Streamability of Nested Word Transductions*

Emmanuel Filiot[1], Olivier Gauwin[2], Pierre-Alain Reynier[3], and Frédéric Servais[4]

1   Université Libre de Bruxelles
2   Université de Mons
3   LIF, Aix-Marseille Univ & CNRS, France
4   Hasselt University and Transnational University of Limburg

──── **Abstract** ────

We consider the problem of evaluating in streaming (*i.e.* in a single left-to-right pass) a nested word transduction with a limited amount of memory. A transduction T is said to be height bounded memory (HBM) if it can be evaluated with a memory that depends only on the size of T and on the height of the input word. We show that it is decidable in coNPTime for a nested word transduction defined by a visibly pushdown transducer (VPT), if it is HBM. In this case, the required amount of memory may depend exponentially on the height of the word. We exhibit a sufficient, decidable condition for a VPT to be evaluated with a memory that depends quadratically on the height of the word. This condition defines a class of transductions that strictly contains all determinizable VPTs.

1   Introduction

Memory analysis is an important tool for ensuring system robustness. In this paper we focus on the analysis of programs processing *nested words* [2], *i.e.*, words with a recursive structure, like program traces, XML documents, or more generally unranked trees. On huge inputs, a *streaming* mode is often used, where the nested word is read only once, from left to right. This corresponds to a depth-first left-to-right traversal when the nested word is considered as a tree. For such programs, *dynamic analysis* problems have been addressed in various contexts. For instance, runtime verification detects dynamically, and as early as possible, whether a property is satisfied by a program trace [17, 6]. On XML streams, some algorithms outputting nodes selected by an XPath expression at the earliest possible event have also been proposed [7, 12]. These algorithms allow minimal buffering [3].

In this paper, we investigate *static analysis* of memory usage for a special kind of programs on nested words, namely programs defined by *transducers*. We assume that the transducers are *functional* and *non-deterministic*. Non-determinism is required as input words are read from left to right in a single pass and some actions may depend on the future of the stream. For instance, the XML transformation language XSLT uses XPath for selecting nodes where local transformations are applied, and XPath queries relies on

---

non-deterministic moves along tree axes, such as a move to any descendant. We require our transducers to be *functional*, as we are mainly interested by transformation languages like XSLT, XQuery and XQuery Update Facility, for which any transformation maps each XML input document to a unique output document.

Visibly pushdown transducers (VPTs) form a subclass of pushdown transducers adequate for dealing with nested words and streaming evaluation, as the input nested word is processed from left to right. They are visibly pushdown automata [2] extended with arbitrary output words on transitions. VPTs capture interesting fragments of the aforementioned XML transformation languages that are amenable to efficient streaming evaluation, such as all editing operations (insertion, deletion, and relabeling of nodes, as used for instance in XQuery Update Facility) under all regular tests. Like for visibly pushdown automata, the stack behavior of VPTs is imposed by the type of symbols read by the transducer. Those restrictions on stack operations allow to decide functionality and equivalence of functional VPTs in PTime and ExpTime respectively [11].

Some transductions defined by (functional and non-deterministic) VPTs cannot be evaluated efficiently in streaming. For instance, swapping the first and last letter of a word can be defined by a VPT as follows: guess the last letter and transform the first letter into the guessed last letter, keep the value of the first letter in the state, and transform any value in the middle into itself. Any deterministic machine implementing this transformation requires to keep the entire word in memory until the last letter is read. It is not reasonable in practice as for instance XML documents can be very huge.

Our aim is thus to identify decidable classes of transductions for various memory requirements that are suitable to space-efficient streaming evaluation. We first consider the requirement that a transducer can be implemented by a program using a *bounded memory* (BM), *i.e.* computing the output word using a memory independent of the size of the input word. However when dealing with nested words in a streaming setting, the bounded memory requirement is quite restrictive. Indeed, even performing such a basic task as checking that a word is well-nested or checking that a nested word belongs to a regular language of nested words requires a memory dependent on the height (the level of nesting) of the input word [19]. This observation leads us to the second question: decide, given a transducer, whether the transduction can be evaluated with a memory that depends only on the size of the transducer and the height of the word (but not on its length). In that case, we say that the transduction is *height bounded memory* (HBM). This is particularly relevant to XML transformations as XML documents can be very long but have usually a small depth [5]. HBM does not specify *how* memory depends on the height. A stronger requirement is thus to consider HBM transductions whose evaluation can be done with a memory that depends *polynomially* on the height of the input word.

Contributions First, we give a general space-efficient evaluation algorithm for functional VPTs. After reading a prefix of an input word, the number of configurations of the (non-deterministic) transducer as well as the number of output candidates to be kept in memory may be exponential in the size of the transducer and the height of the input word (but not in its length). Our algorithm produces as output the longest common prefix of all output candidates, and relies on a compact representation of sets of configurations and remaining output candidates (the original output word without the longest common prefix). We prove that it uses a memory linear in the height of the input word, and linear in the maximal length of a remaining output candidate.

We prove that BM is equivalent to subsequentiability for finite state transducers (FSTs), which is known to be decidable in PTime. BM is however undecidable for arbitrary push-

down transducers but we show that it is decidable for VPTs in coNPTime.

Like BM, HBM is undecidable for arbitrary pushdown transductions. We show, via a non-trivial reduction to the emptiness of pushdown automata with bounded reversal counters, that it is decidable in coNPTime for transductions defined by VPTs. In particular, we show that the previously defined algorithm runs in HBM iff the VPT satisfies some property, which is an extension of the so called *twinning property* for FSTs [9] to nested words. We call it the *horizontal twinning property*, as it only cares about configurations of the transducers with stack contents of identical height. This property only depends on the transduction, *i.e.* is preserved by equivalent transducers.

When a VPT-transduction is height bounded memory, the memory needed may be exponential in the height of the word. We introduce a refinement of the twinning property that takes the height of the configurations into account, hence called *matched twinning property*. A VPT satisfying this property is called *twinned*. We prove that the evaluation of *twinned transductions* with our algorithm uses a memory *quadratic* in the height of the input word. We show that it is decidable in coNPTime whether a VPT is twinned. Moreover, the most challenging result of this paper is to show that being twinned depends only on the transduction and not on the VPT that defines it. Thus, this property indeed defines a class of transductions. As a consequence of this result, all subsequentializable VPTs are twinned, because subsequential VPTs trivially satisfy the matched twinning property. The class of twinned transductions captures a strictly larger class than subsequentializable VPTs while staying in the same complexity class for evaluation, i.e. polynomial space in the height of the input word when the transducer is fixed.

