A Robust Class of Languages of 2-Nested Words

Séverine Fratani
LIS, Aix-Marseille Univ, CNRS, Marseille, France

Guillaume Maurras
LIS, Aix-Marseille Univ, CNRS, Marseille, France

Pierre-Alain Reynier
LIS, Aix-Marseille Univ, CNRS, Marseille, France

Abstract

Regular nested word languages (a.k.a. visibly pushdown languages) strictly extend regular word languages, while preserving their main closure and decidability properties. Previous works have shown that considering languages of 2-nested words, *i.e.* words enriched with two matchings (a.k.a. 2-visibly pushdown languages), is not as successful: the corresponding model of automata is not closed under determinization. In this work, inspired by homomorphic representations of indexed languages, we identify a subclass of 2-nested words, which we call 2-wave words. This class strictly extends the class of nested words, while preserving its main properties. More precisely, we prove closure under determinization of the corresponding automaton model, we provide a logical characterization of the recognized languages, and show that the corresponding graphs have bounded treewidth. As a consequence, we derive important closure and decidability properties. Last, we show that the word projections of the languages we define belong to the class of linear indexed languages.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Formal languages and automata theory

Keywords and phrases Nested word, Determinization, Indexed languages

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.50

Funding Partly supported by the DeLTA ANR project (ANR-16-CE40-0007)

1 Introduction

The class of regular languages constitutes a cornerstone of theoretical computer science, thanks to its numerous closure and decidability properties. A long line of research studied extensions of this class, while preserving its robustness. Context free languages (CFL for short) constitute a very important class: they admit multiple presentations, by means of pushdown automata, context-free grammars and more, and have led to numerous applications. Unfortunately, CFL do not satisfy several important properties enjoyed by regular languages. More precisely, the corresponding automaton model, namely pushdown automata, does not admit determinization. In addition, the class of CFL is not closed under intersection nor complement, and universality, inclusion and equivalence are undecidable properties.

A simple way to patch this is by considering as input the word together with the inherent matching relation, resulting in what is known as a *nested word* [3]. Indeed, as soon as a word belongs to a CFL, one can identify a matching relation on (some of) the positions of the word, whatever the presentation of the CFL. For instance, if the CFL is given as a pushdown automaton, then this relation associates push/pop positions. Another way to define the matching relation is to use an alternative presentation of CFL given in [16], which refines the Chomsky-Schützenberger theorem. Following this work, one can show that a language L is a CFL iff there exists a regular language R, a Dyck language D_2 over two pairs of brackets, and two homomorphisms h (h is non-erasing), g such that $L = h(g^{-1}(D_2) \cap R)$. This alternative presentation also leads to a natural matching relation, induced by D_2. The model of *nested* *word automata* naturally extends finite-state automata by allowing to label edges of the matching relation with states (often called hierarchical). This model accepts the so-called class of *regular nested word languages*, allowing to recover most of the nice properties of regular languages. More precisely, nested word automata can always be determinized. The class of regular nested word languages is closed under all the boolean operations, admits an equivalent presentation by means of logic (monadic-second order logic with a binary predicate corresponding to the matching relation), and expected decidability properties (emptiness, universality, inclusion and equivalence). It is worth noticing that another way to present this class is by splitting the alphabet into call/return/internal symbols. This leads to so-called *visibly pushdown languages* [2], and the associated model of visibly pushdown automata.

Several works tried to extend the class of regular nested word languages while preserving its closure and decidability properties. In particular, a focus has been put on words with multiple matching relations. In [19], the authors consider multiple stacks with a semantical restriction of push/pop operations known as k phases. That way, they obtain decidability of the emptiness problem. However, their model of automata cannot be determinized. In [6], the authors consider visibly pushdown automata with multiple stacks, with an ordering on stacks, and prove the closure under complement of their model. Their proof does not rely on the determinization of the model: indeed, as shown in [19], this class cannot be determinized in general. This corrects a previous result published in [5], which states that the general class of 2-stack visibly pushdown automata is closed under complementation, which does not hold, as shown in [4]. The crux in their proof was the use of determinization of 2-stack visibly pushdown automata. In [4], Bollig studies 2-stack visibly pushdown automata in their unrestricted form. He proves the equivalence with the existential fragment of monadic second-order logic, but that quantifier alternation leads to an infinite hierarchy in this setting. As a corollary, the resulting class of languages is not closed under complementation. Another restriction, known as scope-bounded pushdown languages, has been introduced in [20, 21], for which the authors manage to prove that the automaton model can be determinized.

The previous survey of related works shows the difficulty in identifying a class of 2-nested words for which the corresponding automaton model can be determinized. As a consequence, the decidability results presented in these papers require ad-hoc involved proofs. Graphs of bounded treewidth constitute an alternative approach for obtaining decidability properties. Indeed, in [13], the authors show that most of the previous classes with good decidability properties actually correspond to graphs of bounded treewidth, for which MSO decidability follows from [7, 17]. Yet, determinization of nested word automata is the keystone of the nice properties of this model, and thus constitutes a highly desirable feature. In the present work, taking inspiration in indexed languages, we identify a class of 2-nested words for which automata can be determinized. Our class is incomparable with those of [19], [6] and [20, 21]. Intuitively, between two matched positions of the first matching, they bound the number of switches between matchings, while we do not. In addition, the proof of determinizability of [20, 21] is different from ours, as their proof is a kind of superviser that uses determinization of [3] as a subroutine, while ours generalizes the construction of [3] to two nestings.

Indexed languages [1] correspond to the level 2 of the infinite hierarchy of higher-order languages [14]. With numerous applications in computational linguistics, they have been much studied during the seventies and the eighties [15, 8, 9]. Homomorphic characterizations of CFL that we presented before have been extended to (linear) indexed languages in several works, including [22, 10]. One of them (see [10]) shows that L is an indexed language iff there exists a regular language R, two Dyck languages D_2 and D_k over two and k pairs of brackets respectively, and two homomorphisms h, g such that $L = h(g^{-1}(D_2) \cap R \cap D_k)$, with some additional conditions on g. This presentation allows to associate with a word $w \in L$ two matchings, induced by D_2 and D_k, which interact in a very particular way. Graphically speaking, this interaction yields kinds of *waves* (see Figure 1). When restricted to linear indexed languages, the length of these waves is upper bounded by 2, yielding the structure of 2-waves that will be of interest to us in this paper. All these notions will be formally presented in the paper. We also refer the reader to Section 7, in which we explore the relationship between our work and (linear) indexed languages.

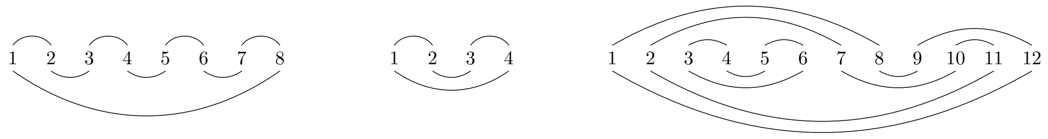


Figure 1 A 4-wave (left), a 2-wave (middle), and a combination of 2-waves (right).

In this paper, we consider 2-nested words whose matchings satisfy the structural restriction of 2-waves; we call them 2-wave words. The main result of this paper is to show that 2-nested word automata are closed under determinization on the class of 2-wave words. While determinization of nested word automata extends the well-known powerset construction with a reference state, our construction is more involved in order to be able to reconcile the labels of the different arches, and we give a detailed proof of correction. This allows us to prove the equivalence of automata with monadic-second order logic on 2-wave words, as well as with its existential fragment. This contrasts with results of [4] which show that on 2-nested words, quantifier alternation yields an infinite hierarchy. We also prove that the graphs associated with 2-wave words are MSO-definable, and have bounded treewidth. As a consequence, we obtain the following decidability results for the class of 2-wave words: satisfiability of MSO, emptiness of automata (in polynomial time), universality, inclusion and equivalence of automata (in exponential time).

