Visibly Pushdown Transducers with Well-nested Outputs*

Pierre-Alain Reynier† and Jean-Marc Talbot

Aix Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille, France

Visibly pushdown transducers (VPTs) are visibly pushdown automata extended with outputs. They have been introduced to model transformations of nested words, i.e. words with a call/return structure. When outputs are also structured and well nested words, VPTs are a natural formalism to express tree transformations evaluated in streaming. We prove the class of VPTs with well-nested outputs to be decidable in PTIME. Moreover, we show that this class is closed under composition and that its type-checking against visibly pushdown languages is decidable.

1. Introduction

Visibly pushdown automata (VPA) [2] (first introduced as input-driven pushdown automata [4]) are pushdown machines defined over a structured alphabet whose stack behavior is synchronized with the structure of the input word. A structured alphabet is actually partitioned into call, internal and return symbols inducing a nesting structure of the input words [2]. Words over a structured alphabet are called nested words. The structure of the input word induces the behaviour of the VPA regarding the stack: when reading a call symbol the machine must push a symbol onto the stack, when reading a return symbol it must pop a symbol from the stack and when reading an internal symbol the stack remains unchanged. Hence, the nesting level at some position in a word corresponds to the height of the stack at this position. The set of nested words accepted by a VPA is a visibly pushdown language (VPL). Such languages enjoys good properties: emptiness is decidable and they are (effectively) closed under Boolean operations. This implies in particular that inclusion is decidable for VPL.

Visibly pushdown transducers (VPTs) [8, 9, 7, 10] extend visibly pushdown automata with outputs. Each transition is equipped with an output word; a VPT thus transforms an input word into an output word obtained as the concatenation of all the output words produced along a successful run of the machine (*i.e.* a sequence of transitions from an initial configuration to some final configuration) on this input. VPTs are a strict subclass of pushdown transducers (PTs) and strictly extend finite state transducers (FST). Several undecidable problems for PTs become decidable for

*This work has been realized thanks to the participation of LabEx Archimède (ANR-11-LABX-0033) and of the foundation A*MIDEX (ANR-11-IDEX-0001-02), funded by the program "Investissements d'Avenir" lead by the ANR.
†This author was partly supported by the ANR project MACARON (ANR-13-JS02-0002).

VPTs (similarly to FST): functionality (in Ptime), k-valuedness (in co-NPtime) and functional equivalence (EXPtime-complete) [7]. However, some decidability results or valuable properties of finite-state transducers do not hold for VPTs [8]: VPTs are not closed under composition and type-checking against VPA (deciding whether the range of a transducer is included into the language of a given VPA) is undecidable. The main reason of these facts is that ranges of VPTs can be arbitrary context-free languages and the inclusion problem for context-free languages into VPL is undecidable [8].

Unranked trees and more generally hedges can be linearized into nested words over a structured alphabet (such as XML documents). In this case, the matching between call and return symbols is perfect yielding well-nested words. So, VPTs are a suitable formalism to express hedge transformations. Moreover, as they process the linearization from left to right, they are also an adequate formalism to model and analyze transformations in streaming, as shown in [6]. As VPTs output strings, operating on well-nested inputs, they define hedge-to-string transformations. But if the output strings are well-nested too, they indeed define hedge-to-hedge transformations [5].

In [7], by means of a syntactical restriction on transition rules, a class of VPTs whose range contains only well-nested words is presented. Although ranges of such class of VPTs may not be VPL languages, this class enjoys nonetheless good properties: it is closed under composition and type-checking against visibly pushdown languages is decidable. This raises a natural question: does these good properties come from the syntactic definition of this particular subclass or from a semantical property of the range of these VPTs, for instance the fact that it contains only well-nested words ?

In this paper, we consider classes of transductions (that is, of relations) over nested words definable by VPTs. We first consider the class of *globally well-nested* transductions, denoted $\mathcal{G}_{\mathsf{wn}}$, which is the class of VPT transductions whose range contains only well-nested words. This class of transductions naturally defines the class gwnVPT of transducers: a VPT is in gwnVPT if the transduction it defines is in $\mathcal{G}_{\mathsf{wn}}$. Using a result from [11, 3], we prove the class gwnVPT to be decidable in Ptime. Unfortunately, the proposed algorithm does not provide a deep understanding of this class: the desired property of well-nestedness is only a global property for words produced on accepting runs. To circumvent this problem, seeking a class closed by composition as large as possible, we introduce *almost well-nested* transductions (denoted $\mathcal{A}_{\mathsf{wn}}$). This class slightly generalizes the class $\mathcal{G}_{\mathsf{wn}}$ by allowing, in the range of the transduction, words that may differ on a bounded number of symbols from well-nested words: there must exist $k \in \mathbb{N}$ such that every output word contains at most k unmatched calls and returns. As done for $\mathcal{G}_{\mathsf{wn}}$, we associate with the class of transductions $\mathcal{A}_{\mathsf{wn}}$ the class of transducers awnVPT. For this class of transducers, it turns out that such a boundedness property on unmatched calls and returns can be considered not only on accepting runs but also on the partial runs on well-nested words. So, although the two classes gwnVPT and awnVPT are defined in a semantical way, we provide criteria on successful computations of VPTs characterizing precisely each of them. From these criteria, we succeed in computing the natural k upper-bounding the difference of outputs from well-nested words. Using this bound, we prove that the two classes gwnVPT and awnVPT enjoy good properties: they are closed under composition and type-checking is decidable against visibly pushdown languages.

The paper is organized as follows: definitions and basic properties of VPTs are presented in Section 2. We introduce in Section 3 the two classes of transductions we consider in this paper as well as the corresponding classes of transducers. Together with the (restricted) class introduced in [7], we prove also that they form a strict hierarchy. Then, we give in Section 4 a precise characterization of the classes gwnVPT and awnVPT by means of some criteria on VPTs. Section 5 describes decision procedure of the considered classes of transducers. Finally, the closure of gwnVPT and awnVPT under composition and the decidability of type-checking are addressed in Section 6.

2. Preliminaries

(Well) nested words The set of all finite words (resp. of all words of length at most n) over a finite alphabet Σ is denoted by Σ^* (resp. $\Sigma^{\leq n}$); the empty word is denoted by ϵ. A *structured alphabet* is a triple $\Sigma = (\Sigma_c, \Sigma_i, \Sigma_r)$ of disjoint alphabets, of call, internal and return symbols respectively. Given a structured alphabet Σ, we always denote by Σ_c, Σ_i and Σ_r its implicit structure, and identify Σ with $\Sigma_c \cup \Sigma_i \cup \Sigma_r$. A *nested word* is a finite word over a structured alphabet.

The set of *well-nested words* over a structured alphabet Σ, denoted by Σ^*_{wn}, is defined by the following grammar:

$$\Sigma^*_{\mathsf{wn}} \ni w ::= \epsilon \mid cw_1rw_2 \mid iw$$

where $c \in \Sigma_c$, $r \in \Sigma_r$ and $i \in \Sigma_i$. E.g. on $\Sigma = (\{c_1, c_2\}, \varnothing, \{r\})$, the nested word c_1rc_2r is well-nested while rc_1r and rc_1rc_2 are not.

For a word w from Σ^*, we define its balance B as the difference between the number of symbols from Σ_c and of symbols from Σ_r occurring in w. Note that if $w \in \Sigma^*_{\mathsf{wn}}$, then $\mathsf{B}(w) = 0$; but the converse is false as examplified by rc_1rc_2.

Lemma 1. *Let $w, w' \in \Sigma^*$. We have $\mathsf{B}(ww') = \mathsf{B}(w) + \mathsf{B}(w') = \mathsf{B}(w'w)$.*

We start with a decomposition property of nested words that can easily be proven by induction on the length of words.

Lemma 2. *Let $w \in \Sigma^*$. Then there exist unique words $(u_i)_{1\leq i\leq m}, (v_j)_{1\leq j\leq m}, w_0$ in Σ^*_{wn}, and unique symbols $r_i \in \Sigma_r, 1 \leq i \leq m$ and $c_j \in \Sigma_c, 1 \leq j \leq n$, such that:*

$$w = u_1r_1 \dots u_mr_mw_0c_1v_1 \dots c_nv_n$$

According to this decomposition of a word $w \in \Sigma^*$, we say that w contains exactly m open returns (letters r_i) and n open calls (letters c_j). This means that these symbols are not matched in w. We denote by $\mathsf{Oc}(w)$ and by $\mathsf{Or}(w)$ the number of open calls and of open returns in w respectively. We define, for any word w, $\mathsf{O}(w)$ as the pair $(\mathsf{Or}(w), \mathsf{Oc}(w)) \in \mathbb{N}^2$.

Lemma 3. *Let $w \in \Sigma^*$, the following equalities hold:*

$$\begin{aligned} \mathsf{Or}(w) &= -\min\{\mathsf{B}(w') \mid w'w'' = w\} \\ \mathsf{Oc}(w) &= \max\{\mathsf{B}(w'') \mid w'w'' = w\} \\ \mathsf{B}(w) &= \mathsf{Oc}(w) - \mathsf{Or}(w) \end{aligned}$$

Example 4. *Let $\Sigma = (\{c_1, c_2\}, \varnothing, \{r\})$. For c_1rc_2r, $\mathsf{B}(c_1) = 1$, $\mathsf{B}(c_1r) = 0$, $\mathsf{B}(c_1rc_2) = 1$, $\mathsf{B}(c_1rc_2r) = 0$. Hence, $\mathsf{Or}(c_1rc_2r) = 0$ and $\mathsf{Oc}(c_1rc_2r) = 0$*

For rc_1r and rc_1rc_2, $\mathsf{B}(r) = -1$, $\mathsf{B}(rc_1) = 0$, $\mathsf{B}(rc_1r) = -1$, $\mathsf{B}(rc_1rc_2) = 0$. Hence, $\mathsf{Or}(rc_1r) = \mathsf{Or}(rc_1rc_2) = 1$, $\mathsf{Oc}(rc_1r) = 0$ and $\mathsf{Oc}(rc_1rc_2) = 1$.

Given $(n_1, n_2) \in \mathbb{N}^2$, we define $||(n_1, n_2)|| = \max(n_1, n_2)$. Then, for a word w, we have $||\mathsf{O}(w)|| = \max\{\mathsf{Or}(w), \mathsf{Oc}(w)\}$; note that $w \in \Sigma^*_{\mathsf{wn}}$ iff $||\mathsf{O}(w)|| = 0$, that is $\mathsf{O}(w) = (0, 0)$.

Given a word $w \in \Sigma^*$, we define the height of w, denoted $\mathsf{height}(w)$, as $\max\{||\mathsf{O}(w_1)|| \mid w = w_1w_2\}$. We denote by $|w|$ the length of w, defined as usual.

