Logic for ω-Pushdown Automata

Manfred Droste, Sven Dziadek, Werner Kuich

Logic for ω-Pushdown Automata

Manfred Droste[a], Sven Dziadek[a,1], Werner Kuich[b,2]

[a] *Institut für Informatik, Universität Leipzig, Germany*
[b] *Institut für Diskrete Mathematik und Geometrie, Technische Unversität Wien, Austria*

Abstract

Context-free languages of infinite words have recently found increasing interest. Here, we will present a second-order logic with the same expressive power as Büchi or Muller pushdown automata for infinite words. This extends fundamental logical characterizations of Büchi, Elgot, Trakhtenbrot for regular languages of finite and infinite words and a more recent logical characterization of Lautemann, Schwentick and Thérien for context-free languages of finite words to ω-context-free languages. For our argument, we will investigate Greibach normal forms of ω-context-free grammars as well as a new type of Büchi pushdown automata which can alter their stack by at most one element and without ϵ-transitions. We show that they suffice to accept all ω-context-free languages. This enables us to use similar results recently developed for infinite nested words.

Keywords:
pushdown automata, ω-words, logic, nested-words, Greibach normal form

1. Introduction

The Büchi-Elgot-Trakhtenbrot Theorem [6, 14, 25] provided a seminal connection between automata and monadic second-order logic for finite words. It was extended to various other structures, like infinite words [7], finite trees [23], finite pictures [16], and finite and infinite nested words [2] and it

Email addresses: droste@informatik.uni-leipzig.de (Manfred Droste), dziadek@informatik.uni-leipzig.de (Sven Dziadek), werner.kuich@tuwien.ac.at (Werner Kuich)

[1] Supported by DFG Research Training Group 1763 (QuantLA)
[2] Partially supported by Austrian Science Fund (FWF): grant no. I1661 N25

also led to practical applications, e.g. in verification of finite-state programs (model checking, cf. [21, 4, 8]). Lautemann, Schwentick and Thérien [18] provided a Büchi-type description for context-free languages of finite words. This was extended to algebraic formal power series in [20], to (even weighted) higher-order pushdown automata in [26], and to weighted pushdown automata in [13].

Already Cohen and Gold [9] developed fundamental results for ω-context-free languages of infinite words, including several equivalent descriptions in terms of grammars, Büchi or Muller automata, and closures under Kleene-like ω-rational operations. The main goal of this paper is to provide a Büchi-type description for ω-context-free languages by a suitable logic, thereby extending Lautemann, Schwentick and Thérien's result to infinite words (and Büchi's result on ω-regular languages to ω-context-free languages). In our proof, we will employ a new type of ω-pushdown automata. These *simple* ω-pushdown automata do not allow ϵ-transitions and have only a very restricted access to the stack: they can only push one symbol, pop one symbol or ignore the stack. For finite words, such pushdown automata were utilized in [12]. Weighted simple reset pushdown automata for finite words were investigated in [10]. We believe that this model of simple pushdown automata can be of independent interest.

Recall that in formal language theory, grammars in Greibach normal forms are of basic importance for context-free languages of finite words. Here, we will first use a Kleene-type result of Cohen and Gold [9] to show that each ω-context-free language has a Büchi-accepting grammar in quadratic Greibach normal form. This enables us to show, as our first main new result, that each ω-context-free language can be accepted by a simple ω-pushdown automaton. A similar construction for context-free languages of finite words occurred within an argument of Blass and Gurevich [5].

Then we show that the languages of simple pushdown automata are, in a natural way, projections of visibly pushdown languages investigated by Alur and Madhusudan [1]. Now we can use their expressive equivalence result for visibly pushdown languages and monadic second-order logic to derive our second main result, the logical description of ω-context-free languages. Since our proof is constructive and the emptiness problem for ω-pushdown automata is decidable (cf. [19]), we can also decide the emptiness of simple ω-pushdown automata and therefore the satisfiability for our matching ω-logic.

Our paper is structured so that we first deal with simple pushdown au-

tomata and grammars in Greibach normal forms. Our language-theoretic results for these can be read independently from the definition of and results for our second-order logic which are developed afterwards.

2. ω-Pushdown Automata

Common definitions of ω-pushdown automata (cf. e.g., Cohen and Gold [9]) extend pushdown automata over finite words by a set of Muller- or Büchi-accepting final states. We do not directly work with this automaton definition because the equivalence proof for this automaton and the logic we will define in Section 4 is not easily possible.

Instead, we propose another automaton model, the simple ω-pushdown automaton. As in [12], we restrict the access to the stack to only allow either to keep the stack unaltered, to push one symbol or to pop one symbol. This will later allow us to employ a simple translation from nested word automata to our automaton model. We believe that this automaton model is also of independent interest.

For an alphabet Γ, let $\mathcal{S}(\Gamma) = (\{\downarrow\} \times \Gamma) \cup \{\#\} \cup (\{\uparrow\} \times \Gamma)$ be the set of *stack commands*.

Definition 1. *A* simple ω-pushdown automaton *(ωSPDA) denotes a 6-tuple* $M = (Q, \Sigma, \Gamma, T, q_0, F)$ *where*

- Q *is a finite set of states,*
- Σ *is a finite input alphabet,*
- Γ *is a finite stack alphabet,*
- $T \subseteq Q \times \Sigma \times Q \times \mathcal{S}(\Gamma)$ *is a set of transitions,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is a set of (Büchi-accepting) final states.*

A *configuration* of an ωSPDA $M = (Q, \Sigma, \Gamma, T, q_0, F)$ is a pair (q, γ), where $q \in Q$ and $\gamma \in \Gamma^*$. We define the transition relation between configurations as follows. Let $\gamma \in \Gamma^*$ and $t \in T$. If $t = (q, \sigma, q', (\downarrow, A))$, we let $(q, \gamma) \vdash_M^t (q', A\gamma)$. If $t = (q, \sigma, q', \#)$, we put $(q, \gamma) \vdash_M^t (q', \gamma)$. Finally, if $t = (q, \sigma, q', (\uparrow, A))$, we let $(q, A\gamma) \vdash_M^t (q', \gamma)$. These three types of transitions are called *push*, *internal* and *pop* transitions, respectively. Note that the stack here grows to the left.

We denote by $\text{state}(q, \sigma, q', s) = q$ the state and by $\text{label}(q, \sigma, q', s) = \sigma$ the *label* of a transition. Both will be extended to infinite sequence of transitions by letting $\text{state}((t_i)_{i\geq 0}) = (\text{state}(t_i))_{i\geq 0} \in Q^\omega$ for the infinite

sequence of states and $\text{label}((t_i)_{i\geq 0}) = (\text{label}(t_i))_{i\geq 0} \in \Sigma^\omega$ for the infinite word constructed from the labels of the transitions.

We call an infinite sequence of transitions $\rho = (t_i)_{i\geq 0}$ with $t_i \in T$ a *run* of $M = (Q, \Sigma, \Gamma, T, q_0, F)$ on $w = \text{label}(\rho)$ iff there exists an infinite sequence of configurations $(q_i, \gamma_i)_{i\geq 0}$ with $\gamma_0 = \epsilon$ such that $(q_i, \gamma_i) \vdash_M^{t_i} (q_{i+1}, \gamma_{i+1})$ for each $i \geq 0$.

For the sequence of states $(q_i)_{i\geq 0}$, let $\text{Inf}((q_i)_{i\geq 0}) = \{q \mid q = q_i \text{ for infinitely many } i \geq 0\}$. The run ρ is called *successful* if $\text{Inf}(\text{state}(\rho)) \cap F \neq \emptyset$.