In Section 2, we introduce the definitions of matchings and nested words, and provide a grammar for 2-wave words. In Section 3, we introduce the automaton model, and present in Section 4 the main result of the paper: the closure under determinization over 2-wave words. Applications to logic and decidability are presented in Sections 5 and 6. Last, a discussion on relations with (linear) indexed languages is given in Section 7.

2 Words and matchings

Words and relations. For any positive integer n, $[n] = \{1, \dots, n\}$ is the set of all positive integers ranging from 1 to n. When referencing a position in a word, integers might be called *positions* or *index*. Σ denotes a finite alphabet. The empty word is denoted ϵ, and the set of finite words on Σ is denoted Σ^*. The length of $w \in \Sigma^*$ is denoted $|w|$. Given a non-empty word $w \in \Sigma^*$, its positions are numbered from 1 to $|w|$. We say that u is a *factor* of w if there exist two words x, y such that $w = xuy$. Given an interval $I \subseteq [|w|]$, we denote by $w_{|I}$ the factor of w corresponding to positions in I. Unless specified otherwise, words and automata introduced in this paper are defined on Σ.

▶ **Definition 1.** *A* matching relation *of length $n \geq 0$ is a binary relation M on $[n]$ such that:*
1. *if $M(i,j)$ then $i < j$, i.e. M is compatible with the natural order on integers*
2. *if $M(i,j)$ and $M(k,l)$ then $\{i,j\} \cap \{k,l\} \neq \emptyset \implies i = k \wedge j = l$, i.e. any integer is related at most once by the matching*
3. *if $M(i,j)$ and $M(k,l)$ then $i < k < j \implies l < j$ i.e. a matching is non-crossing.*

Since a matching is an injective functional relation, we will often use functional notations: $M(i) = j$ or $M^{-1}(j) = i$ rather than $M(i,j)$. If $M(i,j)$, we call i a *call* position, j a *return* position and if $k \in [n]$ is neither a call nor a return position we call it an *internal* position.

If I is a subset of $[n]$ we denote by I^c the subset of call positions of I, and by I^r the subset of return positions and I^{int} the subset of internal position. We say that I is *without pending arch* (wpa) if it has no pending call nor pending return, *i.e.* $M(I^c) \cup M^{-1}(I^r) \subseteq I$. Note that this definition holds when I is an interval, but even for an arbitrary subset of $[n]$. In particular, we say a pair (I_1, I_2) of intervals is without pending arch if $I_1 \cup I_2$ is. In this case, we may also say that (I_1, I_2) is a pair of *matched intervals.*

(2-)Nested words. We first recall the classical definition of nested words:

▶ **Definition 2.** *A* nested word *on Σ is a pair $\omega = (w, M)$ where $w \in \Sigma^*$ and M is a matching of length $|w|$. We write* $\mathsf{NW}(\Sigma)$ *the set of nested words on Σ, and* $\mathsf{NWL}(\Sigma)$ *the set of languages of nested words.*

Words equipped with two matchings are called 2-nested words:

▶ **Definition 3.** *A* 2-nested word *on Σ is a triple $\omega = (w, M_1, M_2)$ where $w \in \Sigma^*$ and M_1, M_2 are matchings of length $|w|$. We write* $\mathsf{2NW}(\Sigma)$ *for the set of 2-nested words on Σ, and* $\mathsf{2NWL}(\Sigma)$ *for the set of languages of 2-nested words.*

▶ **Example 4.** An example of a nested word (resp. two examples of 2-nested words) is depicted on the left (resp. on the middle and right) of Figure 2. For 2-nested words, the matching M_1 is depicted above, while matching M_2 is depicted below.

Given a 2-nested word $\omega = (w, M_1, M_2)$ and an interval $I \subseteq [|w|]$ which is wpa w.r.t. both M_1 and M_2, we denote by $\omega_{|I}$ the 2-nested word consisting of $w_{|I}$ and of the two matchings M_1' and M_2' obtained from M_1 and M_2 by restricting them to I, and then shifting them to the interval $[|I|]$. It is routine to verify that if ω is a 2-nested word, then so is $\omega_{|I}$.

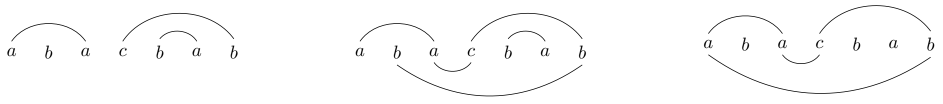


Figure 2 A nested word (left) and two 2-nested words (middle and right).

2-Waves and 2-wave words. In the sequel, we introduce the restriction of 2-nested words on which we will focus. Intuitively, wave structures are graphs obtained from the two matchings consisting of cycles alternating M_1-arches and M_2-arches whose shape evokes waves.

▶ **Definition 5.** *Let n be an integer, and (M_1, M_2) be a pair of matching relations of length n. A sequence of 4 integers $1 \leq i_1 < i_2 < i_3 < i_4 \leq n$ is a* 2-wave *if the following holds:*

- *$M_1(i_1, i_2)$ and $M_1(i_3, i_4)$ (top arches),*
- *$M_2(i_2, i_3)$ (bottom arch), and $M_2(i_1, i_4)$ (support arch)*

A pair (M_1, M_2) is a 2-wave structure *if any arch in $M_1 \cup M_2$ belongs to a 2-wave.*

▶ Remark 6. One could allow 2-wave structures to admit 1-waves, *i.e.* pairs of indices (i_1, i_2) with $i_1 < i_2$, $M_1(i_1, i_2)$ and $M_2(i_1, i_2)$. All our results would also hold for this generalization. However, in order to simplify the presentation of the paper, we do not consider them in this extended abstract.

▶ **Definition 7.** *A 2-wave word is a 2-nested word* $\omega = (w, M_1, M_2)$ *such that* (M_1, M_2) *is a 2-wave structure. We denote by* $\mathsf{WW}_2(\Sigma)$ *the set of 2-wave words over the alphabet* Σ.

▶ **Example 8.** Examples of waves are given on Figure 1. Let us consider the 2-nested words depicted on Figure 2. The one on the middle is not a 2-wave word (the upper arch $(5, 6)$ does not belong to a 2-wave), while the one on the right is.

Grammar. In order to proceed with structural induction, we present an inductive presentation of 2-wave words based on multiple context-free grammars (MCFG for short [18]). To this end, we turn to 2-visibly pushdown languages: the alphabet Σ is duplicated into five copies Σ^c_c, Σ^c_r, Σ^r_c, Σ^r_r and $\Sigma^{\mathrm{int}}_{\mathrm{int}}$, whose disjoint is denoted $\tilde{\Sigma}$. This way, the two matchings are encoded in the types of the symbols: the upper index is for the first and the lower index for the second matching relation. More formally, given $\omega \in \mathsf{WW}_2(\Sigma)$, we denote by $\tilde{\omega} \in \tilde{\Sigma}^*$ its visibly pushdown version.