Definition 5. *For any two pairs (n_1, n_2) and (n'_1, n'_2) of naturals from $\mathbb{N}^2$, we define $(n_1, n_2) \oplus (n'_1, n'_2)$ as the pair*

$$\begin{cases} (n_1, n_2 - n'_1 + n'_2) \text{ if } n_2 \geq n'_1 \\ (n_1 + n'_1 - n_2, n'_2) \text{ if } n'_1 > n_2 \end{cases}$$

Lemma 6. *For any words w, w' from Σ^*, $\mathsf{O}(ww') = \mathsf{O}(w) \oplus \mathsf{O}(w')$.*

Proof. We consider the decompositions of w and w' given by Lemma 2 and derive from them the unique decomposition of ww'. By distinguishing two cases (whether $\mathsf{Oc}(w) \geq \mathsf{Or}(w')$ or not) one easily proves the result. □

Proposition 7. $(\mathbb{N}^2, \oplus, (0, 0))$ *is a monoid, and the mapping* O *is a morphism from* $(\Sigma^*, ., \epsilon)$ *to* $(\mathbb{N}^2, \oplus, (0, 0))$

Proof. Proving that $(\mathbb{N}^2, \oplus, (0, 0))$ is a monoid is straightforward. The fact that O is a morphism follows from Lemma 6. □

Lemma 8. *For any finite sequence of words $v_1, v_2, \ldots v_n$,*

$$\mathsf{Or}(v_1v_2 \ldots v_n) = \max_{1 \leq i \leq n} (\mathsf{Or}(v_i) - \mathsf{B}(v_1v_2 \ldots v_{i-1}))$$

Proof. The proof goes by induction on the length n of the sequence. This is obvious for $n = 1$. Now, we consider $n + 1$ assuming this holds for n: $\mathsf{Or}(v_1v_2 \ldots v_{n+1})$ is equal by definition of $\oplus$ to

$$\begin{cases} \mathsf{Or}(v_1) & \text{if } \mathsf{Oc}(v_1) \geq \mathsf{Or}(v_2 \ldots v_{n+1})) \\ \mathsf{Or}(v_1) + \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{Oc}(v_1) & \text{if } \mathsf{Oc}(v_1) < \mathsf{Or}(v_2 \ldots v_{n+1})) \end{cases}$$

$$\begin{aligned} &\text{Now, } \max(\mathsf{Or}(v_1), \max_{2\leq i\leq n+1}(\mathsf{Or}(v_i) - \mathsf{B}(v_1v_2 \ldots v_{i-1}))) \\ &\quad = \max(\mathsf{Or}(v_1), \max_{2\leq i\leq n+1}(\mathsf{Or}(v_i) - \mathsf{B}(v_2 \ldots v_{i-1})) - \mathsf{B}(v_1)) \\ &\quad = \max(\mathsf{Or}(v_1), \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{B}(v_1)) \quad \text{(by ind. hyp.)} \end{aligned}$$

Depending of the greatest of the two elements; if $\mathsf{Or}(v_1) \geq \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{B}(v_1) = \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{Oc}(v_1) + \mathsf{Or}(v_1)$. We have $\mathsf{Oc}(v_1) \geq \mathsf{Or}(v_2 \ldots v_{n+1})$. If $\mathsf{Or}(v_1) < \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{B}(v_1) = \mathsf{Or}(v_2 \ldots v_{n+1}) - \mathsf{Oc}(v_1) + \mathsf{Or}(v_1)$. Then, $\mathsf{Oc}(v_1) < \mathsf{Or}(v_2 \ldots v_{n+1})$. In both cases, this is equal to $\mathsf{Or}(v_1v_2 \ldots v_{n+1})$. $\square$

Transductions - Transducers Let Σ be a structured (input) alphabet, and Δ be a structured (output) alphabet. A relation over $\Sigma^* \times \Delta^*$ is a *transduction*. We denote by $\mathcal{T}(\Sigma, \Delta)$ the set of these transductions. For a transduction T, the set of words u (resp. v) such that $(u, v) \in T$ is called the *domain* (resp. the *range*) of T. For $L \subseteq \Sigma^*$, we denote by $T(L)$ the set $\{v \in \Delta^* \mid (u, v) \in T \text{ for some } u \in L\}$.

Definition 9. [8] *A visibly pushdown transducer (*VPT*) from Σ to Δ is a tuple $A = (Q, I, F, \Gamma, \delta)$ where Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, Γ a (finite) stack alphabet, and $\delta = \delta_c \uplus \delta_r \uplus \delta_i$ the transition relation where:*

- $\delta_c \subseteq Q \times \Sigma_c \times \Gamma \times \Delta^* \times Q$ *are the* call transitions,
- $\delta_r \subseteq Q \times \Sigma_r \times \Gamma \times \Delta^* \times Q$ *are the* return transitions.
- $\delta_i \subseteq Q \times \Sigma_i \times \Delta^* \times Q$ *are the* internal transitions.

A stack (content) is a word over Γ. Hence, Γ^* is a monoid for the concatenation with $\perp$ (the empty stack) as neutral element. A configuration of A is a pair (q, σ) where $q \in Q$ and $\sigma \in \Gamma^*$ is a stack content. Let $u = a_1 \ldots a_l$ be a (nested) word on Σ, and $(q, \sigma), (q', \sigma')$ be two configurations of A. A *run* ρ of the VPT A over u from (q, σ) to (q', σ') is a (possibly empty) sequence of transitions $\rho = t_1t_2 \ldots t_l \in \delta^*$ such that there exist $q_0, q_1, \ldots q_l \in Q$ and $\sigma_0, \ldots \sigma_l \in \Gamma^*$ with $(q_0, \sigma_0) = (q, \sigma)$, $(q_l, \sigma_l) = (q', \sigma')$, and for each $0 < k \leq l$, we have either (i) $t_k = (q_{k-1}, a_k, \gamma, v_k, q_k) \in \delta_c$ and $\sigma_k = \sigma_{k-1}\gamma$, or (ii) $t_k = (q_{k-1}, a_k, \gamma, v_k, q_k) \in \delta_r$, and $\sigma_{k-1} = \sigma_k\gamma$, or (iii) $t_k = (q_{k-1}, a_k, v_k, q_k) \in \delta_i$, and $\sigma_{k-1} = \sigma_k$. When the sequence of transitions is empty, $(q, \sigma) = (q', \sigma')$.

The length (resp. height) of a run ρ over some word $u \in \Sigma^*$, denoted $|\rho|$ (resp. $\mathsf{height}(\rho)$) is defined as the length of u (resp. as the height of u).

The *output* of ρ (denoted $\mathsf{output}(\rho)$) is the word $v \in \Delta^*$ defined as the concatenation $v = v_1 \ldots v_l$ when the sequence of transitions is not empty and as ϵ

otherwise. We write $(q,\sigma) \xrightarrow{u|v} (q',\sigma')$ when there exists a run on u from (q,σ) to (q',σ') producing v as output. Initial (resp. final) configurations are pairs $(q,\perp)$ with $q \in I$ (resp. with $q \in F$). A configuration (q,σ) is *reachable* (resp. *co-reachable*) if there exist some initial configuration $(i,\perp)$ and a run from $(i,\perp)$ to (q,σ) (resp. some final configuration $(f,\perp)$ and a run from (q,σ) to $(f,\perp)$). A run is *accepting* if it starts in an initial configuration and ends in a final configuration.

A transducer A defines transduction (*ie* a relation) from nested words to nested words, denoted by $[\![A]\!]$, and defined as the set of pairs $(u,v) \in \Sigma^* \times \Delta^*$ such that there exists an accepting run on u producing v as output. Note that since both initial and final configurations have empty stack, A accepts only well-nested words, i.e. $[\![A]\!] \subseteq \Sigma^*_{\mathsf{wn}} \times \Delta^*$.

We denote by $\mathsf{VPT}(\Sigma,\Delta)$ the class of VPTs over the structured alphabets Σ (as input alphabet) and Δ (as output alphabet) and by $\mathcal{VP}(\Sigma,\Delta)$ the class of transductions defined by VPTs from $\mathsf{VPT}(\Sigma,\Delta)$.

Given a VPT $A = (Q,I,F,\Gamma,\delta)$, we let $\mathsf{O}^A_{\max}$ be the maximal number of open calls and of open returns in transition outputs in A, that is $\mathsf{O}^A_{\max} = \max\{||\mathsf{O}(v)|| \mid (p,\alpha,v,\gamma,q) \in \delta_c \cup \delta_r \text{ or } (p,\alpha,v,q) \in \delta_i\}$. Similarly, we define $\mathsf{Out}^A_{\max}$ as the maximal length of transition outputs in A, that is $\mathsf{Out}^A_{\max} = \max\{|v| \mid (p,\alpha,v,\gamma,q) \in \delta_c \cup \delta_r \text{ or } (p,\alpha,v,q) \in \delta_i\}$. Finally, $|S|$ denoting the cardinality of the set S, the size of A is defined by $\Big(3 * |Q| + |\Gamma| + |Q| * |Q| * |\Sigma| * |\Gamma| * \mathsf{Out}^A_{\max}\Big)$.

Visibly pushdown automata We define visibly pushdown automata (VPA) [1] simply as a particular case of VPT; we may think of them as transducers with no output. Hence, only the domain of the transduction matters and is called the language defined by the visibly pushdown automaton. For a VPA A, this language is denoted $L(A)$. Such languages are called *visibly pushdown languages* (VPL).

Properties of computations in $\mathsf{VPT}/\mathsf{VPA}$ For a VPT $A = (Q,I,F,\Gamma,\delta)$, we define two sets $\mathcal{P}^A_{\mathsf{wn}}$ and $\mathcal{C}^A_{\mathsf{wn}}$, subsets of $Q \times Q$ and $(Q \times Q)^2$, whose aim is to abstract computations of A.

