Two-Way Visibly Pushdown Automata and Transducers *

Luc Dartois Emmanuel Filiot
Université Libre de Bruxelles, Belgium
{ldartois,efiliot}@ulb.ac.be

Pierre-Alain Reynier Jean-Marc Talbot
LIF, Aix-Marseille University, CNRS, France
firstname.lastname@lif.univ-mrs.fr

Abstract

Automata-logic connections are pillars of the theory of regular languages. Such connections are harder to obtain for transducers, but important results have been obtained recently for word-to-word transformations, showing that the three following models are equivalent: deterministic two-way transducers, monadic second-order (MSO) transducers, and deterministic one-way automata equipped with a finite number of registers. Nested words are words with a nesting structure, allowing to model unranked trees as their depth-first-search linearisations. In this paper, we consider transformations from nested words to words, allowing in particular to produce unranked trees if output words have a nesting structure. The model of visibly pushdown transducers allows to describe such transformations, and we propose a simple deterministic extension of this model with two-way moves that has the following properties: *i)* it is a *simple computational model*, that naturally has a good evaluation complexity; *ii)* it is *expressive*: it subsumes nested word-to-word MSO transducers, and the exact expressiveness of MSO transducers is recovered using a simple syntactic restriction; *iii)* it has *good algorithmic/closure properties*: the model is closed under composition with a unambiguous one-way letter-to-letter transducer which gives closure under regular look-around, and has a decidable equivalence problem.

Keywords Transductions, Pushdown automata, Logic.

1. Introduction

Pillars of word language theory The theory of languages is one of the deepest and richest theory in computer science, with successful applications such as, computer-aided verification and synthesis. A major reason for this success is the strong connections between models of languages, with quite different flavours, that are based on two important pillars: *computation and logic*. Perhaps one of the most famous example is the effective correspondence for regular languages of finite words between a low-level computational model, finite state automata, and a high-level declarative formalism, monadic second-order logic (MSO). Similar connections have been obtained for other structures (e.g. infinite words, finite and infinite trees, nested words) (Thomas 1997; Comon-Lundh et al. 2007). In some cases, it has been even possible to build a third pillar based on algebra. The class of regular languages for instance is known to be the class of languages with finite syntactic congruence.

The logic/two-way/one-way trinity of word transductions To model functions from (input) words to (output) words, i.e. *word transductions*, and more generally word binary relations, automata have been extended to *transducers*, i.e. automata with outputs. Whenever a transducer reads an input symbol, it can produce on the output a finite word, the final output word being the right concatenation of all the finite words produced along the way. To capture functions mirroring or copying twice the input word, transducers need to read the input word in both directions: this yields the class of *two-way finite state transducers* (2FST). Two-way transducers have appealing properties: they are closed under composition (Chytil and Jákl 1977) and if they are deterministic, their equivalence problem is decidable (in PSpace) (Gurari 1982; Culik and Karhumaki 1987) and the transduction can be evaluated in constant space (for a fixed transducer), the output being produced on-the-fly.

Impressively, in the late 90s, deterministic two-way transducers have been shown in (Engelfriet and Hoogeboom 2001) to correspond to monadic second-order transducers (MSOT), a powerful logical formalism introduced in (Courcelle 1994) in a more general context, with independent motivations. It was the first logic-transducer connection obtained for a class of word transductions expressive enough to capture interesting and desirable transductions. This correspondence has been extended to finite tree transductions (Engelfriet and Maneth 1999, 2003; Bloem and Engelfriet 2000).

Recently, an MSOT-expressive one-way model, *streaming string transducers* (SST), has been introduced in (Alur and Černý 2010, 2011): it uses registers that can store output words and can be combined and updated along the run in a linear (copyless) manner (a register content cannot be used twice in an update). The main advantage of this model is its one-wayness, but the price to pay is the space complexity of evaluation: it depends also on the size of the register contents.

The models MSOT, deterministic 2FST and deterministic SST have the same expressive power, and we refer to this correspondence as *the logic/two-way/one-way trinity*. This trinity has been extended to transductions of infinite words (Alur et al. 2012) and ranked trees (Courcelle and Engelfriet 2012; Alur and D'Antoni 2012). For trees, bi-directionality is replaced by a tree walking ability: the transducer can move along the edges of the tree in any direction. However, to capture MSOT, the transducer needs to have *regular look-around*, i.e. needs to be able to test regular properties of the context of the tree node in which it is currently positioned (Courcelle and Engelfriet 2012). Look-arounds can be removed at

* Emmanuel Filiot is research associate at FNRS. This work is supported by the ARC project *Transform* (French speaking community of Belgium), the Belgian FNRS PDR project *Flare*, and the French ANR project ExtStream. This work has been carried out thanks to the support of the ARCHIMEDE Labex (ANR-11-LABX-0033) and the A*MIDEX project (ANR-11-IDEX-0001-02) funded by the Investissements dAvenir French Government program, managed by the French National Research Agency (ANR).

the price of adding a pushdown store (Courcelle and Engelfriet 2012). For one-way machines, uni-directionality is modeled by fixing the traversal of the tree to be a depth-first left-to-right traversal and, as for words, to capture MSOT, the transducer needs to have registers (Alur and D'Antoni 2012). Tree-walking transducers with look-around, and tree transducers with registers are strictly more expressive than MSOT, but restrictions have been defined that capture exactly MSOT. Finally, let us mention the macro tree transducers, the first computational model shown to capture, with suitable restrictions, MSOT ranked tree transductions (Engelfriet and Maneth 1999, 2003; Bloem and Engelfriet 2000). This model has parallel computations, like a top-down tree automaton, and registers.

Nested words In this paper, we consider transductions of nested words to words. *Nested words* are words with a nesting structure, built over symbols of two kinds: call and return symbols[1]. In particular, nested words can model ordered unranked trees, viewed as their depth-first, left-to-right, linearisation, and in turn are a natural model of tree-structured documents, such as XML documents. *Visibly pushdown automata* (VPA) have been introduced in (Alur and Madhusudan 2009) as a model of regularity for languages of nested words. They are pushdown automata with a constrained stack policy: whenever a call symbol is read, exactly one symbol is pushed onto the stack, and when reading a return symbol, exactly one symbol is popped from the stack. Therefore, at any point, the height of the stack corresponds to the nesting level (call depth) of the word. Roughly, VPA are tree automata over linearised trees, and as such they inherit all the good closure and algorithmic properties of tree automata. However, viewing trees as nested words has raised motivating questions in the context of tree streams, such as streaming XML validation (Picalausa et al. 2011; Segoufin and Sirangelo 2007), streaming XML queries (Kumar et al. 2007; Gauwin et al. 2011), as well as streaming XML transformations (Filiot et al. 2011) (see also (Alur 2016) for other applications of VPA).

By using a matching predicate $M(x, y)$ that holds true if x is a call symbol, y is a return symbol and is the matching return of x, MSO logic can be extended from words to nested words, and it is known to correspond to regular nested word languages (Alur and Madhusudan 2009).

Nested word to word transductions Besides the motivations given before for considering nested words instead of unranked trees, we argue that seeing unranked trees as nested word yields a natural and simple two-way model for transductions of nested words, presented later. On the output, we do not require the words to have a particular structure. It is not a weakness: nested words are words, and the model we introduce in this paper can as well produce output words that are nested.

VPA have been extended with output, yielding the class of *visibly pushdown transducers* (VPT, (Filiot et al. 2010)). When reading an input symbol, VPT can generate a word on the output. VPT have good algorithmic and closure properties, and are well-suited to a streaming context (Filiot et al. 2011). However, VPT suffer from a low expressive power, as they are only one-way, without registers.

Based on MSO for nested words, one can define MSO transducers *à la Courcelle* to define nested word to word transductions. From now on, we refer to such MSO transducers as MSOT. A one-way model has already been defined in (Alur and D'Antoni 2012) that captures exactly MSOT. They extend VPA with registers that can store partial output words. Whenever a call symbol is read, the contents of the registers are pushed onto the stack and all the registers reset. On reading return symbols, they can combine the content of the current registers with the content of the registers stored on the stack, in a copyless fashion. The space complexity of evaluation for such transducers is linear in the length of the input nested word, and they have decidable equivalence problem.

Objective and two-way visibly pushdown transducers Our main goal in this paper is to establish a logic/two-way/one-way trinity for nested word to word transductions. Since the logic/one-way connection has already been shown in (Alur and D'Antoni 2012), we want in particular to define a *two-way computational model* with the following requirements: it must be *conceptually simple*, at least *as expressive as MSOT* and have *decidable equivalence problem*.

To this aim, we introduce *deterministic two-way visibly pushdown transducers* (D2VPT) and show it meets the later requirements. D2VPT read their input in both directions, and their stack behaviour not only depends on the type of symbols they read, but also on the reading mode they are in, either backward or forward. In a forward mode, they behave just like VPT. On the backward mode, they behave like VPT where the call and return types are swapped: when reading a return symbol backward, they push a symbol onto the stack, and when reading a call symbol backward, they pop a symbol from the stack. They can change their mode at any moment, and produce words on the output.

Let us give now an illustrating example of a transduction f_s of nested words, which will be formalised in Example 1. Assume a set of call symbols $\{1, \dots, n\}$ ordered by the total order on natural numbers, and one return symbol $\{r\}$. The transduction f_s sorts an input nested word in ascending order, recursively nesting level by nesting level, according to the order on calls. We assume inputs start and end with special symbols $\triangleright$ and $\triangleleft$ (call and return resp.). E.g., f_s maps $\triangleright 22r1rr1r3r\triangleleft$ to $\triangleright 1r21r2rr3r\triangleleft$ and $\triangleright 23r1r2rr2r3r1r\triangleleft$ to $\triangleright 1r21r2r3rr2r3r\triangleleft$ (see Figure 1). To make f_s a function in case the same call symbol occurs twice at the same level, f_s preserves their order of appearance. The tree representation of this mapping is given in Figure 1 (omitting return symbols). The transduction f_s is easily implemented with a D2VPT T_s. To process a sequence of siblings at level k, T_s works as follows: for i from 0 to n, T_s performs a forward pass on the siblings (note that a sibling is actually a tree whose linearisation is of the form jwr where w is again a sequence of linearised trees). During this forward pass, T_s transforms a sibling jwr into ϵ if $j \neq i$, and into $iw'r$ otherwise, where w' is the result of sorting recursively w. To implement the loop, when T_s has finished the i-th forward pass, i.e. when it reads a return symbol at level $j-1$, it comes back to its matching call and starts from there the $(i+1)$-th forward pass, if $i < n$.

Contributions By linearising input trees, the simple and well-known concept of bi-directionality can be generalised naturally from words to trees. While D2VPT, as we show in this paper, allow one to lift known results from word transductions to nested word to word transductions, we think that D2VPT are an appealing model for the following reasons:

memory efficiency Regarding the complexity of evaluation, for a fixed D2VPT, computing the output word of an input nested word w can be done in space $O(d(w))$, where $d(w)$ is the depth of w. Indeed, only the stack and current state need to be kept in memory when processing an input nested word. It is an appealing property when transforming large but not deep tree-structured documents, such as XML documents in general.

expressiveness At the same time, we show that this efficiency does not entail expressive power: D2VPT can express all MSOT transductions. They are strictly more expressive than MSOT

[1] Sometimes, internal symbols are also considered but in this paper, to ease the presentation, we omit them. This is wlog as an internal symbol a can be harmlessly replaced by a call symbol c_a followed by a return symbol r_a.

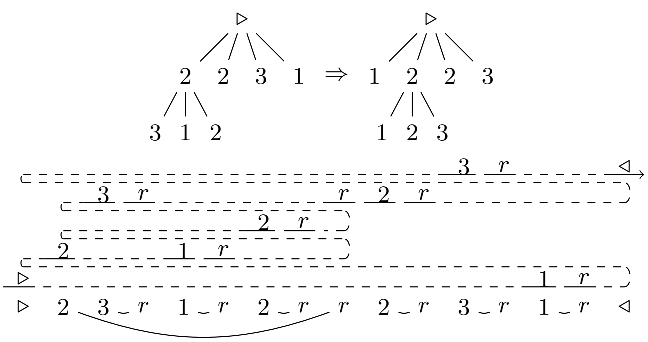

Figure 1. On top, the transformation of the input. Between siblings with the same labeling, the original order is preserved. Below, the run of the transducer. Dashed lines are non producing sequences.

as they can for instance express transduction of *exponential size increase*, while MSOT are only of linear size increase. By putting a simple decidable restriction on D2VPT, called *single-useness*, D2VPT capture exactly MSOT transductions.

algorithmic properties Despite their high expressive power, D2VPT still have *decidable equivalence problem*. We also prove that preprocessing the input of a D2VPT by a letter-to-letter unambiguous VPT does not increase its expressive power, as their composition is again a D2VPT.

The proof of expressiveness relies on an existing correspondence between tree-walking and MSO transducers of ranked trees to words (Courcelle and Engelfriet 2012), and on the classical *first child-next sibling* (fcns for short) encoding of unranked trees into binary trees. As in (Courcelle and Engelfriet 2012), we use an intermediate automata model equipped with MSO look-around, and then show that these look-around tests can be removed. For the latter property, our proof differs from that of (Courcelle and Engelfriet 2012) in which a pushdown stack is used to update information on MSO-types. On binary trees, their model pushes the stack while moving to the first-child, but also while moving to the second child. This latter push corresponds, through the fcns encoding, to pushing a symbol while moving to the next sibling, an operation that is not allowed with a visibly pushdown stack. Hence, in order to prove that look-around tests can be removed in our model, we need a more involved construction, that extends a non-trivial result proven in (Hopcroft and Ullman 1967) for two-way automata on words. Decidability of D2VPT equivalence is done by reduction to deterministic top-down tree to word transducer equivalence, a problem which was opened for long and recently solved in (Seidl et al. 2015).

Application 1: Unranked tree to word walking transducers D2VPT can easily be translated into a pushdown walking model of unranked tree to word transductions. It works exactly as in the ranked tree case of (Courcelle and Engelfriet 2012): one stack symbol is pushed while going downward and popped while going upward. While moving along sibling relations, the stack is untouched. As a consequence of our results, this model, with single-use restriction, captures exactly MSOT. This model is discussed in the last section.

Application 2: Query 2VPA Deterministic two-way VPA have been introduced in (Madhusudan and Viswanathan 2009) as an equi-expressive model for MSO-definable unary queries on nested words. Using (Neven and Schwentick 2002; Niehren et al. 2005), such queries can be shown to be equivalent to unambiguous VPA with special states which select the nested word positions that are answers to the query. As shown in (Madhusudan and Viswanathan 2009), unambiguity can be traded for determinism, at the price of adding two-wayness. This result comes as a consequence of ours: a one-way unambiguous selecting VPA can be seen as a deterministic VPT with look-around, that annotates the input positions selected by the VPA (look-around resolves nondeterminism), which can be transformed into a D2VPT using our results. The main ingredient of the proof of (Madhusudan and Viswanathan 2009) is also a Hopcroft-Ullman construction, but in a setting simpler than ours [2].

Organisation of the paper In Section 2, we introduce two-way VPA and two-way VPA with look-around, define the notion of transition algebra for 2VPA and use this to show that they are equivalent to one-way VPA. As a consequence, they have decidable (exptime-c) emptiness problem. In Section 3, we introduce D2VPT and D2VPT with look-around, show that they are equivalent, and study their algorithmic properties. Section 4 is devoted to the expressiveness of D2VPT, with a comparison to MSOT and to other known models of nested word to word transductions. Due to lack of space, some results are proved in Appendix. Finally, all our expressiveness equivalences are effective.

2. Two-way visibly pushdown automata

2.1 Definitions

We introduce in this section two-way visibly pushdown automata, following the definition of (Madhusudan and Viswanathan 2009).

We consider a structured alphabet Σ defined as the disjoint union of call symbols Σ_c and return symbols Σ_r. The set of words over Σ is Σ^*. As usual, ϵ denotes the empty word. Amongst words, the set of nested words $\mathcal{N}(\Sigma)$ is defined as the least set such that $\epsilon \in \mathcal{N}(\Sigma)$ and if $w_1, w_2 \in \mathcal{N}(\Sigma)$ then both $w_1 w_2$ and $c w_1 r$ (for all $c \in \Sigma_c$ and $r \in \Sigma_r$) belong to $\mathcal{N}(\Sigma)$. In the following, we assume that input words of our models are always nested words. This is not restrictive as all our models can recognize and filter nested words.

For a word $w \in \Sigma^*$, its length is denoted by $|w|$ and we denote by $w(i)$ its ith symbol. Its set of positions is $pos(w) = \{1, \dots, |w|\}$, and for $i, j \in pos(w)$ such that $i < j$, we say that (i, j) is a *matching pair* of w if $w(i) \in \Sigma_c$, $w(j) \in \Sigma_r$ and w can be decomposed into $w = w_1 w(i) w_2 w(j) w_3$, where $w_1, w_3 \in \Sigma^*, w_2 \in \mathcal{N}(\Sigma)$ and $|w_1| = i - 1$, $|w_2| = j - i - 1$. Note that if $w \in \mathcal{N}(\Sigma)$, then necessarily, $w_1 w_3 \in \mathcal{N}(\Sigma)$.

When dealing with two-way machines, we assume the structured alphabet Σ to be extended into $\overline{\Sigma}$ by adding two special symbols $\triangleright, \triangleleft$ in $\overline{\Sigma}_c$ and $\overline{\Sigma}_r$ respectively, and we consider words with left and right markers from $\triangleright\Sigma^*\triangleleft$.

Definition 1. *A two way visibly pushdown automaton (*2VPA *for short) A over $\overline{\Sigma}$ is given by $(Q, q_I, F, \Gamma, \delta)$ where Q is a finite set of states, $q_I \in Q$ is the initial state, $F \subseteq Q$ is a set of final states and Γ is a finite stack alphabet. Given the set $\mathbb{D} = \{\leftarrow, \rightarrow\}$ of directions, the transition relation δ is defined by $\delta^{push} \cup \delta^{pop}$ where*

- $\delta^{push} \subseteq ((Q \times \{\rightarrow\} \times \Sigma_c) \cup (Q \times \{\leftarrow\} \times \Sigma_r)) \times ((Q \times \mathbb{D}) \times \Gamma)$
- $\delta^{pop} \subseteq ((Q \times \{\leftarrow\} \times \Sigma_c \times \Gamma) \cup (Q \times \{\rightarrow\} \times \Sigma_r \times \Gamma)) \times (Q \times \mathbb{D})$

Additionally, we require that for any states q, q' and any stack symbol γ, if $(q, \leftarrow, \triangleright, \gamma, q', d) \in \delta^{pop}$ then $d = \rightarrow$ and if $(q, \rightarrow, \triangleleft, \gamma, q', d) \in \delta^{pop}$ then $d = \leftarrow$.

Informally, a 2VPA has a reading head pointing between symbols (and possibly on the left of $\triangleright$ and on the right of $\triangleleft$). A configuration of the machine is given by a state, a direction d and a

[2] They provide a construction for the composition of a co-deterministic VPA with an unambiguous VPA, while we study that of a D2VPA with an unambiguous VPA.

stack content. The next symbol to be read is on the right of the head if $d = \rightarrow$ and on the left if $d = \leftarrow$. Note that when reading the left marker from right to left $\leftarrow$ (resp. the right marker from left to right $\rightarrow$), the next direction can only be $\rightarrow$ (resp. $\leftarrow$). The structure of the alphabet induces the behaviour of the machine regarding the stack when reading the input word: when reading on the right, a call symbol leads to push onto the stack while a return symbol pops a symbol from the stack. When reading on the left, a dual behaviour holds (hence, at a given position in the input word, the height of the stack is always constant at each visit to that position in the run). Finally, the state and the direction are updated.