In MCFG, non-terminals allow to express tuples of words. We present a grammar with two non-terminals W and H which represent respectively words (denoted $w \in \tilde{\Sigma}^*$), and pairs of words (denoted $(x, y) \in \tilde{\Sigma}^* \times \tilde{\Sigma}^*$). The grammar is defined by the following rules:

$$\begin{array}{lll} \mathsf{W} \ni w & ::= & \epsilon \mid i \mid w_1 w_2 \mid xwy \\ \mathsf{H} \ni (x, y) & ::= & (\epsilon, \epsilon) \mid (x_1 x_2, y_2 y_1) \mid (w_1 x w_1', w_2 y w_2') \mid (axb, cyd) \end{array}$$

where $i \in \Sigma^{\mathrm{int}}_{\mathrm{int}}$ and $(a, b, c, d) \in \Sigma^c_c \times \Sigma^r_c \times \Sigma^c_r \times \Sigma^r_r$.

▶ **Lemma 9.** $L(\mathsf{W}) = \{\tilde{\omega} \in \tilde{\Sigma}^* \mid \omega \in \mathsf{WW}_2(\Sigma)\}$

Proof sketch. We give some hints on how to show the right to left implication. Let $\omega = (w, M_1, M_2) \in \mathsf{WW}_2(\Sigma)$. We show, by induction on $n \le |w|$, the following properties:

- Let $I \subseteq [|w|]$ such that $|I| = n$ and I is wpa, then $\tilde{\omega}_{|I} \in L(\mathsf{W})$.
- Let $I_1, I_2 \subseteq [|w|]$ such that $|I_1| + |I_2| = n$ and (I_1, I_2) is wpa, then $(\tilde{\omega}_{|I_1}, \tilde{\omega}_{|I_2}) \in L(\mathsf{H})$.

The proof decomposes the 2-wave, by distinguishing cases according to the structure of the two matchings. One can then verify that in all cases, one can produce the words using one of the rules of the grammar. ◀

3 2-Nested Word Automata

Nested word automata have been introduced in [3] as an extension of finite-state automata intended to recognize nested words. They label arches of the matching relation with so-called *hierarchical* states: if the position corresponds to a call (resp. a return), then the automaton "outputs" (resp. "receives") the hierarchical state used to label the arch, hence we place it after the input letter (resp. before). This corresponds to push/pop operations performed by a (visibly) pushdown automaton. The extension of this model to multiple matchings is natural, and has already been considered in [4, 6]: with two matchings, automata label edges of both matchings with hierarchical states.

We first introduce some notations. Let us assume that two matching relations M_1, M_2 of length n are given. Then, each index $i \in [n]$ can be labelled in 9 different ways depending on it's call, return and internal status with regard to matchings M_1 and M_2. We say that a position i is a *call-return* if it is a call w.r.t. M_1, and a return w.r.t. M_2. We extend this convention to other possible types of positions (*call-call*, *return-call*, *call-internal*...). Let $I \subseteq [n]$ and $x, y \in \{c, r, \mathrm{int}\}$. Following our graphical representation of 2-nested words, in which M_1 is depicted above the word, and M_2 is depicted below, we denote by I^x_y the subset

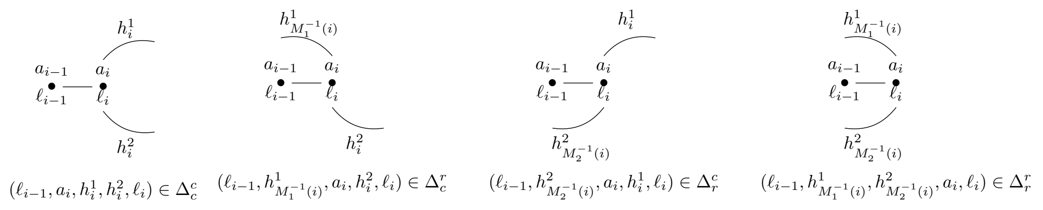

h_i^1
a_{i-1} a_i
ℓ_{i-1} ℓ_i
h_i^2
$(\ell_{i-1}, a_i, h_i^1, h_i^2, \ell_i) \in \Delta_c^c$
$h^1_{M_1^{-1}(i)}$
$(\ell_{i-1}, h^1_{M_1^{-1}(i)}, a_i, h_i^2, \ell_i) \in \Delta_c^r$
$h^2_{M_2^{-1}(i)}$
$(\ell_{i-1}, h^2_{M_2^{-1}(i)}, a_i, h_i^1, \ell_i) \in \Delta_r^c$
$(\ell_{i-1}, h^1_{M_1^{-1}(i)}, h^2_{M_2^{-1}(i)}, a_i, \ell_i) \in \Delta_r^r$

Figure 3 Examples of transition steps.

of I with x status on M_1, and y status on M_2. For instance, given a position $i \in I$, we have $i \in I_r^c$ if i is a call w.r.t. M_1 and a return w.r.t. M_2. We also introduce the following shortcuts: for $x \in \{c, r\}$, $I^x = I_c^x \cup I_r^x \cup I_{\text{int}}^x$ and $I_x = I_x^c \cup I_x^r \cup I_x^{\text{int}}$. For instance, I^c (resp. I_c) denotes the set of positions in I which are a call w.r.t. M_1 (resp. M_2).

▶ **Definition 10.** *A* 2-nested word automaton *is a tuple $A = (Q, Q_0, Q_f, P, \Sigma, \Delta)$ where :*

- *Q is a finite set of states and $Q_0, Q_f \subseteq Q$ are respectively the initial and final states*
- *P is a set of hierarchical states*
- *$\Delta = (\Delta_y^x)_{x,y \in \{c,r,int\}}$ is a set of transitions : $\Delta_y^x \subseteq Q \times P^{\mathsf{in}_{x,y}} \times \Sigma \times P^{\mathsf{out}_{x,y}} \times Q$, where $\mathsf{in}_{x,y}$ (resp. $\mathsf{out}_{x,y}$) is the number of r (resp. c) in $\{x, y\}$. For instance, $\mathsf{in}_{c,r} = \mathsf{in}_{r,c} = 1$ and $\mathsf{out}_{c,c} = 2$.*

▶ Remark 11. In the previous definition, elements in $P^{\mathsf{in}_{x,y}}$ correspond to hierarchical states that label closing arches, which can be interpreted as popped symbols, while elements in $P^{\mathsf{out}_{x,y}}$ correspond to hierarchical states that label opening arches, which can be interpreted as pushed symbols. Elements of Q are called *linear* states, as they follow the edges of the linear order, in contrast to hierarchical states.

▶ **Definition 12** (Run/Language of a 2NWA). *Let $\omega = (a_1 \ldots a_n, M_1, M_2) \in$ 2NW and A be a* 2NWA. *Let $\ell = (\ell_i)_{i \in [\![0,n]\!]}$ be a sequence of states, $h^1 = (h_i^1)_{i \in [n]^c}$ and $h^2 = (h_i^2)_{i \in [n]_c}$ be two sequences of elements of P. For all $i \in [n]$, we write $run_i^A(\omega, \ell, h^1, h^2)$ if one of the following cases holds: (the first four cases are illustrated on Figure 3)*

- **call-call:** *$i \in [n]_c^c$, and $(\ell_{i-1}, a_i, h_i^1, h_i^2, \ell_i) \in \Delta_c^c$*
- **return-call:** *$i \in [n]_c^r$, and $(\ell_{i-1}, h^1_{M_1^{-1}(i)}, a_i, h_i^2, \ell_i) \in \Delta_c^r$*
- **call-return:** *$i \in [n]_r^c$, and $(\ell_{i-1}, h^2_{M_2^{-1}(i)}, a_i, h_i^1, \ell_i) \in \Delta_r^c$*
- **return-return:** *$i \in [n]_r^r$, and $(\ell_{i-1}, h^1_{M_1^{-1}(i)}, h^2_{M_2^{-1}(i)}, a_i, \ell_i) \in \Delta_r^r$*
- **call-internal:** *$i \in [n]_{int}^c$, and $(\ell_{i-1}, a_i, h_i^1, \ell_i) \in \Delta_{int}^c$*
- **internal-call:** *$i \in [n]_c^{int}$, and $(\ell_{i-1}, a_i, h_i^2, \ell_i) \in \Delta_c^{int}$*
- **return-internal:** *$i \in [n]_{int}^r$, and $(\ell_{i-1}, h^1_{M_1^{-1}(i)}, a_i, \ell_i) \in \Delta_{int}^r$*
- **internal-return:** *$i \in [n]_r^{int}$, and $(\ell_{i-1}, h^2_{M_2^{-1}(i)}, a_i, \ell_i) \in \Delta_r^{int}$*
- **internal-internal:** *$i \in [n]_{int}^{int}$, and $(\ell_{i-1}, a_i, \ell_i) \in \Delta_{int}^{int}$*