Definition 10. *Let $A = (Q,I,F,\Gamma,\delta)$ be a VPT. The set $\mathcal{P}^A_{\mathsf{wn}}$ is the set of pairs $(p,q) \in Q \times Q$ such that there exist $u \in \Sigma^*_{\mathsf{wn}}$ and a run $(p,\perp) \xrightarrow{u|v} (q,\perp)$ in A.*

Thanks to the grammar describing the set Σ^*_{wn} and given in the preliminaries on nested words, the set $\mathcal{P}^A_{\mathsf{wn}}$ is obtained as the least subset of $Q \times Q$ satisying the following rules:

$$\frac{}{(p,p) \in \mathcal{P}^A_{\mathsf{wn}}} \qquad\qquad \frac{(p,i,v,p') \in \delta_i \quad (p',q) \in \mathcal{P}^A_{\mathsf{wn}}}{(p,q) \in \mathcal{P}^A_{\mathsf{wn}}}$$

$$\frac{(p,c,\gamma,v,p') \in \delta_c \quad (p',q') \in \mathcal{P}^A_{\mathsf{wn}} \quad (q',r,\gamma,v',q'') \in \delta_r \quad (q'',q) \in \mathcal{P}^A_{\mathsf{wn}}}{(p,q) \in \mathcal{P}^A_{\mathsf{wn}}}$$

Elements of $\mathcal{P}^A_{\mathsf{wn}}$ thus represent horizontal paths in A. In order to study vertical loops in A, we also introduce a notion of vertical path. To this end, we consider the set $\mathcal{C}(\Sigma)$ of contexts over Σ. A context is a pair of words $(u_1, u_2) \in \Sigma^* \times \Sigma^*$ such that $u_1u_2 \in \Sigma^*_{\mathsf{wn}}$. $\mathcal{C}(\Sigma)$ is described by the following grammar:

$$\mathcal{C}(\Sigma) \ni (u_1, u_2) ::= (\epsilon, \epsilon) \mid (cu_1, u_2r) \mid (w_1u_1, u_2w_2)$$

where $w_1, w_2 \in \Sigma^*_{\mathsf{wn}}$, $c \in \Sigma_c$ and $r \in \Sigma_r$.

Definition 11. *Let $A = (Q, I, F, \Gamma, \delta)$ be a* VPT. *The set $\mathcal{C}^A_{\mathsf{wn}}$ is the set of pairs $((p, p'), (q', q)) \in (Q \times Q)^2$ such that there exist a context $(u_1, u_2) \in \mathcal{C}(\Sigma)$ and two runs $(p, \bot) \xrightarrow{u_1|v_1} (p', \sigma)$ and $(q', \sigma) \xrightarrow{u_2|v_2} (q, \bot)$ in A.*

Again, thanks to the description of $\mathcal{C}(\Sigma)$ by means of a grammar, this set is obtained as the least subset of $(Q \times Q)^2$ satisfying the following rules:

$$\frac{}{((p, p), (q, q)) \in \mathcal{C}^A_{\mathsf{wn}}}$$

$$\frac{(p, c, \gamma, v, p_1) \in \delta_c \quad (q_1, r, \gamma, v', q) \in \delta_r \quad ((p_1, p'), (q', q_1)) \in \mathcal{C}^A_{\mathsf{wn}}}{((p, p'), (q', q)) \in \mathcal{C}^A_{\mathsf{wn}}}$$

$$\frac{\{(p, p_1), (q_1, q)\} \subseteq \mathcal{P}^A_{\mathsf{wn}} \quad ((p_1, p'), (q', q_1)) \in \mathcal{C}^A_{\mathsf{wn}}}{((p, p'), (q', q)) \in \mathcal{C}^A_{\mathsf{wn}}}$$

As a consequence of the above inductive definitions of $\mathcal{P}^A_{\mathsf{wn}}$ and $\mathcal{C}^A_{\mathsf{wn}}$, we have:

Proposition 12. *Given a* VPA *A, the sets $\mathcal{P}^A_{\mathsf{wn}}$, $\mathcal{C}^A_{\mathsf{wn}}$ can be computed in polynomial time in the size of A.*

We present now results on runs of visibly pushdown machines. More precisely, these results state first that sufficiently long runs of small height must contain a (horizontal) loop on some well-nested word and secondly that, runs of great height contain a (vertical) loop that let the stack grow.

Lemma 13. *Let A be a* VPT *with set of states Q and $\rho : (p, \bot) \xrightarrow{u|v} (q, \bot)$ be a run of A over some word $u \in \Sigma^*_{\mathsf{wn}}$. Let $h \in \mathbb{N}_{>0}$. We have:*

(*i*) *if $\mathsf{height}(u) \leq h$ and $|u| > 3 * (|Q| - 1)^{h+1}$, then ρ can be decomposed as follows:*

$$\rho : (p, \bot) \xrightarrow{u_1|v_1} (p_1, \sigma) \xrightarrow{u_2|v_2} (p_1, \sigma) \xrightarrow{u_3|v_3} (q, \bot)$$

with u_1u_3 and u_2 well-nested words and $u_2 \neq \epsilon$.

(*ii*) *if $\mathsf{height}(u) \geq |Q|^2$, then ρ can be decomposed as follows:*

$$\rho : (p, \bot) \xrightarrow{u_1|v_1} (p_1, \sigma) \xrightarrow{u_2|v_2} (p_1, \sigma\sigma') \xrightarrow{u_3|v_3} (p_2, \sigma\sigma') \xrightarrow{u_4|v_4} (p_2, \sigma) \xrightarrow{u_5|v_5} (q, \bot)$$

with u_1u_5, u_2u_4 and u_3 well-nested words, and $\sigma' \neq \bot$.

Proof. Let us start by proving point (i): we show that every run ρ on some input word $u \in \Sigma_{\mathsf{wn}}^*$, that does not contain a loop as expressed in point (i), has a length bounded by a function only depending on the height of u. We denote this function f and will identify it using a recurrence relation. First, if the height of u is null, then the length of ρ is at most $|Q|-1$. Now, let us evaluate $f(h+1)$ by means of $f(h)$, for some $h \in \mathbb{N}$. Consider a run ρ on some word u of height $h+1$. As ρ contains no loop, ρ contains at most $|Q|-1$ configurations with empty stack. Let us denote by $w \in \Sigma_{\mathsf{wn}}^*$ the input word between two consecutive such configurations. Either w is equal to some internal symbol, or w is of the form $cw'r$, with $c \in \Sigma_c$, $r \in \Sigma_r$ and $w' \in \Sigma_{\mathsf{wn}}^*$. In the latter case, we can bound the length of the run on w' by the value $f(h)$. As a consequence, we obtain the following recurrence relation:

$$f(h+1) = (|Q|-1)(f(h)+2)$$

Together with the constraint $f(0) = |Q|-1$, one can deduce the following solution: $f(h) = (|Q|-1)(|Q|(|Q|-1)^h - 2)/(|Q|-2)$. It is then easy to verify $f(h) \leq 3(|Q|-1)^{h+1}$, provided that $|Q| \geq 3$.

Let us prove now point (ii): it is possible to extract from ρ on u a sequence of runs $\rho_0, \dots, \rho_{h-1}$ on words $\bar{u}_0, \dots, \bar{u}_{h-1}$ respectively such that:

- $\rho_0 = \rho$, $\bar{u}_0 = u$,
- for all $0 \leq i \leq h-2$, for some $\bar{u}'_{i+1}, \bar{u}''_{i+1} \in \Sigma_{\mathsf{wn}}^*$, $\bar{u}_i = \bar{u}'_{i+1}\bar{u}_{i+1}\bar{u}''_{i+1}$ and $\bar{u}_{i+1} = cwr$ for some $c \in \Sigma_c$, $r \in \Sigma_r$ and $w \in \Sigma_{\mathsf{wn}}^*$,
- for all $0 \leq i \leq h-2$, ρ_{i+1} is the subrun of ρ_i running on $\bar{u}_{i+1}$ and if the starting and the ending configurations of ρ_i are of the form (p_i, σ_i) and (q_i, σ_i) respectively then the starting and ending configurations of ρ_{i+1} are of the form $(p_{i+1}, \sigma_i\gamma)$ and $(q_{i+1}, \sigma_i\gamma)$ for some $\gamma \in \Gamma$.

When $h \geq |Q|^2$, by a simple pigeonhole principle argument, there must be two runs ρ_j, ρ_k having the same state in their starting configurations as well as in their ending configurations. By letting $u_1 = \bar{u}'_0 \dots \bar{u}'_j$, $u_5 = \bar{u}''_j \dots \bar{u}''_0$, $u_3 = \bar{u}_k$, $u_2 = \bar{u}'_{j+1} \dots \bar{u}'_k$ and $u_4 = \bar{u}''_k \dots \bar{u}''_{j+1}$, one obtains a pattern similar to the one in point (ii). $\square$

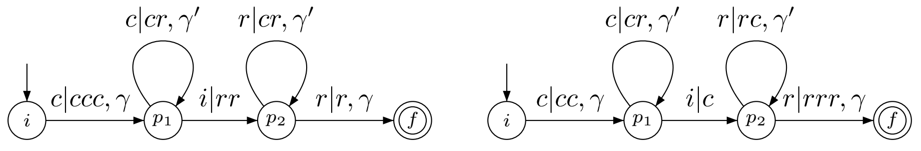
3. Classes of VPT producing (almost) well-nested outputs

We first recall the definition of (locally) well-nested VPT and then we introduce the new classes of globally and almost well-nested VPT. Finally, we prove relationships between these classes.

3.1. *Definitions*

Locally Well-nested VPTs In [7], the class of (locally) well-nested VPT has been introduced. For this class, the enforcement of the well-nestedness of the output is done locally and syntactically at the level of transition rules.

Definition 14 (Locally Well-nested) *Let $A = (Q, I, F, \Gamma, \delta)$ be a* VPT. *A is a* locally well-nested VPT (lwnVPT) *if:*



(a) The VPT A_1. (b) The VPT A_2.

Figure 1. Two VPTs in $\mathcal{VP}(\Sigma, \Sigma)$ with $\Sigma_c = \{c\}$, $\Sigma_r = \{r\}$ and $\Sigma_i = \{i\}$.

- *for any pair of transitions $(q, c, v, \gamma, q') \in \delta_c$, $(p, r, v', \gamma, p') \in \delta_r$, the word vv' is well nested, and*
- *for any transition $(q, a, v, q') \in \delta_i$, the word v is well-nested.*

A VPT *transduction T is* locally well-nested *if there exists a* lwnVPT *A that realizes T (ie $[\![A]\!] = T$). The class of locally well-nested* VPT *transductions is denoted $\mathcal{L}_{\mathsf{wn}}$.*

Proposition 15. [7] *Let A be a locally well-nested* VPT *and (p, σ), (q, σ) two configurations of A. For all well-nested word u, if $(p, \sigma) \xrightarrow{u|v} (q, \sigma)$ then $v \in \Delta^*_{\mathsf{wn}}$.*

Hence, any locally well-nested VPT transduction T is included into $\Sigma^*_{\mathsf{wn}} \times \Delta^*_{\mathsf{wn}}$.

Globally and almost well-nested VPT transductions we introduce now the class of globally well-nested transductions and its weaker variant of almost well-nested transductions. Unlike the definition of $\mathcal{L}_{\mathsf{wn}}$, done at the level of transducers, these definitions are done at the level of transductions and thus, as semantical properties.