Definition 2. *For an ωSPDA $M = (Q, \Sigma, \Gamma, T, q_0, F)$, the language* accepted *by M is denoted by $\mathcal{L}(M) = \{w \in \Sigma^\omega \mid \exists$ successful run of M on $w\}$. A language $L \subseteq \Sigma^\omega$ is called* ωSPDA-recognizable *if there exists an ωSPDA M with $\mathcal{L}(M) = L$.*

For clarity, we abbreviate a run $\rho = (t_i)_{i\geq 0}$ with $(q_0, \gamma_0) \vdash_M^{t_0} (q_1, \gamma_1) \vdash_M^{t_1} \cdots$ where $\text{label}(t_i) = a_i$ by ρ: $(q_0, \gamma_0) \xrightarrow{a_0} (q_1, \gamma_1) \xrightarrow{a_1} \cdots$ such that the word becomes visible.

Example 1. *We define an example automaton $\mathcal{A} = (Q, \Sigma, \{S, B\}, T, S, \{S\})$ with $Q = \{S, M, B\}$, $\Sigma = \{a, b\}$ and the transitions T as depicted in Fig. 1. In state M, the automaton reads a and pushes B. For every B that is popped from the stack, the automaton reads b. When there are no more B on the stack, S is remaining on the stack and b brings the automaton to start from the beginning. As S is the only final state, we have $\mathcal{L}(\mathcal{A}) = \{a^n b^n \mid n \geq 1\}^\omega$.*

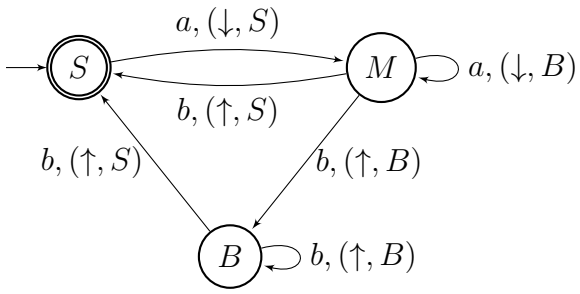

Figure 1: Example 1: Automaton

3. Expressiveness

In this section, we prove that the simple ω-pushdown automata defined above recognize all ω-context-free languages. This means that the restrictions in the automaton model do not change the expressiveness of ω-pushdown automata. Even though this property seems to be very basic, we could not find many results in the literature on this property. For the case of finite words, Blass and Gurevich [5] show how to translate nested-word automata into pushdown automata and only use the three stack commands defined for ωSPDA. As we borrowed the restrictions of our automaton model from nested-word automata, we use a similar strategy here to prove the expressive equivalence also in the infinite case.

First, let us recall some background. The concept of ω-context-free languages has been defined in [9] (for an overview cf. [24]). They are defined to be the languages generated by ω-context-free grammars with Muller-acceptance condition (cf. [22]). It is shown that these languages coincide with the class of languages recognized by general ω-pushdown automata, both for Büchi- and Muller-acceptance condition.

Clearly, every ωSPDA-recognizable language is ω-context-free. The inverse will be shown subsequently.

The ω-context-free grammars are similar to context-free grammars for finite words. We consider only infinite leftmost derivations and define both Büchi- and Muller-acceptance conditions.

Definition 3 (Cohen and Gold [9])**.** *An ω-context-free grammar is a tuple $G = (N, \Sigma, P, S, F)$ where (N, Σ, P, S) is an ordinary context-free grammar for finite words and F defines the acceptance condition: If G is Muller-accepting, we have $F \subseteq 2^N$. If G is Büchi-accepting, we have $F \subseteq N$.*

Let $\delta \colon S \to \dots$ be an infinite derivation of G. We write $\delta \colon S \to_G^\omega w$ if $w \in \Sigma^\omega$ is the infinite word of terminals occurring in the production rules of δ. For $i \geq 0$, let $\delta_N(i) = A_i$ be the non-terminal which is the left-hand side of the rule applied in step i of derivation δ. We define $\mathit{Inf}(\delta) = \{A \mid A = A_i \text{ for infinitely many } i \geq 0\}$.

For a Muller-accepting ω-context-free grammar G, the language generated *by G is defined as*

$$\mathcal{L}(G) = \{w \in \Sigma^\omega \mid \exists \text{ leftmost derivation } \delta \colon S \to_G^\omega w \text{ with } \mathit{Inf}(\delta) \in F\}.$$

For Büchi-accepting ω-context-free grammars,

$$\mathcal{L}(G) = \{w \in \Sigma^\omega \mid \exists \textit{ leftmost derivation } \delta\colon S \to_G^\omega w \textit{ with } \mathit{Inf}(\delta) \cap F \neq \emptyset\}\,.$$

A language $L \subseteq \Sigma^\omega$ is said to be an ω-context-free language if $L = \mathcal{L}(G)$ for a Muller-accepting ω-context-free grammar G.

Clearly, every Büchi-accepting ω-context-free grammar can be translated in a Muller-accepting one. The inverse is not so easily seen.

Lemma 4 ([15])**.** *Every ω-context-free language is generated by a Büchi-accepting ω-context-free grammar.*

This has been shown in [15] by using automata over countable words, i.e., also words with multiple ω-operators are allowed. But it can be shown directly by using a standard idea used already to translate Muller-automata to Büchi-automata (cf. Theorem 4.1.4 in Cohen and Gold [9]). In the sequel, a stronger result will be needed and is provided by Lemma 6 below.

For finite words, the following definition is standard, cf. Autebert, Berstel and Boasson [3] for an overview.

Definition 5. *An ω-context-free grammar $G = (N, \Sigma, P, S, F)$ is in* Greibach normal form *if $P \subseteq N \times \Sigma N^*$. More specifically, G is in* quadratic *(or 2-)* Greibach normal form *if*

$$P \subseteq N \times (\Sigma \cup \Sigma N \cup \Sigma N N)\,.$$

Lemma 6. *Let L be an ω-context-free language. There exists a Büchi-accepting ω-context-free grammar G in quadratic Greibach normal form with $\mathcal{L}(G) = L$.*

The idea of the proof is similar to the idea given in Cohen and Gold [9], Theorem 4.2.2, which shows that for Muller-accepting ω-context-free grammars one can remove rules of the type $A \to \epsilon$. They claim in Theorem 4.2.4 that the same idea can be used to prove that for every ω-context-free language there exists a Muller-accepting ω-context-free grammar in Greibach normal form. We show it here for Büchi-acceptance and for the stricter quadratic Greibach normal form.

Proof. By Theorem 4.1.8 of Cohen and Gold [9], L can be expressed as the Kleene-closure of context-free languages over finite words, i.e., for some $l \in \mathbb{N}$, there exist context-free grammars G_i, G_i' $(1 \leq i \leq l)$ such that $L = \bigcup_{i=1}^{l} \mathcal{L}(G_i)\mathcal{L}(G_i')^\omega$. For $1 \leq i \leq l$, let $G_i = (N_i, \Sigma, P_i, S_i)$ and $G_i' =$

$(N_i', \Sigma, P_i', S_i')$ and we assume all N_i and N_i' to be pairwise distinct. As the G_i and G_i' are context-free grammars for finite words, we can assume they are in quadratic Greibach normal form.