If for all $i \in [n]$, $run_i^A(\omega, \ell, h^1, h^2)$ holds, the triple (ℓ, h^1, h^2) is said to be a run *of A* over *ω; it is an* accepting run *if $\ell_0 \in Q_0$ and $\ell_n \in Q_f$.*

We will write $q \xrightarrow[A]{\omega} q'$ when there exists a triple (ℓ, h^1, h^2) which is a run of A on ω, and whose first (resp. last) element of ℓ is equal to q (resp. q'). If we denote by n the length of ω, and if $I \subseteq [n]$ is an interval wpa, then we write $q \xrightarrow[A]{\omega, I} q'$ as a shortcut for $q \xrightarrow[A]{\omega_{|I}} q'$.

A 2-nested word is accepted by *A if it admits an accepting run. The set of all 2-nested words accepted by A is denoted $L(A)$, and a language of 2-nested words is called* regular *if it is accepted by a 2-nested word automaton. In the sequel, we will also be interested in restricting the language of a* 2NWA *to 2-wave words. Hence, we denote by $L_{\mathsf{WW}_2}(A)$ the set of 2-wave words accepted by A,* i.e. *$L(A) \cap \mathsf{WW}_2(\Sigma)$.*

▶ **Example 13.** Let $A_{ex} = (Q, Q_0, Q_f, P, \{a, b, c, d\}, \Delta)$, with $Q = \{q_a, q_b, q_c, q_d\}$, $Q_0 = \{q_a\}$, $Q_f = \{q_d\}$, $P = Q$ and Δ defined as follows (we only give non-empty transition sets):

$\Delta^c_c = \{(q_a, a, q_a, q_a, q_a)\}$
$\Delta^r_c = \{(q_x, q_a, b, q_b, q_b) \mid x \in \{a, b\}\}$
$\Delta^c_r = \{(q_x, q_b, c, q_c, q_c) \mid x \in \{b, c\}\}$
$\Delta^r_r = \{(q_x, q_c, q_a, d, q_d) \mid x \in \{c, d\}\}$

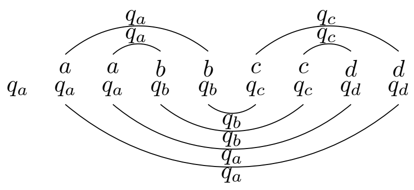

q_a
q_a
q_c
q_c
a a b b c c d d
q_a q_a q_a q_b q_b q_c q_c q_d q_d
q_b
q_b
q_a
q_a

Figure 4 A run of A_{ex} over ω_2.

We illustrate the semantics of 2NWA by giving on Figure 4 a graphical representation of a run of A_{ex} on the 2NW $\omega_2 = (a^2b^2c^2d^2, M_1, M_2)$, with M_1, M_2 depicted on Figure 4. We let the reader check that the projection of $L(A_{ex})$ on Σ^* is equal to $\{a^n b^n c^n d^n \mid n \geq 1\}$, hence not context-free.

▶ **Definition 14** (deterministic 2NWA). *A 2NWA is deterministic iff $Q_0 = \{q_0\}$ and for all $x, y \in \{c, r, int\}$, Δ^x_y induces a function $Q \times P^{\mathsf{in}_{x,y}} \times \Sigma \to P^{\mathsf{out}_{x,y}} \times Q$.*

▶ **Example 15.** The automaton A_{ex} considered in Example 13 is deterministic.

Normal form. To ease further constructions, we present a normal form for 2NWA that requires the hierarchical state of an arch to be equal to the target linear state of its call index. More formally, we say that a 2NWA is in weakly-hierarchical post form (post form for short) if $P = Q$ and for all $x, y \in \{c, r, \text{int}\}$, $\Delta^x_y \subseteq \bigcup_{q \in Q} Q \times Q^{\mathsf{in}_{x,y}} \times \Sigma \times \{q\}^{\mathsf{out}_{x,y}} \times \{q\}$. As a consequence, transitions of an automaton in post form can be simplified: $\Delta^x_y \subseteq Q \times Q^{\mathsf{in}_{x,y}} \times \Sigma \times Q$.

It is worth observing that in a 2NWA in post form, a run is completely characterized by the linear states. Hence, we can omit the hierarchical states in the formula run^A_i, and we can say that a sequence of (linear) states ℓ is a run of A on a 2-nested word ω.

▶ **Example 16.** The automaton A_{ex} considered in Example 13 is in post form.

▶ **Lemma 17.** *Given a* 2NWA $A = (Q, Q_0, Q_f, \Sigma, P, \Delta)$, *we can build a* 2NWA $A' = (Q', q'_0, Q'_f, \Sigma, Q', \Delta')$ *which is in weakly-hierarchical post form and such that $L(A) = L(A')$.*

Closure properties. Applying classical automata constructions to 2NWA, one can prove the following closure properties of regular 2NWL.

▶ **Proposition 18** (See also [4]). *Regular* 2NWL *are closed under union, intersection, and direct and reciprocal image by a non erasing alphabetic morphism.*

4 Determinization of 2NWA over 2-wave words

▶ **Theorem 19.** 2NWA *are determinizable on the subclass of 2-wave words,* i.e., *given a* 2NWA A, *we can build a deterministic* 2NWA A' *such that $L_{\mathsf{WW}_2}(A) = L_{\mathsf{WW}_2}(A')$.*

Thanks to Lemma 17, we start from a 2NWA in post normal form $A = (Q, Q_0, Q_f, \Delta)$. This normal form will allow to lighten the presentation, as less arguments are needed to write the transitions and the runs. We will describe the construction of a deterministic 2NWA A' (not in post normal form), which accepts the same language of 2-wave words.

Introduction to the construction. The determinization of finite-state automata, known as the powerset construction, registers all the states reachable by a run of A, which in the sequel are named *current* states. The determinization procedure of [3] for nested word automata requires the recording of two states. In addition to the current state, they also store the state labelling the call of the arch covering the current position. This state is named *reference* state. Intuitively, it stores where we come from, and will be used when closing the arch to reconcile the global run with what happened below this arch.

In our setting, as we consider a pair of matchings (M_1, M_2) instead of a single one, we need to store triples of states, composed of two reference states (one for M_1 and one for M_2), and one current state. Hence, states of A' will be sets of such triples of states.

The very particular shape of 2-wave words, and in particular the fact that the support arch and the top arches do end up at the same return-return position, ensure that we can gather the information collected along these two paths to compute, in a deterministic fashion, the set of possible current states.

However, when trying to address the setting of 2-wave words, we face another difficulty related to reference states. Indeed, the computations we will perform on each arch of the 2-wave word are somehow disconnected. In order to be sure that they can be reconciled, we will enrich the reference state of the second top arch of the 2-wave with the state of the bottom arch and that of the first top arch. This allows us to check whether the reference associated with the second top arch is *compatible* with that chosen for the first top arch.