Related Work   In the XML context, visibly pushdown automata based streaming processing has been extensively studied for validating XML streams [16, 4, 19]. The validation problem with bounded memory is studied in [4] when the input is assumed to be a well-nested word and in [19] when it is assumed to be a well-formed XML document (this problem is still open). Querying XML streams has been considered in [13]. It consists in selecting a set of tuples of nodes in the tree representation of the XML document. For monadic queries (selecting nodes instead of tuples), this can be achieved by a functional VPT returning the input stream of tags, annotated with Booleans indicating selection by the query. However, functional VPTs cannot encode queries of arbitrary arities. The setting for functional VPTs is in fact different to query evaluation, because the output has to be produced on-the-fly in the right order, while query evaluation algorithms can output nodes in any order: an incoming input symbol can be immediately output, while another candidate is still to be confirmed. This makes a difference with the notion of concurrency of queries, measuring the minimal amount of candidates to be stored, and for which algorithms and lower bounds have been proposed [3]. VPTs also relate to tree transducers [11], for which no comparable work on memory requirements is known. However, the height of the input word is known to be a lower bound for Core XPath filters [13]. As VPTs can express them, this lower bound also applies when evaluating VPTs. When allowing two-way access on the input stream, space-efficient algorithms for XML validation [15] and querying [18] have been proposed.

2   Visibly Pushdown Languages and Transductions

Words and nested words   In this paper, we consider nested words accessed in streaming. Their nesting structure is thus discovered on-the-fly, so we consider a finite alphabet Σ partitioned into three disjoint sets Σ_c, Σ_r and Σ_ι, denoting respectively the *call*, *return* and *internal* alphabets. We denote by Σ^* the set of (finite) words over Σ and by ϵ the empty

word. The length of a word u is denoted by $|u|$. For all words $u, v \in \Sigma^*$, we denote by $u \wedge v$ the longest common prefix of u and v. More generally, for any non-empty finite set of words $V \subseteq \Sigma^*$, the longest common prefix of V, denoted by $\mathsf{lcp}(V)$, is inductively defined by $\mathsf{lcp}(\{u\}) = u$ and $\mathsf{lcp}(V \cup \{u\}) = \mathsf{lcp}(V) \wedge u$. The set of *well-nested* words Σ^*_{wn} is the smallest subset of Σ^* such that $\Sigma^*_\iota \subseteq \Sigma^*_{\mathsf{wn}}$ and for all $c \in \Sigma_c$, all $r \in \Sigma_r$, all $u, v \in \Sigma^*_{\mathsf{wn}}$, $cur \in \Sigma^*_{\mathsf{wn}}$ and $uv \in \Sigma^*_{\mathsf{wn}}$. Let $u = \alpha_1 \ldots \alpha_n \in \Sigma^*$ be a prefix of a well-nested word. A position $i \in \{1, \ldots, n\}$ is a *pending call* if $\alpha_i \in \Sigma_c$ and for all $j \geq i$, $\alpha_i \ldots \alpha_j \notin \Sigma^*_{\mathsf{wn}}$. The *height* of u is the maximal number of pending calls on any prefix of u, i.e.

$$h(u) = \max_{1 \leq i \leq n} |\{k \mid 1 \leq k \leq i, \alpha_k \text{ is a pending call of } \alpha_1 \ldots \alpha_i\}|$$

For instance, $h(crcrcc) = h(ccrcrr) = 2$. In particular, for well-nested words, the height corresponds to the usual height of the nesting structure of the word.

Given two words $u, v \in \Sigma^*$, the *delay* of u and v, denoted by $\Delta(u, v)$, is the unique pair of words (u', v') such that $u = (u \wedge v)u'$ and $v = (u \wedge v)v'$. For instance, $\Delta(abc, abde) = (c, de)$. Informally, in a word transduction, if there are two output candidates u and v during the evaluation, we are sure that we can output $u \wedge v$ and $\Delta(u, v)$ is the remaining suffixes we still keep in memory.

Visibly pushdown transducers (VPTs) As finite-state transducers extend finite-state automata with outputs, visibly pushdown transducers extend visibly pushdown automata [2] with outputs [11]. To simplify notations, we suppose that the output alphabet is Σ, but our results still hold for an arbitrary output alphabet. Informally, the stack behavior of a VPT is similar to the stack behavior of visibly pushdown automata (VPA). On a call symbol, the VPT pushes a symbol on the stack and produces some output word (possibly empty), on a return symbol, it must pop the top symbol of the stack and produce some output word (possibly empty) and on an internal symbol, the stack remains unchanged and it produces some output word. Formally:

▶ **Definition 1.** A *visibly pushdown transducer* (VPT) on finite words over Σ is a tuple $T = (Q, I, F, \Gamma, \delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ the set of final states, Γ is the stack alphabet, $\delta = \delta_c \uplus \delta_r \uplus \delta_\iota$ the (finite) transition relation, with $\delta_c \subseteq Q \times \Sigma_c \times \Sigma^* \times \Gamma \times Q$, $\delta_r \subseteq Q \times \Sigma_r \times \Sigma^* \times \Gamma \times Q$, and $\delta_\iota \subseteq Q \times \Sigma_\iota \times \Sigma^* \times Q$.

