Nivat-Theorem and Logic for Weighted Pushdown Automata on Infinite Words

Manfred Droste
Institut für Informatik, Universität Leipzig, Germany
droste@informatik.uni-leipzig.de

Sven Dziadek
Institut für Informatik, Universität Leipzig, Germany
dziadek@informatik.uni-leipzig.de

Werner Kuich
Institut für Diskrete Mathematik und Geometrie, Technische Universität Wien, Austria
werner.kuich@tuwien.ac.at

Abstract

Recently, weighted \(\omega\)-pushdown automata have been introduced by Droste, Ésik, Kuich. This new type of automaton has access to a stack and models quantitative aspects of infinite words. Here, we consider a simple version of those automata. The simple \(\omega\)-pushdown automata do not use \(\epsilon\)-transitions and have a very restricted stack access. In previous work, we could show this automaton model to be expressively equivalent to context-free \(\omega\)-languages in the unweighted case. Furthermore, semiring-weighted simple \(\omega\)-pushdown automata recognize all \(\omega\)-algebraic series.

Here, we consider \(\omega\)-valuation monoids as weight structures. As a first result, we prove that for this weight structure and for simple \(\omega\)-pushdown automata, Büchi-acceptance and Muller-acceptance are expressively equivalent. In our second result, we derive a Nivat theorem for these automata stating that the behaviors of weighted \(\omega\)-pushdown automata are precisely the projections of very simple \(\omega\)-series restricted to \(\omega\)-context-free languages. The third result is a weighted logic with the same expressive power as the new automaton model. To prove the equivalence, we use a similar result for weighted nested \(\omega\)-word automata and apply our present result of expressive equivalence of Muller and Büchi acceptance.

1 Introduction

Languages of infinite words or \(\omega\)-languages are intensively researched due to their applications in model checking and verification [30, 3, 9]. Context-free languages of infinite words have been investigated in a fundamental study by Cohen and Gold [10].

Weighted languages allow us to model the use of resources. In formal language theory, we consider a word to be in the language or not. Contrary to this, weighted languages relate words to resources such as costs, gains, probabilities, counts, time, and of course Boolean values. There exist generalizations to several language classes (regular, context-free, star-free languages, etc.), to various structures (words, trees, pictures, nested words, infinite words, etc.) and to different weight structures (semirings, valuation monoids, etc.). See [20] for
an overview. While weighted context-free languages already date back to Chomsky and Schützenberger [8], more recently, Droste, Ésik, Kuich [27, 19, 16] generalized context-free languages of infinite words to the weighted setting.

In this paper, we investigate a type of weighted \(\omega\)-pushdown automata called \textit{simple} \(\omega\)-reset pushdown automaton in [12]. They do not allow \(\epsilon\)-transitions and the stack can only be altered by at most one symbol. Simple automata have been shown to be expressively equivalent to general pushdown automata in the unweighted case for finite words [4] and for infinite words [15] (i.e., the language classes accepted by these two kinds of automata coincide). For continuous commutative star-omega semirings we could show in [13, 12, 14] that for every \(\omega\)-algebraic series \(r\), there exists a simple \(\omega\)-reset pushdown automaton with behavior \(r\).

Here, we consider \(\omega\)-valuation monoids as weight structures. They include complete semirings but also discounted and average behavior. Valuation monoids first appeared in [22] but their idea is based on [7]. By an example, we show how a basic web server and its average response time for requests can be modeled by a simple \(\omega\)-pushdown automaton with weights in a suitable \(\omega\)-valuation monoid.

Our first main result is the expressive equivalence of Büchi and Muller acceptance for weighted simple \(\omega\)-pushdown automata; i.e., the classes of behaviors of these two weighted automata models coincide.

Then we show several closure properties for weighted \(\omega\)-pushdown automata. Our second main result is a Nivat-like decomposition theorem [31] that shows that by the help of a morphism, we can express the behavior of every weighted \(\omega\)-pushdown automaton as the intersection of an unweighted \(\omega\)-pushdown automaton and a very simple \(\omega\)-series. Nivat’s theorem was extended to weighted automata of finite words over semirings by [21].

Büchi, Elgot, Trakhtenbrot [5, 26, 33] (BET-Theorem) proved that regular languages are exactly those languages definable by monadic second-order logic. Their result was extended by Lautemann, Schwentick, Thérien [29] to context-free languages. While both these former results are for finite words, we defined a logic that is expressively equivalent to context-free languages of infinite words (cf. [15]). The BET-Theorem has been extended to the weighted setting [17]. Weighted logics allow the logical description of weights of finite words [17, 24, 34] and also of infinite words [25, 18, 22].

In this paper, as the third main result, we extend the BET-Theorem to weighted simple \(\omega\)-pushdown automata. We extend the logic in [29, 11] and prove its equivalence to weighted simple \(\omega\)-pushdown automata. For the proof, we do not reinvent the wheel but use the already existing BET-Theorem for weighted nested \(\omega\)-word automata [11]. The application of a projection allows us to lift the result on weighted nested \(\omega\)-word automata to weighted simple \(\omega\)-pushdown automata. We show how the quantitative behavior of the basic web server example mentioned above can be described in our weighted matching \(\omega\)-MSO logic.

An expressive equivalence result for arbitrary weighted \(\omega\)-pushdown automata, besides our Nivat-like result, remains open at present.

We structure the paper as follows. We give basic definitions and compare Muller and Büchi acceptance in Section 2. Then, we prove the Nivat-like result in Section 3. Section 4 defines the logic. Section 5 summarizes the known results about weighted nested \(\omega\)-word languages and also shows the new projection. In Section 6, we prove our weighted BET-Theorem.
2 Weight Structure and Simple $\omega$-Pushdown Automata

This section introduces our weight structure, the $\omega$-valuation monoids (cf. [22]), and the weighted automata we want to discuss in this paper. At the end of this section, we give our first main result, the comparison of Muller and Büchi acceptance.

An alphabet denotes a finite set of symbols. Let $\mathbb{N}$ be the set of non-negative integers.