We construct the Büchi-accepting ω-context-free grammar $G = (N, \Sigma, P, S, F)$ where $N = \{S\} \cup \bigcup_{i=1}^{l} (N_i \cup N_i' \cup \{\bar{S}_i\})$ and $F = \{\bar{S}_i \mid 1 \leq i \leq l\}$ as follows; here the symbols S and $\bar{S}_i$ are new symbols and are assumed to be neither in N_i nor in N_i'. We define as an intermediate step

$$P_{\text{tmp}} = \{S \to S_i \bar{S}_i,\ \bar{S}_i \to S_i' \bar{S}_i \mid 1 \leq i \leq l\} \cup \bigcup_{i=1}^{l} \left(P_i \cup P_i'\right).$$

The grammar G with P_{tmp} as set of production rules accepts the language L but is not yet in Greibach normal form. To achieve this, we first substitute S_i and S_i' with all possible right-hand sides of their productions to obtain G in Greibach normal form:

$$P = \{S \to \alpha \bar{S}_i,\ \bar{S}_i \to \alpha' \bar{S}_i \mid S_i \to \alpha \in P_i,\ S_i' \to \alpha' \in P_i', 1 \leq i \leq l\} \cup \bigcup_{i=1}^{l} \left(P_i \cup P_i'\right)$$

Unfortunately, α and α' can already contain two non-terminals and therefore, G can contain up to three non-terminals on the right-hand sides of its productions. Thus, G is not yet in quadratic Greibach normal form.

The Büchi-acceptance condition and the definition of F ensure that for every word accepted by G, one of the G_i' is applied infinitely many times. Hence

$$\mathcal{L}(G) = \bigcup_{i=1}^{l} \mathcal{L}(G_i)\mathcal{L}(G_i')^{\omega} = L\,.$$

Now, we apply a standard algorithm (see e.g. Harrison [17], Theorem 4.7.1) to convert P into quadratic Greibach normal form $\bar{G} = (\bar{N}, \Sigma, \bar{P}, \bar{S}, F)$ with $\bar{N} = N \cup N \times N$ and

$$
\begin{aligned}
\bar{P} = & \bigcup_{i=1}^{l} (P_i \cup P'_i) \\
& \cup \{\bar{S} \to \alpha \bar{S}_i && \mid S_i \to \alpha \in P_i, |\alpha| \leq 2, 1 \leq i \leq l\} \\
& \cup \{\bar{S} \to aB(C, \bar{S}_i) && \mid S_i \to aBC \in P_i, 1 \leq i \leq l\} \\
& \cup \{\bar{S}_i \to \alpha \bar{S}_i && \mid S'_i \to \alpha \in P'_i, |\alpha| \leq 2, 1 \leq i \leq l\} \\
& \cup \{\bar{S}_i \to aB(C, \bar{S}_i) && \mid S'_i \to aBC \in P'_i, 1 \leq i \leq l\} \\
& \cup \{(A, \bar{S}_i) \to \alpha \bar{S}_i && \mid A \to \alpha \in P_i \cup P'_i, |\alpha| \leq 2, 1 \leq i \leq l\} \\
& \cup \{(A, \bar{S}_i) \to aB(C, \bar{S}_i) && \mid A \to aBC \in P_i \cup P'_i, 1 \leq i \leq l\}\,.
\end{aligned}
$$

The new non-terminals are pairs (A, B) with production rules that apply a production rule of non-terminal A and add the non-terminal B to its right-hand side.

Technically, because the G_i and G'_i are grammars for finite words, they could derive the empty word ϵ. In this special case, whenever $|\alpha| = 0$, we substitute in the above construction $\alpha \bar{S}_i$ by all right-hand sides of productions for $\bar{S}_i$ and we simply omit rules $\bar{S}_i \to \alpha \bar{S}_i$.

This shortens the production rules to at most two non-terminals on the right-hand side. As only an occurrence of the non-terminal $\bar{S}_i$ in $\bar{G}$ implies an occurrence of $\bar{S}_i$ in a derivation of G, the set of Büchi states F is not changed. It follows that

$$\mathcal{L}(\bar{G}) = \mathcal{L}(G) = L$$

and $\bar{G}$ is in quadratic Greibach normal form. □

Note that the above construction for $\bar{P}$ needs one case less than the original construction in Harrison [17], Theorem 4.7.1. In the original construction, there is a special case, in which for (A, B) there is already a production rule for A with three non-terminals on the right hand side. In $\bar{P}$, such production rules only occur for $\bar{S}$ and for $\bar{S}_i$. But neither $\bar{S}$ nor any of the $\bar{S}_i$ occur in the first position of any pair (A, B).

Example 2. *Let* $\Sigma = \{a, b, c\}$, $L_1 = c^+$ *and* $L'_1 = \{w \in \{a, b\}^+ \mid |w|_a = |w|_b\}$. *The corresponding grammars are* $G_1 = (\{S_1\}, \Sigma, P_1, S_1)$ *where* $P_1 = \{S_1 \to c \mid cS_1\}$ *and* $G'_1 = (N'_1, \Sigma, P'_1, S'_1)$ *where* $N'_1 = \{S'_1, M, N, A, B\}$ *and* P'_1 *contains the rules*

$$
\begin{aligned}
S_1' &\to aB \mid bA \mid aS_1'B \mid bS_1'A \mid aBS_1' \mid bAS_1' \mid aS_1'M \mid bS_1'N\\
M &\to bS_1'\\
N &\to aS_1'\\
A &\to a\\
B &\to b\,.
\end{aligned}
$$

The corresponding grammar for $L = L_1 L_1'^{\omega}$ *is* $\bar{G} = (\bar{N}, \Sigma, \bar{P}, \bar{S}, F)$ *with* $F = \{\bar{S}_1\}$ *and* $\bar{P}$ *contains the rules of* P_1*, of* P_1'*, and additionally*

$$
\begin{aligned}
\bar{S} &\to c\bar{S}_1 \mid cS_1\bar{S}_1\\
\bar{S}_1 &\to aB\bar{S}_1 \mid bA\bar{S}_1 \mid aS_1'(B,\bar{S}_1) \mid bS_1'(A,\bar{S}_1) \mid aB(S_1',\bar{S}_1) \mid bA(S_1',\bar{S}_1) \mid\\
&\qquad aS_1'(M,\bar{S}_1) \mid bS_1'(N,\bar{S}_1)\\
(B,\bar{S}_1) &\to b\bar{S}_1\\
(A,\bar{S}_1) &\to a\bar{S}_1\\
(S_1',\bar{S}_1) &\to aB\bar{S}_1 \mid bA\bar{S}_1 \mid aS_1'(B,\bar{S}_1) \mid bS_1'(A,\bar{S}_1) \mid aB(S_1',\bar{S}_1) \mid bA(S_1',\bar{S}_1) \mid\\
&\qquad aS_1'(M,\bar{S}_1) \mid bS_1'(N,\bar{S}_1)\\
(M,\bar{S}_1) &\to bS_1'\bar{S}_1\\
(N,\bar{S}_1) &\to aS_1'\bar{S}_1\,.
\end{aligned}
$$

We call intermediate steps in a derivation $\delta\colon S \to \alpha_1 \to \alpha_2 \to \dots$ *sentential forms*. Thus, the i-th sentential form of δ is α_i. Similarly, the i-th configuration of a run $\rho\colon \gamma_0 \vdash \gamma_1 \vdash \dots$ is defined to be γ_i.

The following will be the first main result of this paper.

Theorem 7. *Every ω-context-free language is ωSPDA-recognizable.*

Proof. Let L be an ω-context-free language. By Lemma 6, L is generated by some Büchi-accepting ω-context-free grammar $G = (N, \Sigma, P, S, F)$ in quadratic Greibach normal form. We construct an ωSPDA $M = (Q, \Sigma, \Gamma, T, q_0, F)$ with $Q = \Gamma = N$, $q_0 = S$, and

$$
\begin{aligned}
T = &\{(A, a, B, (\downarrow, C)) \mid A \to aBC \in P\} \cup && (1)\\
&\{(A, a, B, \#) \quad\ \mid A \to aB \in P\} \cup && (2)\\
&\{(A, a, B, (\uparrow, B)) \mid A \to a \in P, B \in N\} && (3)
\end{aligned}
$$

for $a \in \Sigma$ and $A, B, C \in N$.