Let $w \in \mathcal{N}(\Sigma)$. We set $w(0) = \triangleright$ and $w(|w|+1) = \triangleleft$. For a move d and $0 \leq i \leq |w|$, we denote by

- $\mathrm{move}(d, i)$ the integer $i - 1$ if $d = \leftarrow$ and $i + 1$ if $d = \rightarrow$.
- $\mathrm{read}(w, d, i)$ the symbol $w(i)$ if $d = \leftarrow$ and $w(i+1)$ if $d = \rightarrow$.

Formally, a stack σ is a finite word over Γ. The empty stack/word over Γ is denoted $\perp$. For a word $\triangleright w \triangleleft$ where $w \in \mathcal{N}(\Sigma)$ and a 2VPA $A = (Q, q_I, F, \Gamma, \delta)$, a *configuration* of A is a triple (q, i, d, σ) where $q \in Q$, $0 \leq i \leq |w| + 1$, $d \in \mathbb{D}$ and σ is a stack. A *run* of A on a word w is a finite non-empty sequence of configurations $(q_0, i_0, d_0, \sigma_0)(q_1, i_1, d_1, \sigma_1) \ldots (q_\ell, i_\ell, d_\ell, \sigma_\ell)$ where for all $0 \leq j \leq \ell$, the configuration $(q_{j+1}, i_{j+1}, d_{j+1}, \sigma_{j+1})$ satisfies $i_{j+1} = \mathrm{move}(i_j, d_j)$ and

- if $\mathrm{read}(w, d_j, i_j) \in \Sigma_c$ and $d_j = \rightarrow$ or $\mathrm{read}(w, d_j, i_j) \in \Sigma_r$ and $d_j = \leftarrow$ then $(q_j, d_j, \mathrm{read}(w, d_j, i_j), q_{j+1}, d_{j+1}, \gamma) \in \delta^{\mathrm{push}}$ and $\sigma_{j+1} = \sigma_j \gamma$.
- if $\mathrm{read}(w, d_j, i_j) \in \Sigma_c$ and $d_j = \leftarrow$ or $\mathrm{read}(w, d_j, i_j) \in \Sigma_r$ and $d_j = \rightarrow$ then $(q_j, d_j, \mathrm{read}(w, d_j, i_j), \gamma, q_{j+1}, d_{j+1}) \in \delta^{\mathrm{pop}}$ and $\sigma_{j+1}\gamma = \sigma_j$.

By the special treatment of $\triangleright$ and $\triangleleft$ ensured by the definition of 2VPA, the indices i_j all belong to $\{0, \ldots, |w|+1\}$. Note also that any configuration is actually a run on the empty word ϵ. A run on a nested word w is accepting whenever $q_0 = q_I$, $i_0 = 0$, $d_0 = \rightarrow$, $\sigma_0 = \perp$ and $q_\ell \in F$, $i_\ell = |w| + 1$, $d_\ell = \rightarrow$, $\sigma_\ell = \perp$.

Note that A being a visibly pushdown automaton, for any two configurations in a run of A at the same position i in the word (q, i, d, σ) and (q', i, d', σ'), the stack σ, σ' have the same height/length.

The language $\mathcal{L}(A)$ defined by A is the set of nested words w from Σ^* such that there exists an accepting run of A on $\triangleright w \triangleleft$.

DEFINITION 2. *A two-way visibly pushdown automaton is*

- deterministic *(*D2VPA *for short) if we may write δ^{push}, δ^{pop} as functions from $((Q \times \{\rightarrow\} \times \Sigma_c) \cup (Q \times \{\leftarrow\} \times \Sigma_r))$ to $(Q \times \mathbb{D}) \times \Gamma$ and from $((Q \times \{\leftarrow\} \times \Sigma_c \times \Gamma) \cup (Q \times \{\rightarrow\} \times \Sigma_r \times \Gamma))$ to $Q \times \mathbb{D}$ respectively.*
- codeterministic *if we may write δ^{push}, δ^{pop} as injective applications, with the same type as in the previous item.*
- unambiguous *iff for any word w, there exists at most one accepting run on w.*

Obviously, if A is (co)deterministic, for any word w from $\mathcal{N}(\bar{\Sigma})$, there exists a unique run on w in A from any fixed configuration. Hence, any (co)deterministic 2VPA is unambiguous. Note also that the determinism of A implies that any configuration can occur only once in some accepting run (otherwise, the machine would loop without reaching a final configuration).

A two-way visibly pushdown automaton is a *(one-way) visibly pushdown automaton* (VPA for short) whenever $d' = d = \rightarrow$ for all $(q, d, \alpha, q', d', \gamma')$ in δ^{push} and for all $(q, d, \alpha, \gamma, q', d')$ in δ^{pop}.

For VPA, we may omit directions in the transition relation, configurations and runs.

Finally, we will denote DVPA the class of deterministic VPA. In this case, the transition relation is defined as a function omitting directions.

2.2 Transition algebra for 2VPA

Nested words from $\mathcal{N}(\Sigma)$ (or $\mathcal{N}(\overline{\Sigma})$) induce a natural algebra $\mathbb{W} = (\mathcal{N}(\Sigma), ., \{f_{c,r} \mid c \in \Sigma_c,\ r \in \Sigma_r\}, \epsilon)$ where '.' is a binary operation, the $f_{c,r}$ form a family of unary operations and ϵ is a constant. The semantics of ϵ is the empty word, of . is concatenation and for any w in $\mathcal{N}(\Sigma)$, $f_{c,r}(w) = cwr$. Obviously, the operators finitely generates $\mathcal{N}(\Sigma)$ which can be seen as the free generated algebra over this signature quotiented by the associativity of '.' and the neutrality of ϵ wrt the concatenation '.'.

The traversal congruence $\sim$ Inspired by works on two-way automata on words (Pécuchet 1985; Shepherdson 1959), we study traversals of a 2VPA A. A traversal of some nested word w abstracts a run of A keeping track only of the fact that it starts reading the word from the left or from the right (depending on the initial direction) in some state p and leaves it in some state q. Now, formally, for any states p, q, and any two directions $d_1, d_2 \in \mathbb{D}$, $((p, d_1), (q, d_2))$ belongs to the traversal of w if there exists a run of A on w starting in the configuration $(p, pos(d_1), d_1, \perp)$ and ending in $(q, pos(d_2), d_2, \perp)$, where

$$\begin{cases} pos(d_1) = 0 \text{ if } d_1 = \rightarrow \quad \text{and} \quad pos(d_1) = |w| \text{ otherwise} \\ pos(d_2) = |w| \text{ if } d_2 = \rightarrow \quad \text{and} \quad pos(d_2) = 0 \text{ otherwise} \end{cases}$$

Note that the reading starts either at the beginning or at the end of w depending on the initial current direction and that the final direction indeed leads to leave the word. One may associate with a nested word the set of its traversals and define a relation $\sim$ on nested words such that $u \sim v$ if u and v have the same traversals.

Obviously, $\sim$ is an equivalence relation over $\mathcal{N}(\Sigma)$ and we denote by $[w]_\sim$ the set of traversals of a nested word w. We prove that $\sim$ is actually a congruence, that is if $w_1 \sim w_2$ and $w_1' \sim w_2'$ then $f_{c,r}(w_1) = cw_1r \sim cw_2r = f_{c,r}(w_2)$ and $w_1.w_1' \sim w_2.w_2'$ for any nested words w_1, w_1', w_2, w_2' in $\mathcal{N}(\Sigma)$.

PROPOSITION 1. *The relation $\sim$ is a congruence of finite index.*

The transition algebra $\mathbb{T}_A$ Based on Proposition 1, the congruence relation $\sim$ induces a finite algebra $\mathbb{T}_A = (\mathrm{Trav}_A, .^{\mathbb{T}_A}, \{f_{c,r}^{\mathbb{T}_A} \mid c \in \Sigma_c, r \in \Sigma_r\}, \epsilon^{\mathbb{T}_A})$ where the support is Trav_A the set of all traversals induced by A, $.^{\mathbb{T}_A}$ is a binary operation which is associative, each $f_{c,r}^{\mathbb{T}_A}$ is a unary operation and $\epsilon^{\mathbb{T}_A}$ is a constant from Trav_A and a neutral element for $.^{\mathbb{T}_A}$. More specifically, $\epsilon^{\mathbb{T}_A} = [\epsilon]_\sim$, $[u]_\sim .^{\mathbb{T}_A} [v]_\sim = [uv]_\sim$ and $f_{c,r}^{\mathbb{T}_A}([u]_\sim) = [cur]_\sim$. These operations are well-defined since $\sim$ is a congruence.

Hence, there exists a unique and canonical morphism $\mu_{\mathbb{T}_A}$ from $\mathbb{W}$, the algebra of nested words, onto $\mathbb{T}_A$, that satisfies $\mu_{\mathbb{T}_A}(w) = [w]_\sim$. We also denote $[w]_\sim$ as $w^{\mathbb{T}_A}$ since it can be considered as the interpretation of w (which is an element $\mathbb{W}$) in $\mathbb{T}_A$.

The correction of this morphism $\mu_{\mathbb{T}_A}$ directly implies:

PROPOSITION 2. *Let $A = (Q, q_I, F, \Gamma, \delta)$ be a* 2VPA. *$\mathcal{L}(A) = \mu_{\mathbb{T}_A}^{-1}(\{m \in Trav_A \mid m \cap (\{(q_I, \rightarrow)\} \times F \times \{\rightarrow\}) \neq \varnothing\})$.*

Note that this statement corresponds to the classical notion of recognizability by some finite algebra.

2.3 From two-way visibly pushdown automata to visibly pushdown automata

In this subsection we give a reduction from 2VPA to VPA. While this result can be inferred from (Madhusudan and Viswanathan 2009), our Shepherdson-inspired approach gives an upper bound on the complexity of the procedure. We first recall the notion of recognizability by finite algebra and show that this notion is equivalent to recognazibility by DVPA. Then we prove the main result of this section appealing to the transition algebra $\mathbb{T}_A$.

Let $\mathbb{A} = (D_{\mathbb{A}}, .^{\mathbb{A}}, (f^{\mathbb{A}}_{c,r})_{(c,r)\in\Sigma_c\times\Sigma_r}, \epsilon^{\mathbb{A}})$ be a finite algebra such that $.^{\mathbb{A}}$ is associative having $\epsilon^{\mathbb{A}}$ as neutral element. There exists a unique morphism $\mu_{\mathbb{A}}$ from the algebra of nested words $\mathbb{W}$ onto $\mathbb{A}$.

DEFINITION 3. *A language $\mathcal{L} \subseteq \mathcal{N}(\Sigma)$ is recognized by $\mathbb{A}$ if there exists a set $\mathcal{L}_{\mathbb{A}} \subseteq D_{\mathbb{A}}$ such that $\mathcal{L} = \mu_{\mathcal{A}}^{-1}(\mathcal{L}_{\mathbb{A}})$.*

As an example, as shown in Proposion 2, a language $\mathcal{L}$ defined by a 2VPA is recognized by the transition algebra $\mathbb{T}_A$. We show that recognability by finite algebra implies DVPA recognizability.

LEMMA 1. *If $\mathcal{L}$ is recognized by a finite algebra $\mathbb{A}$ then it is recognizable by a* DVPA *$B_{\mathbb{A}}$. Moreover, the size of $B_{\mathbb{A}}$ is polynomial in the size of $D_{\mathbb{A}}$, the support of $\mathbb{A}$.*

Proof. For $\mathbb{A}$ and the set $\mathcal{L}_{\mathbb{A}} \subseteq D_{\mathbb{A}}$, we define the DVPA $B_{\mathbb{A}} = (D_{\mathbb{A}}, \epsilon^{\mathbb{A}}, \mathcal{L}_{\mathbb{A}}, \Sigma_c \times D_{\mathbb{A}}, \delta_{B_{\mathbb{A}}})$ where $\delta_{B_{\mathbb{A}}} = \delta^{\text{push}}_{B_{\mathbb{A}}} \cup \delta^{\text{pop}}_{B_{\mathbb{A}}}$ and $\delta^{\text{push}}_{B_{\mathcal{A}}}(m^{\mathbb{A}}, c) = (\epsilon^{\mathbb{A}}, (c, m^{\mathbb{A}}))$, $\delta^{\text{pop}}_{B_{\mathcal{A}}}(m'^{\mathbb{A}}, r, (c, m^{\mathbb{A}})) = m^{\mathbb{A}} \circ f^{\mathbb{A}}_{c,r}(m'^{\mathbb{A}})$. Obviously, $B_{\mathbb{A}}$ is deterministic. Its correctness can be proved by induction on nested words showing for all $w \in \mathcal{N}(\Sigma)$, there exists a run in $B_{\mathbb{A}}$ on w from $(m^{\mathbb{A}}, 0, \bot)$ to $(m'^{\mathbb{A}}, |w|, \bot)$ iff $m'^{\mathbb{A}} = m^{\mathbb{A}}.^{\mathbb{A}}\mu_{\mathbb{A}}(w)$. And so, for an accepting run on w from $(\epsilon^{\mathbb{A}}, 0, \bot)$ to $(m'^{\mathbb{A}}, |w|, \bot)$ with $m'^{\mathbb{A}} \in \mathcal{L}_{\mathbb{A}}$, $m'^{\mathbb{A}} = \mu_{\mathbb{A}}(w)$. Hence, $\mathcal{L}(B_{\mathbb{A}}) = \mu_{\mathbb{A}}^{-1}(\mathcal{L}_{\mathbb{A}})$. Finally, note that the number of states of $B_{\mathcal{A}}$ is precisely the cardinality of the support of $\mathcal{A}$. □

We can now come to the main result of this section.

THEOREM 1. *For any* 2VPA *A, one can compute (in exponential time) a* DVPA *B such that $\mathcal{L}(A) = \mathcal{L}(B)$ and the size of B is exponential in the size of A.*

Proof. One can build from the 2VPA A the elements of $\{[w]_\sim \mid w \in \mathcal{N}(\Sigma)\}$ and thus, the transition algebra $\mathbb{T}_A$, in exponential time. Then, by Lemma 1, a VPA $B_{\mathbb{T}_A}$ is built from $\mathbb{T}_A$. The correctness follows from Proposition 2 for $\mathcal{L}_{\mathbb{T}_A} = \{m^{\mathbb{T}_A} \in \text{Trav}_A \mid m^{\mathbb{T}_A} \cap (\{(q_I, \rightarrow)\} \times F \times \{\rightarrow\}) \neq \varnothing\}$. □

COROLLARY 1. *For any* 2VPA *A, deciding the emptiness of A (ie $\mathcal{L}(A) = \varnothing$) is* EXPTIME-C. *The same result holds for* D2VPA.

Proof. We prove the upper-bound for 2VPA and the lower bound for D2VPA. For the upper-bound, it suffices to build from A in exponential time a equivalent DVPA B possibly exponentially larger than A (Theorem 1). Then, emptiness of B can be tested in polynomial time (Alur and Madhusudan 2009).

The proof of the lower bound proceeds by a reduction of the emptiness problem of intersection of k deterministic top-down tree automata, that is known to be EXPTIME-C.

2.4 2VPA with look-around

As we will later on need the notion of look-around for transducers, we introduce it first for automata to ease the presentation. Hence, we extend the model of 2VPA with look-around. The feature will add a guard to each transition of the machine. This guard will require to be satisfied for the transition to be applied.

DEFINITION 4. *A* 2VPA *with look-around (*$2\mathsf{VPA}^{\mathsf{LA}}$ *for short) is given by a triple (A, λ, B) such that A is a* 2VPA *and B a unambiguous* VPA *and λ is a mapping from the transitions of A to the states of B.*

The notion of runs is adapted to take into account look-around as follows: in any run on some nested word w, for any two successive configurations $(q_j, i_j, d_j, \sigma_j)(q_{j+1}, i_{j+1}, d_{j+1}, \sigma_{j+1})$ obtained by a transition t, we require that there exists a unique accepting run on w in B and that this run contains a configuration of the form $(\lambda(t), \text{read}(w, d_j, i_j), \sigma)$.

The definition of accepting runs remains the same and the language defined by such machines is defined accordingly.

The notion of one-wayness extends trivially to 2VPA with look-around. For determinism, we ask the look-around to be disjoint on transitions with the same left hand-side: for any two different transitions of A, $t_1 = (q, d, a, q'_1, d'_1, \gamma_1)$, $t_2 = (q, d, a, q'_2, d'_2, \gamma_2)$ in δ_c (resp. $t_1 = (q, d, a, \gamma, q'_1, d'_1)$, $t_2 = (q, d, a, \gamma, q'_2, d'_2)$ in δ_r), it holds that $\lambda(t_1) \neq \lambda(t_2)$.

Non-surprisingly, 2VPA are closed under look-around:

THEOREM 2. *Given a* $2\mathsf{VPA}^{\mathsf{LA}}$ *(A, λ, B), there exists a* VPA *A' such that $\mathcal{L}((A, \lambda, B)) = \mathcal{L}(A')$.*

3. Two-way visibly pushdown transducers

3.1 Definitions

Let Σ, Δ be two finite alphabets such that Σ is structured. Two-way visibly pushdown transducers (2VPT) from Σ to Δ extend 2VPA over Σ with a one-way-left-to-right output tape. They are defined as a pair $T = (A, \mathsf{O})$ where A is a 2VPA over Σ and O is a morphism from the set of rules of A to words in Δ^*.

A run of a 2VPT $T = (A, \mathsf{O})$ on an input word $w \in \mathcal{N}(\Sigma)$ is a run ρ of A on w. We say the run is accepting if it is in A. A run ρ may be simultaneously a run on a word w and on a word $w' \neq w$, however, when the underlying input word w is given, there is a unique sequence of transitions $t_1 t_2 \dots t_n$ associated with ρ and w. In this case, the output produced by the run ρ on w is defined as the word $v = \mathsf{O}(t_1)\mathsf{O}(t_2)\dots\mathsf{O}(t_n) \in \Delta^*$. This word is denoted by $\mathsf{out}^w(\rho)$. If ρ contains a single configuration, then we let $\mathsf{out}^w(\rho) = \epsilon$. The transduction defined by T is the relation

$$[\![T]\!] = \{(w, \mathsf{out}^w(\rho)) \in \mathcal{N}(\Sigma)\times\Delta^* \mid \rho \text{ is an accepting run of } T \text{ on } w\}.$$

We say that T is *functional* if $[\![T]\!]$ is a function, and that T is *deterministic* (resp. *unambiguous*) if A is deterministic (resp. unambiguous). The class of deterministic two-way visibly pushdown transducers is denoted D2VPT. Observe that if T is deterministic or unambiguous, then it is trivially functional. Last, when T is functional, we may interpret the relation $[\![T]\!]$ as a partial function on $\mathcal{N}(\Sigma)$: given a word $w \in \mathcal{N}(\Sigma)$, denote by $[\![T]\!](w)$ the unique word $v \in \Sigma^*$ such that $(w, v) \in [\![T]\!]$, whenever it exists. To ease readability, we may simply write T to denote $[\![T]\!]$ when it is clear from the context, for example when considering composition of functions.