Construction. We will define the deterministic 2NWA $A' = (Q', \{q'_0\}, Q'_f, P', \delta)$ in the following way. We first introduce the reference states for M_1 and M_2: [1]

$\overline{\mathcal{R}}^1$	$:= Q$	the reference for positions at the surface of the first top arch
$\overline{\mathcal{R}}^2$	$:= Q^3$	the reference for positions at the surface of the second top arch
$\underline{\mathcal{R}}$	$:= Q$	the reference for M_2

We denote by $\overline{\mathcal{R}}$ the union $\overline{\mathcal{R}}^1 \cup \overline{\mathcal{R}}^2$. This allows us to define:

$$Q' := 2^{\overline{\mathcal{R}}^1 \times \underline{\mathcal{R}} \times Q} \cup 2^{\overline{\mathcal{R}}^2 \times \underline{\mathcal{R}} \times Q} \qquad q'_0 := \{(q_0, q_0, q_0);\ q_0 \in Q_0\}$$

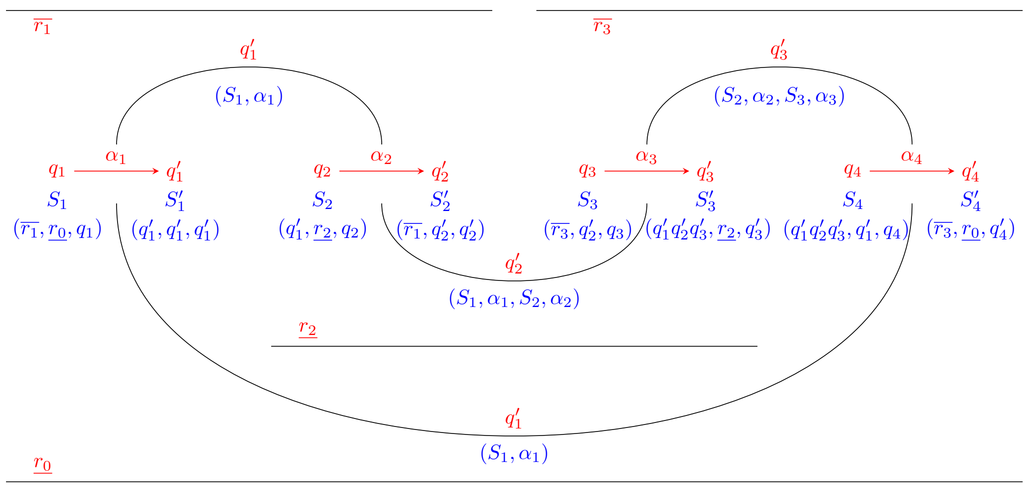
$$Q'_f := \{S \in Q';\ S \cap Q \times Q \times Q_f \neq \emptyset\} \qquad P' := (Q' \times \Sigma) \cup (Q' \times \Sigma)^2$$

The transition function δ for A' is defined as follows, by distinguishing cases according to the nature of the symbol. The construction is illustrated on Figure 5. In order to ease the writing, the arguments of the formula are not explicitly written.

- $\delta_{\text{int}}(S, \alpha) := \{(\overline{r}, \underline{r}, q');\ \exists q\ (\overline{r}, \underline{r}, q) \in S \wedge (q, \alpha, q') \in \Delta_{\text{int}}\}$
- $\delta_c^c(S_1, \alpha_1) := (S'_1, (S_1, \alpha_1), (S_1, \alpha_1))$ where

$$S'_1 := \{(q'_1, q'_1, q'_1);\ \exists \overline{r}_1, \underline{r}_0, q_1\ \phi_1\}$$
$$\phi_1 := (\overline{r}_1, \underline{r}_0, q_1) \in S_1 \wedge (q_1, \alpha_1, q'_1) \in \Delta_c^c.$$

[1] Observe that the reference state for the second top arch is a triple of states, as explained before.



$\overline{r_1}$
q_1'
(S_1, α_1)
q_3'
$\overline{r_3}$
$(S_2, \alpha_2, S_3, \alpha_3)$
$q_1 \xrightarrow{\alpha_1} q_1'$
$q_2 \xrightarrow{\alpha_2} q_2'$
$q_3 \xrightarrow{\alpha_3} q_3'$
$q_4 \xrightarrow{\alpha_4} q_4'$
S_1 S_1' S_2 S_2' S_3 S_3' S_4 S_4'
$(\overline{r_1}, \underline{r_0}, q_1)$
(q_1', q_1', q_1')
$(q_1', \underline{r_2}, q_2)$
$(\overline{r_1}, q_2', q_2')$
$(\overline{r_3}, q_2', q_3)$
$(q_1'q_2'q_3', \underline{r_2}, q_3')$
$(q_1'q_2'q_3', q_1', q_4)$
$(\overline{r_3}, \underline{r_0}, q_4')$
q_2'
$(S_1, \alpha_1, S_2, \alpha_2)$
$\underline{r_2}$
q_1'
(S_1, α_1)
$\underline{r_0}$

Figure 5 Illustration of the determinization. The (original) run in A is depicted in red, while the (new) one in A' is in blue. Elements of states of A' are illustrated by triples depicted below them.

- $\delta_c^r(S_2, (S_1, \alpha_1), \alpha_2) := (S_2', (S_1, \alpha_1, S_2, \alpha_2))$ where

$$S_2' := \{(\overline{r}_1, q_2', q_2');\ \exists \underline{r}_0, \underline{r}_2, q_1, q_1', q_2\ \phi_2\}$$
$$\phi_2 := \phi_1 \wedge (q_1', \underline{r}_2, q_2) \in S_2 \wedge (q_2, q_1', \alpha_2, q_2') \in \Delta_c^r$$

- $\delta_r^c(S_3, (S_1, \alpha_1, S_2, \alpha_2), \alpha_3) := (S_3', (S_2, \alpha_2, S_3, \alpha_3))$ where [2]

$$S_3' := \{(q_1'q_2'q_3', \underline{r}_2, q_3');\ \exists \overline{r}_1, \overline{r}_3, \underline{r}_0, q_1, q_2, q_3\ \phi_3\}$$
$$\phi_3 := \phi_2 \wedge \overline{r}_1 \preceq \overline{r}_3 \wedge (\overline{r}_3, q_2', q_3) \in S_3 \wedge (q_3, q_2', \alpha_3, q_3') \in \Delta_r^c$$

- $\delta_r^r(S_4, (S_2, \alpha_2, S_3, \alpha_3), (S_1, \alpha_1), \alpha_4) := S_4'$ where

$$S_4' := \{(\overline{r}_3, \underline{r}_0, q_4');\ \exists \overline{r}_1, \underline{r}_2, q_1, q_1', q_2, q_2', q_3, q_3', q_4\ \phi_4\}$$
$$\phi_4 := \phi_3 \wedge (q_1'q_2'q_3', q_1', q_4) \in S_4 \wedge (q_4, q_3', q_1', \alpha_4, q_4') \in \Delta_r^r$$

▶ Remark 20. Formulas $(\phi_j)_{j\in[4]}$ are not quantified at all, hence all their variables are free. Their objective is to link parameters extracted from the triplets of states of A'.

Proof of correctness. We fix a 2-wave word $\omega = (w, M_1, M_2)$, with $w = a_1 \dots a_n$. The following proposition states that on an interval without pending arch, runs of A' exactly capture possible runs of A, while keeping track of the reference states, as explained before.