A monoid $(\mathcal{D}, +, 0)$ is called complete, if it is equipped with sum operations $\sum_I : \mathcal{D}^I \to \mathcal{D}$ for all families $(a_i \mid i \in I)$ of elements of $\mathcal{D}$, where $I$ is an arbitrary index set, such that the following conditions are satisfied:

\begin{enumerate}[(i)]
  \item $\sum_{i \in \emptyset} d_i = 0$, $\sum_{i \in \{k\}} d_i = d_k$, $\sum_{i \in \{j,k\}} d_i = d_j + d_k$ for $j \neq k$, and
  \item $\sum_{j \in J} \left( \sum_{i \in I_j} d_i \right) = \sum_{i \in I} d_i$ if $\bigcup_{j \in J} I_j = I$ and $I_j \cap I_k = \emptyset$ for $j \neq k$.
\end{enumerate}

This means that a monoid $\mathcal{D}$ is complete if it has infinitary sum operations (i) that are an extension of the finite sums and (ii) that are associative and commutative (cf. [28]).

For a set $S$ we denote by $C \subseteq \mathbb{N}$, $D \subseteq S$ and $C^\omega$ a finite set of symbols. Let $\mathcal{D}^\omega = \bigcup_{C \subseteq \mathbb{N}} \mathcal{D}^C$. An $\omega$-valuation monoid $(\mathcal{D}, +, \text{Val}^\omega)$ consists of a complete monoid $(\mathcal{D}, +, 0)$ and an $\omega$-valuation function $\text{Val}^\omega : (\mathcal{D}^\omega)^\omega \to \mathcal{D}$ such that $\text{Val}^\omega(d_i)_{i \in \mathbb{N}} = 0$ whenever $d_i = 0$ for some $i \in \mathbb{N}$. A product $\omega$-valuation monoid ($\omega$-pv-monoid) is a tuple $(\mathcal{D}, +, \text{Val}^\omega, \omega, 0, 1)$ where $(\mathcal{D}, +, \text{Val}^\omega, 0)$ is an $\omega$-valuation monoid, $\omega : \mathcal{D}^2 \to \mathcal{D}$ is a product function and further $\mathbb{I} \in \mathcal{D}$, $\text{Val}^\omega(\mathbb{I}^\omega) = \mathbb{I}$ and $0 \circ d = d \circ 0 = 0$, $1 \circ d = d \circ 1 = d$ for all $d \in \mathcal{D}$.

A monoid $(\mathcal{D}, +, 0)$ is called idempotent if $d + d = d$ for all $d \in \mathcal{D}$. An $\omega$-valuation monoid $(\mathcal{D}, +, \text{Val}^\omega, 0)$ is equally called idempotent if its underlying monoid $(\mathcal{D}, +, 0)$ is idempotent.

In [11, 22], $\omega$-valuation monoids are classified by specific properties. More specific $\omega$-valuation monoids will later lead to more loose restrictions on our logic. Due to space constraints, we omit properties on $\omega$-valuation monoids here and refer the interested reader to [11]. Additionally, we will only present one possible restriction on our logic.

Example 1 ($\omega$-valuation monoids). The first two examples are inspired by [7].

1. Let $\widehat{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ and $-\infty + \infty = -\infty$. Then $(\widehat{\mathbb{R}}, \sup, \text{lim avg}, +, -\infty, 0)$ is an $\omega$-pv-monoid where $\text{lim avg}(d_i)_{i \in \mathbb{N}} = \lim \inf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} d_i$.

2. Let $\widehat{\mathbb{R}} = \{ x \in \mathbb{R} \mid x \geq 0 \} \cup \{-\infty\}$. Then $(\widehat{\mathbb{R}}, \sup, \text{disc}, +, -\infty, 0)$ for $0 < \lambda < 1$ is an $\omega$-pv-monoid where $\text{disc}(d_i)_{i \in \mathbb{N}} = \lim_{n \to \infty} \sum_{i=0}^{n} \lambda^i d_i$.

3. Any complete semiring $(S, \oplus, \otimes, 0, 1)$ is an $\omega$-pv-monoid $(S, \oplus, \otimes, 0, 1)$.

As it simplifies the logical characterization, we follow [23, 15] and use a restricted form of pushdown automaton. We call it simple $\omega$-pushdown automaton. For the unweighted setting, we proved in [15] that this automaton model is expressively equivalent to general $\omega$-pushdown automata; for finite words, this equivalence is hidden in [4]. For the weighted case and for continuous semirings, we show a corresponding result for finite words in [13]. For weights in continuous semirings and for infinite words, we showed in [12, 14] that all $\omega$-algebraic series are recognized by weighted simple $\omega$-pushdown automata.

Simple $\omega$-pushdown automata are realtime, i.e. they do not use $\varepsilon$-transitions. Additionally, we restrict transitions in a way to only allow either to keep the stack unaltered, to push one symbol or to pop one symbol. Thus, let $\mathcal{S}(\Gamma) = \{\downarrow\} \times \Gamma \cup \{\#\} \cup \{\uparrow\} \times \Gamma$ be the set of stack commands for a stack alphabet $\Gamma$. Note that this implies that the automaton can only read the top of the stack when popping it. Additionally, for technical reasons, we start runs with an empty stack and therefore allow to push onto the empty stack.
Definition 2. An (unweighted) $\omega$-pushdown automaton ($\omega$PDA) over the alphabet $\Sigma$ is a tuple $M = (Q, \Gamma, T, I, F)$ where
- $Q$ is a finite set of states,
- $\Gamma$ is a finite stack alphabet,
- $T \subseteq Q \times \Sigma \times Q \times \mathcal{S}(\Gamma)$ is a set of transitions,
- $I \subseteq Q$ is the set of initial states,
- $F \subseteq Q$ is a set of (Büchi-accepting) final states.

Definition 3. A weighted $\omega$-pushdown automaton ($\omega$WPDA) over the alphabet $\Sigma$ and the $\omega$-valuation monoid $(D, +, \text{Val}, 0)$ is a tuple $M = (Q, \Gamma, T, I, F, wt)$ where
- $M$ is called a $\omega$WPDA over $\Sigma$,
- $wt: T \rightarrow D$ is a weight function.