We consider classes of one-way visibly pushdown transducers, obtained by considering the corresponding classes of one-way visibly pushdown automata. The notions of functional, deterministic and unambiguous transducers are naturally defined for these transducers, and we denote by (D)VPT the class of (deterministic) one-way visibly pushdown transducers. Last, we say that a VPT $T = (A, \mathsf{O})$ from Σ to Δ is *letter-to-letter* if Δ is a structured alphabet and if O maps every call transition of A to an element of Δ_c and every return transition of A to an element of Δ_r.

2VPT (resp. D2VPT) can be extended with look-around, as we did for 2VPA. Formally, a two-way visibly pushdown transducer with look-around ($2\mathsf{VPT}^{\mathsf{LA}}$ for short) is a pair $T = (A', \mathsf{O})$ where $A' = (A, \lambda, B)$ is a $2\mathsf{VPA}^{\mathsf{LA}}$ and O is a morphism from the set of rules of A to words in Δ^*. We say that such a machine is deterministic if the $2\mathsf{VPA}^{\mathsf{LA}}$ A is deterministic, the resulting class being denoted by $\mathsf{D2VPT}^{\mathsf{LA}}$.

EXAMPLE 1. *We now formally express the transduction given in the introduction (see Figure 1). Let $Q = \{q_1, \dots, q_n\} \cup \{q_{i,j} \mid 1 \leq i, j \leq n$ or $i = \triangleright\} \cup \{q_f\}$ be the set of states with initial state q_1 and final state q_f, a set of stack symbols $\Gamma = \{\bot\} \cup \{i \mid i =$*

$1,\ldots,n\}$, *and for all* $i,j,k \in \{\triangleright, 1,\ldots,n\}$, *we have the rules:*

$$\begin{array}{llllll}
q_i, \rightarrow & \xrightarrow{i|i,+i} & q_1, \rightarrow & q_n, \rightarrow & \xrightarrow{r|r,-j} & q_j, \rightarrow \\
q_i, \rightarrow & \xrightarrow{(j,r)} & q_i, \rightarrow \ \textit{if } j \neq i & q_{i,j}, \leftarrow & \xrightarrow{(j,r)} & q_{i,j}, \rightarrow \\
q_i, \rightarrow & \xrightarrow{r|\epsilon,-j} & q_{i,j}, \leftarrow \ \textit{if } i < n & q_{i,j}, \rightarrow & \xrightarrow{k|\epsilon,+j} & q_{i+1}, \rightarrow
\end{array}$$

The markers are treated as letters, except that they push $\perp$ *instead of* $\triangleright$ *and upon popping* $\perp$ *in state* q_n*, the transducer goes to* q_f *and accepts. The transitions labeled by* (j,r) *are macros corresponding to moves along matching relation, which can easily be implemented.*

Evaluation Observe here that if a transformation is given as a D2VPT T, then one can evaluate it using a memory linear in the depth of the input word w (we assume w can be accessed as we want on some media). Indeed, one simply needs to store the current configuration of T, given as a state and a stack content.

3.2 Closure under composition

We prove in this subsection that 2VPT are closed by composition with a letter-to-letter unambiguous VPT, extending a similar result for transducers on words (Hopcroft and Ullman 1967). This will reveal useful to show that D2VPT are closed under look-around. First, we extend to nested words a result that was known for finite transducers:

LEMMA 2. *Any unambiguous* VPT T *can be written as the composition of two* VPT $T_1 \circ T_2$*, where* T_1 *is deterministic and* T_2 *is letter-to-letter and co-deterministic. Furthermore, if* T *is letter-to-letter, so is* T_1.

THEOREM 3. *Given a letter-to-letter* DVPT A *and a* 2VPT B*, we can construct a* 2VPT C *that realizes the composition* $C = B \circ A$*. If furthermore* B *is deterministic, then so is* C.

Proof. We first notice that since we are considering visibly pushdown machines and the first machine is letter-to-letter, the stacks of both machines are always synchronized, meaning that they have the same height on each position. Then, let us remark that when the 2VPT moves to the right, we can do the simulation in a straight forward fashion by simulating it on the production of the one-way. It becomes more involved when it moves to the left. We then need to rewind the run of the one-way, and nondeterminism can arise. To bypass this, let us recall that a similar construction from (Hopcroft and Ullman 1967) exists for classical transducers, and that the rewinding is done through a back and forth reading of the input, backtracking the run up to a position where the nondeterminism is cleared, and then moving back to the current position. The method is to compute the set of possible candidates for the previous state, and keep moving to the left until we reach a position i where there is only one path left leading to the starting position j. Afterward, we simply follow this path along another one from position $i+1$. As we know that they will merge at position j, we can stop at position $j-1$ with the correct state. If we reach the beginning of the word with multiple candidates, we do the same procedure, the correct path being the one starting from the initial state.

This cannot be done as such on pushdown transducers since rewinding the run might lead to popping the stack, and losing information. However, if at each push position, we push not only the stack symbols but also the current state, we are able, when rewinding the run, to clear the nondeterminism as soon as we pop this information by using it as a *local initial state*, limiting the back and forth reading to the current subhedge. The overall construction can be seen as a classical Hopcroft-Ullman construction on hedges, abstracted as words over the left-to-right traversals of their subhedges, which are called summaries in (Alur and Madhusudan 2009) (see

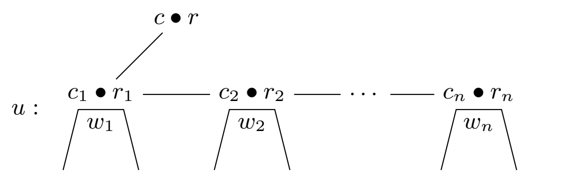

Figure 2. The nested word $cc_1w_1r_1c_2w_2r_2\ldots c_nw_nr_nr$ is abstracted as a word u over letters (c_i, S_i, r_i) where S_i is the summary of w_i. The position labelled by c serves as initial position of the word and the corresponding state was pushed to the stack upon reading it.

Figure 2). These summaries can be computed on-demand by a one-way automaton.

Finally, note that to apply this construction, we need to push this local initial state each time we enter a subhedge, whether we enter from the right or from the left. This can be maintained since when entering from the left, it simply corresponds to the current state and when entering from the right, this state is computed by the Hopcroft-Ullman construction. Note also that the Hopcroft-Ullman routine is deterministic, and consequently the construction preserves determinism. ☐

THEOREM 4. *Let* A *be a* D2VPT *and* $relab$ *be an unambiguous letter-to-letter* VPT*. Then the composition* $A \circ relab$ *can be defined by a* D2VPT.

Proof. The proof is straightforward using previous results. First, Lemma 2 states that $relab$ can be decomposed in $T_1 \circ T_2$, where T_1 is a deterministic VPT and T_2 is a co-deterministic one, and both are letter-to-letter, i.e $A \circ relab = A \circ T_1 \circ T_2$. Now Theorem 3 states that we can construct a D2VPT A' that realizes the composition $A \circ T_1$. Finally, as a co-deterministic VPT can be seen as a deterministic one going right-to-left, a symmetric construction of Theorem 3 on $A' \circ T_2$ gives a D2VPT that realizes $A \circ relab$. ☐

A look-around can be viewed as an MSO formula with one free variable, and it is satisfied iff the formula is satisfied at this position. In (Madhusudan and Viswanathan 2009), the authors consider MSO queries on nested words. An MSO query is an MSO formula with one free variable that annotates the positions of the input word that satisfies it. They proved, using a Hopcroft-Ullman argument, that MSO queries were also implemented by D2VPA. Theorem 4 proves that looks-around can be done on the fly while following the run of an other D2VPA. Since a look-around can be encoded as an unambiguous letter-to-letter VPT, we get the following corollary, that subsumes the result by (Madhusudan and Viswanathan 2009).

COROLLARY 2. $\mathsf{D2VPT} = \mathsf{D2VPT}^{\mathsf{LA}}$.

3.3 Decision problems

We consider the following type-checking problem: given a VPA A_1 on Σ, a finite-state automaton A_2 on Δ, and a D2VPT T from $\mathcal{N}(\Sigma)$ to Δ^*, decide whether for every word $w \in \mathcal{L}(A_1)$, $[\![T]\!](w)$ belongs to $\mathcal{L}(A_2)$. This property is denoted by $T(A_1) \subseteq A_2$[3]. The equivalence problem asks whether given two D2VPT as input, they define the same transduction. We prove the following result:

THEOREM 5. *1. The inverse image of a regular language of words by a* D2VPT *is recognizable by a* VPA.

2. The type-checking problem for D2VPT *is* EXPTIME*-complete.*

3. The equivalence problem for D2VPT *is decidable.*

[3] If A_2 is a VPA, the problem is known to be undecidable even for T a DVPT (Raskin and Servais 2008).

Proof. We prove the three results independently.

(1) Given a D2VPT $T = (A, \mathsf{O})$ and an automaton on words B, we can define a 2VPA A' as a product construction of A and B which simulates B on the production by O. States of A' are simply pairs of states of A and B, and A' recognizes $\{w \in \mathcal{N}(\Sigma) \mid [\![T]\!](w) \in \mathcal{L}(B)\} = [\![T]\!]^{-1}(\mathcal{L}(B))$. Observe that the construction is linear in the sizes of A and B, and that as B may be non-deterministic, A' may also be non-deterministic.

(2) EXPTIME membership: as in the proof of the previous item, we can build a 2VPA A whose size is linear in the sizes of T and A_2, and such that $\mathcal{L}(A) = [\![T]\!]^{-1}(\mathcal{L}(A_2))$. Thus, $T(A_1) \subseteq A_2$ holds iff $\mathcal{L}(A_1) \subseteq \mathcal{L}(A)$ holds. This can be checked in EXPTIME thanks to Theorem 1.

EXPTIME hardness: we reduce the problem of emptiness of a D2VPA A. From A, we build a D2VPT $T = (A, \mathsf{O})$ such that O maps every transition of A to the empty word ϵ. Then, we let A_1 be a VPA such that $\mathcal{L}(A_1) = \mathcal{N}(\Sigma)$ and A_2 such that $\mathcal{L}(A_2) = \emptyset$. Then $T(A_1) \subseteq A_2$ holds iff $\mathcal{L}(A) = \emptyset$.

(3) As proved in Section 4, D2VPT are included in the class of deterministic hedge-to-string transducers with look-ahead, *i.e.* deterministic top-down tree-to-string transducers with look-ahead, run on the first-child-next-sibling encoding of the input hedge. The equivalence problem for these machines has recently been proven decidable in (Seidl et al. 2015). □

4. Expressiveness of Two-Way Visibly Pushdown Transducers

In this section, we study the expressiveness of D2VPT by comparing them with Courcelle's MSO-transductions casted to nested words, the one-way model of (Alur and D'Antoni 2012), and a top-down model for hedges, inspired by top-down tree-to-string transducers.

4.1 MSO-definable Transductions

We first define MSO for nested words and words, as done in (Alur and Madhusudan 2009), and then MSO-transductions from nested words to words, based on Courcelle's MSO-definable graph transductions (Courcelle 1994).

MSO on nested words and words Let Σ be a structured alphabet. A nested word $w \in \mathcal{N}(\Sigma)$ is viewed as a structure with *pos*(w) as domain, over the successor predicate $S(x, y)$ interpreted as pairs $(i, i+1)$ for $i \in pos(w)\backslash\{|w|\}$, the label predicates $\sigma(x)$ for $\sigma \in \Sigma$, interpreted by the positions labeled by σ, and the matching predicate $M(x, y)$ interpreted as the set of matching pairs in w.

Monadic second-order logic (MSO) extends first-order logic with quantification overs sets. First-order variables $x, y, \dots$ are interpreted by positions of words, while second-order variables $X, Y, \dots$ are interpreted by sets of positions. *MSO formulas for nested words over* Σ are defined by the following grammar:

$$\varphi ::= \sigma(x) \mid x \in X \mid S(x, y) \mid M(x, y) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $\sigma \in \Sigma$. The semantics of an MSO formula is defined in a classical way, and for φ an MSO formula, $w \in \mathcal{N}(\Sigma)$, ν a valuation of the free variables of φ into positions and sets of positions of w, we write $w, \nu \models \varphi$ to mean that w is a model of φ under the valuation ν. When φ is a sentence, we just write $w \models \varphi$. We denote by $\mathsf{MSO}_{\mathsf{nw}}[\Sigma]$ the set of MSO formulas for nested words over Σ (and just $\mathsf{MSO}_{\mathsf{nw}}$ when Σ is clear from the context). Since we are interested in transductions from nested words to words, we also define MSO for words. Similarly as nested words, words are seen as structures but in that case we do not have the matching pair predicate $M(x, y)$. MSO formulas on words are defined accordingly to this smaller signature.

EXAMPLE 2. *We interpret* $\mathsf{MSO}_{\mathsf{nw}}[\Sigma]$ *on nested words rather that on words in* Σ^*. *It is not a restriction since checking whether a given relation* $M(x, y)$ *is a valid matching relation is definable by an MSO formula* ϕ_{wn}. *This formula expresses that* M *is a bijection between call and return symbols, and that it is well-nested (there is no crossing), as follows:*

$$\neg\exists x_c, x_r, y_c, y_r.M(x_c, x_r) \wedge M(y_c, y_r) \wedge x_c \prec y_c \prec x_r \prec y_r \\ \wedge\, bij(M) \wedge \forall x, y.M(x, y) \to x \prec y$$

where $\prec$ *is the transitive closure of* S *(well-known to be MSO-definable) and* $bij(M)$ *expresses that* M *maps bijectively call and return symbols (it is trivially MSO-definable).*

MSO transducers from nested words to words MSO-transducers define (partial) functions from nested words to word structures. The output word structure is defined by taking a fixed number k of copies of the input structure domain. Nodes of these copies can be filtered out by $\mathsf{MSO}_{\mathsf{nw}}$ formulas with one free first-order variable. In particular, the nodes of the c-th copy are the input positions that satisfy some given $\mathsf{MSO}_{\mathsf{nw}}$ formula $\phi^c_{pos}(x)$. The label predicates $\sigma(x)$ and the successor predicate $S(x, y)$ of the output structure are defined by $\mathsf{MSO}_{\mathsf{nw}}$ formulas with respectively one and two free first-order variables, interpreted over the input structure. Formally, *an MSO-transducer from nested words to words* is a tuple $T =$

$$(k, \phi_{dom}, (\phi^c_{pos}(x))_{1 \le c \le k}, (\phi^c_\sigma(x))_{\substack{1 \le c \le k \\ \sigma \in \Sigma}}, (\phi^{c,d}_S(x, y))_{1 \le c, d \le k})$$

where $k \in \mathbb{N}$ and the formulas ϕ_{dom}, ϕ^c_{pos}, ϕ^c_a and $\phi^{c,d}_S$ are $\mathsf{MSO}_{\mathsf{nw}}$ formulas. We denote by $\mathsf{MSO}[\mathsf{nw2w}]$ the class of MSO-transducers from nested words to words.

An MSO-transducer T defines a function from nested word structures over Σ to word structures over Σ, denoted by $[\![T]\!]$. The domain of $[\![T]\!]$ consists of all nested word structures u such that $u \models \phi_{dom}$. Given a nested word structure $u \in dom([\![T]\!])$, the *output structure* v such that $(u, v) \in [\![T]\!]$ is defined by the domain $D^v \subseteq pos(u) \times \{1, \dots, k\}$ such that $D^v = \{(i, c) \mid i \in pos(u),\ c \in \{1, \dots, k\},\ u \models \phi^c_{pos}(i)\}$, a node $(i, c) \in D^v$ of the output structure is labeled $a \in \Sigma$ if $u \models \phi^c_a(i)$, and a node $(j, d) \in D^v$ is the successor of a node $(i, c) \in D^v$ if $u \models \phi^{c,d}_S(i, j)$. Note that the output structure is not necessarily a word, because for instance, nothing guarantees that an output node is labeled by a unique symbol, or that the successor relation forms a linear order on the positions. However, it is not difficult to see that it is decidable whether an $\mathsf{MSO}[\mathsf{nw2w}]$ transducer produces only words (see for instance (Filiot 2015)).

We say that a function f from nested words to words is MSO-definable if there exists an $T \in \mathsf{MSO}[\mathsf{nw2w}]$ such that $[\![T]\!] = f$. By definition of $\mathsf{MSO}[\mathsf{nw2w}]$ transducers, for any MSO-definable function f there exists $k \in \mathbb{N}$ such that for all $u \in Dom(f)$, $|f(v)| \le k.|u|$ (by taking k as the number of copies of the $\mathsf{MSO}[\mathsf{nw2w}]$ transducer defining f). We say in that case that f is of *linear-size increase*.

EXAMPLE 3. *This example transforms a nested word into the sequence of calls of maximal depth (the leaves). E.g.,* $c_1c_2r_2c_3c_4r_4r_3r_1$ *is mapped to* c_2c_4. *This transformation is MSO-definable. The domain is defined by the formula* ϕ_{wn} *(see Example 2). One needs only one copy of the input word, whose positions are filtered out by the formula* $\phi^1_{pos}(x) = \exists y.M(x, y) \wedge S(x, y)$ *which holds true iff* x *is a call position and its successor position* y *is its matching return position. The labels are preserved:* $\phi^1_a(x) = a(x)$ *for all* $a \in \Sigma$. *Finally, the successor relation is defined by* $\phi^{1,1}_S(x, y) = \phi^1_{pos}(x) \wedge \phi^1_{pos}(y) \wedge x \prec y \wedge \neg\exists z.\phi^1_{pos}(z) \wedge x \prec z \prec y$.

4.2 Logical equivalences

An $\mathsf{MSO}[\mathsf{nw2w}]$ T is said to be *order-preserving* if for any word u of the domain of T, any positions i, j of u and any copies c, d of T, if $u \models \phi_S^{c,d}(i,j)$ then $i \leq j$. This means that the output arrows can not point to the right. It is emphasized by the next theorem, which echoes a similar result on words proved in (Bojanczyk 2014; Filiot 2015).

THEOREM 6. *An order-preserving transduction is definable in* $\mathsf{MSO}[\mathsf{nw2w}]$ *if, and only if, it is definable by a functional*[4] VPT.

In the following, we show that $\mathsf{D2VPT}$ are strictly more expressive than $\mathsf{MSO}[\mathsf{nw2w}]$, and define a restriction that capture exactly $\mathsf{MSO}[\mathsf{nw2w}]$. The fact that $\mathsf{D2VPT}$ are more expressive than $\mathsf{MSO}[\mathsf{nw2w}]$ can be easily shown, based on a similar result for ranked trees established in (Courcelle and Engelfriet 2012). Since $\mathsf{D2VPT}$ can, using their stack, express transductions of exponential-size increase, while MSO-transductions are of linear-size increase, they are strictly more expressive than $\mathsf{MSO}[\mathsf{nw2w}]$.

To capture exactly $\mathsf{MSO}[\mathsf{nw2w}]$, one defines the *single-use restriction* for $\mathsf{D2VPT}$ (and $\mathsf{D2VPT}^{\mathsf{LA}}$). Intuitively, this restriction requires that when a $\mathsf{D2VPT}$ passes twice at the same position with the same state, then necessarily the transitions fired from these states produces ϵ.

