPTIME-complete. Second, we have introduced a new class of alternating finite-state automata, namely the class of *2-AJA*, and we have shown that basic decision problems for *2-AJA* are EXPTIME-complete. Finally, *2-AJA* have been used to obtain exponential-time completeness for satisfiability and pushdown model checking of a linear-time fixpoint calculus with past modalities, called *VP-μTL*, which subsumes the linear-time μ-calculus and the logic CARET [3].

We believe that *2-AJA* represent an elegant and interesting formulation of the theory of *VPL* on infinite words. In particular, boolean operations are easy and basic decision problems are equivalent from the complexity point of view and EXPTIME-complete. Moreover, *2-AJA* represent an intuitive extension of standard alternating finite-state automata on infinite words. Finally, and more importantly, *2-AJA* make easy the temporal reasoning about the past, and linear-time context-free temporal logics with past modalities such as CARET and the fixpoint calculus *VP-μTL* can be easily and linearly translated into *2-AJA*. Note that we have not considered (one-way or two-way) *nondeterministic* jump automata, since we can show that they capture only a proper subclass of the class of *VPL* (we defer further details to the full version of this paper).

References

1. Alur, R., Chaudhuri, S., Madhusudan, P.: A fixpoint calculus for local and global program flows. In: Proc. 33rd POPL, pp. 153–165. ACM Press, New York (2006)
2. Alur, R., Chaudhuri, S., Madhusudan, P.: Languages of nested trees. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 329–342. Springer, Heidelberg (2006)
3. Alur, R., Etessami, K., Madhusudan, P.: A Temporal Logic of Nested Calls and Returns. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 467–481. Springer, Heidelberg (2004)
4. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proc. 36th STOC, pp. 202–211. ACM Press, New York (2004)
5. Alur, R., Madhusudan, P.: Adding nesting structure to words. In: Developments in Language Theory, pp. 1–13 (2006)
6. Ball, T., Rajamani, S.: Bebop: a symbolic model checker for boolean programs. In: Havelund, K., Penix, J., Visser, W. (eds.) SPIN Model Checking and Software Verification. LNCS, vol. 1885, pp. 113–130. Springer, Heidelberg (2000)
7. Bouajjani, A., Esparza, J., Maler, O.: Reachability Analysis of Pushdown Automata: Application to Model-Checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
8. Chatterjee, K., Ma, D., Majumdar, R., Zhao, T., Henzinger, T.A., Palsberg, J.: Stack size analysis for interrupt-driven programs. In: Cousot, R. (ed.) SAS 2003. LNCS, vol. 2694, pp. 109–126. Springer, Heidelberg (2003)
9. Chen, H., Wagner, D.: Mops: an infrastructure for examining security properties of software. In: Proc. 9th CCS, pp. 235–244. ACM Press, New York (2002)
10. Esparza, J., Kucera, A., Schwoon, S.: Model checking LTL with regular valuations for pushdown systems. Information and Computation 186(2), 355–376 (2003)
11. Loding, C., Madhusudan, P., Serre, O.: Visibly pushdown games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2004. LNCS, vol. 3328, pp. 408–420. Springer, Heidelberg (2004)
12. Muller, D.E., Schupp, P.E.: Alternating Automata on Infinite Trees. Theoretical Computer Science 54, 267–276 (1987)

13. Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the Theorems of Rabin, McNaughton and Safra. Theoretical Computer Science 141(1-2), 69–107 (1995)
14. Vardi, M.Y.: A temporal fixpoint calculus. In: Proc. 15th Annual POPL, pp. 250–259. ACM Press, New York (1988)
15. Vardi, M.Y.: Reasoning about the past with two-way automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)
16. Walukiewicz, I.: Pushdown processes: Games and Model Checking. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 62–74. Springer, Heidelberg (1996)