Definition 4. A Muller-accepting $\omega$-pushdown automaton over the alphabet $\Sigma$ is a tuple $M = (Q, \Gamma, T, I, F)$ where $Q, \Gamma, T, I$ are defined as for $\omega$PDA, but $F \subseteq 2^Q$ is a set of Muller-accepting subsets of $Q$. Similarly, a weighted Muller-accepting $\omega$-pushdown automaton over the alphabet $\Sigma$ and the $\omega$-valuation monoid $D$ is a tuple $M = (Q, \Gamma, T, I, F, wt)$.

A configuration of an $\omega$PDA or $\omega$WPDA is a pair $(q, \gamma)$, where $q \in Q$ and $\gamma \in \Gamma^*$. We define the transition relation between configurations as follows. Let $\gamma \in \Gamma^*$ and $t \in T$. For $t = (q, a, q', (\downarrow, A))$, we write $(q, \gamma) \xrightarrow{\rho} (q', A\gamma)$. For $t = (q, a, q', \#)$, we write $(q, \gamma) \xrightarrow{\rho} (q', \gamma)$. Finally, for $t = (q, a, q', (\uparrow, A))$, we write $(q, A\gamma) \xrightarrow{\rho} (q', \gamma)$. These three types of transitions are called push, internal and pop transitions, respectively.

We denote by label$(q, a, q', s) = a$ the label and by state$(q, a, q', s) = q$ the state of a transition. Both, as well as the function $wt$ will be extended to infinite sequences of transitions by letting label$(\omega(t_i), i \geq 0) = \text{label}(\omega(t_i))_{i \geq 0}$ for the infinite word constructed from the labels and similar for state$(\omega(t_i), i \geq 0) \in Q^\omega$ and for $wt(\omega(t_i), i \geq 0) \in D^\omega$.

An infinite sequence of transitions $\rho = (t_i)_{i \geq 0}$ with $t_i \in T$ is called a run of the $\omega$WPDA or $\omega$PDA $M$ on $w = \text{label}(\rho)$ iff there exists an infinite sequence of configurations $(p_i, \gamma_i)_{i \geq 0}$ with $p_0 \in I$ and $\gamma_0 = \epsilon$ such that $(p_i, \gamma_i) \xrightarrow{\rho} (p_{i+1}, \gamma_{i+1})$ for each $i \geq 0$. We abbreviate a run $\rho = (t_i)_{i \geq 0}$ with $(p_0, \gamma_0) \xrightarrow{\rho} (p_1, \gamma_1) \xrightarrow{\rho} \cdots$ such that the word becomes visible.

For an infinite sequence of states $(q_i)_{i \geq 0}$, let Inf$(q_i)_{i \geq 0} = \{q \mid q = q_i \text{ for infinitely many } i \geq 0\}$ be the set of states that occur infinitely often. For Büchi-accepting automata, a run $\rho$ is called successful if Inf(state$(\rho)$) \cap $F \neq \emptyset$. For Muller-accepting automata, a run $\rho$ is called successful if Inf(state$(\rho)$) \subseteq $F$. For an $\omega$PDA $M = (Q, \Gamma, T, I, F)$, the language accepted by $M$ is denoted by $L(M) = \{w \in \Sigma^\omega \mid \exists$ successful run of $M$ on $w\}$. A language $L \subseteq \Sigma^\omega$ is called $\omega$PDA-recognizable if there exists an $\omega$PDA $M$ with $L(M) = L$. For an $\omega$WPDA $M$, we introduce the following function $\|M\|: \Sigma^\omega \rightarrow D$ which is called the behavior of $M$ and which is defined by $\|M\|(w) = \sum(\text{Val}(wt(\rho)))$ where $\rho$ successful run of $M$ on $w$.

An $\omega$PDA or $\omega$WPDA $M$ over $\Sigma$ is called unambiguous if there exists at most one successful run of $M$ on every word $w \in \Sigma^\omega$. If there exists an unambiguous $\omega$WPDA $M$ with $L(M) = L$, the language $L$ is called unambiguous.

Any function $s: \Sigma^\omega \rightarrow D$ is called a series over $\Sigma$ and $D$. The set of all such series is denoted by $D(\langle \Sigma^\omega \rangle)$. Every series $s: \Sigma^\omega \rightarrow D$ which is the behavior of some $\omega$WPDA over $D$ is called a $\omega$WPDA-recognizable.

An $\omega$WPDA $M = (Q, \Gamma, T, I, F, wt)$ that only uses internal transitions, i.e., for which $\Gamma = \emptyset$ and for all transitions $t = (q, a, q', s) \in T$ holds $s = \#$, is called a weighted finite automaton, or short $\omega$WFA. Series that are the behavior of some $\omega$WFA are called $\omega$WFA-recognizable.
Example 5: Weighted $\omega$-pushdown automaton over the alphabet $\Sigma = \{\text{req}, \text{ans}, \text{call}, \text{ret}, \text{wait}\}$ and the $\omega$-valuation monoid $\bar{R}$. The value after the “:” are the used weights 0 and 1.

**Figure 1** Example 5: Weighted $\omega$-pushdown automaton over the alphabet $\Sigma = \{\text{req}, \text{ans}, \text{call}, \text{ret}, \text{wait}\}$ and the $\omega$-valuation monoid $\bar{R}$. The value after the “:” are the used weights 0 and 1.

**Figure 2** Proof of Theorem 6: The states 1, 2, 3, and 4 stand for the set of states that are initial and final, initial but not final, final but not initial, or neither initial nor final, respectively. Groups 1 and 3 are copied into $\bar{1}$ and $\bar{3}$. Transitions into $\bar{1}$ and $\bar{3}$ are only allowed from originally non-accepting states.

**Example 5 ($\omega$WPDA).** We extend the $\omega$-pv-monoid 1 of Example 1 as $(\bar{R}, \sup, \text{specialavg}, +, -\infty, 0)$ where we define a new $\omega$-valuation function to count and take the average of the counted values. Let $h$ be a function that maps natural numbers to strings as follows.