DEFINITION 5 (Single-use restriction). *A* $\mathsf{D2VPT}$ *(resp.* $\mathsf{D2VPT}^{\mathsf{LA}}$*)* $T = (A, \mathsf{O})$ *with* $A = (Q, q_I, F, \Gamma, \delta)$ *a* $\mathsf{2VPA}$ *(resp.* $\mathsf{2VPA}^{\mathsf{LA}}$*) is* single-use *with respect to a set* $P \subseteq Q$ *if any transition* t *from a state* $q \notin P$ *satisfies* $\mathsf{O}(t) = \epsilon$*, and if for all runs* $r = (q_0, i_0, d_0, \sigma_0) \dots (q_\ell, i_\ell, d_\ell, \sigma_\ell)$ *of* T *on a word* w *and all states* $p \in P$*,* r *does not visit twice the same position in state* p*, i.e. if* $(q_\alpha, i_\alpha) = (q_\beta, i_\beta)$ *for* $\alpha \neq \beta$*, then* $q_\alpha = q_\beta \notin P$*.*

A $\mathsf{D2VPT}$ *(resp.* $\mathsf{D2VPT}^{\mathsf{LA}}$*) is* single-use *if it is single-use w.r.t. some set* $P \subseteq Q$*, and* strongly single-use *if it is single-use w.r.t.* Q*.*

We denote by $\mathsf{D2VPT}_{\mathsf{su}}$ (resp. $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$) the class of single-use $\mathsf{D2VPT}$ (resp. $\mathsf{D2VPT}^{\mathsf{LA}}$). By reduction to the D2VPA emptiness, we get:

PROPOSITION 3. *Deciding the single use property on a* $\mathsf{2VPT}$ *is* EXPTIME-C.

A single-use restriction was already defined in (Courcelle and Engelfriet 2012) for deterministic tree-walking transducers with look-around to capture MSO-transductions from trees to trees (and words). It requires that in any accepting run, every node is visited at most once by a state. It is therefore more restrictive than our single-restriction and, as a matter of fact, corresponds to what we call the strongly single-use restriction. However, the following result shows that the strongly single-use restriction is not powerful enough, in our context, to capture all MSO-definable transductions, even with regular look-arounds.

LEMMA 3. *There is an MSO-definable nested word to word transduction* f *which is not definable by strongly single-use* $\mathsf{D2VPT}^{\mathsf{LA}}$.

We now proceed to the first logical equivalence, between our model and MSO-transductions, which is mainly a consequence of results from (Courcelle and Engelfriet 2012).

THEOREM 7. *Let* f *be a transduction from nested words to words. Then* f *is MSO-definable iff it is definable by a (look-around)* $\mathsf{D2VPT}_{\mathsf{su}}$*, i.e.,*

$$\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} = \mathsf{D2VPT}_{\mathsf{su}}.$$

Sketch of proof. We show that both other models are equivalent to $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$. We have already seen that look-around can be removed from $\mathsf{D2VPT}^{\mathsf{LA}}$ (Theorem 2), while preserving their expressive power. Our Hopcroft-Ullman's construction can add exponentially more visits to the same positions, but these visits are only ϵ-producing. In other words, our Hopcroft-Ullman's construction does not preserve the strongly single-use restriction, but it preserves the single-use restriction. As a consequence of this observation and Corollary 2, we obtain that $\mathsf{D2VPT}_{\mathsf{su}} = \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$.

To show $\mathsf{MSO}[\mathsf{nw2w}] \subseteq \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$, we rely on the equivalence of (Courcelle and Engelfriet 2012) between deterministic binary tree to word walking transducers with look-around (DTWT^{la}) and MSO-transductions from binary trees to words ($\mathsf{MSO}[\mathsf{b2w}]$). Informally, DTWT^{la} can follow the directions of binary trees (1st child, 2nd child and parent) and take their transitions based on regular look-around information. Due to determinism, they are always strongly single-use, in the sense that any position is not visited twice by the same state. Such a machine, running on first-child next-sibling encoding of nested words, is easily encoded into an equivalent $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$. In this encoding, a nested word over Σ is encoded as a binary tree over $(\Sigma_c \times \Sigma_r) \cup \{\perp\}$, inductively defined as $\mathsf{fcns}(cw_1rw_2) = (c,r)(\mathsf{fcns}(w_1), \mathsf{fcns}(w_2))$ and $\mathsf{fcns}(\epsilon) = \perp$. In this encoding, moving to a 1st child corresponds to moving from c to w_1, which can be done by a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$, and moving to a 2nd child corresponds to moving from c to w_2. This can be done also by a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$, but it needs to traverse all the word cw_1r, while producing ϵ only. Similarly, one can encode moves to parent nodes. The two latter moves implies that the $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ is not strongly single-use anymore, but it remains single-use: the extra moves are all ϵ-producing. The result follows as $\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{MSO}[\mathsf{b2w}] \circ \mathsf{fcns}$.

To show $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} \subseteq \mathsf{MSO}[\mathsf{nw2w}]$, we rely on another correspondence shown in (Courcelle and Engelfriet 2012), between $\mathsf{MSO}[\mathsf{b2w}]$ and deterministic (visibly) pushdown binary tree to word walking transducers with look-around of linear-size increase ($\mathrm{DPTWT}^{la}_{lsi}$). These transducers extend DTWT^{la} with a pushdown store with a visibly condition: when moving to a child, they push one symbol, and moving up, they pop one symbol. The lsi restriction is semantical: they restrict the class to transducers that define lsi transductions. Any $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ defines an lsi transduction, and can be easily encoded into a $\mathrm{DPTWT}^{la}_{lsi}$ running on fcns encodings, which mimics the moves of the $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$. Again, the result follows by the equality $\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{MSO}[\mathsf{b2w}] \circ \mathsf{fcns}$. ☐

4.3 Comparison with other transducer models

In this section, we relate $\mathsf{D2VPT}$ to two other transducer models, namely streaming tree-to-string transducers and deterministic hedge-to-string transducers with look-ahead. Streaming tree-to-string transducers with a simple copyless restriction of updates will serve as the third edge of our trinity. Deterministic hedge-to-string transducers with look-ahead is a natural model for which equivalence is known to be decidable.

Streaming tree-to-string transducers are deterministic one-way machines (Alur and D'Antoni 2012) equipped with registers storing words. We fix a finite alphabet Δ and, given two finite sets $\mathcal{X}$ and $\mathcal{Y}$, denote by $\mathcal{U}(\mathcal{X}, \mathcal{Y})$ the set of mappings from $\mathcal{X}$ to $(\Delta \cup \mathcal{Y})^*$.

DEFINITION 6. *A* streaming tree-to-string transducer S *(*STST *for short) is a deterministic machine defined over a structured alphabet* Σ *and given by the tuple* $(Q, q_I, \Gamma, \mathcal{X}, \delta, \mu_F)$ *where* Q *is a finite set of states,* $q_I \in Q$ *is the initial state,* Γ *is a finite set of stack symbols and* $\mathcal{X}$ *is a finite set of registers. Finally,* μ_F *is a partial mapping from* Q *to* $(\Delta \cup \mathcal{X})^*$ *and* $\delta = \delta^{push} \uplus \delta^{pop}$ *where* $\delta^{push} : Q \times \Sigma_c \to Q \times \Gamma \times \mathcal{U}(\mathcal{X}, \mathcal{X})$ *and* $\delta^{pop} : Q \times \Sigma_r \times \Gamma \to Q \times \mathcal{U}(\mathcal{X}, \mathcal{X} \cup \mathcal{X}')$*,* $\mathcal{X}'$ *being a disjoint copy of* $\mathcal{X}$*.*

[4] Within the class of VPT, the class of functional VPT is decidable in PTime (Filiot et al. 2010)

Let $\mathcal{V}^{\Delta}_{\mathcal{X}}$ be the set of mappings from from $\mathcal{X}$ to Δ^*. These mappings are extended to $(\mathcal{X}\cup\Delta)^*$ by considering them as identity over Δ. An accepting run of a STST S on a nested word w is a (non-empty) sequence $(q_0, \theta_0, \sigma_0, w_0) \dots (q_\ell, \theta_\ell, \sigma_\ell, w_\ell)$ of quadruples from $Q \times \mathcal{V}^{\Delta}_{\mathcal{X}} \times (\Gamma \times \mathcal{V}^{\Delta}_{\mathcal{X}})^* \times \Sigma^*$ such that $q_0 = q_I$, $w_0 = w, w_\ell = \epsilon$, θ_0 is the mapping θ_ϵ which associates ϵ to any X in $\mathcal{X}$, σ_0, σ_ℓ are equal to $\perp$ the empty stack and for all $0 \leq i < \ell$, one has either

- $w_i = cw_{i+1}$ and there exists $(q_i, c, q_{i+1}, \gamma, \nu) \in \delta^{push}$, $\theta_{i+1} = \theta_\epsilon$ and $\sigma_{i+1} = \sigma_i(\gamma, \theta_i \circ \nu)$,
- $w_i = rw_{i+1}$ and there exists $(q_i, r, \gamma, q_{i+1}, \nu) \in \delta^{pop}$, $\sigma_i = \sigma_{i+1}(\gamma, \theta)$ and $\theta_{i+1} = \theta' \circ \theta_i \circ \nu$, where $\theta' \in \mathcal{V}^{\Delta}_{\mathcal{X}'}$ is defined by $\theta'(X') = \theta(X)$ for all $X \in \mathcal{X}$.

The semantics $[\![S]\!]$ of the STST S is a partial mapping from $\mathcal{N}(\Sigma)$ to Δ^* such that $[\![S]\!](w) = v$ if there exists an accepting run on w in S ending in some configuration $(q_\ell, \theta_\ell, \perp, \epsilon)$ and $v = \theta_\ell(\mu_F(q_\ell))$.

Using a restriction on the updates $\mathcal{U}$ used in STST (so-called copyless updates), (Alur and D'Antoni 2012) proved that copyless STST and MSO[nw2w] are expressively equivalent. As a consequence, we obtain the logic/two-way/one-way trinity announced in the introduction:

THEOREM 8. MSO[nw2w] $=$ D2VPT$_{\mathsf{su}}$ $=$ copyless STST

A well-known class of transducers running on ranked trees is the class of deterministic top-down tree transducers with look-ahead. This class can be defined to output strings. We consider now the extension of this class to unranked trees, or more precisely sequences of unranked trees, that is, hedges.

DEFINITION 7. *An* hedge automaton *(*HA *for short) over the structured alphabet* Σ [5] *is a tuple* (Q, F, δ) *where* Q *is a finite set of states,* $F \subseteq Q$ *is a set of final states and* δ *is a transition relation such that* $\delta \subseteq Q \times \Sigma_c \times \Sigma_r \times Q \times Q$.

An hedge automaton is said to be bottom-up deterministic if whenever (q, c, r, q_1, q_2) and (q', c, r, q_1, q_2) belongs to δ, it holds that $q = q'$. The semantics of an HA B is given by means of sets $\mathcal{L}^B_q \subseteq \mathcal{N}(\Sigma)$ defined for each $q \in Q$ inductively as follows: (i) $\epsilon \in \mathcal{L}^B_q$ for all q and (ii) $cwrw' \in \mathcal{L}^B_q$ if $(q, c, r, q_1, q_2) \in \delta$ and $w \in \mathcal{L}^B_{q_1}$, $w' \in \mathcal{L}^B_{q_2}$. The language defined by an HA B is then $\bigcup_{q\in F} L^B_q$. Note that when B is bottom-up deterministic whenever $q_1 \neq q_2$, it holds that $\mathcal{L}^B_{q_1} \cap \mathcal{L}^B_{q_2} = \varnothing$.

DEFINITION 8. *A deterministic* hedge-to-string transducer *with look-ahead (*dH2S$^{\mathsf{LA}}$*)* H *over the structured alphabet* Σ *and the output alphabet* Δ *is given by a tuple* (Q, I, F, δ, B) *where* Q *is a finite set of states,* $q_I \in Q$ *is an initial state,* $F \subseteq Q$ *is a set of final states,* B *is a deterministic bottom-up hedge automaton with states* Q'*, and* δ *is a transition relation given by a partial mapping*

$$\delta : Q \times \Sigma_c \times \Sigma_r \times Q' \times Q' \to \Delta^*_Q$$

Δ_Q *is the finite set of symbols* $(\Delta \cup \{q(x_i) \mid 1 \leq i \leq 2, q \in Q\})^*$.

The semantics of a dH2S$^{\mathsf{LA}}$ is first given by a partial mapping $[\![H]\!]$ from $\mathcal{N}(\Sigma) \times Q$ onto Δ^* defined inductively as: (i) $[\![H]\!](\epsilon, q) = \epsilon$ if $q \in F$, and (ii) for $w = cw_1rw_2$ with $w_1, w_2 \in \mathcal{N}(\Sigma)$, $[\![H]\!](w, q) = \omega[q_i(x_{i_j}) \leftarrow [\![H]\!](w_{i_j}, q_i)]$ where $\omega[q_i(x_{i_j}) \leftarrow [\![H]\!](w_{i_j}, q_i)]$ denotes the word ω in which each occurrence of $q_i(x_{i_j})$ has been replaced by $[\![H]\!](w_{i_j}, q_i)$ if $\delta(q, c, r, q', q'') = \omega$, $w_1 \in \mathcal{L}^{q'}_B$ and $w_2 \in \mathcal{L}^{q''}_B$ and undefined otherwise.

Then, the transduction $[\![H]\!]$ defined by H is given by $\{(w, s) \mid w \in \mathcal{N}(\Sigma),\ s = [\![H]\!](w, q_I)\}$.

THEOREM 9. D2VPT $\subsetneq$ STST *and* D2VPT $\subsetneq$ dH2S$^{\mathsf{LA}}$

Sketch of proof. The two results rely on a same intermediate model that extends the transition algebra described in Section 2. This algebra allows to describe the possible traversals of a D2VPA. One can extend it to D2VPT by storing in matrices the words produced by traversals. This yields an infinite algebra, realized by a finite set of operations. We use this to describe effective translations into STST and dH2S$^{\mathsf{LA}}$.

As an illustration, in order to build of an equivalent STST, the set of variables considered is the set $\Xi = \{x^{(p,d),(p',d')} \mid (p, d), (p', d') \in Q \times \mathbb{D}\}$, *i.e.* one variable for each traversal. This generalizes the construction described in (Alur and Černý 2010; Alur et al. 2012) in order to translate a deterministic two-way transducer (on words) into a streaming string transducer.

The fact that the inclusions are strict relies on a simple argument based on size increase: on nested words of bounded depth, D2VPT are linear-size increase, while STST and dH2S$^{\mathsf{LA}}$ are not. □

5. Discussion

Unranked tree to word transductions Since unranked trees t can be linearised into nested words $lin(t)$, our result also gives a model for unranked tree to word transductions. If one denotes by MSO[u2w] the transductions from unranked trees to words definable by an MSO transducer (over the signature of unranked trees that has the child and next-sibling predicates), it is easy to show that MSO[u2w] $=$ D2VPT$_{\mathsf{su}}$ $\circ\, lin$.

One could argue that D2VPT for realising transductions of unranked trees is not an adequate model, because it performs unnecessary ϵ-producing moves to navigate, for instance, from a node n to its next-sibling. Indeed, the D2VPT needs to walk through the whole subtree rooted at n.

First, while it is true from an operational point of view, we think that the simplicity of D2VPT makes them a good candidate as a specification model of unranked tree transductions, and to this aim, it is easy to define, as we did for next-sibling moves (rules $q \xrightarrow{(c,r)} p$), macros that realise moves given by the predicates of unranked trees (and their inverse). Second, for instance in the context of stream processing of XML documents, it cannot be always assumed that the input document is given by its DOM (with the unranked tree predicates) as sometimes, it is just stored as plain text, i.e. as its linearisation.

Finally and most importantly, our result allows one to get an extension of a known model of ranked tree to word transductions, to unranked tree to word transductions, namely, *deterministic pushdown unranked tree to word walking transducers* (DPUWT). To avoid technical details, we define formally this model only in Appendix, and rather give intuitions here. DPUWT can walk through the unranked tree following the next-sibling and first-child predicates (and their inverse), while producing words on the output. They are also equipped with a pushdown store with a visibly condition: whenever they go down the tree by one level, they have to push one symbol onto the stack, and going up, they pop one symbol. They let the stack unchanged when moving horizontally between siblings. With the single-use restriction, defined similarly as for D2VPT, we get that MSO[u2w] $=$ DPUWT$_{su}$. Therefore, if the input is given by an unranked tree, one can rather use a DPUWT or a D2VPT on the linearisation.

[5] Usually, such automata are given over a classical unstructured but unary alphabet. However, for having a uniform presentation, we choose wlog this definition which corresponds somehow to consider a pair from $\Sigma_c \times \Sigma_r$ as single symbol.

Nested word to nested word transductions As we claimed earlier, $\mathsf{D2VPT_{su}}$ can be used to define unranked tree transformations represented as nested word to nested word transducers, that is, as nested word to word tranduscers with a structured output alphabet.

On the logical side, $\mathsf{MSO}[\mathsf{nw2w}]$ transductions can be extended with binary formulas $\varphi^{c,d}_M(x,y)$ aiming at representing the matching relation existing on output nested words. As checking whether a relation denotes a matching relation is MSO definable (see Example 2), one can decide whether any input nested word is indeed transformed by the $\mathsf{MSO}[\mathsf{nw2w}]$ transducer into a nested word by testing the validity of the sentence obtained from the logical definition of the matching M (Example 2) by replacing the predicate M with $\bigvee_{c,d}\varphi^{c,d}_M$. So, starting from an $\mathsf{MSO}[\mathsf{nw2w}]$ transducer with a matching relation defined on its output, one may forget this matching and view this transducer as an ordinary $\mathsf{MSO}[\mathsf{nw2w}]$ transducer; this machine turns out to be equivalent in the sense that remaining call and returns symbols induce uniquely the erased matching. Finally, by the results presented in this paper, one can from this $\mathsf{MSO}[\mathsf{nw2w}]$ transducer build an equivalent $\mathsf{D2VPT_{su}}$ whose range will indeed contain only nested words and thus, defines an unranked tree transformation.

Let us point out that our results do not entail the trinity for tree-to-tree transformations: the class of D2VPT which produce only nested words/trees as output may be a good candidate to complete the missing part (the equivalence between MSO transformations and streaming tree transducers has already been established in (Alur and D'Antoni 2012)). Nonetheless, deciding this class seems to be challenging and moreover, there is actually no guarantee that it corresponds to the other two cited members of this trinity.

Input streaming In an input streaming scenario, one assumes that the input nested word is given as a stream of call and return symbols. In such a scenario, one wants to transform the input stream as soon as possible, on-the-fly, and it is not reasonable to load the whole stream in memory. An interesting question is whether a given $\mathsf{D2VPT}$ really needs its two-way ability ? In other words, can we decide whether a given $\mathsf{D2VPT}$ is equivalent to a (one-way) VPT? For words and two-way finite transducers, this question has been shown to be decidable in (Filiot et al. 2013). As future work, we want to extend this result to $\mathsf{D2VPT}$.