A *configuration* of a VPT is a pair $(q, \sigma) \in Q \times \Gamma^*$. A *run* of T on a word $u = a_1 \ldots a_l \in \Sigma^*$ from a configuration (q, σ) to a configuration (q', σ') is a finite sequence $\rho = \{(q_k, \sigma_k)\}_{0 \leq k \leq l}$ such that $q_0 = q$, $\sigma_0 = \sigma$, $q_l = q'$, $\sigma_l = \sigma'$ and for each $1 \leq k \leq l$, there exist $v_k \in \Sigma^*$ and $\gamma_k \in \Gamma$ such that either $(q_{k-1}, a_k, v_k, \gamma_k, q_k) \in \delta_c$ and $\sigma_k = \sigma_{k-1}\gamma_k$ or $(q_{k-1}, a_k, v_k, \gamma_k, q_k) \in \delta_r$ and $\sigma_{k-1} = \sigma_k \gamma_k$, or $(q_{k-1}, a_k, v_k, q_k) \in \delta_\iota$ and $\sigma_k = \sigma_{k-1}$. The word $v = v_1 \ldots v_l$ is called an *output* of ρ. 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. We denote by $\bot$ the empty word on Γ. A configuration (q, σ) is *accessible* (resp. is *co-accessible*) if there exist $u, v \in \Sigma^*$ and $q_0 \in I$ (resp. $q_f \in F$) such that $(q_0, \bot) \xrightarrow{u/v} (q, \sigma)$ (resp. such that $(q, \sigma) \xrightarrow{u/v} (q_f, \bot)$). A transducer T is *reduced* if every accessible configuration is co-accessible. Given any VPT, computing an equivalent reduced VPT can be performed in polynomial time [8][1]. A transducer T defines the binary word relation $[\![T]\!] = \{(u, v) \in \Sigma^* \times \Sigma^* \mid \exists q \in I, q' \in F, (q, \bot) \xrightarrow{u/v} (q', \bot)\}$.

A *transduction* is a binary relation $R \subseteq \Sigma^* \times \Sigma^*$. We say that a transduction R is a VPT-transduction if there exists a VPT T such that $R = [\![T]\!]$. For any input word $u \in \Sigma^*$, we denote by $R(u)$ the set $\{v \mid (u, v) \in R\}$. Similarly, for a VPT T, we denote by $T(u)$ the

---

[1] The reduction of VPAs in [8] trivially extends to VPTs.
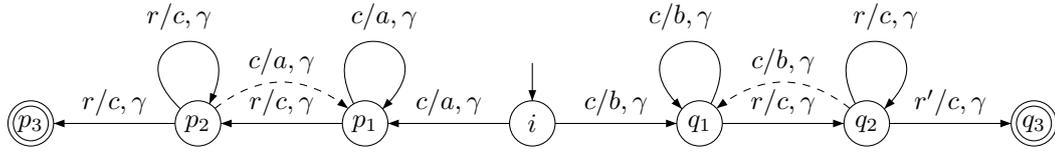
■ **Figure 1** A functional VPT with $\Sigma_c = \{c\}$, $\Sigma_r = \{r, r'\}$ and $\Sigma_\iota = \{a, b\}$

set $[\![T]\!](u)$. A transduction R is *functional* if for all $u \in \Sigma^*$, $R(u)$ has size at most one. If R is functional, we identify $R(u)$ with the unique image of u if it exists. A VPT T is functional if $[\![T]\!]$ is functional, and this can be decided in PTime [11]. The class of functional VPTs is denoted by fVPT. The *domain* of T (denoted by $Dom(T)$) is the domain of $[\![T]\!]$. The domain of T contains only well-nested words, which is not necessarily the case of the codomain.

▶ **Example 2.** Consider the VPT T of Fig. 1 represented in plain arrows. The left and right parts accept the same input words except for the last letter of the word. The domain of T is $Dom(T) = \{c^n r^n \mid n \geq 2\} \cup \{cc^n r^n r' \mid n \geq 1\}$. Any word $c^n r^n$ is translated into $a^n c^n$, and any word $cc^n r^n r'$ is translated into $b^{n+1} c^{n+1}$. Therefore the translation of the first sequence of calls depends on the last letter r or r'. This transformation cannot be evaluated with a bounded amount of memory, but with a memory which depends on the height n of the input word.

Finite state transducers (FSTs)   A *finite state transducer* (FST) on an alphabet Σ is a tuple (Q, I, F, δ) where Q is a finite set, $I, F \subseteq Q$ and $\delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ with the standard semantics. This definition corresponds to the usual definition of *real-time* FSTs, as there is no ϵ-transitions. We always consider real-time FSTs in this paper, so we just call them FSTs.

A *subsequential* FST (resp. VPT) is a pair (T, Ψ) where T is an (input) deterministic FST (resp. VPT) and $\Psi : F \to \Sigma^*$. The outputs of u by (T, Ψ) are the words $v.\Psi(q)$ whenever there is a run of T on u producing v and ending up in some accepting state q.

Given an integer $k \in \mathbb{N}$ and a VPT T, one can define an FST, denoted by $\mathsf{FST}(T, k)$, which is the restriction of T to input words of height less than k. The transducer is naturally constructed by taking as states the configurations (q, σ) of T such that $|\sigma| \leq k$.

Turing Transducers   In order to formally define the complexity classes for evaluation that we target, we introduce a *deterministic* computational model for word transductions that we call *Turing Transducers*. Turing transducers have three tapes: one read-only left-to-right input tape, one write-only left-to-right output tape, and one standard working tape. Such a machine naturally defines a transduction: the input word is initially on the input tape, and the result of the transduction is the word written on the output tape after the machine terminates in an accepting state. We denote by $[\![M]\!]$ the transduction defined by M. The space complexity is measured on the working tape only.

3   Online Evaluation Algorithm of VPT-Transductions

We present an online algorithm LcpIn to evaluate functional word transductions defined by fVPTs. For clarity, we present this algorithm under some assumptions, without loss of generality. First, input words of our algorithms are words $u \in \Sigma^*$ concatenated with a special symbol $\$ \notin \Sigma$, denoting the end of the word. Second, we only consider input words without internal symbols, as they can easily be encoded by successive call and return symbols. Third, input words are supposed to be valid, in the sense that they produce an output. It is indeed easy to extend our algorithms in order to raise an error message when the input is not in the domain, *i.e.* when no run of the VPT applies on the input.