$h: \mathbb{N} \to \{0, 1\}^*$, $n \mapsto 011\ldots1$ $n$-times

Then we extend $h$ to infinite sequences of natural numbers $h: \mathbb{N}^\omega \to \{0, 1\}^\omega$ in the natural way. We will consider its inverse where we have for instance $h^{-1}(011100110011110\ldots) = 324\ldots$. Then let $\text{specialavg} = \lim \text{avg} \circ h^{-1}$. For $w \notin (01^*0)^\omega$ we set $\text{specialavg}(w) = -\infty$.

Now, we define an automaton $A$ as shown in Figure 1. We let $A = ((1, 2), \{X, Y\}, T, (1, 1, \text{wt}))$ be an $\omega$WPDA over the alphabet $\Sigma = \{\text{req}, \text{ans}, \text{call}, \text{ret}, \text{wait}\}$, where $T$ is defined as shown in the Figure and the weights are indicated after the colon symbol.

The automaton simulates some kind of (web) server that takes requests from clients and answers them. For every request, the server has to call some amount of other services and await their returns. Only when all calls have been returned, the server answers the original request. This is a context-free property. Only runs that always eventually return to state 1 to serve new clients are considered valid.

Every call, return, or wait takes one second to operate and this operation time is accounted for in the weight. The specialavg operation sums up all the waiting time per request and returns the long run average response time.

We now state our first main result.

**Theorem 6.** Let $s: \Sigma^\omega \to D$ be a series. The following are equivalent:

- $s$ is recognizable by a Büchi-accepting $\omega$WPDA,
- $s$ is recognizable by a Muller-accepting $\omega$WPDA.

**Proof Idea.** In the direction Büchi to Muller acceptance, the standard approach works also in the weighted case.
For the other direction, the standard approach usually employs a special set of accepting states that have to be traversed to be accepted. This construction needs to be adjusted as it creates infinitely many possible runs in the Büchi automaton for every run of the Muller automaton.

A solution to this problem was presented in [25] whose construction allows exactly one entry point into the special set of accepting states. Entering the group of accepting states is forbidden from a state that is already accepting. In this way, the only successful runs are the ones that switch from the original states to the new group of accepting states at the last possible moment. In contrast to [25], we cannot assume an initially normalized automaton to solve the remaining question of the initial states that are also final.

Instead, the automaton decides non-deterministically if it will eventually see a non-final entry point into the special set of accepting states. Entering the group of accepting states is possible only if the automaton has a non-final accepting state. In this way, the only successful runs are forbidden from a state that is already accepting. In this way, the only successful runs are infinitely many possible runs in the Büchi automaton for every run of the Muller automaton.

To allow final states of both original Büchi-accepting automata to be visited alternately, we use the standard construction for intersecting unweighted Büchi automata for

\[ L = \{ a^i b^j c^k d^l \mid i = j \text{ or } j = k \} \]

and intersect it with the
We further define \( \Delta \), there exist an alphabet \( \Delta \). The language \( |D| \subseteq \omega \). If \( D \) is idempotent, the series \( D^N (\langle \Sigma^\omega \rangle) \) with \( N \) meaning Nivat denote the set of series \( s \) over \( \Sigma \) and \( D \) such that there exist an alphabet \( \Delta \), mappings \( h: \Delta \to \Sigma \) and \( r: \Delta \to D \) and an \( \omega \)-PDA-recognizable language \( L \subseteq \Delta^\omega \) such that

\[
s = h ((\text{Val} \circ r) \cap L).
\]

We further define \( D^N \) like \( D^N \) with the difference that \( L \) is an unambiguous (deterministic, respectively) \( \omega \)-PDA-recognizable language.

**Example 11.** We extend the \( \omega \)-PDA-monoid 1 of Example 1 as \( (\hat{R}, \sup, \text{partialavg}, +, -\infty, 0) \) where we add a new value \( d \) that will later be ignored, i.e., \( \hat{R} = \hat{R} \cup \{d\} \). We set \( \sup(-\infty, d) = d \) and \( \sup(r, d) = r \) for every \( r \in \hat{R} \). We define a new \( \omega \)-valuation function to ignore \( d \) and take the average of the remaining values. Let now \( h \) be defined as follows.

\[
h: \hat{R} \to \hat{R}^*, \quad r \mapsto r, \quad \text{for } r \in \hat{R}
\]

\[
d \mapsto \varepsilon
\]

Then we extend \( h \) to infinite sequences \( h: \hat{R}^\omega \to \hat{R}^\omega \) in the natural way. Now let \( \text{partialavg} = \lim \text{avg} \circ h \) and \( \Sigma = \{a, b\} \). We make the following definitions:

- \( \Delta = \Sigma \times \{0, 1, \ldots, 6\} \),
- \( L = \{\langle \sigma_1, d_1 \rangle \langle \sigma_2, d_2 \rangle \langle \sigma_3, d_3 \rangle \cdots | d_i = i \mod 7, \sigma_i \in \Sigma\} \),
- \( r(b, i) = d \) for all \( i \in \{0, \ldots, 6\} \) and \( r(a, i) = \begin{cases} 1, & \text{if } 5 \leq i \leq 6 \\ 0, & \text{otherwise} \end{cases} \)
- \( h(\sigma, i) = \sigma \)

The language \( L \subseteq \Delta^\omega \) is obviously \( \omega \)-PDA-recognizable. As we will see in the following theorem, the series \( s = h ((\text{Val} \circ r) \cap L) \in \hat{R}^N (\langle \Sigma^\omega \rangle) \) is \( \omega \)-PDA-recognizable because \( \hat{R} \) is idempotent. The series \( s \) calculates the greatest accumulation point of the ratio of events \( a \) happening at the weekend (days 5 and 6) compared to all occurrences of events \( a \).

The following is the second main Nivat-like decomposition result.

**Theorem 12.** Let \( \Sigma \) be an alphabet and \( (D, +, \text{Val}, 0) \) an \( \omega \)-valuation monoid. Then,

\[
D^N (\langle \Sigma^\omega \rangle) = D^N (\langle \Sigma^\omega \rangle) = D^N (\langle \Sigma^\omega \rangle) \subseteq D^N (\langle \Sigma^\omega \rangle).
\]

If \( D \) is idempotent, \( D^N (\langle \Sigma^\omega \rangle) = D^N (\langle \Sigma^\omega \rangle) \).