References

R. Alur. Nested words, 2016. URL `https://www.cis.upenn.edu/~alur/nw.html`.

R. Alur and P. Černý. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8, pages 1–12, 2010.

R. Alur and P. Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *POPL*, pages 599–610, 2011.

R. Alur and L. D'Antoni. Streaming tree transducers. In *ICALP (2)*, volume 7392 of *LNCS*, pages 42–53. Springer, 2012.

R. Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3), 2009.

R. Alur, E. Filiot, and A. Trivedi. Regular transformations of infinite strings. In *LICS*, pages 65–74, 2012.

R. Bloem and J. Engelfriet. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *J. Comput. Syst. Sci.*, 61(1):1–50, 2000.

M. Bojanczyk. Transducers with origin information. In *ICALP*, volume 8573 of *LNCS*, pages 26–37. Springer, 2014.

M. Chytil and V. Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *ICALP*, volume 52 of *LNCS*, pages 135–147. Springer, 1977.

H. Comon-Lundh, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. online, Nov. 2007. URL `http://www.grappa.univ-lille3.fr/tata/`.

B. Courcelle. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126(1):53–75, 1994.

B. Courcelle and J. Engelfriet. Book: Graph structure and monadic second-order logic. A language-theoretic approach. *Bulletin of the EATCS*, 108: 179, 2012.

K. Culik and J. Karhumaki. The equivalence problem for single-valued two-way transducers (on NPDT0L languages) is decidable. *SIAM J. on Computing*, 16(2):221–230, 1987.

J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic*, 2(2):216–254, 2001.

J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and mso definable tree translations. *Information and Computation*, 154 (1):34–91, 1999.

J. Engelfriet and S. Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. of Computing*, 32(4):950–1006, 2003.

E. Filiot. Logic-automata connections for transformations. In *ICLA*, volume 8923 of *LNCS*, pages 30–57. Springer, 2015.

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.

E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. Streamability of nested word transductions. In *FSTTCS*, volume 13 of *LIPIcs*, pages 312–324, 2011.

E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *LICS*, pages 468–477. IEEE, 2013.

O. Gauwin, J. Niehren, and S. Tison. Queries on XML streams with bounded delay and concurrency. *Information and Computation*, 209(3): 409–442, 2011.

E. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM J. on Computing*, 11, 1982.

J. E. Hopcroft and J. D. Ullman. An approach to a unified theory of automata. *BELLTJ: The Bell System Technical Journal*, 46:1793–1829, 1967.

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

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

F. Neven and T. Schwentick. Query automata over finite trees. *Theoretical Computer Science*, 275, 2002.

J. Niehren, L. Planque, J.-M. Talbot, and S. Tison. N-ary queries by tree automata. In *DBLP*, volume 3774 of *LNCS*, pages 217–231. Springer, 2005.

J. Pécuchet. Automates boustrophedon, semi-groupe de Birget et monoide inversif libre. *RAIRO - ITA*, 19(1):71–100, 1985.

F. Picalausa, F. Servais, and E. Zimányi. XEvolve: an XML schema evolution framework. In *SAC*, pages 1645–1650. ACM, 2011.

J. Raskin and F. Servais. Visibly pushdown transducers. In *ICALP*, volume 5126 of *LNCS*, pages 386–397. Springer, 2008. doi: 10.1007/978-3-540-70583-3_32. URL `http://dx.doi.org/10.1007/978-3-540-70583-3_32`.

L. Segoufin and C. Sirangelo. Constant-memory validation of streaming XML documents against DTDs. In *ICDT*, volume 4353 of *LNCS*, pages 299–313. Springer, 2007.

H. Seidl, S. Maneth, and G. Kemper. Equivalence of deterministic top-down tree-to-string transducers is decidable. In *FOCS*, pages 943–962. IEEE, 2015.

J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

W. Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words. Springer, Berlin, 1997.

A. Appendix

A.1 Two-way visibly pushdown automata

PROPOSITION 1. *The relation $\sim$ is a congruence of finite index.*

Proof. We consider R the set of binary relations over $Q \times \mathbb{D}$. Obviously, R is finite. As traversals are subsets of R, $\sim$ is of finite index. Let us now prove that $\sim$ is a congruence relation for the binary operation . and the unary ones, $f_{c,r}$ (for $c \in \Sigma_c, r \in \Sigma_r$).

From $\sim$, we define four equivalence relations $\sim_{\mathsf{ll}}, \sim_{\mathsf{lr}}, \sim_{\mathsf{rl}}, \sim_{\mathsf{rr}}$ on $Q \times Q$ such that for $(\alpha, \beta \in \{\mathsf{l}, \mathsf{r}\})$, we have $u \sim_{\alpha\beta} v$ if

$$[u]_\sim \cap (Q \times \{bdir(\alpha)\} \times Q \times \{edir(\beta)\}) =$$
$$[v]_\sim \cap (Q \times \{bdir(\alpha)\} \times Q \times \{edir(\beta)\})$$

where $bdir(\mathsf{l}) = edir(\mathsf{r}) = \rightarrow$ and $bdir(\mathsf{r}) = edir(\mathsf{l}) = \leftarrow$.

Intuitively, (p, q) belongs to $[w]_{\sim_{\mathsf{ll}}}$ (respectively to $[w]_{\sim_{\mathsf{lr}}}$) if there exists a run of A on w starting reading w from the *left* side, *ie*, with direction $\rightarrow$ in state p and leaves the word on the *left*, *ie*, with direction $\leftarrow$ (resp. on the *right*, *ie*, with direction $\rightarrow$) in state q.

The relation $\sim$ is uniquely determined by the four relations $\sim_{\mathsf{ll}}, \sim_{\mathsf{lr}}, \sim_{\mathsf{rl}}, \sim_{\mathsf{rr}}$ and in particular, $\sim$ is a congruence iff all the $\sim_{\mathsf{ll}}, \sim_{\mathsf{lr}}, \sim_{\mathsf{rl}}, \sim_{\mathsf{rr}}$ are congruences.

Let us first notice that for ϵ, one has $[\epsilon]_{\sim_{\mathsf{ll}}} = [\epsilon]_{\sim_{\mathsf{rr}}} = \varnothing$ whereas $[\epsilon]_{\sim_{\mathsf{lr}}}$, $[\epsilon]_{\sim_{\mathsf{rl}}}$ are the identity relation.

Let us consider u, u', v, v' in $\mathcal{N}(\Sigma)$ and assume that $u \sim u'$ (and thus, $u \sim_{\mathsf{ll}} u'$, $u \sim_{\mathsf{lr}} u'$, $u \sim_{\mathsf{rl}} u'$, $u \sim_{\mathsf{rr}} u'$) and $v \sim v'$. We consider $u.v$ and $u.'v'$ and prove that $u.v \sim u.'v'$.

From the definition of runs and traversals, one has

$$\begin{array}{l}
[u.v]_{\sim_{\mathsf{ll}}} = [u]_{\sim_{\mathsf{ll}}} \cup [u]_{\sim_{\mathsf{lr}}} \circ ([v]_{\sim_{\mathsf{ll}}} \circ [u]_{\sim_{\mathsf{rr}}})^* \circ [v]_{\sim_{\mathsf{ll}}} \circ [u]_{\sim_{\mathsf{rl}}} \\
{[u.v]_{\sim_{\mathsf{lr}}}} = [u]_{\sim_{\mathsf{lr}}} \circ ([v]_{\sim_{\mathsf{ll}}} \circ [u]_{\sim_{\mathsf{rr}}})^* \circ [v]_{\sim_{\mathsf{lr}}} \\
{[u.v]_{\sim_{\mathsf{rl}}}} = [v]_{\sim_{\mathsf{rl}}} \circ ([u]_{\sim_{\mathsf{rr}}} \circ [v]_{\sim_{\mathsf{ll}}})^* \circ [u]_{\sim_{\mathsf{rl}}} \\
{[u.v]_{\sim_{\mathsf{rr}}}} = [v]_{\sim_{\mathsf{rr}}} \cup [v]_{\sim_{\mathsf{rl}}} \circ ([u]_{\sim_{\mathsf{rr}}} \circ [v]_{\sim_{\mathsf{ll}}})^* \circ [u]_{\sim_{\mathsf{rr}}} \circ [v]_{\sim_{\mathsf{lr}}}
\end{array}$$

Hence, $[u.v]_{\sim_{\alpha\beta}} = [u'.v']_{\sim_{\alpha\beta}}$ for all $\alpha, \beta \in \{\mathsf{l}, \mathsf{r}\}$ and so, $u.v \sim u'.v'$. Let us point out that these definition are similar to those defined for words in the case of two-way finite state automata (Shepherdson 1959) and that $[(u.v).w]_{\sim_{\alpha\beta}} = [u.(v.w)]_{\sim_{\alpha\beta}}$ and $[u.\epsilon]_{\sim_{\alpha\beta}} = [\epsilon.u]_{\sim_{\alpha\beta}}$ for all $\alpha, \beta \in \{\mathsf{l}, \mathsf{r}\}$.

Now, let us consider u, u' in $\mathcal{N}(\Sigma)$ and assume that $u \sim u'$. We consider $cur = f_{c,r}(u)$ and $cu'r = f_{c,r}(u')$ and show that $cur \sim cu'r$. Expressing traversals on cur is much more intricate. To ensure that traversals abstract properly runs, we need to forget about stack contents and thus, reason again only on nested words when composing sub-runs of cwr. Hence, new notations are needed: we let for d in $\{\leftarrow, \rightarrow\}$ and $w \in \mathcal{N}(\Sigma)$

$$Z^c_{\mathsf{ll},d} = \left\{(p, q) \mid (p, \rightarrow, c, p', \leftarrow, \gamma) \in \delta^{\text{push}} \,,\ (p', \leftarrow, \gamma, c, q, d) \in \delta^{\text{pop}}\right\}$$

$$Z^{cw}_{\mathsf{ll},d} = \bigcup_{\gamma\in\Gamma} \left(\bigcup_{(\overrightarrow{p} \xrightarrow{c,+\gamma} \overrightarrow{p'})} \{(p, p')\} \circ [w]_{\sim_{\mathsf{ll}}} \circ \bigcup_{(\overleftarrow{q'} \xrightarrow{c,-\gamma} (q,d))} \{(q', q)\} \right)$$

$$Z^r_{\mathsf{rr},d} = \left\{(p, q) \mid (p, \leftarrow, r, p', \rightarrow, \gamma) \in \delta^{\text{push}} \,,\ (p', \rightarrow, r, \gamma, q, d) \in \delta^{\text{pop}}\right\}$$

$$Z^{cw}_{\mathsf{rr},d} = \bigcup_{\gamma\in\Gamma} \left(\bigcup_{(\overleftarrow{p} \xrightarrow{r,+\gamma} \overleftarrow{p'})} \{(p, p')\} \circ [w]_{\sim_{\mathsf{rr}}} \circ \bigcup_{(\overrightarrow{q'} \xrightarrow{r,-\gamma} (q,d))} \{(q', q)\} \right)$$

The expressions $Z^c_{\mathsf{ll},d}$ and $Z^{cw}_{\mathsf{ll},d}$ stands both for left-to-left traversal reading twice the initial letter c; the former one represents a back-and-forth move on c whereas $Z^{cw}_{\mathsf{ll},d}$ implies that between the readings of c a left-to-left traversal of w is performed. If the last direction d is $\leftarrow$ then the reading head leaves the word, otherwise the next reading will be c again. The expressions $Z^r_{\mathsf{rr},d}$ and $Z^{wr}_{\mathsf{rr},d}$ are defined dually.

$$T^{cwr}_{\mathsf{lr},d} = \bigcup_{\gamma\in\Gamma} \left(\bigcup_{(\overrightarrow{p} \xrightarrow{c,+\gamma} \overrightarrow{q}} \{(p, q)\} \right) \circ [w]_{\sim_{\mathsf{lr}}} \circ \left(\bigcup_{(\overrightarrow{p} \xrightarrow{r,-\gamma} (q,d))} \{(p, q)\} \right)$$

$$T^{cwr}_{\mathsf{rl},d} = \bigcup_{\gamma\in\Gamma} \left(\bigcup_{(\overleftarrow{p} \xrightarrow{r,+\gamma} \overleftarrow{q}} \{(p, q)\} \right) \circ [w]_{\sim_{\mathsf{rl}}} \circ \left(\bigcup_{(\overleftarrow{p} \xrightarrow{c,-\gamma} (q,d)} \{(p, q)\} \right)$$

The expression $T^{cwr}_{\mathsf{lr},d}$ represents a direct traversal from left-to-right, going once through c and r.

Finally, the classes $[cwr]_{\sim_{\mathsf{ll}}}$, $[cwr]_{\sim_{\mathsf{lr}}}$, $[cwr]_{\sim_{\mathsf{rl}}}$ and $[cwr]_{\sim_{\mathsf{rr}}}$ are defined in Figure 3.

Hence, we indeed have that $[f_{c,r}(u)]_{\sim_{\alpha\beta}} = [cur]_{\sim_{\alpha\beta}} = [cu'r]_{\sim_{\alpha\beta}} = [f_{c,r}(u')]_{\sim_{\alpha\beta}}$ for all $\alpha, \beta \in \{\mathsf{l}, \mathsf{r}\}$. ☐

COROLLARY 1. *For any* 2VPA *A, deciding the emptiness of A (ie $\mathcal{L}(A) = \varnothing$) is* EXPTIME-C*. The same result holds for* D2VPA*.*

Proof. We detail the hardness proof by reduction from the emptiness of k DVPA. The latter problem can be shown to be EXPTIME-hard from the EXPTIME-hardness of intersection emptiness of k deterministic top-down tree automata and a polynomial translation of deterministic top-down tree automata into DVPA (Alur and Madhusudan 2009). We can then encode this problem as emptiness of a D2VPA as follows: the D2VPA simulates one after the other the DVPA's $A_1, \ldots, A_k$; once the word is read from left to right simulating A_i, if a final state of A_i is reached, one enters a state that move the reading head at the beginning of the word and then switches to the initial state of A_{i+1} to read the input word once again. Hence, starting initially in the initial state of A_1, if the final state of A_k is reached then the input nested word belongs to all the A_i's. ☐

A.2 Two-way visibly pushdown transducers

LEMMA 2. *Any unambiguous* VPT *T can be written as the composition of two* VPT *$T_1 \circ T_2$, where T_1 is deterministic and T_2 is letter-to-letter and co-deterministic. Furthermore, if T is letter-to-letter, so is T_1.*

Proof. It has been proved in (**?**) that every unambiguous VPT can be transformed into a DVPT equipped with a look-ahead limited to the current hedge. Formally, such a transducer is defined as a triple (T, A, λ) where T is a DVPT, A is a VPA with no initial states, and λ is a mapping from call transitions of T to states of A. Given a state p of A, we denote by A_p the VPA defined from A by letting $\{p\}$ be the set of initial states. A call transition t of T can then be fired at some position of an input word w only if the longest nested subword of w from this position belongs to $\mathcal{L}(A_p)$.

Intuitively, the decomposition of a DVPT with look-ahead works as follows: the co-deterministic letter-to-letter VPT does a first pass enriching the alphabet with the results of the look-ahead tests. Then the deterministic VPT simulates the DVPT with look-ahead using this additional information.

Formally, let (T, A, λ) be a DVPT with look-ahead from Σ to Δ, with $A = (Q, F, \Gamma, \delta)$. We first define the structured alphabet Σ' as the disjoint union of the set of call symbols $\Sigma_c \times 2^Q$, and the set of return symbols Σ_r. We define the co-deterministic letter-to-letter VPT $T_2 = (A_2, \mathsf{O}_2)$ from Σ to Σ', where A_2 is defined as the co-determinisation of A. Formally, let us denote by id_X the set $\{(q, q) \mid q \in X\}$. We define $A_2 = (2^{Q\times Q}, I_2, \mathsf{id}_F, \Sigma_r \times 2^{Q\times Q}, \alpha)$ where $I_2 = \{S \subseteq Q \times Q \mid S \cap I \times F \neq \emptyset\}$ and the transitions of A_2 are defined as follows:

$$
\begin{aligned}
[cwr]_{\sim_{\mathsf{ll}}} &= \left(Z^{c/cw}_{\mathsf{ll},\rightarrow} \cup \left(T^{cwr}_{\mathsf{lr},\leftarrow} \circ (Z^{r/wr}_{\mathsf{rr},\leftarrow})^* \circ T^{cwr}_{\mathsf{rl},\rightarrow}\right)\right)^* \left(Z^{c/cw}_{\mathsf{ll},\leftarrow} \cup \left(T^{cwr}_{\mathsf{lr},\leftarrow} \circ (Z^{w/wr}_{\mathsf{rr},\leftarrow})^* \circ T^{cwr}_{\mathsf{rl},\leftarrow}\right)\right) \\
[cwr]_{\sim_{\mathsf{lr}}} &= \left(Z^{r/wr}_{\mathsf{ll},\rightarrow} \cup \left(T^{cwr}_{\mathsf{lr},\leftarrow} \circ (Z^{r/wr}_{\mathsf{rr},\leftarrow})^* \circ T^{cwr}_{\mathsf{rl},\rightarrow}\right)\right)^* \left(T^{cwr}_{\mathsf{lr},\rightarrow} \cup \left(T^{cwr}_{\mathsf{lr},\leftarrow} \circ (Z^{r/wr}_{\mathsf{rr},\leftarrow})^* \circ Z^{r/wr}_{\mathsf{rr},\rightarrow}\right)\right) \\
[cwr]_{\sim_{\mathsf{rl}}} &= \left(Z^{r/wr}_{\mathsf{rr},\leftarrow} \cup \left(T^{cwr}_{\mathsf{rl},\rightarrow} \circ (Z^{c/cw}_{\mathsf{ll},\rightarrow})^* \circ T^{cwr}_{\mathsf{lr},\leftarrow}\right)\right)^* \left(T^{cwr}_{\mathsf{rl},\leftarrow} \cup \left(T^{cwr}_{\mathsf{rl},\rightarrow} \circ (Z^{c/cw}_{\mathsf{ll},\rightarrow})^* \circ Z^{c/cw}_{\mathsf{ll},\leftarrow}\right)\right) \\
[cwr]_{\sim_{\mathsf{rr}}} &= \left(Z^{c/cw}_{\mathsf{rr},\leftarrow} \cup \left(T^{cwr}_{\mathsf{rl},\rightarrow} \circ (Z^{c/cw}_{\mathsf{ll},\rightarrow})^* \circ T^{cwr}_{\mathsf{lr},\leftarrow}\right)\right)^* \left(Z^{r/wr}_{\mathsf{rr},\rightarrow} \cup \left(T^{cwr}_{\mathsf{rl},\rightarrow} \circ (Z^{c/cw}_{\mathsf{ll},\rightarrow})^* \circ T^{cwr}_{\mathsf{lr},\rightarrow}\right)\right)
\end{aligned}
$$

Figure 3. Relations of the transition congruence on cwr, where $Z^{c/cw}_{\mathsf{ll},d}$ and $Z^{r/wr}_{\mathsf{ll},d}$ are defined respectively by $Z^{c}_{\mathsf{ll},d} \cup Z^{cw}_{\mathsf{ll},d}$ and $Z^{r}_{\mathsf{ll},d} \cup Z^{wr}_{\mathsf{ll},d}$.

$$
\begin{tikzcd}
p \arrow[rrr, dashed, "Update(c,S,r)"] \arrow[dr, "{c,+\gamma}"'] & & & q \\
& p'' \arrow[r, "S''"] & q'' \arrow[ur, "{r,-\gamma}"'] &
\end{tikzcd}
$$

- Return transitions: $\mathsf{id}_Q \xrightarrow{r,-(r,S)} S$, with $r \in \Sigma_r$ and $S \subseteq Q \times Q$
- Call transitions: $S \xrightarrow{c,+(r,S')} S''$ with $c \in \Sigma_c$, $r \in \Sigma_r$ and $S, S', S'' \subseteq Q \times Q$, iff $S = \mathsf{update}(c, S'', r) \circ S'$ where $\mathsf{update}(c, S'', r) = \{(p,q) \in Q \times Q \mid \exists \gamma \in \Gamma. \exists (p'', q'') \in S''. p \xrightarrow{c,+\gamma} p''$ and $q'' \xrightarrow{r,-\gamma} q\}$

The mapping O_2 associates ϵ to every return transition, and associates $\{p \in Q \mid \exists q \in F.(p,q) \in S\}$ to the call transition $S \xrightarrow{c,+(r,S')} S''$. This set corresponds to the set of states p such that the look-ahead constraint p is satisfied.