Definition 16 (Globally Well-nested) *A* VPT *transduction T is* globally well-nested *if $T(\Sigma^*_{\mathsf{wn}}) \subseteq \Delta^*_{\mathsf{wn}}$. The class of globally well-nested* VPT *transductions is denoted $\mathcal{G}_{\mathsf{wn}}$.*

A VPT *A is* globally well-nested *if its transduction $[\![A]\!]$ is. The class of globally well-nested* VPT *is denoted* gwnVPT.

Definition 17 (Almost Well-nested) *A* VPT *transduction T is* almost well-nested *if there exists k in $\mathbb{N}$ such that for every word $(u, v) \in T$, it holds that $||\mathsf{O}(v)|| \leq k$. The class of almost well-nested* VPT *transductions is denoted $\mathcal{A}_{\mathsf{wn}}$.*

A VPT *A is* almost well-nested *if its transduction $[\![A]\!]$ is. The class of almost well-nested* VPT *is denoted* awnVPT.

3.2. *Comparison of the different classes*

Classes of transductions $\mathcal{G}_{\mathsf{wn}}$ and $\mathcal{A}_{\mathsf{wn}}$ are defined by semantical conditions on the defined relations. This yields a clear correspondence between the classes $\mathcal{G}_{\mathsf{wn}}$ and gwnVPT on one side and $\mathcal{A}_{\mathsf{wn}}$ and awnVPT on the other side. This is not the case for $\mathcal{L}_{\mathsf{wn}}$: two examples of VPTs are given in Figure 1. It is easy to verify that

$A_1, A_2 \in \mathsf{gwnVPT}$. But, none of these transducers belongs to lwnVPT. However, one can easily build a transducer A_2' such that $[\![A_2]\!] = [\![A_2']\!]$ and $A_2' \in \mathsf{lwnVPT}$. Indeed one can perform the following modifications:

- the transition (p_1, i, c, p_2) becomes (p_1, i, ϵ, p_2)
- the transition $(p_2, r, rc, \gamma', p_2)$ becomes $(p_2, r, cr, \gamma', p_2)$
- the transition (p_2, r, rrr, γ, f) becomes $(p_2, r, crrr, \gamma, f)$

On the contrary, as we prove below, the transduction $[\![A_1]\!]$ does not belong to $\mathcal{L}_{\mathsf{wn}}$: there exists no transducer $A_1' \in \mathsf{lwnVPT}$ such that $[\![A_1']\!] = [\![A_1]\!]$.

Proposition 18. *The following inclusion results hold:*

- *For transducers:* $\mathsf{lwnVPT} \subsetneq \mathsf{gwnVPT} \subsetneq \mathsf{awnVPT}$
- *For transductions:* $\mathcal{L}_{\mathsf{wn}} \subsetneq \mathcal{G}_{\mathsf{wn}} \subsetneq \mathcal{A}_{\mathsf{wn}}$

Proof. The non-strict inclusions are straightforward. The strict inclusions $\mathsf{gwnVPT} \subsetneq \mathsf{awnVPT}$ and $\mathcal{G}_{\mathsf{wn}} \subsetneq \mathcal{A}_{\mathsf{wn}}$ follow from the constraint on the range. The strict inclusion $\mathsf{lwnVPT} \subsetneq \mathsf{gwnVPT}$ is witnessed by A_2, as explained above.

We prove now the strict inclusion $\mathcal{L}_{\mathsf{wn}} \subsetneq \mathcal{G}_{\mathsf{wn}}$, and therefore consider the transducer A_1. Observe that $[\![A_1]\!] \in \mathcal{G}_{\mathsf{wn}}$, we show that $[\![A_1]\!] \notin \mathcal{L}_{\mathsf{wn}}$. First note that $[\![A_1]\!] = \{(cc^k ir^k r, ccc(cr)^k rr(cr)^k r) \mid k \in \mathbb{N}\}$ and that

(Fact 1) the transduction defined by A_1 is injective
(Fact 2) any word of the output can be decomposed as $w_1 rr w_2$ where $w_1 = ccc(cr)^k$ and $w_2 = (cr)^k r$ for some natural k and for each w_1 with fixed k there exists a unique w_2 such that $w_1 rr w_2$ is in the range of A_1 (and symmetrically for w_2 and w_1).
(Fact 3) in any word of the output, the number of occurrences of the subword rr is upper-bounded by 3 and no subword $ccccc$ occurs.

By contradiction, suppose that there exists $A_1' \in \mathsf{lwnVPT}$ such that $[\![A_1']\!] = [\![A_1]\!]$.

Now, for k sufficiently large and depending only on the fixed size of A_1', A_1' has an accepting run for this input of the form given in the point (ii) of Lemma 13 and additionnaly, v_1v_5, v_2v_4, v_3 are well-nested due to Proposition 15.

Now, assume that $v_2 = \epsilon$ and $v_4 = \epsilon$. Then, using a simple pumping argument over the pair (u_2, u_4), one would obtain a different input producing the same output, contradicting (Fact 1) as $[\![A_1]\!] = [\![A_1']\!]$. So, $v_2 \neq \epsilon$ or $v_4 \neq \epsilon$.

Our aim is to identify in which v_j's the pattern rr mentionned previously occurs. Now, by case inspection :
Case where the pattern rr is in v_1: $v_2v_3v_4v_5$ is a suffix of w_2. By a pumping argument over the pair (u_2, u_4) using that either $v_2 \neq \epsilon$ or $v_4 \neq \epsilon$, one would obtain a different w_2' such that $w_1 rr w_2'$ is in the range of A_1' (and thus, A_1), contradicting (Fact 2).
Case where the pattern rr is in v_5 or split as the last letter of v_1 and the first letter of v_2 (resp. as the last letter of v_4 and the first letter of v_5): follow the lines of the previous case.

Case where the pattern rr is in v_2 or v_4: using a pumping over the pair (u_2, u_4), one could obtain outputs with unboundedly many subwords rr, contradicting (Fact 3).
Case where the pattern rr is split as the last letter of v_2 and the first letter of v_3: v_3 is well-nested. Hence, it can not start with some return symbol. Contradiction.
Case where the pattern rr is split as the last letter of v_3 and the first letter of v_4 As v_3 is well-nested, it must be of the form $c(cr)^k r$. So, $v_1 v_2 = cc$. If $v_2 \neq \epsilon$, v_2 is equal to c or cc. Using a pumping over the pair (u_2, u_4), one could obtain outputs containing the subword $ccccc$, contradicting (Fact 3). Otherwise, if $v_2 = \epsilon$, then by a pumping argument over the pair (u_2, u_4) using that $v_4 \neq \epsilon$, one would obtain a different w_2' such that $w_1 rr w_2'$ is in the range of A_1' (and thus, A_1), contradicting (Fact 1).
Case where the pattern rr is in v_3: v_3 is well-nested and contains as a suffix after this pattern rr a prefix of w_2: depending on the form of this prefix:

- the prefix is of the form $(cr)^{k'}$: v_3 must be of the form $cc(cr)^{k'} rr (cr)^{k'}$ and thus, $k' = k$. So, $v_1 v_2 = c$. We then conclude as in the previous case.
- the prefix is of the form $(cr)^{k'} c$: this is not possible as $v_3 \in \Sigma_{\mathsf{wn}}^*$.
- the prefix is of the form $(cr)^k r$: v_3 must be of the form $ccc(cr)^{k'} rr (cr)^{k'}$. It implies that $v_2 = \epsilon$ and $v_4 = \epsilon$. Contradiction.

Hence, there is no situation in which rr can occur maintaining A_1' to be locally well-nested. Therefore, such A_1' cannot exist. □

4. Characterizations

We give now criteria on VPTs that aim to characterize the classes gwnVPT and awnVPT. For the two classes, these criteria entail that on both horizontal or vertical loops containing a reachable and a co-reachable configurations, the balance of output words must be equal to 0.

Definition 19. *Let A be a* VPT*. Let us consider the following criteria:*

(C1) For all states p, i, f such that i is initial and f is final, for any stack σ, then any accepting run

$$(i, \perp) \xrightarrow{u_1 | v_1} (p, \sigma) \xrightarrow{u_2 | v_2} (p, \sigma) \xrightarrow{u_3 | v_3} (f, \perp)$$

with $u_1 u_3, u_2 \in \Sigma_{\mathsf{wn}}^*$ satisfies $\mathsf{B}(v_2) = 0$.

(C2) For all states p, q, i, f such that i is initial and f is final, for any stack σ, σ', then any accepting run

$$(i, \perp) \xrightarrow{u_1 | v_1} (p, \sigma) \xrightarrow{u_2 | v_2} (p, \sigma\sigma') \xrightarrow{u_3 | v_3} (q, \sigma\sigma') \xrightarrow{u_4 | v_4} (q, \sigma) \xrightarrow{u_5 | v_5} (f, \perp)$$

with $u_2 u_4, u_3 \in \Sigma_{\mathsf{wn}}^*$ and $\sigma' \neq \perp$ satisfies $\mathsf{B}(v_2) + \mathsf{B}(v_4) = 0$ and $\mathsf{B}(v_2) \geq 0$.

Intuitively, if one of the two criteria from Definition 19 is violated then one can, by a pumping argument, exhibit an arbitrary long sequence of well-nested words on which the VPT A will produce a sequence of output words whose number of unmatched calls or returns will strictly increase.

The following result follows from Propositions 23 and 25 that we prove in the rest of this section.

Theorem 20. *A VPT A is almost well-nested iff A verifies criteria $(C1)$ and $(C2)$.*

The following lemma is an immediate consequence of the definitions.

Lemma 21. *Let $X \subseteq \Sigma^*$ such that the set $\mathsf{B}(X) = \{\mathsf{B}(u) \mid u \in X\}$ is infinite. Then the set $\{\mathsf{O}(u) \mid u \in X\}$ is infinite as well.*

Lemma 22. *Let $u \in \Sigma^*$ and k be a strictly positive integer. Then $\mathsf{O}(u^k)$ is equal to $(\mathsf{Or}(u), (\mathsf{Oc}(u) - \mathsf{Or}(u)) * (k-1) + \mathsf{Oc}(u))$ if $\mathsf{Oc}(u) \geq \mathsf{Or}(u)$ and to $(\mathsf{Or}(u) + (\mathsf{Or}(u) - \mathsf{Oc}(u)) * (k-1), \mathsf{Oc}(u))$ otherwise.*

Proof. By definition of $\oplus$ and by induction on k, computing $\mathsf{O}(u) \oplus \mathsf{O}(u^k)$. □

Proposition 23. *Let A be a VPT. If A does not satisfy $(C1)$ or $(C2)$, then A is not almost well-nested.*

Proof. Let us assume that A does not satisfy $(C1)$. Hence there exists an accepting run as described in criterion $(C1)$ such that $\mathsf{B}(v_2) \neq 0$. We then build by iterating the loop on word u_2 accepting runs for words of the form $u_1(u_2)^k u_3$ for any natural k, producing output words $v_1(v_2)^k v_3$. Let us denote this set by X. As $\mathsf{B}(v_2) \neq 0$ and by Lemma 1, the set $\mathsf{B}(X)$ is infinite. Lemma 21 entails that A is not almost well-nested.