(a) VPT T_1. **(b)** After reading c. **(c)** After reading cc. **(d)** After reading ccr_1.
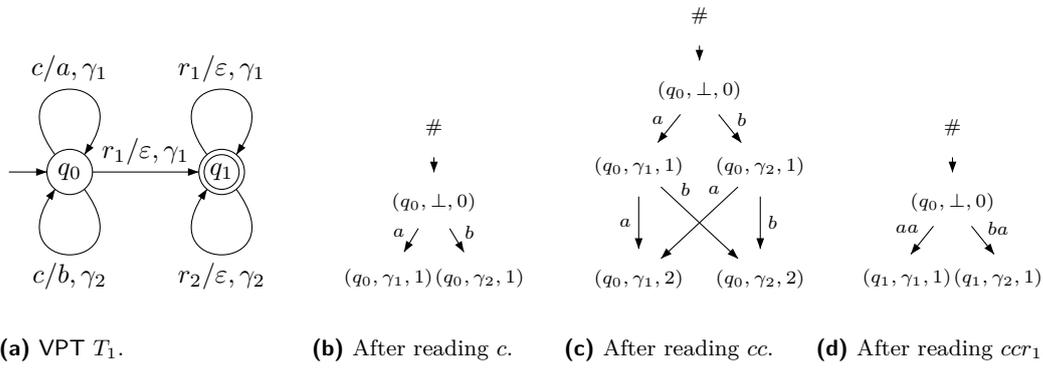
■ **Figure 2** Data structure used by LCPIN.

The core task of this algorithm is to maintain the configuration for each run of the fVPT T on the input u, and produce its output on-the-fly. Therefore, the algorithm LCPIN only applies on reduced fVPTs. Indeed, as T is reduced, functionality ensures that, for a given input word u, and for every accessible configuration (q, σ) of T, there is at most one v such that $(q_i, \bot) \xrightarrow{u/v} (q, \sigma)$ with $q_i \in I$. Hence, a configuration is a triple (q, σ, w) where q is the current state of the run, σ its corresponding stack content, and w the part of the output that has been read but not output yet. We call such a configuration *d-configuration* and write $\mathsf{Dconfs}(T) = Q \times \Gamma^* \times \Sigma^*$ for the set of d-configurations of T. Algorithm LCPIN relies on two main features.

Compact representation First, the set of current d-configurations is stored in a compact structure that shares common stack contents. Consider for instance the VPT T_1 in Fig. 2a. After reading cc, current d-configurations are $\{(q_0, \gamma_1 \gamma_1, aa), (q_0, \gamma_1 \gamma_2, ab), (q_0, \gamma_2 \gamma_1, ba), (q_0, \gamma_2 \gamma_2, bb)\}$. Hence after reading c^n, the number of current d-configurations is 2^n. However, the transition used to update a d-configuration relates the stack symbol and the output word. For instance, the previous set is the set of tuples $(q_0, \eta_1 \eta_2, \alpha_1 \alpha_2)$ where (η_i, α_i) is either (γ_1, a) or (γ_2, b). Based on this observation, we propose a data structure avoiding this blowup. As illustrated in Fig. 2b to 2d, this structure is a directed acyclic graph (DAG). Nodes of this DAG are tuples (q, γ, i) where $q \in Q$, $\gamma \in \Gamma$ and $i \in \mathbb{N}$ is the depth of the node in the DAG. Each edge of the DAG is labelled with a word, so that a branch of this DAG, read from the root $\#$ to the leaf, represents a d-configuration (q, σ, v): q is the state in the leaf, σ is the concatenation of stack symbols in traversed nodes, and v is the concatenation of words on edges. For instance, in the DAG of Fig. 2c, the branch $\# \to (q_0, \bot, 0) \xrightarrow{b} (q_0, \gamma_2, 1) \xrightarrow{a} (q_0, \gamma_1, 2)$ encodes the d-configuration $(q_0, \gamma_2 \gamma_1, ba)$ of the VPT of Fig. 2.(a). However, this data structure cannot store any set of accessible d-configurations of arbitrary functional VPTs: at most one delay w has to be assigned to a d-configuration. This is why we need T to be reduced.

Computing outputs Second, after reading a prefix u' of a word u, LCPIN will have output the common prefix of all corresponding runs, i.e. $\mathsf{lcp}_{\mathsf{in}}(u', T) = \mathsf{lcp}(\mathsf{reach}(u'))$ where $\mathsf{reach}(u') = \{v \mid \exists (q_0, q, \sigma) \in I \times Q \times \Gamma^*, (q_0, \bot) \xrightarrow{u'/v} (q, \sigma)\}$. When a new input symbol is read, the DAG is first updated. Then, a bottom-up pass on this DAG computes $\mathsf{lcp}_{\mathsf{in}}(u', T)$ in the following way. For each node, let ℓ be the largest common prefix of labels of outgoing edges. Then ℓ is removed from these outgoing edges, and concatenated at the end of labels of incoming edges. At the end, the largest common prefix of all output words on branches is the largest common prefix of words on edges outgoing from the root node $\#$.

Let $\mathsf{out}_{\neq}(u')$ be the maximal size of outputs of T on u' where their common prefix is

removed: $\text{out}_{\neq}(u') = \max_{v \in \text{reach}(u')} |v| - |\text{lcp}_{\text{in}}(u', T)|$ and $\text{out}_{\neq}^{\max}(u)$ its maximal value over prefixes of u: $\text{out}_{\neq}^{\max}(u) = \max_{u' \text{ prefix of } u} \text{out}_{\neq}(u')$. To summarize, one can in polynomial time reduce T if necessary, and then build the Turing transducer associated with the algorithm LCPIN. We prove the following complexity result:

▶ **Proposition 3.** *Given an* fVPT *T, one can build in* PTIME *a Turing transducer, denoted* LCPINTT(T), *which, on an input stream $u \in \Sigma^*$, runs in space complexity* $O((h(u) + 1) \cdot \text{out}_{\neq}^{max}(u))$.

In addition, when T is reduced, we can detail how the constant depends on the size of T. The space used by LCPINTT(T) for computing $T(u)$ is in $O(|Q|^2 \cdot |\Gamma|^2 \cdot (h(u) + 1) \cdot \text{out}_{\neq}^{\max}(u))$.

4   Bounded Memory Evaluation Problems

Bounded Memory Transductions

We first consider transductions that can be evaluated with a constant amount of memory if we fix the machine that defines the transduction:

▶ **Definition 4.** A (functional) transduction $R \subseteq \Sigma^* \times \Sigma^*$ is *bounded memory (BM)* if there exists a Turing transducer M and $K \in \mathbb{N}$ such that $[\![M]\!] = R$ and on any input word $u \in \Sigma^*$, M runs in space complexity at most K.