The DVPT T_1 from Σ' to Δ can then easily be derived from the DVPT with look-ahead T as the look-aheads tests can be checked on the enriched alphabet Σ'. □

THEOREM 3. *Given a letter-to-letter* DVPT A *and a* $\mathsf{2VPT}$ B*, we can construct a* $\mathsf{2VPT}$ C *that realizes the composition* $C = B \circ A$*. If furthermore* B *is deterministic, then so is* C*.*

Proof. We first notice that since we're considering visibly pushdown machines, the stacks of both machines are always synchronized, meaning that they have the same height on each position. Then, let us remark that when the $\mathsf{2VPT}$ moves to the right, we can do the simulation in a straight forward fashion by simulating it on the production of the one-way, which we can compute. It becomes more involved when it moves to the left. We then need to rewind the run of the one-way, and nondeterminism can arise. To bypass this, let us recall that a similar construction from (Hopcroft and Ullman 1967) exists for classical transducers, and that the rewinding is done through a back and forth reading of the input, backtracking the run up to a position where the nondeterminism is cleared, and then moving back to the current position.

The main idea is that if we were to consider a hedge as a word over subhedges (see Figure 4), we can use the Hopcroft-Ullman construction, given that we know the initial state, i.d. the state in which the one-way enters the hedge. To overcome this, we will ensure the invariant that the stack contains not only the stack symbols from the two transducers, but also at each step it contains the state in which the one-way enters a hedge. Remark that thanks to this, upon moving to the left of a call letter, the state of the one-way is directly given by the information in the stack.

We now explain how we can treat subhedges as letters. First, while the subhedge alphabet is infinite, we are actually interested in their behaviour in the one-way. Thus we consider an automaton not over the subhedge, but over their summaries, which are finite. We can thus compute a finite automaton of the summaries, and apply the Hopcroft-Ullman construction on it. Consequently, we need to

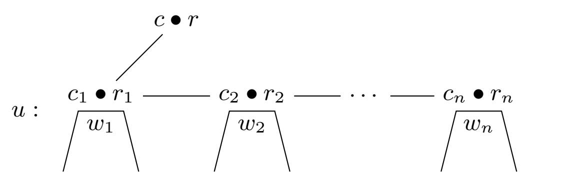

Figure 4. The well nested word $cc_1w_1r_1c_2w_2r_2 \ldots c_nw_nr_n$ can be seen as a word u over letters (c_i, S_i, r_i) where S_i is the summary of w_i. The position labelled by c serves as initial position of the word.

be able to compute the summaries of a given subhedge. This is easily done on the fly using the determinisation procedure of the VPAs. Finally, note that applying the Hopcroft-Ullman construction to the automaton of summaries gives the state in which the one-way enters the previous subhedge (when rewinding a run). This allows us to maintain the invariant, and by reading this subhedge we can compute the state of the one-way at the previous position (from where we started).

Note that the Hopcroft-Ullman routine is deterministic, and consequently the construction preserves determinism.

Formal construction. Let $A = ((Q, i, F, \Gamma, \delta), \mathsf{O}_1)$ be a letter-to-letter DVPT and $B = ((P, j, G, \Theta, \alpha), \mathsf{O}_2)$ be a $\mathsf{2VPT}$ that can be composed with A. We assume that A works on the alphabet equipped with left and right markers and preserves them. Note that it can easily be extended if it is not the case.

We construct $C = ((N, k, H, \Omega, \beta), \mathsf{O}_3)$ a $\mathsf{2VPT}$ that realizes the composition.

- $N = N_m \uplus N_b \uplus N_f \uplus N_s$ where N_m, N_b and N_f correspond to the classical sets of the Hopcroft-Ullman construction, and N_s is used to compute the summary of a subhedge. We have the main mode $N_m = P \times Q$, the back mode $N_b = P \times Q \uplus P \times Q^Q$ and the further mode $N_f = P \times Q^2$, while $N_s = Q^Q$. Note that there are also other states like $read$ or states from $P \times Q \times \{end\}$ that were omitted. The total size of the omitted states is linear in P and Q.
- $k = (i, j)$ is the initial state.
- $H = F \times G$ is the set of final state.
- Ω can similarly to N be written as the disjoint union of stack alphabets for the different modes. We have $\Omega_m = Q \times \Gamma \times \Theta$, $\Omega_b = P \times (Q^Q \uplus Q) \times \Sigma_r$, $\Omega_f = P \times ((Q \times \Gamma) \uplus (Q \times \Gamma)^2)$ and $\Omega_s = Q^Q \times \Sigma_r$.

We now give the transition function β. Lowercase letters denote element of its uppercase counterpart. The direction of a transition is given by the sense of an arrow, and the resulting direction is omitted if it doesn't change. Push transitions are denoted with a $+$ symbol while pop transitions are denoted by a $-$ symbol. For example, we write $(q, \leftarrow, r, q', \leftarrow, \gamma)$ in δ^{push} as $q' \xleftarrow{r,+\gamma} q$ and $(q, \rightarrow, r, q', \leftarrow, \gamma)$ in δ^{pop} as $q \xrightarrow{r,-\gamma} q', \leftarrow$.

- The first three items describe the cases when we are able to directly advance in the two runs. These are the simpler cases. The fourth corresponds to the end of the Hopcroft-Ullman construction, where all the needed information was computed. In these cases, the production of O_3 is the one of the corresponding transition of O_2. Note that in all other cases, the production of O_3 will be empty and thus omitted.
  - $(p, q) \xrightarrow{c,+(q,\gamma,\theta)} (p', q'), d$ if $\mathsf{O}_1(q \xrightarrow{c,+\gamma} q') = c'$ and $p \xrightarrow{c',+\theta} p', d$.
  - $(p, q) \xrightarrow{r,-(q,\gamma,\theta)} (p', q'), d$ if $\mathsf{O}_1(q \xrightarrow{r,-\gamma} q') = r'$ and $p \xrightarrow{r',-\theta} p', d$.
  - $(p', q'), d \xleftarrow{c,-(q',\gamma,\theta)} (p, q)$ if $\mathsf{O}_1(q' \xrightarrow{c,+\gamma} q) = c'$ and $p', d \xleftarrow{c',-\theta} p$.
  - $(p', q'), d \xleftarrow{r,+(q,\gamma,\theta)} (p, q, q', \gamma)$ where there exists q'' such that $\mathsf{O}_1(q' \xrightarrow{r,-\gamma} q'') = r'$ and $p', d \xleftarrow{r',+\theta} p$.
- When B moves to the left on a recall letter, we engage in the Hopcroft-Ullman construction. In order to do that, we need to compute the summary of the subhedge we are about to read. Note that a similar transition happens when the automaton on summaries rewinds one more step. Thus we have the following transitions:
  - $\mathsf{id}_Q \xleftarrow{r,+(p,q,r)} (p, q)$.
  - $\mathsf{id}_Q \xleftarrow{r,+(p,R,r)} (p, R)$.
- Computing a summary amounts to determining a VPA. Note that we stop when we reach the height we are interested in, which is where the stack first contains a state of B, which is handle by the next item. Given a summary S and c, r a call and return letter respectively, we define $Update(c, S, r) = \{(q, q') \mid \exists (q_1, q_2) \in S \text{ and } \gamma\ q \xrightarrow{c,+\gamma} q_1 \text{ and } q_2 \xrightarrow{r,-\gamma} q'\}$. This will reveal to be useful in the remainder of the construction.
  - $\mathsf{id}_Q \xleftarrow{r,+(S,r)} S$.
  - $S'' \xleftarrow{c,-(S',r)} S$ where $S'' = S' \circ Update(c, S, r)$.
- After reading the first subhedge, we get to the point where the top stack symbol is of the form (p, q, r). If there is only one candidate, then there is no ambiguity. Otherwise, we start rewinding the runs.
  - $(p, q', end), \rightarrow \xleftarrow{c,-(p,q,r)} S$ if q' is the only state such that (q', q) belongs to $Update(c, S, r)$.
  - $(p, R) \xleftarrow{c,-(p,q,r)} S$ where $R = \{(q', q') \mid (q', q) \in Update(c, S, r)\}$.
- After reading the following subhedges, similar subcases appear, depending on whether the nondeterminism is cleared or not. If there is only one candidate left, we store a state leading to from the next subhedge, as well as a state leading to another candidate. They will be used to know when we got to the correct position. Otherwise we just update the set of partial runs.
  - $(p, q, q'), \rightarrow \xleftarrow{c,-(p,R,r)} S$ if $R(q)$ is defined, $R \circ Update(c, S, r) \subseteq Q \times \{R(q)\}$ and if $R(q')$ is defined and different from $R(q)$.
  - $(p, R') \xleftarrow{c,-(p,R,r)} S$ where $R' = R \circ Update(c, S, r)$ and $R' \not\subseteq Q \times \{q\}$ for any q.
- It can happen that the nondeterminism has not been cleared until we reach the beginning of the hedge. In the same way that the Hopcroft-Ullman uses the initial state, we then use the information on the top of the stack to decide the candidate.
  - $(p, q, \theta, q'), \rightarrow \xleftarrow{c,-(q,\gamma,\theta)} (p, R)$ where if q'' is such that $q \xrightarrow{c,+\gamma} q''$, both $R(q')$ and $R(q'')$ are defined and different.
- Due to the definition, the model of 2VPT does not allow for direct u-turns. Consequently, the u-turns have been parametrized by specific states in the previous cases. We know explicit how we handle them:
  - $(p, q, end) \xrightarrow{c,+(p,q,\gamma)} q'$ where $q \xrightarrow{c,+\gamma} q'$. We also have a subroutine that follows run of A on this subhedge until it ends.
  - $(p, q, q') \xrightarrow{c,+(p,q,q')} read$ where the state $read$ is a subroutine that only reads the subhedge until it pops the stacked information.
  - $read \xrightarrow{r,-(p,q,q')} (p, q, q')$. At the end of the $read$ subroutine, we start following two runs in parallel, in the same way as in the next subcase.
  - $(p, q, \theta, q') \xrightarrow{c,+(q,\gamma,\theta)} (p, q'', q')$ where $q \xrightarrow{c,+\gamma} q''$.
- When we are in states of N_f, i.d. states of the form (p, q, q'), we simply follow the two runs in parallel, stacking p and the current states on the current height and the stack letters of both runs at each step nonetheless. This subroutine ends upon popping a stack letter that contains p where the two runs collide, meaning we reached the original position. We now explicit what happens on this position:
  - $q \xrightarrow{r,-(p,q',\gamma)} (p, q', q, \gamma), \leftarrow$.
  - $(q, q') \xrightarrow{r,-(p,q_1,\gamma,q_2,\gamma')} (p, q_1, q, \gamma), \leftarrow$ if there exists q'' such that $q \xrightarrow{r,-\gamma} q''$ and $q' \xrightarrow{r,-\gamma'} q''$.

□

A.3 Expressiveness of Two-Way Visibly Pushdown Transducers

THEOREM 6. *An order-preserving transduction is definable in* MSO[nw2w] *if, and only if, it is definable by a functional*[6] VPT.

Proof. The proof relies on the similar result for finite words from (Filiot 2015) and the equivalence between VPA and $\mathsf{MSO}_{\mathsf{nw}}$ from (**?**). Let T be a functional VPT. From (**?**), we know that we can construct an equivalent unambiguous VPT T' realizing the same function. Using (**?**), we can construct an $\mathsf{MSO}_{\mathsf{nw}}$ formula φ of the form $\exists X_1 \ldots \exists X_{|Q|}\ \psi(X_1, \ldots, X_{|Q|})$ that recognizes $dom(T')$. Moreover, given u in $dom(T')$, there exists a unique assignment of the variables X_i satisfying ψ, such that a variable $x \in X_i$ if, and only if, x quantify a position j such that the unique accepting run of T' on u is in state q_i on position j. Using φ, we can then easily construct an MSO[nw2w] transduction T'' using $|Q|$ copies. The domain formula is φ, position formulas are $\phi^q_{pos}(x) = \varphi \wedge x \in X_q$. The successor transition is given by $\phi^{q,q'}_S(x, y) = S(x, y) \wedge \phi^q_{pos}(x) \wedge \phi^{q'}_{pos}(y)$ and we label the q copy of a node by the possibly empty production of the transducer

[6] Within the class of VPT, the class of functional VPT is decidable in PTime (Filiot et al. 2010)

in state q reading the label of the node. We have, for v a production of T, $\phi_v^q(x) = \bigvee_{a\in A_{q,v}} a(x)$ where $A_{q,v} = \{a \in A \mid \exists q'\ q \xrightarrow{a|v} q'\}$. Note that labeling by possibly empty words is not restrictive as MSO transductions are closed under composition, and a simple transduction can extend words into linear graphs and compress the ϵ-labeled paths.

Now given an order-preserving MSO[nw2w] T, we construct an unambiguous VPT that recognizes the same function. As T is order-preserving, for every $u = u_1 \dots u_n$ in $dom(T)$, we can decompose $T(u)$ in $v_1 \dots v_n$ where v_i corresponds to the production from position i. Let us call B the finite set of all possible v_i appearing in a such decomposition. For any v in B, we use the formulas of T to construct a formula $\phi_v(x)$ that holds on an input word u and a position i if in the decomposition of $T(u)$, $v_i = v$. For any sequence $I = (c_1, \dots, c_k)$ of $|v|$ different copies of T, we define $\phi_I^v(x) = \bigwedge_{i<k} \phi_S^{c_i,c_{i+1}}(x,x) \wedge \bigwedge_{i\leq k} \phi_{pos}^{c_i}(x) \wedge \phi_{v_i}^{c_i}(x) \wedge \bigwedge_{x\notin I} \neg\phi_{pos}^c(x)$. The formula $\phi_v(x)$ is simply defined as the disjunction of the formulas $\phi_I^v(x)$ on all possible sequences I.

Then using these formulas, we construct a formula ψ over the finite alphabet $\Sigma\times B$ that recognizes the language $\mathcal{L} = \{(u, T(u)) \mid u \in dom(T)\}$. We define $\psi = \phi'_{dom} \wedge \forall x\ (a,v)(x) \to \phi_v(x)$ where ϕ'_{dom} is obtained from ϕ_{dom} by replacing every predicate $a(x)$ by $\bigvee_{v\in B}(a,v)(x)$. Now thanks to (?), we can construct a DVPA that recognizes $\mathcal{L} = \mathcal{L}(\psi)$. Finally, we transform it into a VPT by replacing transitions reading (a, v) into transitions reading a and producing v. Since T realizes a function, we obtain a functional VPT, concluding the proof. ☐

PROPOSITION 3. *Deciding the single use property on a* 2VPT *is* EXPTIME-C.

Proof. We prove that this problem is equivalent to deciding the emptiness of a D2VPA, which concludes the proof thanks to Corollary 1.

Let us first remark that if A is single use, it is single use with respect to the set of all states that can produce a non empty word. Let A be a 2VPT on an input alphabet Σ. We define a 2VPA B on the marked alphabet $\Sigma \times \{0, 1\}$ as follows. The transducer B first reads its input to ensure that there is exactly one position with a 1. It then nondeterministically chooses a producing state q and simulates A on its input. It finally accepts if it visits the marked position twice in state q. Then A is single use if, and only if, the language recognised by B is empty. Since the size of B is linear in the size of A, deciding the single use property is Exptime.

Conversely, let B be a 2VPA. We construct a 2VPT A as follows. All existing transitions of B are set to produce the empty word, and every accepting transition is replaced by a back and forth move on the last position, producing a single letter. Then the producing transitions can only be fired in A if there is a run of B that fires an accepting transition. If it is the case, then the corresponding run on A will visit the state q twice in the last position while producing non empty words. Thus the language recognised by B is empty if, and only if, A is single-use. As the size of A is linear in the size of B, we get the Exptime-hardness of the single use problem. ☐

LEMMA 3. *There is an MSO-definable nested word to word transduction f which is not definable by strongly single-use* $\mathsf{D2VPT}^{\mathsf{LA}}$.

Proof. We explicit a transformation that is definable by an MSO[nw2w] transduction but not by a strongly single-use $\mathsf{D2VPT}^{\mathsf{LA}}$.

Consider an alphabet Σ with some special letters c and r from Σ_c and Σ_r respectively. We define the transformation f which associates to a word $w_0cw_1cw_2 \dots w_{n-1}cw_nrw'_{n-1} \dots w'_2rw'_1rw_0$ where all w_i, w'_i are non empty nested words and do not any contain c, for $0 \leq i \leq n$, the word $w_0w'_0w_1w'_1 \dots w_{n-1}w'_{n-1}w_n$. Its

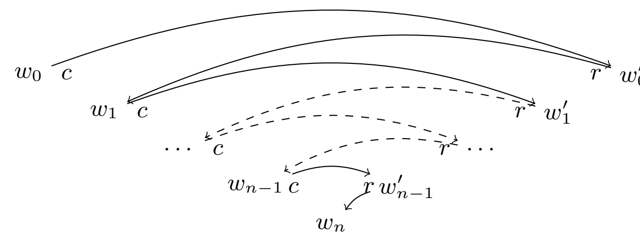

Figure 5. The transformation f alterns n times between positions left and right of w_n. Thus it has to read w_n at least $2n$ times.

domain is then the set of nested words where any c is matched by an r, and all letters c appear successively nested on a given branch. The transformation is illustrated in Figure 5.

Before giving the MSO[nw2w] that defines f, we explain how it is not definable by a strongly single use $\mathsf{D2VPT}^{\mathsf{LA}}$. As the w_i and w'_i are unbounded, they cannot be guessed by look-around. Thus a machine realizing it has to visit these subwords in the order they are output. But each walk from w_i to w'_i has to cross w_n. Thus w_n is read at least $2n$ times. As n is not bounded, f cannot be realized by a strongly single use $\mathsf{D2VPT}^{\mathsf{LA}}$.