Assume now that A does not satisfy $(C2)$. Hence, there exists an accepting run as described in this criterion such that either (i) $\mathsf{B}(v_2) + \mathsf{B}(v_4) = b \neq 0$ or (ii) $\mathsf{B}(v_2) < 0$. In the case of (i), from this run, one can build by pumping accepting runs for words of the form $u_1(u_2)^k u_3(u_4)^k u_5$ for any natural k, producing output words $v_1(v_2)^k v_3(u_4)^k v_5$. As before, Lemmas 1 and 21 imply that A is not almost well-nested.

Now, for (ii) assuming that $\mathsf{B}(v_2) + \mathsf{B}(v_4) = 0$. As $\mathsf{B}(v_2) < 0$, it holds that $\mathsf{B}(v_4) > 0$ and thus, $\mathsf{Or}(v_2) > \mathsf{Oc}(v_2)$, $\mathsf{Or}(v_4) < \mathsf{Oc}(v_4)$. From the run of the criterion, one can build by pumping accepting runs for words of the form $u_1(u_2)^k u_3(u_4)^k u_5$ for any natural k, producing output words $v_1(v_2)^k v_3(v_4)^k v_5$. Now, we consider $\mathsf{O}(v_1(v_2)^k v_3(v_4)^k v_5)$ which, by associativity of $\oplus$, is equal to $\mathsf{O}(v_1) \oplus \mathsf{O}((v_2)^k) \oplus \mathsf{O}(v_3) \oplus \mathsf{O}((u_4)^k) \oplus \mathsf{O}(v_5)$. Now, by Lemma 22, it is equal to

$$\begin{aligned}\mathsf{O}(v_1) \oplus (\mathsf{Or}(v_2) + (\mathsf{Or}(v_2) - \mathsf{Oc}(v_2)) * (k-1), \mathsf{Oc}(v_2)) \oplus \mathsf{O}(v_3) \oplus \\ (\mathsf{Or}(v_4), (\mathsf{Oc}(v_4) - \mathsf{Or}(v_4)) * (k-1) + \mathsf{Oc}(v_4)) \oplus \mathsf{O}(v_5)\end{aligned}$$

It is easy to see that for k varying, the described pairs are unbounded, falsifying the almost well-nestedness of A. □

Let us prove now the converse. Given a VPT $A = (Q, I, F, \Gamma, \delta)$, we define the integer $N_A = 3 * (|Q| - 1)^{(2|Q|^2+1)}$.

Lemma 24. *Let A be a VPT. If A satisfies the criteria $(C1)$ and $(C2)$, then for any accepting run ρ such that $|\rho| \geq N_A$, there exists an accepting run ρ' such that $|\rho'| < |\rho|$ and $||\mathsf{O}(\mathsf{output}(\rho'))|| \geq ||\mathsf{O}(\mathsf{output}(\rho))||$.*

Proof. Let $A = (Q, I, F, \Gamma, \delta)$ and ρ be an accepting run for some word u outputing v such that $|\rho| \geq N_A$. We distinguish two cases, depending on $\mathsf{height}(\rho)$:

Case where $\mathsf{height}(\rho) \leq 2|Q|^2$: in this case, by definition of N_A, we can apply the point (i) of Lemma 13 twice and prove that ρ is of the following form:

$$(i, \perp) \xrightarrow{u_1|v_1} (p, \sigma) \xrightarrow{u_2|v_2} (p, \sigma) \xrightarrow{u_3|v_3} (q, \sigma') \xrightarrow{u_4|v_4} (q, \sigma') \xrightarrow{u_5|v_5} (f, \perp)$$

with $u_2, u_4 \in \Sigma_{\mathsf{wn}}^* \setminus \{\epsilon\}$. Then, by criterion $(C1)$, we have $\mathsf{B}(v_2) = \mathsf{B}(v_4) = 0$.

In the following, we prove that at least one of u_2 and u_4 can be removed from u while preserving the value $\mathsf{Or}(u)$: let us denote by v' (resp. v'') the resulting output word once u_2 (resp. u_4) is removed from the input. Observe also that removing one of this part of the run does not modify the balance $\mathsf{B}(.)$ of the output of any prefix $v_1 \dots v_l$ of the output sequence as $\mathsf{B}(v_2) = \mathsf{B}(v_4) = 0$. Now, we appeal to Lemma 8; depending on the v_i defining $\mathsf{Or}(v_1 \dots v_5)$ as a maximum value,

- if the maximum is reached at a position v_i different from v_2, then as $\mathsf{B}(v_1 v_3 \dots v_5) = \mathsf{B}(v_1 \dots v_5)$, $\mathsf{Or}(v) = \mathsf{Or}(v')$ allowing to remove u_2,
- the reasoning is similar if the maximum is reached for v_4 and thus, $\mathsf{Or}(v) = \mathsf{Or}(v'')$ allowing to remove u_4.

Finally as the overall balance of the output is preserved and as $\mathsf{Oc}(v) = \mathsf{B}(v) + \mathsf{Or}(v)$, we obtain $\mathsf{O}(v) = \mathsf{O}(v')$ (resp. $\mathsf{O}(v'')$), yielding the result.

Case where $\mathsf{height}(\rho) > 2|Q|^2$: in this case, we can apply the point (ii) of Lemma 13 twice and prove that ρ is of the following form:

$$(i, \perp) \xrightarrow{u_1|v_1} (p_1, \sigma) \xrightarrow{u_2|v_2} (p_1, \sigma\sigma_1) \xrightarrow{u_3|v_3} (q_1, \sigma\sigma_1\sigma_2) \xrightarrow{u_4|v_4} (q_1, \sigma\sigma_1\sigma_2\sigma_3)$$
$$\xrightarrow{u_5|v_5} (q_2, \sigma\sigma_1\sigma_2\sigma_3) \xrightarrow{u_6|v_6} (q_2, \sigma\sigma_1\sigma_2) \xrightarrow{u_7|v_8} (p_2, \sigma\sigma_1) \xrightarrow{u_8|v_8} (p_2, \sigma) \xrightarrow{u_9/v_9} (f, \perp),$$

with $u_1u_9, u_2u_8, u_3u_7, u_4u_6, u_5 \in \Sigma_{\mathsf{wn}}^*$ and $\sigma_1, \sigma_3 \neq \perp$.

Then the two following runs can be built: the one obtained by removing the parts of ρ on u_2 and u_8, and the one obtained by removing the parts of ρ on u_4 and u_6, yielding runs whose length is strictly smaller than $|\rho|$. Let us denote these two runs by ρ' and ρ'' respectively, and their outputs by v' and v''. As A verifies the criterion $(C2)$, we have that $\mathsf{B}(v) = \mathsf{B}(v') = \mathsf{B}(v'')$, as $\mathsf{B}(v_2) + \mathsf{B}(v_8) = \mathsf{B}(v_4) + \mathsf{B}(v_6) = 0$ and B is commutative. In order to obtain the result, we study $\mathsf{Or}(v)$.

By Lemma 8,

$$\mathsf{Or}(v) = \max_{i=1}^{9} \{\mathsf{Or}(v_i) - \mathsf{B}(v_1 \dots v_{i-1})\}$$

Then, we distinguish two cases:

- If the maximum corresponding to $\mathsf{Or}(v)$ is not obtained for $i \in \{4,5,6\}$, then we prove that $\mathsf{Or}(v'') \geq \mathsf{Or}(v)$: when we remove parts of ρ on u_4 and u_6, we have:

  $$\mathsf{Or}(v'') = \max\{\mathsf{Or}(v_1), \mathsf{Or}(v_2) - \mathsf{B}(v_1), \mathsf{Or}(v_3) - \mathsf{B}(v_1v_2), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_2v_3),\\ \mathsf{Or}(v_7) - \mathsf{B}(v_1v_2v_3v_5), \mathsf{Or}(v_8) - \mathsf{B}(v_1v_2v_3v_5v_7), \mathsf{Or}(v_9) - \mathsf{B}(v_1v_2v_3v_5v_7v_8)\}$$

  As $\mathsf{B}(v_4) + \mathsf{B}(v_6) = 0$ and the maximum corresponding to $\mathsf{Or}(v)$ is not obtained for $i \in \{4,5,6\}$, we obtain:

  $$\mathsf{Or}(v'') = \max\{\mathsf{Or}(v), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_2v_3)\}$$

  This proves $\mathsf{Or}(v'') \geq \mathsf{Or}(v)$.
- If the maximum corresponding to $\mathsf{Or}(v)$ is obtained for $i \in \{4,5,6\}$, then we prove that $\mathsf{Or}(v') \geq \mathsf{Or}(v)$: when we remove parts of ρ on u_2 and u_8, denoting by v' the resulting output word, we have:

  $$\mathsf{Or}(v') = \max\{\mathsf{Or}(v_1), \mathsf{Or}(v_3) - \mathsf{B}(v_1), \mathsf{Or}(v_4) - \mathsf{B}(v_1v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_3v_4),\\ \mathsf{Or}(v_6) - \mathsf{B}(v_1v_3v_4v_5), \mathsf{Or}(v_7) - \mathsf{B}(v_1v_3v_4v_5v_6), \mathsf{Or}(v_9) - \mathsf{B}(v_1v_3v_4v_5v_6v_7)\}$$

  As the maximum corresponding to $\mathsf{Or}(v)$ is obtained for $i \in \{4,5,6\}$, we can write:

  $$\begin{aligned}\mathsf{Or}(v) &= \max\{\mathsf{Or}(v_4) - \mathsf{B}(v_1v_2v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_2v_3v_4), \mathsf{Or}(v_6) - \mathsf{B}(v_1v_2v_3v_4v_5)\}\\ &= \max\{\mathsf{Or}(v_4) - \mathsf{B}(v_1v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_3v_4), \mathsf{Or}(v_6) - \mathsf{B}(v_1v_3v_4v_5)\} - \mathsf{B}(v_2)\end{aligned}$$

  By criterion $(C2)$, we have $\mathsf{B}(v_2) \geq 0$ and thus:

  $$\max\{\mathsf{Or}(v_4) - \mathsf{B}(v_1v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_3v_4), \mathsf{Or}(v_6) - \mathsf{B}(v_1v_3v_4v_5)\} \geq \mathsf{Or}(v)$$