Intuitively, the non-terminals in the grammar are simulated by states in the automaton. The second non-terminal on the right side of the productions

is pushed to the stack to store it for later. Whenever a final production is processed (Eq. (3)), it is checked which non-terminal is waiting on the stack to be processed. The Büchi-accepting final states of M are the same as in G. As the grammar only allows derivations where non-terminals in F occur infinitely often, the automaton will only allow runs where the same is true for states in F.

Claim: There exists a derivation $\delta\colon S \to \cdots \to a_1 \dots a_i A_1 \dots A_j \to \dots$ of G if and only if there exists a run $\rho\colon (S, \epsilon) \xrightarrow{a_1} \dots \xrightarrow{a_i} (A_1, A_2 \dots A_j) \to \dots$ of M, with $i \geq 0$.

We prove the claim by an inductive construction of steps i in the derivation δ and in the run ρ:

Let $i = 0$. Then the derivation $\delta\colon S$ is still in start state. As $q_0 = S$, the start of the corresponding run is $\rho\colon (q_0, \epsilon)$. The same argument holds for the other direction.

Let $i > 0$. We distinguish three cases:

1. Let the i-th rule be $A \to a_i$. To get the sentential form $a_1 \dots a_i A_1 \dots A_j$ in the i-th step, the $i-1$-th sentential form has to be $a_1 \dots a_{i-1} A A_1 \dots A_j$. Then, by induction hypothesis, there exists the $i-1$-th configuration $(A, A_1 \dots A_j)$ in the run ρ.
   By construction, there exists a transition $(A, a_i, A_1, (\uparrow, A_1)) \in T$. It follows that a possible i-th configuration is $(A_1, A_2 \dots A_j)$ and the word read until then is $a_1 \dots a_i$. The direction from run to derivation works similarly.
2. Let the i-th rule be $A \to a_i A_1$. To get the i-th sentential form as assumed in the claim, the $i-1$-th sentential form has to be $a_1 \dots a_{i-1} A A_2 \dots A_j$. Then, by induction hypothesis, there exists the $i-1$-th configuration $(A, A_2 \dots A_j)$ in the run ρ.
   By construction, there exists a transition $(A, a_i, A_1, \#) \in T$. It follows that a possible i-th configuration is $(A_1, A_2 \dots A_j)$ and the word read until then is $a_1 \dots a_i$. The other direction works similarly.
3. Let the i-th rule be $A \to a_i A_1 A_2$. To get the i-th sentential form as assumed in the claim, the $i-1$-th sentential form has to be $a_1 \dots a_{i-1} A A_3 \dots A_j$. Then, by induction hypothesis, there exists the $i-1$-th configuration $(A, A_3 \dots A_j)$ in the run ρ.
   By construction, there exists a transition $(A, a_i, A_1, (\downarrow, A_2)) \in T$. It follows that a possible i-th configuration is $(A_1, A_2 \dots A_j)$ and the word read until then is $a_1 \dots a_i$. The other direction works similarly.

This proves the claim.

Let $w = a_1 a_2 \ldots \in \Sigma^\omega$. Now,

$$
\begin{aligned}
w \in \mathcal{L}(M) \text{ iff } & \exists \text{ run } \rho \text{ of } M \text{ on } w \text{ and } \mathrm{Inf}(\mathrm{state}(\rho)) \cap F \neq \emptyset \\
\text{iff } & \exists \text{ run } \rho \colon (S, \epsilon) \xrightarrow{a_1} \ldots \xrightarrow{a_i} (A_1, A_2 \ldots A_j) \to \ldots \text{ of } M \\
& \qquad \text{and } \mathrm{Inf}(\mathrm{state}(\rho)) \cap F \neq \emptyset \\
\text{iff } & \exists \text{ derivation } \delta \colon S \to \cdots \to a_1 \ldots a_i A_1 \ldots A_j \to \ldots \\
& \qquad \text{of } G \text{ and } \mathrm{Inf}(\delta) \cap F \neq \emptyset \\
\text{iff } & \exists \text{ successful derivation } \delta \text{ of } G \text{ on } w \\
\text{iff } & w \in \mathcal{L}(G)\,.
\end{aligned}
$$

The equivalence of the second and third line is due to the claim above. □

Example 3. *Let $G = (N, \Sigma, P, S, F)$ be a Büchi-accepting ω-context-free grammar with $N = \{S, M, B\}$, $\Sigma = \{a, b\}$, $F = \{S\}$ and P contains the following rules:*

$$
\begin{aligned}
S &\to aMS \\
M &\to b \mid aMB \\
B &\to b
\end{aligned}
$$

Then G is in quadratic Greibach normal form. Note that the non-terminal M derives a string $a^n b^{n+1}$ for $n \in \mathbb{N}$ and the non-terminal S prepends another a. Thus, $\mathcal{L}(G) = \{a^n b^n \mid n \geq 1\}^\omega$. By the construction of the proof of Theorem 7, G can be transformed into the ωSPDA of Example 1. Note that Eq. 3 generates a rule for every non-terminal in the grammar. As there are no transitions that push M on the stack, we omit its pop-rule here.

Now we summarize our results obtained so far.

Corollary 8. *Let $L \subseteq \Sigma^\omega$. The following are equivalent:*

- *(i) L is ωSPDA-recognizable,*
- *(ii) L is accepted by some ω-pushdown automaton,*
- *(iii) L is contained in the ω-Kleene closure of context-free languages,*
- *(iv) L is generated by some Muller-accepting ω-context-free grammar,*
- *(v) L is generated by some Büchi-accepting ω-context-free grammar in 2-Greibach normal form.*

Proof. The notions of ω-Kleene closure and general ω-pushdown automata were given in Cohen and Gold [9]; ω-Kleene closure was also used in the proof of Lemma 6.

The proof is performed by the following steps:
$(i) \Rightarrow (ii)$: trivial by definition,
$(ii) \Leftrightarrow (iii) \Leftrightarrow (iv)$: shown by Cohen and Gold [9], Theorem 4.1.8,
$(iii) \Rightarrow (v)$: follows from Lemma 6,
$(v) \Rightarrow (i)$: follows from Theorem 7. □

4. Logic for ω-Pushdown Automata

The goal of this section is to find a Büchi-type logical formalism that is expressively equivalent to ωSPDA. This extends the work of Lautemann, Schwentick and Thérien [18] who defined a logic for context-free languages over finite words. They use first-order logic together with one second-order variable that has to define a matching. They also show that monadic second-order logic (MSO) with the one matching variable is equivalent as well. Their proof uses context-free grammars in a symmetric version of the Greibach normal form. Here we use a direct translation from automata and therefore, we will use the monadic second-order approach.

This section is guided by the procedure of Droste and Perevoshchikov [12]. Their main result is a logical characterization of timed pushdown languages for finite words.

Let $w \in \Sigma^\omega$ be an ω-word. The set of all positions of w is $\mathbb{N}$. A binary relation $M \subseteq \mathbb{N} \times \mathbb{N}$ is a *matching* (cf. [18]) if

- M is compatible with $<$, i.e., $(i,j) \in M$ implies $i < j$,
- each element i belongs to at most one pair in M,
- M is non-crossing, i.e., $(i,j) \in M$ and $(k,l) \in M$ with $i < k < j$ implies $i < l < j$.