Now we define a MSO[nw2w] T that realizes f. In order to do that, we define a binary predicate $H(x, y)$ which holds is x and y are call or return positions of a same hedge. Let $H_{tc}(X)$ be defined by the formula:

$$\forall x \in X\ \Sigma_c(x) \to \big(\forall y\ M(x,y) \vee (S(y,x) \wedge \Sigma_r(y)) \to y \in X\big)$$
$$\wedge\ \Sigma_r(x) \to \big(\forall y\ M(y,x) \vee (S(x,y) \wedge \Sigma_c(y)) \to y \in X\big)$$

with $\Sigma_l(x) = \bigvee_{\sigma\in\Sigma_l} \sigma(x)$ for $l = c, r$. Then a set X satisfies $H_{tc}(X)$ if, and only if, it is closed by the relation *belong to the same hedge*. We then simply set $H(x, y) = \forall X\ x \in X \wedge H_{tc}(X) \to y \in X$. We also define the parent relation $P(x, y) = \exists z\ H(x, z) \wedge (\forall z'\ H(z, z') \to z \leq z') \wedge S(y, z)$ which holds if y is the call corresponding to the parent of x.

We can now define the domain formula $\phi_{dom} = \forall x\ c(x) \to (\forall y\ (M(x,y) \to r(y)) \wedge (P(x,y) \to c(y)) \wedge (H(x,y) \wedge c(y) \to x = y)$ stating exactly what was mentioned earlier. The transducer T uses 1 copy, the position formula $\phi_{pos}(x) = \neg(c(x) \vee (\exists y\ M(y,x) \wedge c(y)))$ simply erases the c labeled positions and their matching, the labeling formulas simply maintain the labels, and finally the successor formula $\phi_S(x, y)$ is defined by:

$$\exists z\ S(x,z)\ \wedge \neg(c(z) \vee (\exists w\ M(w,z) \wedge c(w))) \wedge y = z)$$
$$\vee\ \big(c(z) \wedge \exists z'\ M(z,z') \wedge S(z',y)\big)$$
$$\vee\ \big(\exists z', z''\ M(z',z) \wedge c(z') \wedge Next_c(z',z'') \wedge S(z'',y)\big)$$

where $Next_c(x, y) = x < y \wedge c(y) \wedge \forall z\ x < z < y \to \neg c(z)$. ☐

THEOREM 7. *Let f be a transduction from nested words to words. Then f is MSO-definable iff it is definable by a (look-around)* $\mathsf{D2VPT}_{\mathsf{su}}$*, i.e.,*

$$\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} = \mathsf{D2VPT}_{\mathsf{su}}.$$

Proof. We prove the equivalence $\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$.

Proof overview This result is based on several results of (Courcelle and Engelfriet 2012), on the class of deterministic tree-to-word walking transducers (DTWT), possibly augmented with visibly pushdown stack (then denoted DPTWT) and a regular look-around ability (denoted by an exponent la), and possibly restricted to linear-size increase the class of linear-size increase transductions (denoted by subscript lsi), or to strongly single-use (denoted by subscript ssu). We will define the most general model formally in the sequel.

Let us also denote by $\mathsf{MSO}[\mathsf{b2w}]$ the class of MSO-definable transductions from (ranked) trees to words. Then, it is shown in (Courcelle and Engelfriet 2012) that

$$\mathsf{MSO}[\mathsf{b2w}] = \mathrm{DTWT}^{la} = \mathrm{DPTWT}^{la}_{lsi}$$

The inclusion $\mathsf{MSO}[\mathsf{nw2w}] \subseteq \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ is proved using the equality $\mathsf{MSO}[\mathsf{b2w}] = \mathrm{DTWT}^{la}$. Due to determinism, DTWT^{la} are always strongly single-use (otherwise they could be stuck in a loop), i.e., $\mathrm{DTWT}^{la} = \mathrm{DTWT}^{la}_{ssu}$ (see (Courcelle and Engelfriet 2012), in which it is just called single-use). Using a first-child next-sibling encoding of nested words w into binary trees $\mathsf{fcns}(w)$, we have $\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{MSO}[\mathsf{b2w}] \circ \mathsf{fcns}$, and therefore $\mathsf{MSO}[\mathsf{nw2w}] = \mathrm{DTWT}^{la}_{ssu} \circ \mathsf{fcns}$. Then, we show that $\mathrm{DTWT}^{la}_{ssu} \circ \mathsf{fcns} \subseteq \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ by simulating DTWT^{la}_{ssu} that runs on fcns encoding of nested words by $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$. In particular, when simulating tree walking moves, one do not preserve the strong single-use restriction, but the resulting $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ is single-use.

To show inclusion $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} \subseteq \mathsf{MSO}[\mathsf{nw2w}]$, we use the equality $\mathsf{MSO}[\mathsf{b2w}] = \mathrm{DPTWT}^{la}_{lsi}$. Using fcns encoding, we get that $\mathsf{MSO}[\mathsf{nw2w}] = \mathrm{DPTWT}^{la}_{lsi} \circ \mathsf{fcns} = (\mathrm{DPTWT}^{la} \circ \mathsf{fcns}) \cap \mathrm{LSI}$, where LSI denotes the class of linear-size increase transductions. Then, we establish the inclusion $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} \subseteq (\mathrm{DPTWT}^{la} \circ \mathsf{fcns}) \cap LSI$. The single-use restriction of $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ ensures that they define only transductions in LSI. Then, a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ can be simulated by a DPTWT^{la} running on fcns encodings of nested words. Due to the encoding, pushdown moves of the D2VPT are simulated by pushdown moves to the 1st child by the $DPTWT$ and the look-around are translated in a straightforward fashion. The resulting DPTWT does not use the stack while moving to 2nd children.

As a matter of fact, putting things altogether, our result also shows that DPTWT^{la} could be strengthen when they run on binary trees, to the following pushdown behaviour, while retaining MSO-expressiveness: they only need to push a symbol when moving to left children, and not when moving to right children.

To summarize, we show the following chain of inclusions:

$$\begin{array}{ccc} \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} & \subseteq (1) & (\mathrm{DPTWT}^{la} \circ \mathsf{fcns}) \cap \mathrm{LSI} \\ \cup\!\!\mid (5) & & \cap\!\!\mid (2) \\ \mathrm{DTWT}^{la}_{su} \circ \mathsf{fcns} & \supseteq (4) & \mathsf{MSO}[\mathsf{b2w}] \circ \mathsf{fcns} = \mathsf{MSO}[\mathsf{nw2w}]\ (3) \end{array}$$

where (1) and (5) are shown in this paper, (3) is immediate, (2) and (4) come from (Courcelle and Engelfriet 2012). We now proceed to the detailed proof.

Tree Walking Transducers Let Λ be a ranked alphabet of binary and constant symbols (i.e.Λ is partitioned into Λ_2 and Λ_0). A tree t of Λ is a term inductively defined by $t ::= f(t,t) \mid a$, where $f \in \Lambda_2$ and $a \in \Lambda_0$. We denote by Trees_Λ the set of trees over Λ. The set of nodes, denoted by N_t of a tree $t \in \mathrm{Trees}_\Lambda$, is a prefix-closed subset of $\{1,2\}^*$ inductively defined as $N_a = \{\epsilon\}$ and $N_{f(t_1,t_2)} = \{\epsilon\} \cup \{i.\pi \mid \pi \in N_{t_i}, i \in \{1,2\}\}$. For a node $n \in N_t$, we denote by $t(n)$ the label of node n in t. Let Σ be a finite (unranked) alphabet. A tree to word transduction is a function from Trees_Λ into Σ^*.

Let us explain informally the different classes of tree walking transducers we consider in this proof. A deterministic tree to word walking transducer walks through the edges of a binary tree (starting from the root node), and writes a word from left to right on some output tape. In a state q of a tree t and at a node n, depending on the label of n, and the state q, the transducer can move either to the father of n (if it exists, otherwise the run rejects), the first or second child of n (if it exists, otherwise the run rejects), change its internal state to a new state, and produces some partial word on the output. It can also decide to stop the walk by going to a stopping state q_s.

Such transducers can be augmented with look-around. We define look-around by an unambiguous bottom-up tree automaton. Prior to starting the computation of the tree walking transducer, the tree, if accepted by the look-around automaton, is labeled by the states of the accepting run of the automaton. Then, transitions are taken depending also on the look-around states.

Finally, walking transducers can be augmented by a (visibly) pushdown store. Initially at the root the pushdown stack contains an initial symbol γ_0, and whenever the transducer goes one step downward, it has to push one symbol on the stack. If it moves one step upward, it has to pop one symbol. At any moment, it can also read the top symbol of the stack.

Formally, a *deterministic pushdown tree to word walking transducer with look-around* from Trees_Λ to Σ^* is a tuple $T = (L, Q, q_0, q_s, \Gamma, \gamma_0, R)$ where L is an unambiguous bottom-up tree automaton[7] over a finite set of states P (the look-around automaton), Q is a finite set of states, q_0 is the initial state, q_s the stopping state, Γ is a finite stack alphabet with initial symbol γ_0, R is a transition function such that

$$R : Q \times P \times \Lambda \times \Gamma \to \Sigma^* \times (\{q_s\} \cup \Gamma \times Q \times \{1,2\} \cup Q \times \{-1\})$$

A configuration of T on a tree t is a triple $(q, n, \beta, u) \in Q \times N_t \times \Gamma^+ \times \Sigma^*$. For all trees $t \in \mathrm{Trees}_\Lambda$, if t is accepted by the look-around automaton, we define $\to_t$ a binary relation between consecutive configurations as follows: for all $q, q' \in Q$, all $n, n' \in N_t$, all $\beta, \beta' \in \Gamma^*$, all $\gamma \in \Gamma$, all $u, v \in \Sigma^*$, $(q, n, \beta\gamma, u) \to_t (q', n', \beta', uv)$ if the accepting run of L labels n by a state $p \in P$ such that $(q, p, t(n), \gamma) \in Dom(R)$ and either

- (*stopping move*) $R(q, p, t(n), \gamma) = (v, q_s)$ and $q' = q_s$, $n' = n$, $\beta' = \beta\gamma$, or
- (*downward move*) $R(q, p, t(n), \gamma) = (v, \gamma', q', i)$ for $i \in \{1, 2\}$ and $\beta' = \beta\gamma\gamma'$, $t(n) \in \Lambda_2$, and n' is the i-th child of n, or
- (*upward move*) $R(q, p, t(n), \gamma) = (v, q', -1)$ and $n \neq \epsilon$ (i.e. n is not the root node), $\beta' = \beta$, and n' is the father of n.

A run of T on a tree t is a finite sequence of configurations $c_0 c_1 \dots c_m$ such that $c_i \to_t c_{i+1}$ for all $i = 0, \dots, m-1$. It is accepting if $c_0 = (q_0, \epsilon, \gamma_0, \epsilon)$ and $c_m = (q_s, n, \beta, u)$ for some $n \in N_t$, $\beta \in \Gamma^+$, $u \in \Sigma^*$. Since R is a function and L is unambiguous, there exists at most one accepting run per input tree t, and we call u the output of t. The transduction realized by T is the set of pairs (t, u) such that t is accepted by the look-around automaton, and there exists an accepting run of T on t whose output is u. The class of deterministic pushdown tree to word walking transducers [8] is denoted by DPTWT^{la}.

[7] We refer the reader to (Comon-Lundh et al. 2007) for a definition of bottom-up tree automata

[8] We have slightly changed the definition of (Courcelle and Engelfriet 2012) to simplify our presentation, but in an equivalent way, and have specialized it to the tree-to-word setting. In (Courcelle and Engelfriet 2012), look-around are MSO-formulas on trees, with one free first-order variable, attached to the transitions of the transducer: a transition can be fired only if its look-around formula holds at the current node. It is known that such an MSO formula $\phi(x)$ is equivalent to an unambiguous bottom-up tree automaton A_ϕ (Niehren et al. 2005; Neven and Schwentick 2002) in the following sense: the automaton as a special set of selecting states S, such that on a tree t accepted by the automaton, a node n is such that $t \models \phi(n)$ iff this node is labeled by a state of S in the accepting run of the automaton on t. If $\phi_1(x), \dots, \phi_n(x)$ are the look-around formulas appearing on the transitions of the tree walking transducer, then by taking the product of the unambiguous automata A_{ϕ_i}, one obtains an unambiguous automaton A_{la}

The class DTWT^{la} denotes the class of deterministic tree to word walking transducers with look-around (without pushdown store), defined similarly as DPTWT^{la} but with a pushdown alphabet Γ that consists of one symbol $\{\gamma_0\}$ only. In that case, we can omit the pushdown symbols in the transitions, except the initial pushdown symbol γ_0 that allows to know whether the current node is the root of the tree or not. Instead of keeping the symbol γ_0 in the transitions, we use a boolean value which is true if the current node is the root, and false otherwise. Therefore, in DTWT^{la}, the transitions function has the following type:

$$R : Q \times P \times \Lambda \to \Sigma^* \times (\{q_s\} \cup Q \times \{-1, 1, 2\})$$

We say that a DTWT^{la} is *strongly single-use* if one any accepting run $c_0 \dots c_n$ on a tree, a node $n \in N_t$ is not visited twice in the same state. As a matter of fact, it turns out that DTWT^{la} are always strongly single-use (Courcelle and Engelfriet 2012).

$\mathsf{MSO}[\mathsf{b2w}]$ is defined similarly as $\mathsf{MSO}[\mathsf{nw2w}]$, except that MSO formulas over the signature $\{S_1(x, y), S_2(x, y), (a(x))_{a\in\Lambda}\}$ are used to define the transduction, where $S_i(x, y)$ holds if y is the i-th child of x. It is easy to show that $\mathsf{MSO}[\mathsf{nw2w}]$ and $\mathsf{MSO}[\mathsf{b2w}]$ are equivalent, modulo suitable encodings. In particular, we will need the following equality:

$$\mathsf{MSO}[\mathsf{nw2w}] = \mathsf{MSO}[\mathsf{b2w}] \circ \mathsf{fcns} \qquad (1)$$

where fcns encodes nested words into binary trees. Let us describe this encoding formally: Let $\Sigma = \Sigma_c \uplus \Sigma_r$ be a structured alphabet. We define the binary alphabet $\Lambda_2 = \Sigma_c \times \Sigma_r$, and the unary alphabet $\Lambda_0 = \{\perp\}$. We define fcns the encoding of nested words to binary trees inductively as follows, for all $w, w' \in \mathcal{N}([)\Sigma]$, $c \in \Sigma_c, r \in \Sigma_r$:

$$\mathsf{fcns}(cwrw') = (c, r)(\mathsf{fcns}(w), \mathsf{fcns}(w')) \qquad \mathsf{fcns}(\epsilon) = \perp$$

We are now equipped to show the following two inclusions, which will prove the desired result:

$$(2)\ \ \mathrm{DTWT}^{la} \circ \mathsf{fcns} \subseteq \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$$

$$(3)\ \ \mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}} \subseteq (\mathrm{DPTWT}^{la} \circ \mathsf{fcns}) \cap LSI$$

Proof of inclusion (2) Let $T = (L, Q, q_0, q_s, R)$ be DTWT^{la}. We construct a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ T' such that $[\![T]\!] \circ \mathsf{fcns} = [\![T']\!]$.

Transitions of T are simulated only on call symbols, i.e. the moves of T are simulated by moves in T' between call symbols. For instance, let $c_1c_2r_2c_3c_4r_4r_3c_5r_5r_1$ be a nested word, whose fcns encoding is the tree

$$(c_1, r_1)((c_2, r_2)(\perp, (c_3, r_3)((c_4, r_4), (c_5, r_5)(\perp, \perp))))$$

If T moves from (c_3, r_3) to its father (c_2, r_2), then T' will move from c_3 to c_2. If T moves from (c_3, r_3) to its first child (c_4, r_4), then T' will move from c_3 to c_4. If T moves from (c_3, r_3) to its second-child (c_5, r_5), then T' will move from c_3 to c_5.

Therefore, one-step moves of T in a binary encoding t_w of a nested word w are simulated by sequences of moves of T' in w. It is easy to see however that those sequences of moves can be achieved by a $\mathsf{D2VPT}$. The trivial case is a first-child move: it suffices to move one-step to the right in w. For the second-child move, T' has to move to the next call symbol at the right of current one, at the same nesting depth: this is done by pushing one special symbol γ_2 when reading the first call, and moving to the right, until γ_2 is popped. For a father move, it suffices to move left: if the previous symbol is a call, then T' has arrived to the call corresponding to the father node. Otherwise, the left symbol is a return symbol: again, a special symbol γ_f is pushed, to know, when T' walks left, whenever it is at the same depth as the initial call symbol. If after popping γ_f, a call symbol is read again, then it corresponds to the father. Note that these walks do not produce anything on the output.

The look-around L of T is transformed into a look-around of T' such that, if L labels a tree node labelled (c, r) by a state p, then T' will label the call symbol c by the state p, as well as the call symbol r. It is possible, since bottom-up tree automata and visibly pushdown automata correspond modulo first-child next-sibling encodings, while preserving unambiguity (Alur and Madhusudan 2009). Therefore, if P is the set of states of L, then the set of states of the look-around automaton of T' is $P \times \Sigma_r$.

Then, a transition $(q, p, (c, r), u, (q', d))$ where $d \in \{-1, 1, 2\}$ is simulated by T' by a sequence of transitions (depending on whether $d = -1$, $d = 1$ or $d = 2$) that starts in state q (with look-around state (p, r)) and ends in state q', and performs moves as explained before.

There is a last additional technical difficulty: fcns encodings contain the symbol $\perp$, unlike the encoded nested words. Therefore, T' may move to $\perp$, while T cannot. Moves to nodes labeled $\perp$ can be simulated easily by T' by adding ϵ-transitions, which can in turn be removed while preserving determinism. It is not difficult but unnecessarily technical.

Finally, since T is necessarily single-use (due to non-determinism), T' is also single-use (the extra states added to simulate one-step moves of T by several moves of T' may be used several times at the same tree node, but the transitions fired from those states are ϵ-producing).

Proof of inclusion (3) Due to the single-use restriction, any $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ transduction is LSI. It remains to show that a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ can be simulated by a DPTWT^{la}. By using again the correspondence between (unambiguous) visibly pushdown automata and (unambiguous) bottom-up tree automata, one can simulate their look-arounds. Since DPTWT^{la} have the ability to push stack symbols in both directions (first-child or second-child), it is not difficult to construct a DPTWT^{la} that simulates a $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$. As a matter of fact, pushing symbols when moving to the second-child is not necessary to simulate $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$: indeed, a second-child in a fcns encoding correspond to a next-sibling in the nested word, and $\mathsf{D2VPT}^{\mathsf{LA}}_{\mathsf{su}}$ do not use their stack for processing symbols that are at the same depth (they do not push "horizontally"). ☐

A.4 Inclusion into streaming transducers and hedge-to-string transducers

THEOREM 9. $\mathsf{D2VPT} \subsetneq \mathsf{STST}$ *and* $\mathsf{D2VPT} \subsetneq \mathsf{dH2S}^{\mathsf{LA}}$

Proof. The proofs of these two inclusions share a same intermediate formal description of transformation. It turns out that this representation will be an extension of the finite transition algebra $\mathbb{T}_A$ for some $\mathsf{D2VPA}$ A.