▶ **Proposition 21.** *Let $[\![s, f]\!] \subseteq [n]$ be wpa, and $S, S' \in Q'$ such that $S \xrightarrow[A']{\omega, [\![s,f]\!]} S'$, then:*

$$\forall \overline{r} \in \overline{\mathcal{R}},\ \underline{r} \in \underline{\mathcal{R}},\ q' \in Q,\ \Big((\overline{r}, \underline{r}, q') \in S' \iff \exists q \in Q\ (\overline{r}, \underline{r}, q) \in S \wedge q \xrightarrow[A]{\omega, [\![s,f]\!]} q' \Big)$$

Before sketching the proof of this proposition, we fix some notations. Let $[\![s, f]\!] \subseteq [n]$ be an interval wpa, and $S, S' \in Q'$ such that $S \xrightarrow[A']{\omega, [\![s,f]\!]} S'$. Observe that as A' is deterministic,

[2] $\preceq$ denotes the prefix partial order on strings.

it has a unique run on the 2-wave word ω restricted to $[\![s,f]\!]$. We let $(L_i)_{i\in[\![s-1,f]\!]}$ denote the (linear) states of this run. In particular, we have $L_{s-1} = S$ and $L_f = S'$. As A' is deterministic, hierarchical states are completely determined from linear ones.

In order to prove Proposition 21, we separate the direct from the indirect case. For the indirect case, it is easy to show by induction that any run of A on ω (restricted to $[\![s,f]\!]$) will appear in the run of A' on ω (restricted to $[\![s,f]\!]$). This only requires to define carefully the corresponding reference states.

▶ **Lemma 22.** *Let $\ell = (\ell_i)_{i\in[\![s-1,f]\!]}$ be a run of A on ω (restricted to $[\![s,f]\!]$) such that $\ell_{s-1} = q$, $\ell_f = q'$ and $(\overline{r}, \underline{r}, q) \in S = L_{s-1}$. Then we can define reference mappings $\overline{R}$ and $\underline{R}$ from $[\![s-1,f]\!]$ to $\overline{\mathcal{R}}$ and $\underline{\mathcal{R}}$ respectively, such that $\overline{R}(f) = \overline{r}$, $\underline{R}(f) = \underline{r}$, and for all $i \in [\![s-1,f]\!]$, $(\overline{R}(i), \underline{R}(i), \ell_i) \in L_i$.*

The direct implication will be proven by induction on $[\![s,f]\!]$. More precisely, we rely on the grammar allowing to describe all 2-wave words given in Section 2. We use a slight abuse of notation here, as the grammar is based on the visibly pushdown presentation, while we work here with 2-nested words, but we believe the correspondence is without ambiguity. The three first cases of the grammar (internal, empty word, and concatenation of 2-wave words) are easy. The last case decomposes the 2-wave word ω as $\omega = x\omega' y$, with (x, y) being a pair of "matched" words, *i.e.* given by the non-terminal H. Intuitively, the grammar gives a decomposition of the wpa interval associated with ω into a wpa interval corresponding to ω', and a pair of matched intervals corresponding to (x, y). Hence, this requires to study pairs of matched intervals, *i.e.* a pair of intervals which is without pending arch. To this end, we introduce a notation for a "partial" run on a pair of matched intervals:

▶ **Definition 23.** *Let $(I_1 = [\![i_1,j_1]\!], I_2 = [\![i_2,j_2]\!])$ be a pair of matched intervals w.r.t. ω. Let q_1, q_2, q'_1, q'_2 be four states. We let $J = I_1 \cup I_2 \cup \{i_1 - 1, i_2 - 1\}$. We write $q_1, q_2 \xrightarrow[A]{\omega, I_1, I_2} q'_1, q'_2$ if there exists $(\ell_i)_{i\in J}$ such that $\forall i \in J$, $run^A_i(\omega, \ell)$ and $\forall k \in [2]$, $q_k = \ell_{i_k-1}$ and $q'_k = \ell_{j_k}$*

We are now ready to state the following lemma:

▶ **Lemma 24.** *Let (i_1, i_2, i_3, i_4) be a 2-wave such that $i_1 \in [\![s,f]\!]$ and $(\overline{r}_3, \underline{r}_0, q'_4) \in L_{i_4}$. For any $\overline{r}_1, \underline{r}_2, q_1, q'_1, q_2, q'_2, q_3, q'_3, q_4$ satisfying ψ_4 we have $q_1, q_3 \xrightarrow[A]{\omega, [\![i_1,i_2]\!], [\![i_3,i_4]\!]} q'_2, q'_4$.*

Lemma 24 confirms the intuition that a 2-wave is an encapsulation in its sort: any q_1, q'_2 and q_3 yielded by $(\overline{r}_3, \underline{r}_0, q'_4) \in L_{i_4}$ via the construction of L_{i_4} (e.g. via ψ_4) define with q'_4 a run of A on ω on the subset of positions given by $[\![i_1,i_2]\!]$ and $[\![i_3,i_4]\!]$.

We explain how to conclude the proof of Proposition 21. Starting from the 2-wave word ω, we obtained a decomposition $\omega = x\omega' y$, with (x, y) being a pair of "matched" words produced by H. We can show in addition that extremal positions of x and y exactly correspond to a 2-wave, in the sense of the premises of Lemma 24. Combining it with the induction hypothesis (direct implication of Proposition 21) applied on ω', we can exhibit the expected run in A.

Proof of Theorem 19. First, observe that by construction, A' is deterministic and complete. Let $\omega \in \mathsf{WW}_2(\Sigma)$. As A' is deterministic and complete, it has a unique run on ω starting from q'_0. Let $(L_i)_{i\in[\![0,n]\!]}$ be this run, with $L_0 = q'_0$. We proceed by equivalence:

$$\omega \in L(A') \iff L_n \in Q'_f \iff L_n \cap Q \times Q \times Q_f \neq \emptyset \iff \exists(\overline{r}, \underline{r}, q_f) \in L_n\ q_f \in Q_f$$
$$\iff \exists q_0, q_f \in Q_f\ (\overline{r}, \underline{r}, q_0) \in q'_0 \text{ and } q_0 \xrightarrow[A]{\omega} q_f \iff \omega \in L(A). \quad ◀$$

Closure under complementation. It is proved in [19] that regular 2NWL are not closed under complementation, since they are not determinizable. The definition of 2NWL they use is sightly different from ours because the two matchings do not share positions, but the same negative result can be shown for our definition. For example, there is no deterministic automaton recognizing the language $\{(a^{n+m}b^m a^n b^m a^n, \{(i, i+2(n+m)) \mid 1 \leq i \leq n+m\}, \{(i, i+n+m) \mid 1 \leq i \leq n+m\}) \mid n, m \geq 0\}$. However, thanks to our determinization result for regular WW_2 languages, we get:

▶ **Proposition 25.** *Regular languages of* 2*-wave words are closed under complementation.*

5 Logical characterization

We show that languages of 2-wave words definable in monadic second order logic (MSO) are exactly regular languages of 2-wave words. Let us fix $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, $\mathcal{V}_1$ is the set of first order variables (whose elements will be written using lower-cases) and $\mathcal{V}_2$ is the set of second order variables (whose elements will be written using upper-cases).

The monadic second-order logic of 2-nested words (MSO$(\Sigma, <, M_1, M_2)$) is given by

$$\phi := Q_a(x) \mid x < y \mid X(x) \mid M_i(x, y) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x\ \phi \mid \exists X\ \phi$$

where $a \in \Sigma$, $x, y \in \mathcal{V}_1$, $X \in \mathcal{V}_2$, $i \in \{1, 2\}$.