Let $\mathrm{Match}(\mathbb{N})$ denote the set of all matchings in $\mathbb{N} \times \mathbb{N}$.

Let V_1, V_2 denote countable and pairwise disjoint sets of first-order and second-order variables. We fix a *matching variable* $\mu \notin V_1 \cup V_2$. Let $\mathcal{V} = V_1 \cup V_2 \cup \{\mu\}$.

We will define the logic in two steps. In the first layer of the logic, $\omega\mathrm{MSO}(\Sigma)$, the matching variable μ is unbounded. The second layer, $\omega\mathrm{ML}(\Sigma)$, existentially bounds this variable. We will show in Section 6 that $\omega\mathrm{ML}(\Sigma)$ is expressively equivalent to ωSPDA. As an intermediate step in the corresponding proof, we will use the fact from Section 5 that $\omega\mathrm{MSO}(\Sigma)$ is expressively equivalent to visibly pushdown ω-automata.

Definition 9. *Let Σ be an alphabet. The set $\omega MSO(\Sigma)$ of matching ω-MSO*

$$
\begin{array}{ll}
(w,\sigma) \models P_a(x) & \text{iff } a_{\sigma(x)} = a \\
(w,\sigma) \models x \leq y & \text{iff } \sigma(x) \leq \sigma(y) \\
(w,\sigma) \models x \in X & \text{iff } \sigma(x) \in \sigma(X) \\
(w,\sigma) \models \mu(x,y) & \text{iff } (\sigma(x),\sigma(y)) \in \sigma(\mu) \\
(w,\sigma) \models \neg\varphi & \text{iff } (w,\sigma) \not\models \varphi \\
(w,\sigma) \models \varphi \vee \psi & \text{iff } (w,\sigma) \models \varphi \text{ or } (w,\sigma) \models \psi \\
(w,\sigma) \models \exists x.\, \varphi & \text{iff } \exists j \in \mathbb{N}.\, (w,\sigma[x/j]) \models \varphi \\
(w,\sigma) \models \exists X.\, \varphi & \text{iff } \exists J \subseteq \mathbb{N}.\, (w,\sigma[X/J]) \models \varphi
\end{array}
$$

Table 1: The semantics of $\omega\mathrm{MSO}(\Sigma)$ formulas

formulas is defined by the extended Backus-Naur form (EBNF)

$$\varphi ::= P_a(x) \mid x \leq y \mid x \in X \mid \mu(x,y) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\, \varphi \mid \exists X.\, \varphi$$

where $a \in \Sigma$, $x, y \in V_1$ *and* $X \in V_2$.

Positions in the word will later be assigned to variables in φ. Here, $P_a(x)$ is a unary predicate indicating that the x-th letter of the word is a. Furthermore, $\mu(x,y)$ says that x and y will be matched.

A $(w,\mathcal{V})$-*assignment* is a mapping $\sigma\colon \mathcal{V} \to \mathbb{N} \cup 2^{\mathbb{N}} \cup \mathrm{Match}(\mathbb{N})$ such that $\sigma(V_1) \subseteq \mathbb{N}$, $\sigma(V_2) \subseteq 2^{\mathbb{N}}$ and $\sigma(\mu) \in \mathrm{Match}(\mathbb{N})$.

Let σ be a $(w,\mathcal{V})$-assignment. For $x \in V_1$ and $j \in \mathbb{N}$, the update $\sigma[x/j]$ is the $(w,\mathcal{V})$-assignment σ' with $\sigma'(x) = j$ and $\sigma'(y) = \sigma(y)$ for all $y \in \mathcal{V} \setminus \{x\}$. The update $\sigma[X/J]$ for $X \in V_2$ and $J \subseteq \mathbb{N}$ and the update $\sigma[\mu/M]$ for $M \in \mathrm{Match}(\mathbb{N})$ are defined similarly.

Let $\varphi \in \omega\mathrm{MSO}(\Sigma)$. Furthermore, let $w = a_0 a_1 \ldots \in \Sigma^\omega$ and σ be a $(w,\mathcal{V})$-assignment. We define $(w,\sigma) \models \varphi$ inductively over the structure of φ as shown in Table 1, where $a \in \Sigma$, $x, y \in V_1$ and $X \in V_2$. The logical counterparts $\wedge$, $\rightarrow$, $\forall x.\, \phi$ and $\forall X.\, \phi$ can be gained in the usual way from negation and the existing operators.

We now define $\mathrm{MATCHING}(\mu) \in \omega\mathrm{MSO}(\Sigma)$ which ensures that μ is

matching. Let

$$\begin{aligned}\mathrm{MATCHING}(\mu) = \forall x \forall y.\, (\mu(x,y) \to x < y) \wedge \\ \forall x \forall y \forall k.\, \big((\mu(x,y) \wedge k \neq x \wedge k \neq y) \to \neg\mu(x,k) \wedge \neg\mu(k,x) \wedge \neg\mu(y,k) \wedge \neg\mu(k,y)\big) \wedge \\ \forall x \forall y \forall k \forall l.\, \big((\mu(x,y) \wedge \mu(k,l) \wedge x < k < y) \to x < l < y\big)\,,\end{aligned}$$

where $x \neq y$, $x < y$ and $i < j < k$ have the usual translation.

Definition 10. *We let $\omega ML(\Sigma)$, the set of formulas of* matching ω-logic *over Σ, be the set of all formulas ψ of the form*

$$\psi = \exists \mu.\, (\varphi \wedge \mathit{MATCHING}(\mu))\,,$$

for short $\psi = \exists^{match} \mu.\, \varphi$, where $\varphi \in \omega MSO(\Sigma)$.

Let $w \in \Sigma^\omega$ and σ be a $(w, \mathcal{V})$-assignment. Then, $(w, \sigma) \models \psi$ if there exists a matching $M \subseteq \mathbb{N}^2$ such that $(w, \sigma[\mu/M]) \models \varphi$.

Let $\psi \in \omega\mathrm{ML}(\Sigma)$. We denote by $\mathrm{Free}(\psi) \subseteq \mathcal{V}$ the set of *free variables* of ψ. A formula ψ with $\mathrm{Free}(\psi) = \emptyset$ is called a *sentence*. For a sentence ψ, the validity of $(w, \sigma) \models \psi$ does not depend on σ. Therefore, σ will be omitted and we only write $w \models \psi$. We denote by $\mathcal{L}(\psi) = \{w \in \Sigma^\omega \mid w \models \psi\}$ the language *defined* by ψ. A language $L \subseteq \Sigma^\omega$ is *ωML-definable* if there exists a sentence $\psi \in \omega\mathrm{ML}(\Sigma)$ such that $\mathcal{L}(\psi) = L$.

The following will be the second main result.

Theorem 11. *Let Σ be an alphabet and $L \subseteq \Sigma^\omega$ an ω-language. Then L is ωML-definable if and only if L is ωSPDA-recognizable.*

After some preparations, this theorem will be proved in Section 6.

5. Visibly Pushdown ω-Languages

It turns out that the $\omega\mathrm{MSO}(\Sigma)$ formulas correspond exactly to the MSO-logic defined for visibly pushdown ω-languages [1]. In fact, without considering the existential quantification over the matching relation $\exists^{\mathrm{match}}\mu$, the matching must explicitly be encoded in the words; the result is a nested word. For the convenience of the reader, we recall nested words and visibly pushdown languages [1, 2] in this section.