We recall that elements from the algebra $\mathbb{T}_A$ are binary relations over $Q \times \mathbb{D}$ where Q is the set of states of A and can thus be depicted as Boolean square matrices $\mathcal{M}_{\mathbb{T}_A}$ over $Q \times \mathbb{D}$. Hence, the morphism $\mu_{\mathbb{T}_A}$ associates with each word w from $\mathcal{N}(\Sigma)$ a matrix from $\mathcal{M}_{\mathbb{T}_A}$ such that $\mu_{\mathbb{T}_A}(w)((p_1, d_1), (p_2, d_2))$ is true if there exists a run on w from (p_1, d_1) to (p_2, d_2) in A.

One may extend this notion to transducers as follows. For a $\mathsf{D2VPT}$ A, we consider square matrices $\mathcal{N}_A$ over $Q \times \mathbb{D}$ whose

such that, on a tree t accepted by A_{la}, the state label of a node n of t in the accepting run of A_{la} contains enough information to decide which look-arounds $\phi_i(x)$ hold at that node or not.

Another modification of the definition of (Courcelle and Engelfriet 2012) is that we do not have 0-moves, i.e. transitions that stay at the same node. This is circumvented by adding the possibility of producing several symbols by a single transition, in contrast to (Courcelle and Engelfriet 2012).

Finally, our transducers produce words over Σ^*, while in (Courcelle and Engelfriet 2012), they produce unary trees, i.e. a sequence of unary symbols followed by a constant symbol. As a consequence, when a constant is produced, the transducers of (Courcelle and Engelfriet 2012) stop. In our definition, we rather have added a stopping state.

values range over subsets of Δ^*. One can define a mapping μ from $\mathcal{N}(\Sigma)$ to $\mathcal{N}_A$ such that for all words w from $\mathcal{N}(\Sigma)$, $\mu(w)$ is a matrix N_A satisfying that $N_A((p_1,d_1),(p_2,d_2))$ is equal to L if for each v in L, there exists a run on w from (p_1,d_1) to (p_2,d_2) in A producing v. Note in fact that A being deterministic, L is either a singleton or the empty set. One can actually prove that one can define an (infinite) algebra $\mathbb{N}_A = (\mathcal{N}_A, .^{\mathbb{N}_A}, (f_{c,r})_{(c,r)\in\Sigma_c\times\Sigma_r}, \epsilon^{\mathbb{N}_A})$ such that $.^{\mathbb{N}_A}$ is associative and $\epsilon^{\mathbb{N}_A}$ is its neutral element. Moreover, the considered mapping μ turns out to $\mu^{\mathbb{N}_A}$ be the canonical morphism from $\mathbb{W}$ to $\mathbb{N}_A$. It is worth noticing that for all (p,d), (p',d'), $\mu_{\mathbb{T}_A}(w)((p,d),(p',d'))$ is false iff $\mu_{\mathbb{N}_A}(w)((p,d),(p',d')) \neq \emptyset$.

The operations $\epsilon^{\mathbb{N}_A}$, $.^{\mathbb{N}_A}$ and $f_{c,r}^{\mathbb{N}_A}$ can be represented as matrices as well. To do so, let us first consider the two sets of symbols $\Xi^\alpha = \{x_\alpha^{(p,d),(p',d')} \mid (p,d),(p',d') \in Q\times\mathbb{D}\}$ for $\alpha\in\{1,2\}$. Then, let $\hat{\mathcal{N}}_A$ be the set of matrices defined over $Q\times\mathbb{D}$ such that for $\hat{N}_A$ in $\hat{\mathcal{N}}_A$, for all (p_1,d_1), (p_2,d_2), $\hat{N}_A((p_1,d_1),(p_2,d_2))$ is either the empty set $\varnothing$ or a singleton set included into the set of words $(\Delta\cup\Xi^1\cup\Xi^2)^*$. Moreover, for $\epsilon^{\mathbb{N}_A}$, the matrix is precisely the one with $\{\epsilon\}$ on its main diagonal and $\varnothing$ everywhere else. For $f_{c,r}^{\mathbb{N}_A}$, the elements of the matrix are actually included into $(\Delta\cup\Xi^1)^*$. The operation $.^{\mathbb{N}_A}$ deals with two matrices $\hat{N}_A^1$ and $\hat{N}_A^2$ and produces the matrix $\hat{N}_A''$ satisfying that for all (p_1,d_1), (p_2,d_2), $\hat{N}_A''((p_1,d_1),(p_2,d_2))$ is obtained from $\hat{N}_A^{'.'}$ the matrix of $.^{\mathbb{N}_A}$ by replacing everywhere in $\hat{N}_A^{'.'}((p_1,d_1),(p_2,d_2))$ the symbol $x_\alpha^{(p,d),(p',d')}$ by $\hat{N}_A^\alpha((p,d),(p',d'))$ for $\alpha\in\{1,2\}$. The application for $f_{c,r}^{\mathbb{N}_A}$ represented by some matrix $\hat{N}_A^{f_{c,r}}$ is similar with a single matrix as operand.

The matrices $\hat{N}_A^{'.'}$ and $\hat{N}_A^{f_{c,r}}$ can be defined by means of expressions similar to the ones defining recursively the equivalence classes of traversals. Hence, these matrices are defined by means of unions, concatenations and Kleene star entrywise; the fact that entries of the matrices contain at most singletons and that Kleene star can be expressed as finite concatenations relies on the determinism of A.

From the infinite algebra $\mathbb{N}_A$ and more specifically the matrices representation of the operators $\hat{N}_A^{'.'}$ and $\hat{N}_A^{f_{c,r}}$ from this algebra, we are going now to build a streaming tree-to-string transducer on the one side and a dH2S on this other side.

For streaming tree-to-string transducers, the idea is to define from a D2VPT A such a machine S_A to simulate the computation of $\mu^{\mathbb{N}_A}(w)$ for any word w or more precisely to compute the value associated to $((q_I,\rightarrow),(q_f,\rightarrow))$ in this matrix, q_I being the initial state of the D2VPT A and q_f the (final) state reached by A after reading w from q_I.

We recall that we can define from A the finite algebra $\mathbb{T}_A$ whose domain is Trav_A and we consider $\Xi^\alpha = \{x_\alpha^{(p,d),(p',d')} \mid (p,d),(p',d')\in Q\times\mathbb{D}\}$ for $\alpha\in\{1,2\}$.

For a D2VPA A, the STST S_A is defined by $(\mathrm{Trav}_A, \epsilon^{\mathbb{T}_A}, \Sigma_c\times \mathrm{Trav}_A, \Xi^1, \delta_{S_A}, \mu_F^{S_A})$ where

- $\delta_{S_A}^{push}(m^{\mathbb{T}_A}, c) = (\epsilon^{\mathbb{T}_A}, (c, m^{\mathbb{T}_A}), \nu_{Id})$
- $\delta_{S_A}^{pop} = (m'^{\mathbb{T}_A}, r, (c, m^{\mathbb{T}_A})) = (m^{\mathbb{T}_A}.^{\mathbb{T}_A} f_{c,r}^{\mathbb{T}_A}(m'^{\mathbb{T}_A}), \nu_{c,r})$

where ν_{Id} is the identity on Ξ^1 and $\nu_{c,r}$ associate with each $x^{(p,d),(p',d')}$ the expression from $N((p,d),(p',d'))$ where the matrix N is defined as $N_{Id}^1.^{\mathbb{N}_A} f_{c,r}^{\mathbb{N}_A}(N_{Id}^2)$, the matrices N_{Id}^α satisfying for all (p_1,d_1), (p_2,d_2), $N_{Id}^\alpha((p_1,d_1),(p_2,d_2)) = x_\alpha^{(p_1,d_1),(p_2,d_2)}$ (for all $\alpha\in\{1,2\}$).

Let us now consider the case of deterministic hedge-to-string transducer. We first define the bottom-up deterministic look-ahead automaton B_A as $(\mathrm{Trav}_A, \{\epsilon_A^{\mathbb{T}}\}, \delta_{B_A})$ where δ_{B_A} is the set of rules of the form $\{(m^{\mathbb{T}_A}.^{\mathbb{T}_A} f_{c,r}^{\mathbb{T}_A}(m'^{\mathbb{T}_A}), c, r, m^{\mathbb{T}_A}, m'^{\mathbb{T}_A})$ with $m^{\mathbb{T}_A}, m'^{\mathbb{T}_A}\in \mathrm{Trav}_A$.

Now, we define the dH2S H_A as follows: the set of states Q_{H_A} is $\{q_I\}\cup \mathrm{Trav}_A\times(Q\times\mathbb{D})^2$, q_I is the initial state, and the set of final states is Q_{H_A}. Now, for the transition function, we define

$$\begin{aligned}&\delta(q_I, c, r, n_1^{\mathbb{T}_A}, n_2^{\mathbb{T}_A}) =\\ &\qquad \omega[x_\alpha^{(q_1,d_1),(q_2,d_2)} \leftarrow (n_\alpha^{\mathbb{T}_A}, ((q_1,d_1),(q_2,d_2)))(x_\alpha)]\\ &\delta((m^{\mathbb{T}_A}, ((p,d),(p',d'))), c, r, m^{\mathbb{T}_A}) =\\ &\qquad \omega[x_\alpha^{(q_1,d_1),(q_2,d_2)} \leftarrow (m_\alpha^{\mathbb{T}_A}, ((q_1,d_1),(q_2,d_2)))(x_\alpha)]\end{aligned}$$

where

- for any $n_1^{\mathbb{T}_A}, n_2^{\mathbb{T}_A}$ such that $(f_{c,r}^{\mathbb{T}_A}(n_1^{\mathbb{T}_A}).^{\mathbb{T}_A} n_2^{\mathbb{T}_A})((q_I,\rightarrow),(q_f,\leftarrow))$ is true for some $q_f\in F$.
- for any $m_1^{\mathbb{T}_A}$, $m_2^{\mathbb{T}_A}$ such that $f_{c,r}^{\mathbb{T}_A}(m_1^{\mathbb{T}_A}).^{\mathbb{T}_A} m_2^{\mathbb{T}_A} = m^{\mathbb{T}_A}$ and $((p,d),(p',d'))\in m^{\mathbb{T}_A}$ and
- ω is equal to the word $N((p,d),(p',d'))$, N being the matrix $f_{c,r}^{\mathbb{N}_A}(N_{Id}^1).^{\mathbb{N}_A} N_{Id}^2$, where the matrix N_{Id}^α satisfies for all $\alpha\in\{1,2\}$, for all (p_1,d_1), (p_2,d_2), $N_{Id}^\alpha((p_1,d_1),(p_2,d_2)) = x_\alpha^{(p_1,d_1),(p_2,d_2)}$.

Let us prove now that the inclusions are strict. The transformation serving as a counter-example is the same for the two classes. We consider a transformation T over the input alphabet $\Sigma_c=\{c\}$ and $\Sigma_r=\{r\}$ and the output alphabet $\{a\}$. This transformation takes as an input words such as $(cr)^n$ for any natural n and outputs a^{2^n-1}. T is given by the dH2S with a single state q and with a universal look-ahead automaton with q' as unique state by

$$\delta(q,c,r,q') = aq(x_1)q(x_1)$$

The transformation T can also be defined by a STST with a single state q, a unique register variable X and a unique stack symbol γ. The transitions are given by

- $\delta^{push}(q,c) = (q,\gamma,\{X\mapsto aXX\})$
- $\delta^{pop}(q,r,\gamma) = (q,\{X\mapsto X\})$

The transformation T cannot be realized by some D2VPT; indeed, for such a machine with stack alphabet Γ on the shallow inputs of the domain, the possible stacks occurring in runs are either $\perp$ or the form γ for $\gamma\in\Gamma$. Hence, the possible behaviours of such D2VPT are similar to the ones of a deterministic finite state transducer. It is known that deterministic finite state transducers realize only functions that are linear-size increase; this is not the case of the transformation T. $\square$

A.5 Unranked Tree Walking Transducers

Unranked Trees Let Λ be a finite set of symbols. *Unranked trees* t over Λ are defined inductively as $t ::= a \mid a(t_1,\ldots,t_n)$, for all $a\in\Lambda$, all $n\geq 1$. Unranked trees over Λ can be identified (modulo renaming of nodes) with structures over the signature $\mathbb{U}_\Lambda$ that consists of the *first-child* predicate $fc(x,y)$ that relates a node x to its first-child y, the *next-sibling* predicate $ns(x,y)$ that relates a node x to its next-sibling y in a sequence of unranked trees, and $a(x)$, for all $a\in\Lambda$, that holds true in node x if it is labeled a. In addition, we also add a *parent* predicate $parent(x,y)$ that relates a node to its parent.

For instance, the unranked tree $a(b,c(a),c)$ is identified with the structure whose set of nodes is $\{\epsilon,1,2,3,21\}$, where the first-child predicate is $\{(\epsilon,1),(2,21)\}$, the next-sibling predicate is $\{(1,2),(2,3)\}$, the a predicate is $\{\epsilon,21\}$, the b predicate is $\{1\}$ and the c predicate is $\{2,3\}$. The parent predicate is given by $\{(1,\epsilon),(2,\epsilon),(3,\epsilon),(21,2)\}$.

Unranked Tree Walking Transducers They are defined similarly as ranked tree walking transducers, except that they move along the next-sibling and first-child predicates. They are equipped with a (visibly) pushdown store such that whenever they go down the first-child, they have to push some symbol, whenever they go up to the parent of a node, they have pop one symbol from the stack. However, when they move horizontally along next-sibling predicates, they do not touch the stack. Before applying a transition, they can test whether the current node is the root, is the first-child of some node, the last-child, or a leaf. Their move have to be consistent with the result of such a test. They are also equipped with *stay* moves that stay at the same tree node.

Formally, a *deterministic pushdown unranked tree to word walking transducer* (DPT$_u$WT) from unranked trees over Λ to Σ^* is a tuple $T = (Q, q_0, q_s, \Gamma, \gamma_0, R)$ where Q is a finite set of states, q_0 is the initial state, q_s the stopping state, Γ is a finite stack alphabet with initial symbol γ_0, R is a transition function such that

$$R : Q\times\Lambda\times\Gamma\times\{0,1\}^4 \to \Sigma^*\times(\{q_s\}\cup\Gamma\times Q\times\{\downarrow\}\cup Q\times\{\rightarrow,\leftarrow,\uparrow,\circlearrowleft\})$$

A configuration of T on a tree t with set of nodes N_t is a triple $(q, n, \beta, u) \in Q \times N_t \times \Gamma^+ \times \Sigma^*$. We define $\rightarrow_t$ a binary relation between consecutive configurations as follows: Let $n \in N_t$ labeled $a \in \Lambda$. Let $\bar{b} = (b_{fc}, b_{lc}, b_r, b_l) \in \{0,1\}^4$ such that $b_{fc} = 1$ iff n is a first-child, $b_{lc} = 1$ iff n is a last-child, $b_r = 1$ iff n is the root, $b_l = 1$ iff n is a leaf. Then, for all $q, q' \in Q$, all $n' \in N_t$, all $\beta, \beta' \in \Gamma^*$, all $\gamma \in \Gamma$, all $u, v \in \Sigma^*$, $(q, n, \beta\gamma, u) \rightarrow_t (q', n', \beta', uv)$ if

- (*stopping move*) $R(q, a, \gamma, \bar{b}) = (v, q_s)$ and $q' = q_s$, $n' = n$, $\beta' = \beta\gamma$, or
- (*downward move*) $R(q, a, \gamma, \bar{b}) = (v, \gamma', q', \downarrow)$, $\beta' = \beta\gamma\gamma'$, and $fc(n, n')$, or,
- (*upward move*) $R(q, a, \gamma, \bar{b}) = (v, q', \uparrow)$ and $\beta' = \beta$, and $parent(n, n')$, or,
- (*left sibling move*) $R(q, a, \gamma, \bar{b}) = (v, q', \leftarrow)$ and $\beta' = \beta$, and $ns(n', n)$, or,
- (*right sibling move*) $R(q, a, \gamma, \bar{b}) = (v, q', \rightarrow)$ and $\beta' = \beta$, and $ns(n, n')$.
- (*stay move*) $R(q, a, \gamma, \bar{b}) = (v, q', \circlearrowleft)$ and $\beta' = \beta$, and $n = n'$.

A run of T on an unranked tree t is a finite sequence of configurations $c_0c_1 \dots c_m$ such that $c_i \rightarrow_t c_{i+1}$ for all $i = 0, \dots, m-1$. It is accepting if $c_0 = (q_0, r, \gamma_0, \epsilon)$, where r is the root node of t, and $c_m = (q_s, n, \beta, u)$ for some node n of t, $\beta \in \Gamma^+$, and $u \in \Sigma^*$. Since R is function, there exists at most one accepting run per input tree t, and we call u the output of t. The transduction realized by T is the set of pairs (t, u) such that there exists an accepting run of T on t whose output is u.

Equivalence between* D2VPT *and DPT$_u$WT Modulo nested word linearisation of unranked trees, the two models are equivalent. Let us briefly sketch why.

Assume T is a DPT$_u$WT and let us construct an equivalent D2VPT T'. First notice that when T is positioned at some node n, its stack height is exactly the depth of node n in the tree, as well as the depth of the call and return symbols corresponding to n in the linearisation. Also note that T can always read the top symbol of the stack, while T' only reads it when it pops a symbol. This issue can be overcome by always keeping in the state of T' the top stack symbol. It remains to see how T' can simulate the moves of T and its tests (root, leaf, etc.). We assume that if T is positioned at some tree node n, then T' is positioned at the call position c_n corresponding to n in the linearisation of the input tree. Then, if T moves from n to its next-sibling n', T' has to traverse the whole linearisation of the subtree rooted at n. It can be easily done by pushing a special symbol when reading c_n forward, which is popped once the matching return position of c_n is met. Simulating previous-sibling moves is done symmetrically. Suppose now that a tree node n' is the parent of a tree node n. In the linearisation, it means that there is a (sub) nested word of the form $c_{n'}w_1c_nw_2r_nw_3r_{n'}$, where w_1, w_2, w_3 are nested words. To simulate a move of T' from n to n', T' has to move backward from c_n to $c_{n'}$, traversing w_1. Again, by using a special stack symbol when traversing w_1, T' can detect when it reads $c_{n'}$: It is the first time it does not popped the special stack symbol. To simulate a stay move, T' just move one-step forward and one-step backward.

Finally, we have to show how T' can simulate the tests (root, leaf, etc.). By using a special bottom stack symbol, T' can know when it is at the root. The other tests can easily be performed by T': For instance, to detect that T' is positioned at a call position that corresponds to a first-child, it suffices to go one-step backward and check whether the previous symbol is call.

Conversely, let T be a D2VPT whose input are assumed to be linearisations of unranked trees. To construct an equivalent DPT$_u$WT T', one again has to show how the moves of T are simulated by moves of T'.

If T moves forward by reading a call symbol c_n, then its next position can be either that of a call symbol $c_{n'}$ (which means that n' is the first-child of n), or that of return symbol r_n (which means that n is a leaf). Using a test, T can decide whether it is at a leaf or not. In the first case, it uses a stay transition and in the second case, it uses a first-child transition.

Other cases are treated similarly: For instance, if T moves forward by reading a return symbol r_n, then if the next symbol is a call symbol $c_{n'}$, it means that n' is the next-sibling of n, and if the next symbol is a return symbol $r_{n'}$, it means that n' is a parent of n. Using tests, T can decide what moves to perform, either next-sibling or parent.