The semantics is defined over 2-nested words in a natural way. The first-order variables are interpreted over positions of the nested word, while set variables are interpreted over sets of positions; $Q_a(x)$ holds if the symbol at the position interpreted for x is a, $x < y$ holds if the position interpreted for x is lesser than the position interpreted for y, and $M_i(x, y)$ holds if the positions interpreted for x and y are related by a nesting edge of matching M_i.

Let us define some fragments of MSO$(\Sigma, <, M_1, M_2)$. The set FO$(\Sigma, <, M_1, M_2)$ of first order (FO) formulas is the set of all formulas in MSO$(<, M_1, M_2)$ that do not contain any second-order quantifier. Furthermore, the set EMSO$(\Sigma, <, M_1, M_2)$ of existential MSO (EMSO) formulas consists of all formulas of the form $\exists X_1 \ldots \exists X_n \phi$, with $\phi \in$ FO$(\Sigma, <, M_1, M_2)$. If $\mathcal{L}$ is a logic (MSO, EMSO or FO), and ϕ is a closed formula in $\mathcal{L}(\Sigma, <, M_1, M_2)$, the *language defined by* ϕ, denoted $L(\phi)$ is the set of all 2-nested words that are a model for ϕ, and $L_{\mathsf{WW}_2}(\phi)$ is the set of all 2-wave words that are a model for ϕ, that is, $L_{\mathsf{WW}_2}(\phi) = L(\phi) \cap \mathsf{WW}_2(\Sigma)$.

In [4], Bollig shows the equivalence between automata and EMSO for the whole class of 2-nested words. However, he shows that quantifier alternation yields an infinite hierarchy. In our setting, this hierarchy collapses, and we obtain the equivalence between MSO and EMSO. The following characterization extends that given in [3] for regular nested word languages. Its proof follows classical lines, and relies on Propositions 18 and 25.

▶ **Theorem 26.** *A language of* 2*-wave words is definable in EMSO iff it is definable in MSO iff it is regular.*

6 Decision problems

The analysis we did so far of 2NWA over 2-wave words allows to establish the following decidability results:

▶ **Theorem 27.** *Let A and B be two* 2NWA *over Σ. The following holds:*
- *Determining whether $L_{\mathsf{WW}_2}(A) = \emptyset$ can be decided in polynomial time.*
- *Determining whether $L_{\mathsf{WW}_2}(A) = \mathsf{WW}_2(\Sigma)$ can be decided in exponential time.*
- *Determining whether $L_{\mathsf{WW}_2}(A) \subseteq L_{\mathsf{WW}_2}(B)$ (resp. $L_{\mathsf{WW}_2}(A) = L_{\mathsf{WW}_2}(B)$) can both be decided in exponential time.*

Proof sketch. The first result follows from the grammar presented in Section 2 that produces all 2-wave words. More precisely, we maintain a set W of pairs of states (for non-terminal W) and a set H of quadruplets of states (for non-terminal H). Intuitively, $(p, q) \in W$ means that there exists a 2-wave word ω and a run $p \xrightarrow[A]{\omega} q$. We initialize them with identity relations and saturate them using the rules of the grammar, and corresponding transitions of A. The other results are direct consequences from closure under complement using the determinization procedure, with the observation that the latter has exponential complexity. ◀

On the treewidth of 2-wave words. Any 2NW $\omega = (w, M_1, M_2)$ of length n can be seen as a graph whose set of vertices is $[n]$ and set of edges is $M_1 \cup M_2 \cup \{(i, i+1)\}_{i \in [n-1]}$, then an alternative approach to decidability is by using Courcelle's Theorem on graphs of bounded treewidth. Indeed, using the grammar presented in Section 2 for 2-wave words (Lemma 9), we show:

▶ **Lemma 28.** *For all $\omega \in \mathsf{WW}_2(\Sigma)$, ω has treewidth at most 11.*

In addition, it is easy to verify that the class of 2-wave words can be expressed in MSO. As a consequence, we obtain using [7, 17]:

▶ **Proposition 29.** *Given a formula $\phi \in MSO(<, M_1, M_2)$, checking whether there exists $\omega \in \mathsf{WW}_2(\Sigma)$ that satisfies ϕ is decidable.*

7 Relation to indexed languages

Let $\mathcal{L}$ be a logic (FO, EMO, or MSO), we denote by $\exists \text{Match}\mathcal{L}(\Sigma, <, M)$ the class of all word languages $L = \{u \in \Sigma^* \mid \exists M (u, M) \in L_M\}$ such that L_M is definable in $\mathcal{L}(\Sigma, <, M)$. Similarly, we denote by $\exists \mathsf{WW}_2\mathcal{L}(\Sigma, <, M_1, M_2)$ the class of languages $L = \{u \in \Sigma^* \mid \exists (u, M_1, M_2) \in L_{\mathsf{WW}_2}(\phi)\}$ for $\phi \in \mathcal{L}(\Sigma, <, M_1, M_2)$.

It is proved in [12] that CFL $= \exists\text{Match FO}(\Sigma, <, M) = \exists\text{Match MSO}(\Sigma, <, M)$. A key of the proof is the fact that CF grammars can be put in Greibach double normal form. Then each production has the form $X \to aub$, and the derivation tree of a word can be encoded by matching letters delimiting each production.

Consider now indexed grammars that generate Indexed Languages (IL). They are CF grammars where each non-terminal symbol carries a pushdown stack (often denoted X^ω where X is the non terminal and ω the stack). Grammars contain *push productions* of the form $X \to Y^p$ saying that any X^ω can be rewritten $Y^{p\omega}$; *pop productions* of the form $X^p \to Y$ saying that any $X^{p\omega}$ can be rewritten Y^ω and *copy productions* of the form $X \to YZ$ saying that any X^ω can be rewritten $Y^\omega Z^\omega$.

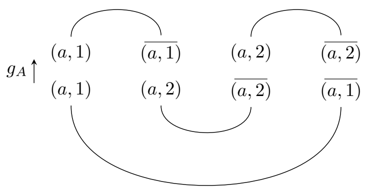
The derivation tree of a word can be encoded using two matching relations: one delimiting productions (corresponding to M_2), and one for the stack moves (corresponding to M_1). The nested structure thus obtained is a wave structure, where each wave follows a pushed symbol amongst the different copies. In particular, a 2-wave corresponds to a symbol which has been popped but has never been copied. Indexed grammars which never process copies are called *linear indexed grammars* and generate *linear indexed languages* (LIL). In such grammars, copy productions have the form $X \to X^\bullet Y$ or $X \to XY^\bullet$ where $\bullet$ marks the symbol on which the stack is transmitted.

The next Proposition establishes a formal connection between regular languages of 2-wave words and linear indexed languages:

▶ **Proposition 30.** *Languages in $\exists\mathsf{WW}_2 MSO(\Sigma, <, M_1, M_2)$ are linear indexed languages.*

Proof sketch. We use the homomorphic characterization of linear indexed languages given in [22]. We denote by $\mathcal{D}_A$ the Dyck language over alphabet A with inverse letters in $\bar{A} = \{\bar{a} \mid a \in A\}$. Then, a language L is linear indexed iff there exists an alphabet A, a regular language R and a morphism h such that $L = h(R \cap \mathcal{D}_{\Gamma_A} \cap g_A^{-1}(\mathcal{D}_{\Gamma_A}))$, where: $A_i = \{(a,i) \mid a \in A\}$, for $i = 1,2$, $\Gamma_A = A_1 \cup A_2$ and g_A is the morphism defined by

$$g_A : \begin{array}{ll} (a,1) \mapsto (a,1) & \overline{(a,1)} \mapsto \overline{(a,2)} \\ (a,2) \mapsto \overline{(a,1)} & \overline{(a,2)} \mapsto (a,2) \end{array}$$


$g_A \uparrow$
$(a,1)$ $\overline{(a,1)}$ $(a,2)$ $\overline{(a,2)}$
$(a,1)$ $(a,2)$ $\overline{(a,2)}$ $\overline{(a,1)}$

Remark that from a given a word u in $\mathcal{D}_{\Gamma_A} \cap g_A^{-1}(\mathcal{D}_{\Gamma_A})$, we can construct a 2NW (u, M_1, M_2) where arches in M_2 correspond to $\mathcal{D}_{\Gamma_A}$, and M_1 to $g_A^{-1}(\mathcal{D}_{\Gamma_A})$.