  In addition, by Lemma 8, we have $\mathsf{Or}(v) \geq \mathsf{Or}(v_1)$ and $\mathsf{Or}(v) \geq \mathsf{Or}(v_9) - \mathsf{B}(v_1v_2v_3v_4v_5v_6v_7v_8)$. By criterion $(C2)$, the latter property can be written as $\mathsf{Or}(v) \geq \mathsf{Or}(v_9) - \mathsf{B}(v_1v_3v_4v_5v_6v_7)$ thanks to the property $\mathsf{B}(v_2) + \mathsf{B}(v_8) = 0$. Using these facts, we can simplify the previous expression and obtain:

  $$\begin{aligned}\mathsf{Or}(v') &= \max\{\mathsf{Or}(v_3) - \mathsf{B}(v_1), \mathsf{Or}(v_4) - \mathsf{B}(v_1v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_3v_4),\\ &\quad \mathsf{Or}(v_6) - \mathsf{B}(v_1v_3v_4v_5), \mathsf{Or}(v_7) - \mathsf{B}(v_1v_3v_4v_5v_6)\}\\ &= \max\{\mathsf{Or}(v_3) - \mathsf{B}(v_1v_2), \mathsf{Or}(v_4) - \mathsf{B}(v_1v_2v_3), \mathsf{Or}(v_5) - \mathsf{B}(v_1v_2v_3v_4),\\ &\quad \mathsf{Or}(v_6) - \mathsf{B}(v_1v_2v_3v_4v_5), \mathsf{Or}(v_7) - \mathsf{B}(v_1v_2v_3v_4v_5v_6)\} + \mathsf{B}(v_2)\\ &= \mathsf{Or}(v) + \mathsf{B}(v_2)\end{aligned}$$

  The last equality follows from the fact the maximum corresponding to $\mathsf{Or}(v)$ is obtained for $i \in \{4,5,6\}$. This concludes this case proving that $\mathsf{Or}(v') \geq \mathsf{Or}(v)$.

Now, in both cases, as for any word w, we have $\mathsf{Oc}(w) = \mathsf{B}(w) + \mathsf{Or}(w)$, we have also respectively $\mathsf{Oc}(v'') \geq \mathsf{Oc}(v)$ and $\mathsf{Oc}(v') \geq \mathsf{Oc}(v)$. This entails, according to the case to be considered, $||\mathsf{O}(v'')|| \geq ||\mathsf{O}(v)||$ or $||\mathsf{O}(v')|| \geq ||\mathsf{O}(v)||$. □

Proposition 25. *Let A be a* VPT. *If A satisfies $(C1)$ and $(C2)$, then every accepting run $(i,\perp) \xrightarrow{u|v} (f,\perp)$ of A verifies $||\mathsf{O}(v)|| \leq N_A.\mathsf{O}^A_{\max}$.*

Proof. If $|v| \leq N_A$ then the result is trivial; otherwise, assuming the existence of a minimal counter-example of this statement, a contradiction follows from Lemma 24. $\square$

Now we can show a precise characterization of transducers from gwnVPT amongst those in awnVPT.

Definition 26. *Let A be a* VPT*. We consider the following criterion:*

(D) For all $(u, v) \in [\![A]\!]$, if $|u| \leq N_A$ then $v \in \Delta^*_{\mathsf{wn}}$.

Theorem 27. *Let A be a* VPT*. Then A is globally well-nested iff A verifies criteria $(C1)$, $(C2)$ and (D).*

Proof. The left-to-right implication is trivial using Proposition 23. For the other direction, we prove that for all $u \in \Sigma^*_{\mathsf{wn}}$ and all $v \in \Delta^*$ such that $(u, v) \in [\![A]\!]$, it holds that $v \in \Delta^*_{\mathsf{wn}}$. The proof goes by induction on the length of u. For the base case, if $|u| \leq N_A$, then the criteria (D) gives the result. For the induction step, we assume the result for any input word of length smaller than k ; we consider a word u such that $|u| = k + 1$ and a word v such that $(u, v) \in [\![A]\!]$. By Lemma 24, there exists a word u' such that $|u'| \leq k$ and for any v' such that $(u', v') \in [\![A]\!]$, $||\mathsf{O}(v)|| \leq ||\mathsf{O}(v')||$. As, by induction hypothesis, v' is well-nested, ie $||\mathsf{O}(v')|| = 0$, we have $||\mathsf{O}(v)|| = 0$. So, v is well-nested. $\square$

5. Deciding the classes of almost and globally well-nested VPT

In this section, we prove that given a VPT A, it is decidable to know whether $[\![A]\!] \in \mathcal{A}_{\mathsf{wn}}$ and whether $[\![A]\!] \in \mathcal{G}_{\mathsf{wn}}$.

Proposition 28. *Given a* VPT *$A = (Q, I, F, \Gamma, \delta)$ and states p, q of A, deciding whether there exists some stack σ such that (p, σ) is reachable and (q, σ) is co-reachable can be done in* PTIME*.*

Proof. This is an immediate consequence of the observation that such pairs (p, q) are exactly the elements obtained as the projection on the second and third components of the set $\mathcal{C}^A_{\mathsf{wn}} \cap I \times Q \times Q \times F$, which can be computed in PTIME thanks to Proposition 12. $\square$

Theorem 29. *Let A be a* VPT*. Whether $[\![A]\!] \in \mathcal{A}_{\mathsf{wn}}$ can be decided in* PSPACE*.*

Proof. By Theorem 20, deciding the class awnVPT amounts to decide criteria $(C1)$ and $(C2)$. Therefore we propose a non-deterministic algorithm running in polynomial space, yielding the result thanks to Savitch theorem.

In this proof, by abuse of notation, given a run ρ whose output is some word v, we may simply write $\mathsf{B}(\rho)$ to denote $\mathsf{B}(v)$.

We claim that A satisfies $(C1)$ and $(C2)$ if and only if it satisfies these criteria on "small instances", defined as follows:

- Criterion $(C1)$: consider only words u_2 such that $\mathsf{height}(u_2) \leq |Q|^2$ and $|u_2| \leq 2.|Q|^{|Q|^2}$.
- Criterion $(C2)$: consider only stacks σ' such that $|\sigma'| \leq |Q|^2$ and words u_2, u_4 of height at most $2.|Q|^2$ and length at most $|Q|^2.|Q|^{|Q|^2}$.

The non-deterministic algorithm follows from the claim: in order to exhibit a witness of the fact that $A \notin \mathsf{awnVPT}$, the algorithm guesses whether $(C1)$ or $(C2)$ is violated, and a pair of states (p, q) and one or two runs, according to the criterion, of exponential size, which can be done in polynomial space. Using Proposition 28 it also satisfies that there exists a stack σ such that (p, σ) is reachable and (q, σ) is co-reachable.

To prove this claim, we show, by induction on $u \in \Sigma^*_{\mathsf{wn}}$, that for every run $(p, \bot) \xrightarrow{u|v} (q, \bot)$ that can be completed into an accepting run, and for every decomposition of this run according to criterion $(C1)$ or $(C2)$, the property stated by the corresponding criterion is fulfilled.

Cases $u = \epsilon$ and $u = a \in \Sigma_i$: The result directly follows from the hypothesis as they are small words.

Case $u = u_1 u_2$ with $u_1, u_2 \in \Sigma^*_{\mathsf{wn}} \setminus \{\epsilon\}$. Consider a run $\rho : (p, \bot) \xrightarrow{u_1 u_2|v} (q, \bot)$. First, consider a decomposition of ρ according to criterion $(C2)$. This decomposition is necessarily either completely in the sub-run on u_1, or in that on u_2 (this follows from the observation that in $(C2)$, the matched loops modify the stack: $\sigma' \neq \bot$). The result follows by induction. Second, consider a decomposition of ρ according to criterion $(C1)$. If the decomposition is completely in the sub-run on u_1, or in that on u_2, then the result follows by induction. Otherwise, the decomposition of ρ looks as follows: $\rho : (p, \bot) \xrightarrow{u'_1|v'_1} (p_1, \bot) \xrightarrow{u''_1|v''_1} (p_2, \bot) \xrightarrow{u'_2|v'_2} (p_1, \bot) \xrightarrow{u''_2|v''_2} (q, \bot)$ where $u_i = u'_i u''_i$ for $i \in \{1, 2\}$. Let us denote by ρ_1 the run $(p_1, \bot) \xrightarrow{u''_1|v''_1} (p_2, \bot)$ and by ρ_2 the run $(p_2, \bot) \xrightarrow{u'_2|v'_2} (p_1, \bot)$. The loop under consideration is represented by the run $\rho_1 \rho_2$.

By induction hypothesis applied on u_1, any decomposition of ρ_1 according to criteria $(C1)$ and $(C2)$ fulfills these criteria. Using Lemma 13, one can identify such decompositions if the height or the length of the run is large enough. Using the according criterion and the definition of the mapping $\mathsf{B}(.)$, one can remove the identified loop while preserving the value of $\mathsf{B}(.)$. Applying iteratively this process, we can build from ρ_1 a run ρ'_1 such that $\mathsf{B}(\rho'_1) = \mathsf{B}(\rho_1)$, $\mathsf{height}(\rho'_1) < |Q|^2$ and $|\rho'_1| < |Q|^{|Q|^2}$.

A similar construction can be done for ρ_2, yielding some run ρ'_2.

By construction of ρ'_1 and ρ'_2, we have $\mathsf{B}(\rho'_1 \rho'_2) = \mathsf{B}(\rho_1 \rho_2)$. Moreover, it is routine to verify that the run $\rho'_1 \rho'_2$ satisfies the constraints of the hypothesis on its height and length, and thus $\mathsf{B}(\rho'_1 \rho'_2) = 0$.

Case *cur*. We consider some run ρ as follows $\rho : (p, \bot) \xrightarrow{c|v_0} (p', \gamma) \xrightarrow{u|v} (q', \gamma) \xrightarrow{r|v_4} (q, \bot)$.

We first consider a decomposition of ρ according to criterion $(C1)$. If it is completely in the sub-run on u, then the result follows by induction. Otherwise, we have $p = q$ and the run ρ itself satisfies the conditions of criterion $(C1)$. Using standard horizontal and vertical pumping and the induction hypothesis on u, we can assume that the height of u is strictly less than $|Q|^2$ and that the length of u is strictly less than $|Q|^{(Q|^2}$ (otherwise by a pumping reasoning we can exhibit a decomposition that satisfies either criterion $(C1)$ or criterion $(C2)$ and using the induction hypothesis remove this part of u). Then, the word cur satisfies our initial requirements on the height and the size, and we thus have $\mathsf{B}(v_0vv_4) = 0$, as expected.