It is not difficult (see [10]) to verify that for FST-transductions, bounded memory is characterized by subsequentializability, which is decidable in PTIME [20]. Moreover, BM is undecidable for pushdown transducers, since it is as difficult as deciding whether a pushdown automaton defines a regular language. For VPTs, BM is quite restrictive as it imposes to verify whether a word is well-nested by using a bounded amount of memory. This can be done only if the height of the words of the domain is bounded by some constant which depends on the transducer only:

▶ **Proposition 5.** *Let T be a functional* VPT *with n states.*

1. $[\![T]\!]$ *is BM iff (i) for all $u \in Dom(T)$, $h(u) \leq n^2$, and (ii)* FST(T, n^2) *is BM;*
2. *It is decidable in* CONPTIME *whether* $[\![T]\!]$ *is BM.*

Sketch. The first assertion is obvious by using simple pumping techniques to show that bounded memory implies bounded height. In the sequel, we define the class of height bounded memory transductions, and show it is decidable in CONPTIME. On words of bounded height, this class collapses with bounded memory transductions.   ◀

Height Bounded Memory Transductions

As we have seen, bounded memory is too restrictive to still benefit from the extra expressiveness of VPT compared to FST, namely the ability to recognize nested words of unbounded height. In this section, we define a notion of bounded memory which is well-suited to VPTs.

▶ **Definition 6.** A (functional) transduction $R \subseteq \Sigma^* \times \Sigma^*$ is *height bounded memory (HBM)* if there exists a Turing transducer M and a function $f : \mathbb{N} \to \mathbb{N}$ such that $[\![M]\!] = R$ and on any input word $u \in \Sigma^*$, M runs in space at most $f(h(u))$.

Note that this definition ensures that the machine cannot store all the input words on the working tape in general. The VPT in Fig. 2a is not in BM, but is in HBM: the stack content suffices (and is necessary) to determine the output. When the structured alphabet contains only internal letters, HBM and BM coincide, thus it is undecidable whether a pushdown transducer is HBM. The remainder of this section is devoted to the proof that HBM is decidable for fVPTs.

BM functional FST-transductions (or equivalently subsequentializable FSTs) are characterized by the so called *twinning property* [9], which is decidable in PTime [20]. We introduce a similar characterization of HBM fVPTs-transductions, called the *horizontal twinning property* (HTP). The restriction of the horizontal twinning property to FSTs is equivalent to the usual twinning property for FSTs (see [10]). Intuitively, the HTP requires that two runs on the same input cannot accumulate increasing output delay on loops.

▶ **Definition 7.** Let T be an fVPT. T satisfies the *horizontal twinning property* (HTP) if for all $u_1, u_2, v_1, v_2, w_1, w_2 \in \Sigma^*$ such that u_2 is well-nested, for all $q_0, q_0' \in I$, for all $q, q' \in Q$, and for all $\sigma, \sigma' \in \Gamma^*$ such that (q, σ) and (q', σ') are co-accessible,

$$\text{if} \begin{cases} (q_0, \bot) \xrightarrow{u_1/v_1} (q, \sigma) \xrightarrow{u_2/v_2} (q, \sigma) \\ (q_0', \bot) \xrightarrow{u_1/w_1} (q', \sigma') \xrightarrow{u_2/w_2} (q', \sigma') \end{cases} \quad (1) \quad \text{then } \Delta(v_1, w_1) = \Delta(v_1 v_2, w_1 w_2).$$

▶ **Example 8.** Consider the VPT of Fig. 1 (including dashed arrows). It does not satisfy the HTP, as the delays increase when looping on *crcr...* Without the dashed transitions, the HTP is satisfied.

▶ **Lemma 9.** *The HTP is decidable in* CoNPTime *for* fVPTs.

Proof. First, let us show that an fVPT T does not satisfy the HTP if and only if there exist $u_1, u_2, v_1, v_2, w_1, w_2 \in \Sigma^*$, $q_0, q_0' \in I$, $q, q' \in Q$, and $\sigma, \sigma' \in \Gamma^*$ such that (q, σ) and (q', σ') are co-accessible, satisfy (1), and such that either we have (i) $|v_2| \neq |w_2|$, or (ii) $|v_2| = |w_2|$, $|v_1| \leq |w_1|$ and not $v_1 v_2 \preceq w_1 w_2$. Indeed, one can easily check that it is a necessary condition. To prove that it is a sufficient condition, suppose we have elements that satisfy (1) with $\Delta(v_1, w_1) \neq \Delta(v_1 v_2, w_1 w_2)$ but conditions (i) and (ii) do not hold. Wlog, we can assume that $|v_1| \leq |w_1|$, therefore we have $|v_2| = |w_2|$, $|v_1| \leq |w_1|$, $v_1 v_2 \preceq w_1 w_2$ and $\Delta(v_1, w_1) \neq \Delta(v_1 v_2, w_1 w_2)$. One can verify (see [10]) that there exists $k \in \mathbb{N}$ such that replacing u_2 with $u_2' = u_2^k$ yields a system that satisfies (ii).

Second, let T be an fVPT, we define a pushdown automaton with bounded reversal counters [14], A, such that the language of A is empty if and only if T satisfies the HTP. More precisely, A accepts the words $u = u_1 u_2 u_3 \in Dom(T)$ such that there exist $v_1, v_2, w_1, w_2 \in \Sigma^*$, $q_0, q_0' \in I$, $q, q' \in Q$, and $\sigma, \sigma' \in \Gamma^*$ that satisfy (1) and either (i) or (ii). A simulates in parallel any two runs of T on the input word (product automaton). It guesses the end of u_1 and stores the states q and q' of the first and second run (in order to be able to check that the simulated runs of T are in state q, resp. q' after reading u_2). Non-deterministically, it checks whether (i) or (ii) holds. To check (i), it uses two counters, one for each run. It does so by, after reading u_1, increasing the counters by the length of the output word of each transition of the corresponding run. Then, when reaching the end of u_2 it checks that both counters are different (by decreasing in parallel both counters and checking they do not reach 0). Similarly, using two other counters, A checks that (ii) holds as follows. Note that condition (ii) implies that there is a position p such that the p-th letter a_1 of $v_1 v_2$ and the p-th letter a_2 of $w_1 w_2$ are different. The automaton A guesses the position $p \in \mathbb{N}$ of the mismatch, and initializes both counters to the value p. Then, while reading $u_1 u_2$, it decreases each counter by the length of the output words of the corresponding run. When

a counter reaches 0, A stores the output letter of the corresponding run. Finally, A checks that $a_1 \neq a_2$, and that both configurations are co-accessible. T satisfies the HTP iff the language of A is empty. The latter is decidable in CONPTIME [11]. ◀