The proof is then similar to that of Chomsky-Schützenberger Theorem applied to pushdown automata: the regular language R encodes runs on linear states, $\mathcal{D}_{\Gamma_A}$ and $g_A^{-1}(\mathcal{D}_{\Gamma_A})$ ensure the consistency of hierarchical states and h projects the run on its word. ◀

However, we don't know if the equality holds, mainly because we don't know if linear indexed languages can be put in a Greibach double normal like form.

We can also ask whether the logical characterization of CFL extends to indexed languages. The appropriate structure seems to be 2NW whose edges form *waves* of unbounded length. Intuitively, a k-wave generalizes a 2-wave as follows: it consists of $2k$ indices in increasing order, with k top arches, $k-1$ bottom arches, and an additional support arch. An example of a 4-wave is depicted on Figure 1. This yields the notion of k-wave word (WW$_k$), and that of wave word (WW), which is simply the union of WW$_k$ over k. As an example, the 2-nested word depicted on Figure 6 is in WW$_4$. So, the question is: does IL $= \exists$WWFO$(\Sigma, <, M_1, M_2) = \exists$WWMSO$(\Sigma, <, M_1, M_2)$ hold? Proposition 31 provides an element of response, and we think that IL $\neq \exists$WWFO$(\Sigma, <, M_1, M_2)$.

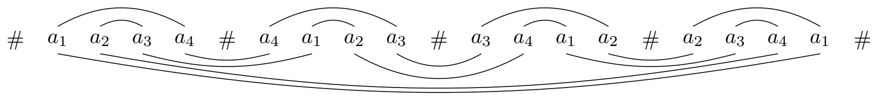

$\#\ a_1\ a_2\ a_3\ a_4\ \#\ a_4\ a_1\ a_2\ a_3\ \#\ a_3\ a_4\ a_1\ a_2\ \#\ a_2\ a_3\ a_4\ a_1\ \#$

Figure 6 Example of a word in L and its associated wave structure.

▶ **Proposition 31.** *There exists a language which is not an indexed language but being* $\exists$WW*EMSO*$(\Sigma, <, M_1, M_2)$*-definable.*

Proof sketch. Consider $\Sigma = A \cup \{\#\}$ for any alphabet A, and the set L of all words of the form $\#u_1\#u_2\#\ldots u_n\#$, for $n \geq 1$, such that for all $i \in [1, n-1]$, $u_i \in A^n$, and if $u_i = a_1 \ldots a_n$, then $u_{i+1} = a_n a_1 \ldots a_{n-1}$. Using the Shrinking Lemma given in [11] for indexed languages, it can easily be proved that L is not an indexed language. In addition, one can write an EMSO-formula defining 2-wave words $(\#u_1\#u_2\#\ldots u_n\#, M_1, M_2)$ such that (M_1, M_2) forms $n/2$ embedded n-waves (see an example on Figure 6). ◀

8 Conclusion

A natural perspective of this work consists in studying its extension to k-wave words. We believe that the determinization property should hold for this class too. This will require to improve our arguments, as the present proof seems intricate to be adapted to k-waves.

For unbounded waves, automata are not determinizable: Bollig proved in [4] that regular 2NWL are not closed under complementation, and are then not determinizable. His proof uses the encoding of grids in 2NWL, and can thus be adapted to (unbounded) wave words.

Another perspective consists in characterizing the subclass of indexed languages capturing languages $\exists\mathsf{WW}L$, when L is a regular WW_2 language, and determine if $\exists\mathsf{WW}L$ is still indexed when L is a regular WW_k language, for $k > 2$.

References

1 Alfred V. Aho. Indexed grammars - an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968. doi:10.1145/321479.321488.

2 Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.

3 Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, 2009. doi:10.1145/1516512.1516518.

4 Benedikt Bollig. On the expressive power of 2-stack visibly pushdown automata. *Log. Methods Comput. Sci.*, 4(4), 2008. doi:10.2168/LMCS-4(4:16)2008.

5 Dario Carotenuto, Aniello Murano, and Adriano Peron. 2-visibly pushdown automata. In Tero Harju, Juhani Karhumäki, and Arto Lepistö, editors, *Developments in Language Theory, 11th International Conference, DLT 2007, Turku, Finland, July 3-6, 2007, Proceedings*, volume 4588 of *Lecture Notes in Computer Science*, pages 132–144. Springer, 2007. doi:10.1007/978-3-540-73208-2_15.

6 Dario Carotenuto, Aniello Murano, and Adriano Peron. Ordered multi-stack visibly pushdown automata. *Theor. Comput. Sci.*, 656:1–26, 2016. doi:10.1016/j.tcs.2016.08.012.

7 Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 313–400. World Scientific, 1997.

8 W. Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20(2):95–207, 1982.

9 J. Engelfriet. Iterated pushdown automata and complexity classes. In *Proceedings of the 14th Symposium on Theory of Computing*, pages 365–373. Association for Computing Machinery, 1983.

10 Séverine Fratani and El Makki Voundy. Epsilon-reducible context-free languages and characterizations of indexed languages. *Inf. Comput.*, 269, 2019. doi:10.1016/j.ic.2019.104444.

11 Robert H. Gilman. A shrinking lemma for indexed languages. *Theor. Comput. Sci.*, 163(1&2):277–281, 1996. doi:10.1016/0304-3975(96)00244-7.

12 Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 1994. doi:10.1007/BFb0022257.

13 P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 283–294. ACM, 2011. doi:10.1145/1926385.1926419.

14 A. N. Maslov. The hierarchy of index languages of arbitrary level. *Dokl. Akad. Nauk SSSR*, 217:1013–1016, 1974.

15 A. N. Maslov. Multilevel pushdown automata. *Problemy Peredači Informacii*, 12(1):55–62, 1976.

16 A. Okhotin. Non-erasing variants of the chomsky-schützenberger theorem. In *Developments in Language Theory*, volume 7410 of *Lecture Notes in Comput. Sci.*, pages 121–129. Springer, 2012.

17 Detlef Seese. The structure of models of decidable monadic theories of graphs. *Ann. Pure Appl. Log.*, 53(2):169–195, 1991. doi:10.1016/0168-0072(91)90054-P.

18 Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theor. Comput. Sci.*, 88(2):191–229, 1991. doi:10.1016/0304-3975(91)90374-B.

19 Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. A robust class of context-sensitive languages. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 161–170. IEEE Computer Society, 2007. doi:10.1109/LICS.2007.9.

20 Salvatore La Torre, Parthasarathy Madhusudan, and Gennaro Parlato. The language theory of bounded context-switching. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics, 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010. Proceedings*, volume 6034 of *Lecture Notes in Computer Science*, pages 96–107. Springer, 2010. doi:10.1007/978-3-642-12200-2_10.

21 Salvatore La Torre, Margherita Napoli, and Gennaro Parlato. Scope-bounded pushdown languages. *Int. J. Found. Comput. Sci.*, 27(2):215–234, 2016. doi:10.1142/S0129054116400074.

22 D. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, 1988. Available as Technical Report MS-CIS-88-74.