**Proof.** First, we show \( D^N (\langle \Sigma^\omega \rangle) \subseteq D^N (\langle \Sigma^\omega \rangle) \): Let \( s \in D^N (\langle \Sigma^\omega \rangle) \). Thus there exists an \( \omega \)-PDA \( M = (Q, \Gamma, T, I, F, wt) \) over \( \Sigma \) such that \( \|M\| = s \). We will show that there exist \( \Delta, h, r \) and \( L \) such that \( s = h ((\text{Val} \circ r) \cap L) \).
Let $\Delta = T$ and let $r = \omega: \Delta \rightarrow D$. We define $h: \Delta \rightarrow \Sigma$ by $h((q, a, q', s)) = a$. Note that the automaton does not allow $\epsilon$-transitions and therefore, $h$ is well-defined. We construct an unweighted $\omega$PDA $M' = (Q, \Gamma, T', I, F)$ over $\Delta$ with
$$T' = \{ (q, (a, p, s), p, p) \mid (q, a, p, s) \in T \}.$$ Note that $M'$ accepts exactly the successful runs of $M$. As there is at most one transition of $M'$ with label $(q, a, p, s)$, $M'$ is deterministic (and unambiguous). Define $L = L(M')$.

Let $w \in \Sigma^\omega$. Therefore,
$$h(Val^\omega \circ r) \cap L(w) = \sum \left( (((Val^\omega \circ r) \cap L)(w') \mid w' \in \Sigma^\omega \text{ and } h(w') = w) = \sum (Val^\omega r(w') \mid w' \in L \text{ and } h(w') = w) = \sum (Val^\omega wt(w') \mid w' \text{ successful run of } M \text{ on } w) = ||M||(w) = s(w).$$

The inclusions $D^\omega_{\text{DET}}(\langle \Sigma^\omega \rangle) \subseteq D^\omega_{\text{UNAMB}}(\langle \Sigma^\omega \rangle) \subseteq D^\omega(\langle \Sigma^\omega \rangle)$ is true by definition. The converse $D^\omega_{\text{UNAMB}}(\langle \Sigma^\omega \rangle) \subseteq D^\omega_{\text{REC}}(\langle \Sigma^\omega \rangle)$ is proven by the closure properties of Lemmas 7, 8 and 9(1).

If $D$ is idempotent, by Lemmas 7, 8 and 9(2), we get $D^\omega(\langle \Sigma^\omega \rangle) \subseteq D^\omega_{\text{REC}}(\langle \Sigma^\omega \rangle)$. □

The inclusion $D^\omega(\langle \Sigma^\omega \rangle) \subseteq D^\omega_{\text{REC}}(\langle \Sigma^\omega \rangle)$ does not hold in general for non-idempotent $D$. For the proof, one can consider an adaptation of the counterexample after Lemma 9.

### 4 Logic for Weighted $\omega$-Pushdown Automata

The third main goal of this paper is a logical characterization of weighted $\omega$-context-free languages. This section introduces this logic. It is based on [15, 17, 29].

Our logic has three components. The first component is a monadic second-order logic (MSO). By Büchi, Elgot, Trakhtenbrot [5, 6, 26, 33], MSO has the same expressive power on finite and infinite words as finite automata.

The second component adds the weights to the logic. Here, this is done by a new layer of formulas that are to be interpreted quantitatively, using the operations of the $\omega$-pv-monoid. Formulas of the unweighted part of the logic will be interpreted as 0 or 1 in the $\omega$-pv-monoid.

The third component is a dyadic second-order predicate – a binary relation that is called matching relation. Every formula will be allowed to use exactly one such predicate to link positions in words. A matching relation has a specific shape that makes it possible to argue about the stack in pushdown automata or the brackets in Dyck languages or even about the nesting in nested words.

Let $w \in \Sigma^\omega$. The set of all positions of $w$ is $\mathbb{N}$. A binary relation $M \subseteq \mathbb{N} \times \mathbb{N}$ is a matching (cf. [29]) if $M$ is compatible with $<$, i.e., $(i, j) \in M$ implies $i < j$, if each element $i$ belongs to at most one pair in $M$, and if $M$ is noncrossing, i.e., $(i, j) \in M$ and $(k, l) \in M$ with $i < k < j$ imply $i < l < j$. Let 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, respectively. We fix a matching variable $\mu \notin V_1 \cup V_2$. Let $V = V_1 \cup V_2 \cup \{\mu\}$. Furthermore, $D$ is always an $\omega$-pv-monoid ($D, +, Val^\omega, 0, 1$).

**Definition 13.** Let $\Sigma$ be an alphabet. The set $\omega\text{MSO}(D, \Sigma)$ of weighted matching $\omega$-MSO formulas over $\Sigma$ and $D$ is defined by the extended Backus-Naur form
$$\beta ::= P_a(x) \mid x \leq y \mid x \in X \mid \mu(x, y) \mid \gamma \beta \mid \beta \lor \beta \mid \exists x. \beta \mid \exists X. \beta$$
$$\varphi ::= d \mid \beta \mid \varphi \lor \varphi \mid \varphi \lor \varphi \mid \oplus_x \varphi \mid \bigoplus X \varphi \mid Val_x \varphi$$
where $a \in \Sigma$, $d \in D$, $x, y \in V_1$ and $X \in V_2$. We call all formulas $\beta$ boolean formulas.
The logical counterparts of the set of all formulas \( \text{Val} \) is
\[
\{ \varphi \oplus \psi \mid \text{Val}(\varphi) = \text{Val}(\psi) \},
\]
where \( \varphi, \psi \in \text{Val} \). The updates \( \text{Val}_x \) of the sequence of infinitely many formulas \( \varphi \), each of them instantiated with a position \( x \in \mathbb{N} \).