We now show that HTP characterizes HBM fVPTs-transductions and therefore by Lemma 9 we get:

▶ **Theorem 10.** *Let T be an* fVPT*. Then $[\![T]\!]$ is HBM iff the HTP holds for T, which is decidable in* CONPTIME*. In this case, the Turing transducer* LCPINTT(T) *runs, on an input stream u, in space complexity exponential in the height of u.*

We can state more precisely the space complexity of LCPINTT(T) when T is reduced. In this case, it is in $O(|Q|^4 \cdot |\Gamma|^{2h(u)+2} \cdot (h(u)+1) \cdot M)$, where $M = \max\{|v| \mid (q, a, v, \gamma, q') \in \delta\}$.

Sketch. We prove that $[\![T]\!]$ is HBM iff the HTP holds for T. To prove that the HTP is a necessary condition to be in HBM, we proceed by contradiction. We find a counter-example for the HTP and we let K be the height of the input word of this counter-example. It implies that the twinning property for FSTs does not hold for FST(T, K), and therefore FST(T, K) is not BM by Proposition 5. In particular, T is not HBM.

For the converse, it can easily be shown that when T satisfies the HTP, the procedure of [8] that reduces T preserves the HTP satisfiability. In particular, there is a one-to-one mapping between the runs of T and the runs of its reduction that preserves the output words. We then show that for any input word $u \in \Sigma^*$, the maximal delay $\mathsf{out}_{\neq}^{\max}(u)$ between the outputs of u is bounded by $(|Q| \cdot |\Gamma|^{h(u)})^2 M$. This is done by a pumping technique "by width" that relies on the property $\Delta(vv', ww') = \Delta(\Delta(v, w) \cdot (v', w'))$ for any words v, v', w, w'. Thus for an input word for which there are two runs that pass by the same configurations twice at the same respective positions, the delay of the output is equal to the delay when removing the part in between the identical configurations. Finally we apply Proposition 3. ◀

HBM is tight Theorem 10 shows that the space complexity of a VPT in HBM is at most exponential. We give here an example illustrating the tightness of this bound. The idea is to encode the tree transduction $f(t, a) \mapsto f(t, a) \cup f(t, b) \mapsto f(\bar{t}, b)$ by a VPT, where t is a binary tree over $\{0, 1\}$ and $\bar{t}$ is the mirror of t, obtained by replacing the 0 by 1 and the 1 by 0 in t. Thus taking the identity or the mirror depends on the second child of the root f. To evaluate this transformation in a streaming manner, one has to store the whole subtree t in memory before deciding to transform it into t or $\bar{t}$. The evaluation of this transduction cannot be done in polynomial space as there are a doubly exponential number of trees of height n, for all $n \geq 0$.

HBM vs Subsequentializable fVPTs We have seen that a functional transduction defined by an FST T is BM iff T is subsequentializable. We give an example illustrating that for VPTs, being subsequentializable is too strong to characterize HBM. Consider the VPT of Fig. 1 defined by the plain arrows. The transduction it defines is in HBM by Proposition 3, as at any time the delay between two outputs is bounded by the height of the input: $\mathsf{out}_{\neq}^{\max}(u) \leq 2h(u)$. However it is not subsequentializable, as the transformation of c into a or b depends on the last return.

5 Quadratic Height Bounded Memory Evaluation

In the previous section, we have shown that a VPT-transduction is in HBM iff the horizontal twinning property holds, and if it is in HBM, the algorithm of Section 3 uses a memory

at most exponential in the height of the word, and this bound is tight. To avoid this exponential cost, we identify in this section a subclass of HBM containing transductions for which the evaluation algorithm of Section 3 uses a memory *quadratic in the height of the word*. Therefore, we strengthen the horizontal twinning property by adding some properties for well-matched loops. Some of our main and challenging results are to show the decidability of this property and that it depends only on the transduction, i.e. is preserved by equivalent transducers. We show that subsequential VPTs satisfy this condition and therefore our class *subsumes* the class of subsequentializable transducers.

The property is a strengthening of the horizontal twinning property that we call the *matched twinning property (MTP)*. Intuitively, the MTP requires that two runs on the same input cannot accumulate increasing output delay on well-matched loops. They can accumulate delay on loops with increasing stack but this delay has to be caught up on the matching loops with descending stack.

▶ **Definition 11.** Let $T = (Q, I, F, \Gamma, \delta)$ be an fVPT. T satisfies the *matched twinning property* (MTP) if for all $u_i, v_i, w_i \in \Sigma^*$ ($i \in \{1, \dots, 4\}$) such that u_3 is well-nested, and $u_2 u_4$ is well-nested, for all $i, i' \in I$, for all $p, q, p', q' \in Q$, and for all $\sigma_1, \sigma_2 \in \perp.\Gamma^*$, for all $\sigma_1', \sigma_2' \in \Gamma^*$, such that (q, σ_1) and (q', σ_2) are co-accessible:

$$\text{if } \begin{cases} (i, \perp) & \xrightarrow{u_1/v_1} & (p, \sigma_1) & \xrightarrow{u_2/v_2} & (p, \sigma_1\sigma_1') & \xrightarrow{u_3/v_3} & (q, \sigma_1\sigma_1') & \xrightarrow{u_4/v_4} & (q, \sigma_1) \\ (i', \perp) & \xrightarrow{u_1/w_1} & (p', \sigma_2) & \xrightarrow{u_2/w_2} & (p', \sigma_2\sigma_2') & \xrightarrow{u_3/w_3} & (q', \sigma_2\sigma_2') & \xrightarrow{u_4/w_4} & (q', \sigma_2) \end{cases}$$

then $\Delta(v_1 v_3, w_1 w_3) = \Delta(v_1 v_2 v_3 v_4, w_1 w_2 w_3 w_4)$. We say that a VPT T is *twinned* whenever it satisfies the MTP.