Consider now a decomposition of ρ according to criterion $(C2)$. If it is completely in the sub-run on u, then the result follows by induction. Otherwise, there exists a decomposition of u as $u = u_1u_2u_3$, with $u_1u_3, u_2 \in \Sigma^*_{\mathsf{wn}}$, and there exists a run $(p', \perp) \xrightarrow{u_1|v_1} (p, \sigma) \xrightarrow{u_2|v_2} (q, \sigma) \xrightarrow{u_3|v_3} (q', \perp)$. First, if $|\sigma| \geq |Q|^2$, then one can find in this run a decomposition according to criterion $(C2)$, and by induction hypothesis on u, the corresponding matched loops have a null effect in the computation of B. When removing these matched loops, we thus obtain new input words u'_1 and u'_3 with output words v'_1 and v'_3 such that $\mathsf{B}(v'_1) + \mathsf{B}(v'_3) = \mathsf{B}(v_1) + \mathsf{B}(v_3)$, and $\mathsf{B}(v_1) \geq \mathsf{B}(v'_1)$. We thus assume now that $|\sigma| < |Q^2|$ and let $k = |\sigma|$.

We decompose u'_1 as follows: $u'_1 = w_0c_1w_1c_2 \dots c_kw_k$. Using standard horizontal and vertical pumping (Lemma 13) and the induction hypothesis on u, we can assume that the height of w_i's is strictly less than $|Q|^2$ and that the length of w_i's is strictly less than $|Q|^{|Q|^2}$. A similar reasoning can be done on word u'_3. Thus, the two words cu'_1 and u'_3r satisfy the conditions on their height and size to ensure that $\mathsf{B}(v_0v'_1) + \mathsf{B}(v'_3v_4) = 0$, and $\mathsf{B}(v_0v'_1) \geq 0$. This entails $\mathsf{B}(v_0v_1) + \mathsf{B}(v_3v_4) = 0$, and $\mathsf{B}(v_0v_1) \geq 0$ as we wanted to prove.

This concludes the induction, and thus the proof. □

The previous algorithm could be extended to handle in addition criterion (D), yielding a PSPACE algorithm to decide whether a VPT A is globally well-nested. However, we can use a recent result to prove that this problem is in PTIME.

Theorem 30. *Let A be a VPT. Whether $[\![A]\!] \in \mathcal{G}_{\mathsf{wn}}$ can be decided in* PTIME.

Proof. This proof heavily relies on results from [3, 11] showing that deciding whether a context-free language is included into a Dyck language can be solved in PTIME.

We first erase the precise symbols of the produced outputs keeping track only of the type of the symbols: we build from A a VPT A' defined on the output alphabet Σ' with $\Sigma'_c = \{\mathsf{(}\}$, $\Sigma'_r = \{\mathsf{)}\}$ and $\Sigma'_i = \emptyset$. A transition of A' is obtained from a transition of A by replacing in output words of the transition of A call symbols by $\mathsf{(}$ and return symbols by $\mathsf{)}$ and removing internal symbols. It is then easy to see that A is in gwnVPT iff A' is in gwnVPT (actually, for each run in A producing v, its corresponding run in A' produces some v' such that $\mathsf{O}(v) = \mathsf{O}(v')$). Then, as shown in [9], one can build in polynomial time a context-free grammar $G_{A'}$ generating the range of A'. Finally, we appeal to [3, 11] to conclude. $\square$

6. Closure under composition and Type-checking

6.1. *Definitions and existing results*

In this section, we consider two natural problems for transducers : the first one is related to composition of transductions. The second problem is the type-checking problem that aims to verify that any output of a transformation belongs to some given type/language. For VPT, the natural class of "types" to consider is VPL.

Definition 31 (Closure under composition) *A class $\mathcal{T}$ of transductions included in $\Sigma^* \times \Sigma^*$ is closed under composition if for all T, T' in $\mathcal{T}$, the transduction $T \circ T'$ is also in $\mathcal{T}$. It is effectively closed under composition if for any transducers A, A' such that $[\![A]\!], [\![A']\!] \in \mathcal{T}$, $A \circ A'$ is computable and $[\![A \circ A']\!]$ is in $\mathcal{T}$.*

A class of transducers T is effectively closed under composition if for any two transducers A, A' in T, $A \circ A'$ is computable and $A \circ A'$ is in T.

Definition 32 (Type-checking (against VPA)) *Given a VPT A and two VPA B, C, decide whether $[\![A]\!](L(B)) \subseteq L(C)$.*

The following results give the status of these properties for arbitrary VPTs and for lwnVPT:

Theorem 33 ([8, 7]) *Regarding closure under composition, we have:*

- *The class $\mathcal{VP}(\Sigma, \Sigma)$ is not closed under composition.*
- *The class lwnVPT is effectively closed under composition.*

In addition, the problem of type checking against VPA is undecidable for (arbitrary) VPT and is EXPTIME-complete for lwnVPT.

6.2. *New results*

Actually, regarding the closure under composition of the class lwnVPT, though it is not explicitly stated, the result proved in [7] is slightly stronger. It is indeed shown that for any VPT A, B such that $A \in$ lwnVPT, there exists an (effectively computable) VPT C satisfying $[\![C]\!] = [\![A]\!] \circ [\![B]\!]$. In addition, if $B \in$ lwnVPT, then $C \in$ lwnVPT.

We extend this positive result to any almost well-nested transducer; one of the main ingredients to prove this result is the set $\mathcal{UPS}_A$ defined for any VPT A as

$$\left\{ (p, p', n_1, n_2) \,\middle|\, \begin{array}{l} \exists \sigma \in \Gamma^*,\ (p, \sigma) \text{ is reachable and } (p', \sigma) \text{ is co-reachable and} \\ \exists u \in \Sigma^*_{\mathsf{wn}}, (p, \bot) \xrightarrow{u|v} (p', \bot) \text{ and } \mathsf{O}(v) = (n_1, n_2) \end{array} \right\}$$

Proposition 34. *Let A in awnVPT. Then the set $\mathcal{UPS}_A$ is finite and computable in exponential time in the size of A.*

$$\frac{(p,p)\in\mathcal{IPC}^A_{\mathsf{wn}}}{(p,p,(0,0))\in\mathcal{S}}$$

$$\frac{(p,i,v,q')\in\delta_i \quad (q',q,(n_1,n_2))\in\mathcal{S} \quad (p,q)\in\mathcal{IPC}^A_{\mathsf{wn}}}{(p,q,\mathsf{O}(v)\oplus(n_1,n_2))\in\mathcal{S}}$$

$$\frac{\begin{array}{c}(p,c,\gamma,v_1,p')\in\delta_c \quad (p',q',(n_1,n_2))\in\mathcal{S} \quad (q',r,\gamma,v_2,q'')\in\delta_r \quad (q'',q,(n_1',n_2'))\in\mathcal{S}\\ (p,q)\in\mathcal{IPC}^A_{\mathsf{wn}}\end{array}}{(p,q,\mathsf{O}(v_1)\oplus(n_1,n_2)\oplus\mathsf{O}(v_2)\oplus(n_1',n_2'))\in\mathcal{S}}$$

Figure 2. A rule system characterizing the set $\mathcal{UPS}_A$.

Proof. For a VPT $A=(Q,I,F,\Gamma,\delta)$, we consider triples of the form $(p,q,(n_1,n_2))$ where $p,q\in Q$ such that $(p,q)\in\mathcal{IPC}^A_{\mathsf{wn}}$ and $(n_1,n_2)\in\mathbb{N}\times\mathbb{N}$. We consider the set $\mathcal{S}$ of such triples, as the least one satisfying the rules from Figure 2.

Using an induction on the structure of runs, one easily proves that $\mathcal{S}=\mathcal{UPS}_A$. In addition, the set $\mathcal{S}$ is finite and can be computed in exponential time. Obviously, rules from Figure 2 can be turned into an algorithm whose iterations will inspect each inference rule for each possible inputs. Such an iteration may add at least one new triple in $\mathcal{S}$. As $\mathcal{S}=\mathcal{UPS}_A$ and by Proposition 25, it is known that in triples $(p,p',(n_1,n_2))$, n_1, n_2 are bounded exponentially in the size of A which ensured termination for the algorithm. □

Theorem 35. *Let A,B be two* VPT*s. If A is almost-well nested, then one can compute in time $2^{2^{O(|A|)}*O(\log_2(|B|))}$ a* VPT *C such that $[\![C]\!]=[\![A]\!]\circ[\![B]\!]$. Moreover, if B is also almost well-nested, then so is C, and if A and B are globally well-nested, then so is C.*

Proof. We present the construction of C. By Proposition 34, $\mathcal{UPS}_A$ is finite and we let K be the computable integer value $\max\{||(n_1,n_2)||\mid(p,p',n_1,n_2)\in\mathcal{UPS}_A\}$.

Given $A=(Q_A,I_A,F_A,\Gamma_A,\delta^A)$ and $B=(Q_B,I_B,F_B,\Gamma_B,\delta^B)$, we define $C=(Q_C,I_C,F_C,\Gamma_C,\delta^C)$ as

$$\begin{array}{ll} Q_C=Q_A\times Q_B\times\Gamma_B^{\leq K} & I_C=I_A\times I_B\times\{\bot\}\\ \Gamma_C=\Gamma_A\times\Gamma_B^{\leq \mathsf{O}^A_{\max}+K} & F_C=F_A\times F_B\times\{\bot\}\end{array}$$

Now for the transition rules δ^C:

- $((p,q,\sigma),i,w,(p',q',\sigma'))\in\delta^C_i$ if there exist a word $v\in\Delta^*$ and a stack $\sigma_0\in\Gamma_B^*$ such that $\sigma=\sigma_0\sigma_1$, $\sigma'=\sigma_0\sigma_1'$, $\mathsf{O}(v)=(|\sigma_1|,|\sigma_1'|)$, and $(p,i,v,p')\in\delta^A_i$ and there exists a run $(q,\sigma_1)\xrightarrow{v|w}(q',\sigma_1')$ in B,
- $((p,q,\sigma),c,w,(\gamma,\sigma_3),(p',q',\sigma_4))\in\delta^C_c$ if there exist a word $v\in\Delta^*$, two stacks $\sigma_0,\sigma_2\in\Gamma_B^*$ and a stack symbol $\gamma\in\Gamma_A$ such that $\sigma=\sigma_0\sigma_1$,

$\mathsf{O}(v) = (|\sigma_1|, |\sigma_2|)$, $\sigma_0\sigma_2 = \sigma_3\sigma_4$, $(p, c, v, \gamma, p') \in \delta_c^A$ and there exists a run $(q, \sigma_1) \xrightarrow{v|w} (q', \sigma_2)$ in B, such a transition exists provided the bounds on the sizes of the different stacks are fulfilled, *i.e.* $|\sigma| \leq K$, $|\sigma_4| \leq K$, and $|\sigma_3| \leq \mathsf{O}^A_{\max} + K$

- $((p, q, \sigma), r, w, (\gamma, \sigma_3), (p', q', \sigma')) \in \delta_r^C$ if there exist a word $v \in \Delta^*$, a stack $\sigma_0 \in \Gamma_B^*$ such that $\sigma_0\sigma_1 = \sigma_3\sigma$, $\sigma_0\sigma_2 = \sigma'$, $\mathsf{O}(v) = (|\sigma_1|, |\sigma_2|)$, $(p, r, v, \gamma, p') \in \delta_r^A$ and there exists a run $(q, \sigma_1) \xrightarrow{v|w} (q', \sigma_2)$ in B such a transition exists provided the bounds on the sizes of the different stacks are fulfilled, *i.e.* $|\sigma| \leq K$, $|\sigma'| \leq K$, and $|\sigma_3| \leq \mathsf{O}^A_{\max} + K$

Intuitively, in a state of C, we store the current states of A and B. In addition, a part of the top of the stack of B is also stored in the state of C to allow the simulation of B. The (finite) amount that needs to be stored in the state is identified using the set $\mathcal{UPS}_A$.