Let \( \varphi \in \omega\text{-MSO}(D, \Sigma) \) be a formula, \( w = a_0 a_1 a_2 \ldots \in \Sigma^\omega \) and let \( \sigma \) be a \( \mathcal{V} \)-assignment. We inductively define \( \sigma(w, \sigma) \equiv \varphi \) if \( \varphi \) is boolean and \( [\varphi] : \Sigma^\omega \times \mathcal{V} \to D \) if \( \varphi \) is non-boolean over the structure of \( \varphi \) as shown in the Table 1, where \( a \in \Sigma, d \in D, x, y \in V_1 \) and \( X \in V_2 \). The logical counterparts \( \land, \neg, \forall x, \forall X, \varphi, x \neq y, x < y \) and \( i < j < k \) can be gained from negation and the existing operators.

Note how formulas \( \varphi \odot \psi \) are evaluated by the product operation \( \odot \) in the \( \omega \)-pv-monoid and also note that our \( \omega \)-WPDA's do not have direct access to this operation. However, the first two layers of our logic, the \( \omega\text{-MSO}(D, \Sigma) \) formulas, will be translated into weighted nested \( \omega \)-word automata and simple series of those automata are closed under intersection and therefore, \( \odot \) can be translated by a product construction.

We now define \( \text{MATCHING}(\mu) \in \omega\text{-MSO}(D, \Sigma) \) which ensures that \( \mu \in \text{Match}(\mathbb{N}) \). Let
\[
\text{MATCHING}(\mu) = \forall x \forall y. (\mu(x, y) \rightarrow x < y) \land
\]
\[
\forall x \forall y \forall k. ((\mu(x, y) \land k \neq x \land k \neq y) \rightarrow \neg \mu(x, k) \land \neg \mu(k, x) \land \neg \mu(y, k) \land \neg \mu(k, y)) \land
\]
\[
\forall x \forall y \forall k \forall l. ((\mu(x, y) \land \mu(k, l) \land x < k < y) \rightarrow x < l < y).
\]

**Definition 14.** The set of formulas of weighted matching \( \omega \)-logic over \( \Sigma \) and \( D \), \( \omega\text{ML}(D, \Sigma) \), denotes the set of all formulas \( \psi \) of the form
\[
\psi = \bigoplus_\mu (\varphi \odot \text{MATCHING}(\mu)),
\]
for short \( \psi = \bigoplus_\mu^\text{match} \varphi \), where \( \varphi \in \omega\text{-MSO}(D, \Sigma) \).

Let \( \psi = \bigoplus_\mu^\text{match} \varphi \), \( w \in \Sigma^\omega \) and let \( \sigma \) be a \( \mathcal{V} \)-assignment. Then,
\[
[\psi](w, \sigma) = \sum_{M \in \text{Match}(\mathbb{N})} ([\varphi](w, \sigma[\mu/M])).
\]
Let \( \psi \in \omega\text{ML}(D, \Sigma) \). We denote by \( \text{Free}(\psi) \subseteq V \) the set of free variables of \( \psi \). A formula \( \psi \) with \( \text{Free}(\psi) = \emptyset \) is called a sentence. For a sentence \( \psi \), \( \llbracket \psi \rrbracket(w, \sigma) \) does not depend on \( \sigma \). It will therefore be omitted and we only write \( \llbracket \psi \rrbracket(w) \) where the series \( \llbracket \psi \rrbracket: \Sigma^\omega \to D \) is called defined by \( \psi \). A series \( s: \Sigma^\omega \to D \) is weighted-\( \omega \)-ML-definable if there exists a sentence \( \psi \in \omega\text{ML}(D, \Sigma) \) such that \( \llbracket \psi \rrbracket = s \).

**Example 15.** Here we define a logical sentence that defines the same series as in Example 5. Consider the same \( \omega\text{-Pushdown Automata} \) as there.

The subformula \( p_{\text{structure}} \) ensures that the first symbol is a request and that requests occur directly after answers. The formula \( p_{\text{matching}} \) relates corresponding call and returns and forbids calls without returns and vice versa. Furthermore, calls must be returned before giving the answer to the clients. Finally, the server has to serve clients infinitely often.

\[
\begin{align*}
\text{next}(x, y) &= x < y \land \neg(\exists z. x \leq z \leq y) \\
\text{first}(x) &= \forall y. x \leq y \\
p_{\text{structure}} &= \forall x. (\text{first}(x) \to P_{\text{req}}(x)) \land \forall x \forall y. \text{next}(x, y) \to \langle P_{\text{ans}}(x) \leftrightarrow P_{\text{req}}(y) \rangle \\
p_{\text{matching}} &= \forall x. P_{\text{call}}(x) \to \exists y. P_{\text{ret}}(y) \land \mu(x, y) \\
&\quad \land \forall y. P_{\text{ret}}(y) \to \exists x. P_{\text{call}}(x) \land \mu(x, y) \\
&\quad \land \forall x. \forall y. [\mu(x, y) \to \neg(\exists z. x \leq z \leq y \land P_{\text{ans}}(z))] \\
&\quad \land \forall x. \forall y. [\mu(x, y) \to \langle (P_{\text{req}}(x) \land P_{\text{ans}}(y)) \lor (P_{\text{call}}(x) \land P_{\text{ret}}(y)) \rangle] \\
p_{\text{inf\_serving}} &= \forall x. \exists y. (x < y \land P_{\text{req}}(y)) \\
\varphi_{\text{unweighted}} &= p_{\text{structure}} \land p_{\text{matching}} \land p_{\text{inf\_serving}} \\
\varphi_{\text{weighted}} &= \text{Val}_x \left[ \langle P_{\text{req}}(x) \lor P_{\text{ans}}(x) \rangle \oplus \langle (P_{\text{call}}(x) \lor P_{\text{ret}}(x) \lor P_{\text{wait}}(x)) \odot 1 \rangle \right]
\end{align*}
\]

Then, we quantify over the matching variable and only consider the weight calculated in \( \varphi_{\text{weighted}} \) if the formula \( \varphi_{\text{unweighted}} \) is true:

\( \psi = \bigoplus_{\mu} \varphi_{\text{unweighted}} \odot \varphi_{\text{weighted}} \)

Finally, we have \( \llbracket \psi \rrbracket = \|A\| \) for the \( \omega\text{WPDA} \ A \) of Example 5.