Note that any twinned VPT also satisfies the HTP (with $u_3 = u_4 = \epsilon$).

▶ **Example 12.** The VPT of Fig. 1 with plain arrows does not satisfy the MTP, as the delay between the two branches increases when iterating the loops. Consider now the VPT obtained by replacing r by r' in the transition (q_1, r, c, γ, q_2). It is obviously twinned, as we cannot construct two runs on the same input which have the form given in the premises of the MTP. However this transducer is not subsequentializable, as the output on the call symbols cannot be delayed to the matching return symbols.

As for the HTP, we can decide the MTP using a reduction to the emptiness of a pushdown automaton with bounded reversal counters. A complete proof can be found in [10].

▶ **Lemma 13.** *The matched twinning property is decidable in* CONPTIME *for* fVPTs.

The most challenging result of this paper is to show that the MTP only depends on the transduction and not on the transducer that defines it. The proof relies on fundamental properties of word combinatorics that allow us to give a general form of the output words $v_1, v_2, v_3, v_4, w_1, w_2, w_3, w_4$ involved in the MTP, that relates them by means of conjugacy of their primitive roots. The proof gives a deep insight into the expressive power of VPTs which is also interesting on its own. As many results of word combinatorics, the proof is a long case study, so that we give it in [10] only.

▶ **Theorem 14.** *Let* T_1, T_2 *be* fVPTs *such that* $[\![T_1]\!] = [\![T_2]\!]$. T_1 *is twinned iff* T_2 *is twinned.*

Sketch. We assume that T_1 is not twinned and show that T_2 is not twinned either. By definition of the MTP there are two runs of the form

$$\begin{cases} (i_1, \perp) & \xrightarrow{u_1/v_1} & (p_1, \sigma_1) & \xrightarrow{u_2/v_2} & (p_1, \sigma_1\beta_1) & \xrightarrow{u_3/v_3} & (q_1, \sigma_1\beta_1) & \xrightarrow{u_4/v_4} & (q_1, \sigma_1) \\ (i_1', \perp) & \xrightarrow{u_1/v_1'} & (p_1', \sigma_1') & \xrightarrow{u_2/v_2'} & (p_1', \sigma_1'\beta_1') & \xrightarrow{u_3/v_3'} & (q_1', \sigma_1'\beta_1') & \xrightarrow{u_4/v_4'} & (q_1', \sigma_1') \end{cases}$$

such that (q_1, σ_1) and (q_1', σ_1') are co-accessible and $\Delta(v_1 v_3, v_1' v_3') \neq \Delta(v_1 v_2 v_3 v_4, v_1' v_2' v_3' v_4')$. We will prove that by pumping the loops on u_2 and u_4 sufficiently many times we will get a similar situation in T_2, proving that T_2 is not twinned. It is easy to show that there exist $k_2 > 0$, $k_1, k_3 \geq 0$, $w_i, w_i' \in \Sigma^*$, $i \in \{1, \ldots, 4\}$, some states $i_2, p_2, q_2, i_2', p_2', q_2'$ of T_2 and some stack contents $\sigma_2, \beta_2, \sigma_2', \gamma_2'$ of T_2 such that we have the following runs in T_2:

$$
\begin{cases}
(i_2, \bot) \xrightarrow{u_1 u_2^{k_1}/w_1} (p_2, \sigma_2) \xrightarrow{u_2^{k_2}/w_2} (p_2, \sigma_2\beta_2) \xrightarrow{u_2^{k_3} u_3 u_4^{k_3}/w_3} (q_2, \sigma_2\beta_2) \xrightarrow{u_4^{k_2}/w_4} (q_2, \sigma_2) \\
(i_2', \bot) \xrightarrow{u_1 u_2^{k_1}/w_1'} (p_2', \sigma_2') \xrightarrow{u_2^{k_2}/w_2'} (p_2', \sigma_2'\beta_2') \xrightarrow{u_2^{k_3} u_3 u_4^{k_3}/w_3'} (q_2', \sigma_2'\beta_2') \xrightarrow{u_4^{k_2}/w_4'} (q_2', \sigma_2')
\end{cases}
$$

such that (q_1, σ_1) and (q_2, σ_2) are co-accessible with the same input word u_5, and (q_1', σ_1') and (q_2', σ_2') are co-accessible with the same input word u_5'. Now for all $i \geq 0$, we let

$$
\begin{aligned}
V^{(i)} &= v_1 (v_2)^{k_1 + i k_2 + k_3} v_3 (v_4)^{k_1 + i k_2 + k_3} & W^{(i)} &= w_1 (w_2)^i w_3 (w_4)^i \\
V'^{(i)} &= v_1' (v_2')^{k_1 + i k_2 + k_3} v_3' (v_4')^{k_1 + i k_2 + k_3} & W'^{(i)} &= w_1' (w_2')^i w_3' (w_4')^i \\
D_1(i) &= \Delta(V^{(i)}, V'^{(i)}) & D_2(i) &= \Delta(W^{(i)}, W'^{(i)})
\end{aligned}
$$

In other words, $D_1(i)$ (resp. $D_2(i)$) is the delay in T_1 (resp. T_2) accumulated on the input word $u_1 (u_2)^{k_1 + i k_2 + k_3} u_3 (u_4)^{k_1 + i k_2 + k_3}$ by the two runs of T_1 (resp. T_2). There is a relation between the words $V^{(i)}$ and $W^{(i)}$. Indeed, since T_1 and T_2 are equivalent and (q_1, σ_1) and (q_2, σ_2) are both co-accessible by the same input word, for all $i \geq 1$, either $V^{(i)}$ is a prefix of $W^{(i)}$ or $W^{(i)}$ is a prefix of $V^{(i)}$. We have a similar relation between $V'^{(i)}$ and $W'^{(i)}$.