A *nested alphabet* is a triple $\tilde{\Sigma} = (\Sigma^\downarrow, \Sigma^\#, \Sigma^\uparrow)$ with $\Sigma^\downarrow$, $\Sigma^\#$ and $\Sigma^\uparrow$ being pairwise disjoint sets of *push*, *internal* and *pop* letters, respectively. Let $\hat{\Sigma} = \Sigma^\downarrow \cup \Sigma^\# \cup \Sigma^\uparrow$.

Definition 12. *A* visibly pushdown ω-automaton *(ωVPA) is a 6-tuple* $M = (Q, \tilde{\Sigma}, \Gamma, T, q_0, F)$ *where*

- Q *is a finite set of states,*
- $\tilde{\Sigma}$ *is a finite nested alphabet,*
- Γ *is a finite stack alphabet,*
- $T = T^{\downarrow} \cup T^{\#} \cup T^{\uparrow}$ *is a set of transitions, with*
  - $T^{\downarrow} \subseteq Q \times \Sigma^{\downarrow} \times Q \times (\{\downarrow\} \times \Gamma)$,
  - $T^{\#} \subseteq Q \times \Sigma^{\#} \times Q \times \{\#\}$,
  - $T^{\uparrow} \subseteq Q \times \Sigma^{\uparrow} \times Q \times (\{\uparrow\} \times \Gamma)$,
- q_0 *is the initial state and*
- F *is a set of (Büchi accepting) final states.*

The following definitions are mostly similar to the ones for ωSPDAs. The only difference is that the definition for T above restricts push transitions to push letters, pop transitions to pop letters and internal transitions to internal letters.

A *configuration* of an ωVPA $M = (Q, \tilde{\Sigma}, \Gamma, T, q_0, F)$ is a pair (q, γ), where $q \in Q$ and $\gamma \in \Gamma^*$. We define the transition relation as follows. Let $\gamma \in \Gamma^*$. For $t = (q, \sigma, q', (\downarrow, A)) \in T^{\downarrow}$, we write $(q, \gamma) \vdash_M^t (q', A\gamma)$. For $t = (q, \sigma, q', \#) \in T^{\#}$, we write $(q, \gamma) \vdash_M^t (q', \gamma)$. Finally, for $t = (q, \sigma, q', (\uparrow, A)) \in T^{\uparrow}$, we write $(q, A\gamma) \vdash_M^t (q', \gamma)$.

We denote by $\text{state}(q, \sigma, q', s) = q$ the state and by $\text{label}(q, \sigma, q', s) = \sigma$ the *label* of a transition. Both are extended to infinite sequence of transitions by letting $\text{state}((t_i)_{i\geq 0}) = (\text{state}(t_i))_{i\geq 0} \in Q^\omega$ for the infinite sequence of states and $\text{label}((t_i)_{i\geq 0}) = (\text{label}(t_i))_{i\geq 0} \in \hat{\Sigma}^\omega$ for the infinite word constructed from the labels of the transitions.

We call an infinite sequence of transitions $\rho = (t_i)_{i\geq 0}$ with $t_i \in T$ a *run* of M on $w = \text{label}(\rho) \in \hat{\Sigma}^\omega$ iff there exists an infinite sequence of configurations $(q_i, \gamma_i)_{i\geq 0}$ with $\gamma_0 = \epsilon$ such that $(q_i, \gamma_i) \vdash_M^{t_i} (q_{i+1}, \gamma_{i+1})$ for each $i \geq 0$.

A run ρ is called *successful* if $\text{Inf}(\text{state}(\rho)) \cap F \neq \emptyset$.

Definition 13. *For an ωVPA $M = (Q, \tilde{\Sigma}, \Gamma, T, q_0, F)$, the language* accepted *by M is denoted by $\mathcal{L}(M) = \{w \in \hat{\Sigma}^\omega \mid \exists$ successful run of M on $w\}$. A language $L \subseteq \hat{\Sigma}^\omega$ is* ωVPA-recognizable with respect to $\tilde{\Sigma}$ *if there exists an ωVPA $M = (Q, \tilde{\Sigma}, \Gamma, T, q_0, F)$ with $\mathcal{L}(M) = L$.*

Note that the ωVPA-recognizable languages with respect to $\tilde{\Sigma}$ form a proper subclass of the ωSPDA-recognizable languages over $\hat{\Sigma}$.

Also note that the definition here differs from the definition in [1] by the way how we handle the empty stack. Alur, Madhusudan allow their automata to check if the stack is empty by reading the special symbol $\perp$. The ωVPA does not allow this check directly. But the ωVPA can duplicate all its states and stack symbols to allow us to distinguish between states with empty stack and those with non-empty stack. Whenever pushing a symbol onto the empty stack, this new stack symbol has to contain the extra information that upon popping that symbol, the automaton has to change to an empty-stack state afterwards.

Let $f : \hat{\Sigma} \to \hat{\Sigma}'$ be a mapping. It *respects nesting* if $f(\Sigma^{\downarrow}) \subseteq \Sigma'^{\downarrow}$, $f(\Sigma^{\#}) \subseteq \Sigma'^{\#}$ and $f(\Sigma^{\uparrow}) \subseteq \Sigma'^{\uparrow}$. The mapping will be extended to words in the natural way.

Theorem 14 (Alur and Madhusudan [1])**.** *Let $L_1 \subseteq \hat{\Sigma}^{\omega}$ and $L_2 \subseteq \hat{\Sigma}^{\omega}$ be ωVPA-recognizable with respect to $\tilde{\Sigma}$. Then $L_1 \cup L_2$, $L_1 \cap L_2$, $\hat{\Sigma}^{\omega} \setminus L_1$ are ωVPA-recognizable with respect to $\tilde{\Sigma}$. If $f\colon \hat{\Sigma} \to \hat{\Sigma}'$ is a mapping that respects nesting, then $f(L_1)$ is ωVPA-recognizable with respect to $\tilde{\Sigma}'$.*

We now discuss how the logic ωMSO has the same expressive power as ωVPAs. Note that all $\omega\mathrm{MSO}(\Sigma)$ formulas contain at least the free variable μ. We therefore have to extend the definitions as follows. Let

$$\mathcal{L}_{\mathcal{V}}(\psi) = \{(w, \sigma) \mid w \in \Sigma^{\omega}, \sigma \text{ is a } (w, \mathcal{V})\text{-assignment}, (w, \sigma) \models \psi\}.$$

In the following, we are only interested in $\mathcal{L}_{\{\mu\}}(\psi)$. The matching relation μ can be encoded into Σ to gain a nested alphabet called the *tagged alphabet* $\mathrm{tag}(\Sigma) = (\{a^{\downarrow} \mid a \in \Sigma\}, \{a^{\#} \mid a \in \Sigma\}, \{a^{\uparrow} \mid a \in \Sigma\})$ where the *tagged* letters $a^{\downarrow}$, $a^{\#}$ and $a^{\uparrow}$ are not occurring in Σ. For that, consider a word $w = a_0 a_1 \cdots$ and a $(w, \{\mu\})$-assignment σ. The encoding for (w, σ) is the *tagged* word $\tilde{w} = \tilde{a}_0 \tilde{a}_1 \cdots$ where $\tilde{a}_i = a_i^{\downarrow}$ if there exists a position y with $\mu(i, y)$, and $\tilde{a}_i = a_i^{\uparrow}$ if there exists a position x with $\mu(x, i)$, and $\tilde{a}_i = a_i^{\#}$ otherwise. This encoding will be extended to sets of $(w, \{\mu\})$-assignments like $\mathcal{L}_{\{\mu\}}(\varphi)$ in the natural way. The underlying alphabet will be called $\hat{\Sigma}_{\mathrm{tag}} = \{\sigma^{\downarrow} \mid \sigma \in \Sigma\} \cup \{\sigma^{\#} \mid \sigma \in \Sigma\} \cup \{\sigma^{\uparrow} \mid \sigma \in \Sigma\}$.