The weighted matching \( \omega \)-logic, \( \omega\text{ML}(D, \Sigma) \), contains exactly one predicate \( \mu \) and exactly one quantification over it. This corresponds to the behavior of pushdown automata where exactly one pushdown tape is used. In contrast, the pushdown automaton uses the \( \omega \)-valuation function \( \text{Val}^\omega \) only once per run and never recursively. As formulas \( \text{Val}_x, \text{Val}_y \varphi \in \omega\text{MSO}(D, \Sigma) \) are not always translatable into automata, we follow [11, 17, 22] and define some possible restrictions of our logic.

The set of *almost boolean formulas* is the smallest set of all formulas of \( \omega\text{MSO}(D, \Sigma) \) containing all constants \( d \in D \) and all boolean formulas which is closed under \( \oplus \) and \( \odot \).

**Definition 16 ([11, 22]).** Let \( \varphi \in \omega\text{MSO}(D, \Sigma) \). We call \( \varphi \)

1. **strongly-\( \odot \)**-restricted if for all subformulas \( \mu \odot \nu \) of \( \varphi \):
   
   either \( \mu \) and \( \nu \) are almost boolean or \( \mu \) is boolean or \( \nu \) is boolean.
2. **Val**-restricted if for all subformulas \( \text{Val}_x \mu \) of \( \varphi \), \( \mu \) is almost boolean.
3. **syntactically** restricted if it is both Val-restricted and strongly-\( \odot \)-restricted.

Let now \( \psi = \bigoplus_{\mu} \varphi \in \omega\text{ML}(D, \Sigma) \). For \( X \in \{ \text{strongly-}\odot, \text{Val}, \text{syntactically} \} \), we also say that \( \psi \) is \( X \)-restricted if \( \varphi \) is \( X \)-restricted.
The following will be the third main result. Regular $\omega$-pv-monoids will be defined in the next section on page 11 as they depend on nested $\omega$-word automata. We will prove the following theorem in Section 6.

\textbf{Theorem 17.} Let $D$ be a regular $\omega$-pv-monoid and $s : \Sigma^\omega \to D$ be a series. The following are equivalent:
1. $s$ is $\omega$-WPDA-recognizable
2. There is a syntactically restricted $\omega$ML($D, \Sigma$)-sentence $\varphi$ with $[\varphi] = s$.

5 Weighted Nested $\omega$-Word Languages

The $\omega$MSO($D, \Sigma$) formulas correspond exactly to weighted nested $\omega$-word languages [11] (cf. [1]). In fact, without considering the existential quantification over the matching relation $\exists \text{match}_\mu$, the matching must explicitly be encoded in the words; the result is a nested word. Because of limited space, we refrain from a detailed definition of weighted nested $\omega$-word automata and refer the reader to [11].

A nested $\omega$-word $nw$ over $\Sigma$ is a pair $(w, \nu) = (a_{0}a_{1}a_{2} \ldots , \nu)$ where $w \in \Sigma^{\omega}$ is an $\omega$-word and $\nu \in \text{Match}(\mathbb{N})$ is a matching relation over $\mathbb{N}$. Let $NW^\omega(\Sigma)$ denote the set of all nested $\omega$-words over $\Sigma$. For two positions $i,j \in \mathbb{N}$ with $\nu(i, j)$, we call $i$ a call position and $j$ a return position. If $i$ is neither call nor return, we call it an internal position. A position $i$ for $i \in \mathbb{N}$ is called top-level if there exist no positions $j, k \in \mathbb{N}$ with $j < i < k$ and $\nu(j, k)$.

A weighted stair Muller nested word automaton ($\omega$WNWA) as defined in [11] is a Muller automaton on nested $\omega$-words $(w, \nu)$ that for every return position has access to the state at the corresponding call position. The stair Muller acceptance condition is a Muller acceptance condition used exclusively on top-level position, i.e., only the states occurring infinitely often in the infinite sequence of top-level positions are considered.

Every function $s : NW^\omega(\Sigma) \to D$ is called a nested $\omega$-word series ($nw$-series). Every $nw$-series $s$ which is the behavior of some $\omega$WNWA over $D$ is called $\omega$WNWA-recognizable.

We will now discuss how $\omega$MSO is an equivalent logic to $\omega$WNWAs. Note that $\omega$MSO($D, \Sigma$) formulas may contain the free variable $\mu$. Given a nested word $nw = (w, \nu)$, we let $\sigma(\mu) = \nu$ and make no difference between $(w, \sigma) \in \Sigma^{\omega} \times \{\mu\} \to \text{Match}(\mathbb{N})$ and the nested word $nw$. We extend the semantics definitions as follows. Let $\varphi \in \omega$MSO($D, \Sigma$) and $\text{Free}(\varphi) \subseteq \{\mu\}$, then we define $[\varphi]_{nw} : NW^\omega(\Sigma) \to D$ by letting

$$[\varphi]_{nw}(w, \nu) = [\varphi](w, \sigma) \quad \text{for } \sigma(\mu) = \nu.$$ 

Let $d \in D$ denote the constant series with value $d$, i.e., $d(nw) = d$ for each $nw \in NW^\omega(\Sigma)$. An $\omega$-pv-monoid $D$ is called regular if all constant series of $D$ are $\omega$WNWA-recognizable. In other words, $D$ is regular if for any alphabet $\Sigma$, we have: For each $d \in D$, there exists an $\omega$WNWA $A_d$ with $\|A_d\| = d$.

Note that for this paper, regularity of $\omega$-pv-monoids is defined by the means of $\omega$WNWAs. In the proof of Theorem 18, this is used in the structural induction as a logical formula $\varphi = d$, for a weight $d$, can otherwise not necessarily be translated into an automaton.