We prove in [10] the following intermediate results: (i) there exists $i_0 \geq 0$ such that for all $i, j \geq i_0$ such that $i \neq j$, $D_1(i) \neq D_1(j)$; (ii) for all $i, j \geq 1$, if $D_1(i) \neq D_1(j)$, then $D_2(i) \neq D_2(j)$. The proofs of those results rely on fundamental properties of word combinatorics and a non-trivial case study that depends on how the words $v_1 (v_2)^{k_1 + i k_2 + k_3} v_3 (v_4)^{k_1 + i k_2 + k_3}$ and $w_1 (w_2)^i w_3 (w_4)^i$ are overlapping. Thanks to (i) and (ii), we clearly get that $D_2(i_0) \neq D_2(i_0 + 1)$, which provides a counter-example for the matched twinning property.   ◄

Subsequential transducers have at most one run per input word, so we get the following:

▶ **Corollary 15.** *Subsequentializable* VPTs *are twinned.*

The MTP is not a sufficient condition to be subsequentializable, as shown for instance by Example 12. Therefore the class of transductions defined by transducers which satisfy the MTP is strictly larger than the class of transductions defined by subsequentializable transducers. However, these transductions are in the same complexity class for evaluation, i.e. polynomial space in the height of the input word for a fixed transducer:

▶ **Theorem 16.** *Let T be an* fVPT. *If T is twinned, then the Turing transducer* LCPINTT(T) *runs, on an input stream u, in space complexity quadratic in the height of u.*

We can state more precisely the space complexity of LCPINTT(T) when T is reduced. In this case, it is in $O\left(|Q|^4 \cdot |\Gamma|^{2|Q|^4 + 2} \cdot (h(u) + 1)^2 \cdot M\right)$, where $M = \max\{|v| : (q, a, v, \gamma, q') \in \delta\}$.

Sketch. Like for the HTP, when T satisfies the MTP, also does the reduced VPT returned by the reduction procedure of [8]. We use a pumping technique to show that for any word $u \in \Sigma^*$ on which there is a run of T, we have $\mathsf{out}_{\neq}^{\max}(u) \leq (h(u) + 1)q(T)$ for some function q, whenever the MTP holds for T. This is done as follows: any such word can be uniquely decomposed as $u = u_0 c_1 u_1 c_2 \ldots c_n u_n$ with $n \leq h(u)$, each u_i is well-nested and each c_i is a call. Then if the u_i are long enough, we can pump them vertically and horizontally without affecting the global delay, by using the property $\Delta(vv', ww') = \Delta(\Delta(v, w).(v', w'))$. Then we can apply Proposition 3.   ◄

6   Conclusion and Remarks

This work investigates the streaming evaluation of nested word transductions, and in particular identifies an interesting class of VPT-transductions which subsumes subsequentializable transductions and can still be efficiently evaluated. The following inclusions summarize the relations between the different *classes* of transductions we have studied:

$$\text{BM fVPTs} \subsetneq \text{Subsequentializable VPTs} \subsetneq \text{twinned fVPTs} \subsetneq \text{HBM fVPTs} \subsetneq \text{fVPTs}$$

Moreover, we have shown that BM, twinned and HBM fVPTs are decidable in CoNPTime.

Further Directions   An important asset of the class of twinned fVPTs w.r.t. the class of subsequentializable VPTs is that it is decidable. It would thus be interesting to determine whether or not the class of subsequentializable VPTs is decidable. In addition, we also plan to extend our techniques to more expressive transducers, such as those recently introduced in [1], which extend VPTs with global variables and are as expressive as MSO-transductions, and can therefore swap or reverse sub-trees. Another line of work concerns the extension of our evaluation procedure, which holds for functional transductions, to finite valued transductions.

—— **References** ——

1   R. Alur and L. D'Antoni. Streaming tree transducers. *CoRR*, abs/1104.2599, 2011.
2   R. Alur and P. Madhusudan. Adding nesting structure to words. *JACM*, 56(3):16:1–16:43, 2009.
3   Z. Bar-Yossef, M. Fontoura, and V. Josifovski. Buffering in query evaluation over XML streams. In *PODS*, pages 216–227. ACM-Press, 2005.
4   V. Bárány, C. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In *STACS*, pages 420–431, 2006.
5   D. Barbosa, L. Mignet, and P. Veltri. Studying the XML web: Gathering statistics from an xml sample. *World Wide Web*, 8:413–438, 2005.
6   A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM TOSEM*, 20, 2011.
7   M. Benedikt and A. Jeffrey. Efficient and expressive tree filters. In *FSTTCS*, volume 4855 of *LNCS*, pages 461–472. Springer Verlag, 2007.
8   M. Caralp, P.-A. Reynier, and J.-M. Talbot. A polynomial procedure for trimming visibly pushdown automata. Technical Report hal-00606778, HAL, CNRS, France, 2011.
9   C. Choffrut. Une Caractérisation des Fonctions Séquentielles et des Fonctions Sous-Séquentielles en tant que Relations Rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977.
10   E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. Streamability of Nested Word Transductions. Technical Report inria-00566409, HAL, CNRS, France, 2011.
11   E. Filiot, J.-F. Raskin, P.-A. Reynier, F. Servais, and J.-M. Talbot. Properties of visibly pushdown transducers. In *MFCS*, volume 6281 of *LNCS*, pages 355–367. Springer, 2010.
12   O. Gauwin, J. Niehren, and S. Tison. Earliest query answering for deterministic nested word automata. In *FCT*, volume 5699 of *LNCS*, pages 121–132. Springer, 2009.
13   M. Grohe, C. Koch, and N. Schweikardt. Tight lower bounds for query processing on streaming and external memory data. *Theor. Comput. Sci.*, 380:199–217, July 2007.
14   T. Harju, O. H. Ibarra, J. Karhumaki, and A. Salomaa. Some decision problems concerning semilinearity and commutation. *JCSS*, 65, 2002.

15   C. Konrad and F. Magniez. Validating XML documents in the streaming model with external memory. Technical Report 1012.3311, arXiv, 2010.

16   V. Kumar, P. Madhusudan, and M. Viswanathan. Visibly pushdown automata for streaming XML. In *WWW*, pages 1053–1062. ACM-Press, 2007.

17   O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

18   P. Madhusudan and M. Viswanathan. Query automata for nested words. In *MFCS*, volume 5734 of *LNCS*, pages 561–573. Springer Berlin / Heidelberg, 2009.

19   L. Segoufin and C. Sirangelo. Constant-memory validation of streaming XML documents against DTDs. In *ICDT*, pages 299–313, 2007.

20   A. Weber and R. Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2):327–340, 1995.