Theorem 15 (Alur and Madhusudan [1])**.** *Let $L \subseteq \hat{\Sigma}^{\omega}_{tag}$. Then, L is ωVPA-recognizable with respect to tag(Σ) iff there is an ωMSO(Σ)-formula φ with Free$(\varphi) = \{\mu\}$ and $\mathcal{L}_{\{\mu\}}(\varphi) = L$.*

The mapping $\pi\colon \hat{\Sigma}_{\text{tag}} \to \Sigma$ removes the "tag" of a tagged letter. Thus, π maps $a^{\downarrow}$, $a^{\#}$ and $a^{\uparrow}$ to a. This can be extended to words by letting $\pi(a_0 a_1 \cdots) = \pi(a_0)\pi(a_1)\cdots$ and to languages $L \subseteq \hat{\Sigma}^{\omega}_{\text{tag}}$ by setting $\pi(L) = \{\pi(w) \mid w \in L\}$.

Lemma 16. *Let $L \subseteq \hat{\Sigma}^{\omega}_{tag}$ be ωVPA-recognizable with respect to tag(Σ). Then $\pi(L) \subseteq \Sigma^{\omega}$ is ωSPDA-recognizable.*

This has been proved in [5] for finite nested words. Here, we present their proof adopted similarly to infinite words.

Proof. Let $A = (Q, \text{tag}(\Sigma), \Gamma, T, q_0, F)$ be an ωVPA with $\mathcal{L}(A) = L$. We construct an ωSPDA $B = (Q, \Sigma, \Gamma, T', q_0, F)$ such that $\mathcal{L}(B) = \pi(L)$. Let

$$T' = \{(q, \pi(a), p, s) \mid (q, a, p, s) \in T\}\,.$$

Now,

$$\begin{aligned}
\mathcal{L}(B) &= \{w \in \Sigma^{\omega} \mid \exists \text{ successful run of } B \text{ on } w\} \\
&= \{w \in \Sigma^{\omega} \mid \exists \text{ successful run } \rho \text{ of } B \text{ on } w = w_0 w_1 \cdots \text{ and} \\
&\qquad\qquad \rho\colon (q_0, \gamma_0) \xrightarrow{w_0} (q_1, \gamma_1) \xrightarrow{w_1} \cdots\} \\
&= \{w \in \Sigma^{\omega} \mid \exists \text{ run } \rho \text{ of } B \text{ on } w \text{ with } \text{Inf}(\text{state}(\rho)) \cap F \neq \emptyset \text{ and} \\
&\qquad\qquad \rho\colon (q_0, \gamma_0) \xrightarrow{w_0} (q_1, \gamma_1) \xrightarrow{w_1} \cdots \text{ and} \\
&\qquad\qquad w = w_0 w_1 \cdots = \pi(v_0)\pi(v_1)\cdots = \pi(v)\} \\
&= \{w \in \Sigma^{\omega} \mid w = \pi(v) \text{ and } \exists \text{ run } \rho' \text{ of } A \text{ on } v = v_0 v_1 \cdots \text{ with} \\
&\qquad\qquad \text{Inf}(\text{state}(\rho')) \cap F \neq \emptyset \text{ and } \rho'\colon (q_0, \gamma_0) \xrightarrow{v_0} (q_1, \gamma_1) \xrightarrow{v_1} \cdots\} \\
&= \{\pi(v) \mid v \in \hat{\Sigma}^{\omega}_{\text{tag}} \text{ and } \exists \text{ successful run } \rho' \text{ of } A \text{ on } v = v_0 v_1 \cdots \text{ with} \\
&\qquad\qquad \rho'\colon (q_0, \gamma_0) \xrightarrow{v_0} (q_1, \gamma_1) \xrightarrow{v_1} \cdots\} \\
&= \pi\{v \in \hat{\Sigma}^{\omega}_{\text{tag}} \mid \exists \text{ successful run of } A \text{ on } v\} \\
&= \pi(\mathcal{L}(A)) = \pi(L)\,.
\end{aligned}$$

□

6. Equivalence of Logic and Automata

This section proves the expressive equivalence of the full logic and ω-pushdown automata.

Let $a = (a_1, \ldots, a_n) \in A_1 \times \ldots \times A_n$ be a tuple. Then we define for $1 \leq i \leq n$ the *i-th projection* of a by $\text{pr}_i(a) = a_i$.

Lemma 17. *Let Σ be an alphabet and $L \subseteq \Sigma^\omega$ be ωSPDA-recognizable. Then L is ωML-definable.*

Proof. By assumption, there exists an ωSPDA $\mathcal{A} = (Q, \Sigma, \Gamma, T, q_0, F)$ with $\mathcal{L}(\mathcal{A}) = L$. Let $T = \{t_1, \ldots, t_m\}$ be an enumeration of the transitions. We define an ωML-sentence ψ such that $\mathcal{L}(\psi) = L$ as follows. Hereby, we proceed similarly to the lines of [11].

First, we will need an auxiliary formula

$$\mathrm{next}(x, y) = x < y \wedge \neg(\exists z.\, x < z \wedge z < y)\,.$$

We define the set of second-order variables $\mathcal{V} = \{X_t \mid t \in T\}$. The following three ωMSO(Σ) formulas ψ_{part}, ψ_{comp} and ψ_{final} will be used. We let

$$\psi_{\text{part}} = \forall x. \bigvee_{t \in T} \big(x \in X_t \wedge \bigwedge_{t' \in T\colon\, t \neq t'} x \notin X_{t'}\big)\,.$$

Then ψ_{part} ensures that the variables X_t form a partitioning of the positions x. Now let

$$\psi_{\text{comp}} = \forall x. \Big(\varphi_{\text{first}}(x) \wedge \bigwedge_{t \in T} \big(x \in X_t \to (\varphi_1(x,t) \wedge \varphi_2(x,t) \wedge \varphi_3(x,t))\big)\Big)\,,$$

and let

$$\begin{aligned}
\varphi_{\text{first}}(x) &= \forall y.\, (x \leq y) \to \bigvee_{t \in T\colon\, \mathrm{pr}_1(t) = q_0} x \in X_t\,,\\
\varphi_1(x, (q, a, q', s)) &= P_a(x)\,,\\
\varphi_2(x, (q, a, q', s)) &= \forall y.\, \big(next(x, y) \to \bigvee_{t' \in T\colon\, \mathrm{pr}_1(t') = q'} y \in X_{t'}\big)\,,\\
\varphi_3(x, (q, a, q', (\uparrow, A))) &= \exists y. \Big(\bigvee_{t \in T\colon\, \mathrm{pr}_4(t) = (\downarrow, A)} y \in X_t \wedge \mu(y, x)\Big)\,.
\end{aligned}$$

Then ψ_{comp} ensures that there exists a run of the automaton. Note how the formula φ_3 applies the matching variable μ to simulate the stack of the automaton.

Finally, ψ_{final} controls the acceptance condition that final states have to occur infinitely often in a successful run:

$$\psi_{\text{final}} = \forall x \exists y. \Big(x < y \wedge \bigvee_{t \in T\colon\, \mathrm{pr}_1(t) \in F} y \in X_t\Big)\,.$$

Now, let $\psi \in \omega\mathrm{ML}(\Sigma)$ be the sentence defined as

$$\psi = \exists^{\mathrm{match}}\mu.\, \psi_{\mathrm{part}} \wedge \psi_{\mathrm{comp}} \wedge \psi_{\mathrm{final}}\,.$$

Then,

$$\begin{aligned}\mathcal{L}(\psi) &= \{w \in \Sigma^\omega \mid \exists \text{ matching } M \text{ s.t. } (w, \emptyset[\mu/M]) \models \psi_{\mathrm{part}} \wedge \psi_{\mathrm{comp}} \wedge \psi_{\mathrm{final}}\} \\ &= \{w \in \Sigma^\omega \mid \exists \text{ successful run of } \mathcal{A} \text{ on } w\} \\ &= \mathcal{L}(\mathcal{A}) = L\,.\end{aligned}$$

□

The other direction uses the corresponding results for visibly pushdown languages.