To prove the correction of this construction, we consider the following definition:

Definition 36. *Given a run $\rho : (p, \bot) \xrightarrow{u|v} (p', \bot)$ of A, we define* Max-bal(ρ) *as the largest non-negative integer n such that ρ can be decomposed as $\rho : (p, \bot) \xrightarrow{u_1|v_1} (p_1, \bot) \xrightarrow{u_2|v_2} (p', \bot)$ and $n = \mathsf{Or}(v_2)$ or $n = \mathsf{Oc}(v_1)$.*

The following lemma states that a run of A and a run of B can be combined to build a run of C, provided there exists a word $v \in \Delta^*$ produced by the run of A, and taken as input of the run of B. Observe also that we may add a stack in the state of C, considered as an unused stack added "below" the run of B, provided the size of the resulting stack stored in the state never exceeds the bound K. This is obtained thanks to the definition of Max-bal. The proof is a rather standard (but technical) induction on the length of u.

Lemma 37. *If we have a run $\rho : (p, \bot) \xrightarrow{u|v} (p', \bot)$ of A, two stacks $\sigma, \sigma' \in \Gamma_B^*$ such that $\mathsf{O}(v) = (|\sigma|, |\sigma'|)$, and a run $(q, \sigma) \xrightarrow{v|w} (q', \sigma')$ of B, then $((p, q, \sigma_0\sigma), \bot) \xrightarrow{u|w} ((p', q', \sigma_0\sigma'), \bot)$ is a run of C, for every stack $\sigma_0 \in \Gamma_B^{\leq K - \text{Max-bal}(\rho)}$.*

The next lemma states the result in the other way: a run in C implies the existence of corresponding runs in A and B.

Lemma 38. *If there exists a run $((p, q, \sigma), \bot) \xrightarrow{u|w} ((p', q', \sigma'), \bot)$ in C, then there exists a word $v \in \Delta^*$ and a stack $\sigma_0 \in \Gamma_B^*$ such that:*

- *there exists a run $\rho : (p, \bot) \xrightarrow{u|v} (p', \bot)$ in A*
- *$\sigma = \sigma_0\sigma_1$, $\sigma' = \sigma_0\sigma_2$, and $\mathsf{O}(v) = (|\sigma_1|, |\sigma_2|)$*
- *there exists a run $(q, \sigma_1) \xrightarrow{v|w} (q', \sigma_2)$ in B*

Theorem 35 follows from Lemma 37 and 38 and from the definition of initial and final states. □

Corollary 39. *The classes $\mathcal{G}_{\mathsf{wn}}$ and $\mathcal{A}_{\mathsf{wn}}$ are (effectively) closed under composition.*

Theorem 40 (Type-checking against VPA) *Given an almost or a globally well-nested* VPT *A and two* VPA *B, C, whether $[\![A]\!](L(B)) \subseteq L(C)$ is* 2EXPTIME*−complete.*

Proof. We prove the upper-bound complexity for A being almost well-nested and the lower bound for A being globally well-nested.

For the upper bound, restricting the domain of A to $L(B)$ is easy: it suffices to compute the product VPA of A and B. Then, VPA being closed under complementation, we compute $\overline{C}$, the complement of C. Note that the size of $\overline{C}$ is at most exponential in the size of C. We then turn $\overline{C}$ into a transducer C' defining the identity relation over $L(\overline{C})$ (this is obvious by simply transforming rules of $\overline{C}$ into rules of transducers outputting their input). Now, by Theorem 35, one can build a transducer defining the composition of $[\![A]\!] \circ [\![C']\!]$ in doubly exponential time in the size of A and B and single exponential time in the size of C. Now, it is sufficient to test whether the VPA underlying this transducer is empty or not.

Now, for the lower bound, we appeal to a result from [11] stating that the inclusion problem of a context-free language into a parenthesis language given respectively as a context-free grammar G and a parenthesis grammar G_P is 2EXPTIME-hard. More precisely, we recall that a context-free grammar is defined by a set of terminals T, a set of non-terminals N (amongst which one is distinguished and called the axiom) and a set P of productions. Parenthesis grammars are a particular case of context-free grammars whose productions are of one of the forms

$$X \to \mathsf{(}\, Y_1, \ldots, Y_n \,\mathsf{)} \qquad X \to a$$

for some non terminal $X, Y_1, \ldots, Y_n$ and $\mathsf{(}$, $\mathsf{)}$, a stand for terminals from T such that a is different from $\mathsf{(}$ and from $\mathsf{)}$.

We prove now that this problem can be reduced in polynomial time to the type-checking problem for gwnVPT. First, note that it is known from [8] that a VPT A_G whose range is precisely $L(G)$, the language defined by G, can be built from G in polynomial time. Note also that for the structured alphabet $\Sigma = (\{\,\mathsf{(}\,\}, \{\,\mathsf{)}\,\}, T \setminus \{\mathsf{(}\,,\,\mathsf{)}\})$, the language $L(G_P)$ defined by the grammar G_P is included into Σ^*_{wn}. Finally, we can show that $L(G_P)$ is actually a VPL. Indeed, we build in polynomial time from G_P a VPA A_{G_P} as follows: states Q of A_{G_P} are strict suffixes of right-hand sides of rules of the form $X \to \mathsf{(}\, Y_1, \ldots, Y_n \,\mathsf{)}$ together with the axiom of the grammar S. The initial state is S and the unique final state is the empty suffix ϵ. The stack alphabet is identical to Q and the transition rules are defined as:

$$\begin{array}{ll} X\omega \xrightarrow{\mathsf{(}\,,\,\omega} \omega' \;\in \delta_c & \text{if } X \to \mathsf{(}\omega' \in G_P \\ X\omega \xrightarrow{a} \omega \;\in \delta_i & \text{if } X \to a \in G_P \\ \mathsf{)} \xrightarrow{\mathsf{)}\,,\,\omega} \omega \;\in \delta_r & \end{array}$$

For the correctness of the construction, one can prove that the configuration $(\alpha_k \ldots \alpha_n, \gamma_1 \ldots \gamma_\ell)$ of A_{G_P} is reached after reading the nested word w iff $S \Rightarrow^*$

$w\alpha_k \dots \alpha_n \gamma_\ell \dots \gamma_1$ is a left derivation of G_P, where the γ_i are strict suffixes of right-hand sides of productions.

We now give the reduction for the inclusion test to the type-checking problem:

$\boldsymbol{L(G) \subseteq L(G_P)}$?

- Build in polynomial time from G the VPT A_G.
- Test whether A_G is globally well-nested in polynomial time (Theorem 30)
- If the test fails, return false
- Build in polynomial time the VPA A_{G_P}
- Return $[\![A_G]\!](\Sigma^*) \subseteq L(A_{G_P})$ (Type-checking) $\square$

7. Conclusion

In this paper, we have considered and precisely characterized the class of VPT with well-nested outputs. We have shown that this class is closed under composition and that its type-checking against VPA is decidable. We have restricted ourselves in this paper to transducers with well-nested domains. We conjecture that this restriction can be easily relaxed and thus, one could consider transducers based on nested word automata [2]. We left open the problem of deciding the class $\mathcal{L}_{\mathsf{wn}}$. As we have described on some examples, this problem is far from being trivial. In [5], a clear relationship between the class lwnVPT and hedge-to-hedge transducers is described; investigating such a relationship for gwnVPT is also an interesting problem.

Bibliography

[1] R. Alur and P. Madhusudan, Visibly Pushdown Languages, *STOC*, (2004), pp. 202–211.

[2] R. Alur and P. Madhusudan, Adding Nesting Structure to Words, *JACM* **56**(3) (2009) 1–43.

[3] A. Bertoni, C. Choffrut and R. Radicioni, The inclusion problem of context-free languages: Some tractable cases, *Int. J. Found. Comput. Sci.* **22**(2) (2011) 289–299.

[4] B. v. Braunmühl and R. Verbeek, Input-driven Languages are Recognized in log n Space, *FCT*, *LNCS* **158**, (Springer, 1983), pp. 40–51.

[5] M. Caralp, E. Filiot, P.-A. Reynier, F. Servais and J.-M. Talbot, Expressiveness of visibly pushdown transducers, *Proceedings Second International Workshop on Trends in Tree Automata and Tree Transducers, TTATT 2013*, *EPTCS* **134** (2013), pp. 17–26.

[6] E. Filiot, O. Gauwin, P.-A. Reynier and F. Servais, Streamability of Nested Word Transductions, *FSTTCS*, *LIPIcs* **13**, (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011), pp. 312–324.

[7] E. Filiot, J.-F. Raskin, P.-A. Reynier, F. Servais and J.-M. Talbot, Properties of Visibly Pushdown Transducers, *MFCS'10*, *LNCS* **6281**, (Springer, 2010), pp. 355–367.

[8] J.-F. Raskin and F. Servais, Visibly pushdown transducers, *ICALP*, *LNCS* **5126** (2008), pp. 386–397.

[9] F. Servais, Visibly pushdown transducers, PhD thesis, Université Libre de Bruxelles (2011).

[10] S. Staworko, G. Laurence, A. Lemay and J. Niehren, Equivalence of deterministic nested word to word transducers, *FCT*, *LNCS* **5699** (2009), pp. 310–322.

[11] A. Tozawa and Y. Minamide, Complexity results on balanced context-free languages, *FOSSACS*, *LNCS* **4423**, (Springer, 2007), pp. 346–360.