Sufficient properties for an $\omega$-pv-monoid to be regular are shown in [22]. Especially left-multiplicative and left-$\text{Val}^{\omega}$-distributive $\omega$-pv-monoids are regular, i.e., if we have $d \circ \text{Val}^{\omega}((d_{i})_{i \in \mathbb{N}}) = \text{Val}^{\omega}((d \circ d_{i})_{i \in \mathbb{N}})$ or $d \circ \text{Val}^{\omega}((d_{i})_{i \in \mathbb{N}}) = \text{Val}^{\omega}(d \circ d_{i}, (d_{i})_{i \geq 0})$ for all $d \in D$ and $(d_{i})_{i \in \mathbb{N}} \in D^{\omega}$, then $D$ is regular because we can easily construct $\omega$WNWAs (and even $\omega$WFAs) for every constant series. All $\omega$-pv-monoids in Example 1 are regular.
Theorem 18 ([11]). Let $D$ be a regular $\omega$-pv-monoid and $s: NW^\omega(\Sigma) \rightarrow D$ be a nw-series. Then the following are equivalent:
1. $s$ is $\omega$WNWA-recognizable.
2. There is a syntactically restricted $\omega$MSO($D, \Sigma$)-formula $\varphi$ with $\text{Free}(\varphi) \subseteq \{\mu\}$ and $\models_{\text{nw}} \varphi = s$.

The mapping $\pi: NW^\omega(\Sigma) \rightarrow \Sigma^\omega$ removes the nesting relation from the nested word, i.e., for $nw = (w, \nu)$, we define $\pi(nw) = w$. This can be extended to nw-series $s: NW^\omega(\Sigma) \rightarrow D$ by setting $\pi(s)(w) = \sum_{nw \in \pi^{-1}(w)} s(nw)$ which equals $\pi(s)(w) = \sum_{M \in \text{Match}(\mathbb{N})} s(w, \emptyset[M])$.

The following is crucial for the rest of the paper.

Lemma 19. Let $s: NW^\omega(\Sigma) \rightarrow D$ be an $\omega$WNWA-recognizable nw-series. Then the series $\pi(s): \Sigma^\omega \rightarrow D$ is $\omega$WPDA-recognizable.

For unweighted languages, there is a similar proof in [4, 15]. Here, the proof is more complicated because the acceptance conditions differ. We have to construct a Büchi-accepting pushdown automaton from a stair Muller nested-word automaton.

Proof. By Theorem 6, it suffices to construct a Muller-accepting $\omega$WPDA from a given $\omega$WNWA. We simulate the transitions of the $\omega$WNWA by pushing states onto the stack. Additionally, we enrich the states by the information if we are top-level or not. This information is also pushed onto the stack for the reconstruction of the top-level property upon popping. Furthermore, we allow the new Muller-accepting $\omega$WPDA to visit arbitrary subsets of states that are not top-level in between the original Muller-accepting states.

6 Equivalence of Logic and Automata

This section proves the equivalence of $\omega$ML($D, \Sigma$) and weighted simple $\omega$-pushdown automata.

Lemma 20. Let $D$ be a regular $\omega$-pv-monoid and $s: \Sigma^\omega \rightarrow D$ be an $\omega$WPDA-recognizable series. Then $s$ is $\omega$ML-definable by a syntactically restricted $\omega$ML($D, \Sigma$)-sentence.

Proof. The proof builds a syntactically restricted $\omega$ML($D, \Sigma$)-sentence $\theta$ such that $[\theta] = s$. The sentence $\theta$ defines exactly the behavior of an $\omega$WPDA. Hereby, we proceed similarly to [15] and [17, 34, 11].

Lemma 21. Let $D$ be a regular $\omega$-pv-monoid and let $\psi$ be a syntactically restricted $\omega$ML($D, \Sigma$)-sentence. Then $[\psi]: \Sigma^\omega \rightarrow D$ is $\omega$WPDA-recognizable.

Proof. Let $\psi = \bigoplus_{\mathcal{F}_{\text{nw}}} \varphi$ for $\varphi \in \omega$MSO($D, \Sigma$). Apply Theorem 18 to infer that $[\varphi]_{\text{nw}}$ is $\omega$WNWA-recognizable. Now, we use the projection $\pi: NW^\omega(\Sigma) \rightarrow \Sigma^\omega$ of Section 5 to get $\pi([\varphi]_{\text{nw}})(w) = \sum_{M \in \text{Match}(\mathbb{N})} [\varphi](w, \emptyset[M]) = [\psi](w)$. By Lemma 19, $[\psi] = \pi([\varphi]_{\text{nw}})$ is $\omega$WPDA-recognizable.

Proof of Theorem 17. This is immediate by Lemmas 20 and 21.

7 Conclusion

We defined $\omega$-pv-monoids and $\omega$-pushdown automata with weights from $\omega$-pv-monoids. We first generalized a fundamental result of unweighted automata theory: Büchi acceptance and Muller acceptance are expressively equivalent; we can show that this remains the case for weighted simple pushdown automata of infinite words.
For the class of languages recognized by our automata, we proved several closure properties and, as our second main result, a Nivat-like decomposition theorem. It states that the weighted languages in our class are induced by an unweighted context-free language and a very simple weighted part; the two components can be intersected and a projection of this intersection gives us the original language.

The third main result is an expressively equivalent logic. This logic has three layers. The first layer basically describes nested $\omega$-word-languages. The first two layers together describe weighted nested $\omega$-word-languages. The third layer existentially quantifies the matching variable and corresponds to a projection from nested words to context-free languages. In this way, we can apply the Büchi-Elgot-Trakhtenbrot-Theorem for weighted regular nested $\omega$-word-languages to obtain our equivalence result.

The present result raises the question how weighted simple $\omega$-pushdown automata on $\omega$-valuation monoids and therefore also our weighted matching $\omega$-logic relate to a corresponding notion of weighted context-free $\omega$-grammars; for weighted simple $\omega$-pushdown automata over commutative complete star-omega semirings, this was described in [12].

In Theorem 17, it would be desirable to generalize the notion of regular $\omega$-pv-monoids to only require $\omega$WPDA instead of $\omega$WNWA. The classical inductive proof method of Theorem 18 not longer works in this case. However it seems that $\omega$-pv-monoids where constant series are $\omega$WPDA-recognizable but not $\omega$WNWA-recognizable are very artificial.

References


Nivat-Theorem and Logic for Weighted $\omega$-Pushdown Automata