Lemma 18. *Let Σ be an alphabet and $L \subseteq \Sigma^\omega$ be ωML-definable. Then L is ωSPDA-recognizable.*

Proof. Let $\psi = \exists^{\mathrm{match}}\mu.\varphi \in \omega\mathrm{ML}(\Sigma)$ be a formula with $\mathcal{L}(\psi) = L$.

We know $\varphi \in \omega\mathrm{MSO}(\Sigma)$ with $\mathrm{Free}(\varphi) = \{\mu\}$. Let $L' = \mathcal{L}_{\{\mu\}}(\varphi)$. By Theorem 15, $L' \subseteq \hat{\Sigma}^\omega$ is ωVPA-recognizable with respect to $\tilde{\Sigma} = (\{\sigma^\downarrow \mid \sigma \in \Sigma\}, \{\sigma^\# \mid \sigma \in \Sigma\}, \{\sigma^\uparrow \mid \sigma \in \Sigma\})$.

The remaining existential quantification over the matching relation μ is exactly the projection π from ωVPA-recognizable languages to ωSPDA-recognizable languages. By Lemma 16, $L = \pi(L')$ is ωSPDA-recognizable. □

Proof of Theorem 11. This theorem is immediate by Lemmas 17 and 18. □

Concluding, we can summarize:

Corollary 19. *Let $L \subseteq \Sigma^\omega$. The following are equivalent:*

(i) L is sPDA-recognizable,
(ii) L is an ω-context-free language,
(iii) L is ωML-definable.

References

[1] R. Alur, P. Madhusudan (2004). Visibly pushdown languages. In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*, pp. 202–211. doi:10.1145/1007352.1007390.

[2] R. Alur, P. Madhusudan (2009). Adding nesting structure to words. *JACM*, vol. 56(3), pp. 16:1–16:43. doi:10.1145/1516512.1516518.

[3] J.-M. Autebert, J. Berstel, L. Boasson (1997). Context-free languages and pushdown automata. In: *Handbook of Formal Languages* (eds. G. Rozenberg, A. Salomaa), vol. 1, chap. 3, pp. 111–174. Springer. doi: 10.1007/978-3-642-59136-5_3.

[4] C. Baier, J.-P. Katoen (2008). *Principles of model checking*. The MIT Press. ISBN 9780262026499.

[5] A. Blass, Y. Gurevich (2006). A note on nested words. *Microsoft Research*.
URL `https://www.microsoft.com/en-us/research/publication/180-a-note-on-nested-words/`

[6] J. R. Büchi (1960). Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, vol. 6, pp. 66–92. doi: 10.1002/malq.19600060105.

[7] J. R. Büchi (1966). Symposium on decision problems: On a decision method in restricted second order arithmetic. In: *Logic, Methodology and Philosophy of Science* (eds. E. Nagel, P. Suppes, A. Tarski), *Studies in Logic and the Foundations of Mathematics*, vol. 44, pp. 1–11. Elsevier. doi:10.1016/S0049-237X(09)70564-6.

[8] E. M. Clarke, T. A. Henzinger, H. Veith, R. P. Bloem (2016). *Handbook of Model Checking*. Springer. doi:10.1007/978-3-319-10575-8.

[9] R. S. Cohen, A. Y. Gold (1977). Theory of ω-languages I: Characterizations of ω-context-free languages. *Journal of Computer and System Sciences*, vol. 15(2), pp. 169–184. doi:10.1016/S0022-0000(77)80004-4.

[10] M. Droste, S. Dziadek, W. Kuich. Weighted simple reset pushdown automata. Submitted.

[11] M. Droste, P. Gastin (2007). Weighted automata and weighted logics. *Theor. Comput. Sci.*, vol. 380(1-2), pp. 69–86. doi: 10.1016/j.tcs.2007.02.055.

[12] M. Droste, V. Perevoshchikov (2015). A logical characterization of timed pushdown languages. In: *Computer Science - Theory and Applications - 10th International Computer Science Symposium in Russia, CSR 2015*,

LNCS, vol. 9139, pp. 189–203. Springer. doi:10.1007/978-3-319-20297-6_13.

[13] M. Droste, V. Perevoshchikov (2015). Logics for weighted timed pushdown automata. In: *Fields of Logic and Computation II*, pp. 153–173. Springer. doi:10.1007/978-3-319-23534-9_9.

[14] C. C. Elgot (1961). Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, vol. 98, pp. 21–51. doi:10.2307/2270940.

[15] Z. Ésik, S. Iván (2011). Büchi context-free languages. *Theoretical Computer Science*, vol. 412(8), pp. 805–821. doi:10.1016/j.tcs.2010.11.026.

[16] D. Giammarresi, A. Restivo, S. Seibert, W. Thomas (1996). Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation*, vol. 125(1), pp. 32–45. doi:10.1006/inco.1996.0018.

[17] M. A. Harrison (1978). *Introduction to Formal Language Theory*. Addison-Wesley. ISBN 0201029553.

[18] C. Lautemann, T. Schwentick, D. Thérien (1994). Logics for context-free languages. In: *International Workshop on Computer Science Logic (CSL 1994)*, *LNCS*, vol. 933, pp. 205–216. Springer. doi:10.1007/BFb0022257.

[19] Y. Lei, F. Song, W. Liu, M. Zhang (2017). On the complexity of ω-pushdown automata. *Science China Information Sciences*, vol. 60(11), p. 112102. doi:10.1007/s11432-016-9026-x.

[20] C. Mathissen (2008). Weighted logics for nested words and algebraic formal power series. In: *Automata, Languages and Programming*, pp. 221–232. Springer. doi:10.1007/978-3-540-70583-3_19.

[21] K. L. McMillan (1993). *Symbolic Model Checking*. Kluwer Academic Publishers. doi:10.1007/978-1-4615-3190-6.

[22] D. E. Muller (1963). Infinite sequences and finite machines. In: *Proceedings of the Fourth Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1963)*, pp. 3–16. IEEE. doi:10.1109/SWCT.1963.8.

[23] J. W. Thatcher, J. B. Wright (1968). Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, vol. 2(1), pp. 57–81. doi:10.1007/BF01691346.

[24] W. Thomas (1990). Automata on infinite objects. In: *Handbook of Theoretical Computer Science* (ed. J. van Leeuwen), vol. B, chap. 4, pp. 133–191. Elsevier. doi:10.1016/B978-0-444-88074-1.50009-3.

[25] B. A. Trakhtenbrot (1961). Finite automata and the logic of single-place predicates. *Doklady Akademii Nauk*, vol. 140(2), pp. 326–329. In Russian.
URL `http://mi.mathnet.ru/dan25511`

[26] H. Vogler, M. Droste, L. Herrmann (2016). A weighted MSO logic with storage behaviour and its Büchi-Elgot-Trakhtenbrot theorem. In: *Language and Automata Theory and Applications - 10th International Conference, (LATA 2016)*, *LNCS*, vol. 9618, pp. 127–139. Springer. doi: 10.1007/978-3-319-30000-9_10.