Regular Languages of Nested Words: Fixed Points, Automata, and Synchronization

Marcelo Arenas[1] **Pablo Barceló**[2] **Leonid Libkin**[3]

[1] Pontificia Universidad Católica de Chile
[2] Universidad de Chile
[3] University of Edinburgh

Abstract. Nested words are a restriction of the class of visibly pushdown languages that provide a natural model of runs of programs with recursive procedure calls. The usual connection between monadic second-order logic (MSO) and automata extends from words to nested words and gives us a natural notion of regular languages of nested words.

In this paper we look at some well-known aspects of regular languages – their characterization via fixed points, deterministic and alternating automata for them, and synchronization for defining regular relations – and extend them to nested words. We show that mu-calculus is as expressive as MSO over finite and infinite nested words, and the equivalence holds, more generally, for mu-calculus with past modalities evaluated in arbitrary positions in a word, not only in the first position. We introduce the notion of alternating automata for nested words, show that they are as expressive as the usual automata, and also prove that Muller automata can be determinized (unlike in the case of visibly pushdown languages). Finally we look at synchronization over nested words. We show that the usual letter-to-letter synchronization is completely incompatible with nested words (in the sense that even the weakest form of it leads to an undecidable formalism) and present an alternative form of synchronization that gives us decidable notions of regular relations.

1 Introduction

The class of *visibly pushdown languages* (VPL) has been been introduced by Alur and Madhusudan [5] as a restriction of the class of context-free languages that subsumes all regular properties and some non-regular properties relevant in program analysis (e.g. stack-inspection properties and pre-post conditions). VPLs in many ways resemble regular languages: they have the same closure properties, and most natural problems related to them are decidable. The intuitive idea of VPLs is that the input alphabet Σ is partitioned into three parts, $\Sigma_c, \Sigma_r, \Sigma_i$, of symbols viewed as procedure calls, returns, and internal operations. A machine model for VPLs is a special pushdown automaton that pushes a symbol onto the stack in a call, pops one symbol in a return, and does not touch the stack when reading an internal symbol.

Nested words [6] replaced the implicit nesting structure of calls and returns by an explicit relation that matches calls and returns. A nested word is thus a word extended with a hierarchical structure on top of its linear structure. An example of such a nested structure of matching calls c_i and returns r_i is given below.

$c_1 \quad c_2 \quad c_3 \quad r_3 \quad c_4 \quad r_4 \quad r_2 \quad c_5 \quad c_6 \quad r_6 \quad r_5 \quad c_7 \quad r_7 \quad r_1$
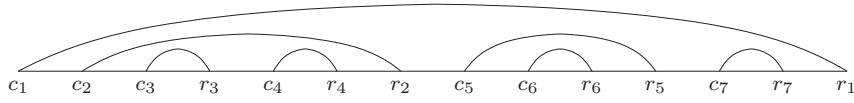
Such structures naturally appear, for instance, in XML documents that are string representations of trees using opening and closing tags [23, 8], or in software verification of programs with stack-based control flow [4, 2]. A *nested word automaton* [6] runs from left to right, similarly to a finite state automaton, but each time it encounters a "return" position, the next state depends not only on the current state but also on the state of the matching "call".

A nice property of nested words and their automata is that they share logical characterizations with the usual (unnested) words: the automaton model has the same expressiveness as monadic second-order logic (MSO) [5, 6]. This gives us a natural and robust notion of *regular languages* of nested words, with the expected closure properties, decision procedures, and logical characterizations.

For finite or infinite unnested words, an alternative way of describing regularity logically is via the modal μ-calculus (cf. [7]). That is, μ-calculus formulae evaluated in the first position of a word define precisely the regular languages. Moreover, μ-calculus formulae with past modalities evaluated in an arbitrary position of a word have precisely the power of MSO formulae with one free variable. As our first result, we extend these equivalences to the case of finite and infinite nested words.

We then look at automata characterizations of VPLs and nested words. Nondeterministic and deterministic automata have previously been considered [5, 6, 18], and [5] showed that automata can be determinized in the finite case, but in the infinite case this is impossible even for automata with a Muller acceptance condition (unlike in the case of the usual ω-words), if one considers VPLs. Then [18] introduced a different automaton model and showed that it admits a determinization procedure over nested words. We expand this in two ways. First we introduce alternation in the case of nested word automata, and prove that alternating automata can still be translated into nondeterministic ones. Second, we then refine the determinization procedure for automata from [18] to show that over infinite nested words, every regular language is definable by a deterministic Muller automaton. This also gives us some corollaries about the structure of regular languages of nested ω-words.

We finally turn our attention to the notion of regular *relations*. Over words, one moves from sets to relations by using letter-to-letter synchronization. That is, an automaton runs over a tuple of words viewing the tuple of ith letters of the words as a single letter of an expanded alphabet [15]. The same approach works for trees, ranked and unranked [11]. The notion of regular relations also leads to a notion of automatic structures [12, 13, 10], i.e. decidable first-order structures over words in which all definable relations are regular.

Here we show that, in contrast to the case of words and trees, the notion of letter-to-letter synchronization is incompatible with nested words: the simplest extension of nested word automata with such synchronization is undecidable. We present an alternative call-return notion of synchronization, and show that it gives us a natural concept of regular relations over nested words.

Related work VPLs were introduced in [5] and nested words in [6]. They can be viewed as special classes of trees (and we shall use this often in the paper); such tree representations were introduced in [5,6] as well. Applications in program analysis are discussed, e.g., in [2,4], and applications in processing tree-structured data in [23,8].

There are several related results on μ-calculus and MSO, e.g. their equality over infinite binary trees [20] or finite unranked trees [9] or expressive-completeness of μ-calculus [16]. We explain in Section 3 why we cannot derive our result from those. Another fixed-point logic VP_μ is defined in [2] to specify properties of executions of programs. It differs from the standard versions of μ-calculus we look at as its fixed points are evaluated not over sets of nodes but over sets of subtrees of the program; further, its expressiveness is known to be different from MSO [3].

Automata for VPLs and nested words were defined in [5,6], and [5] observed that Muller automata are not determinizable. Then [18] noticed that this is due to VPLs having potentially arbitrarily many unmatched calls/returns, and introduced a different automaton model (stair automata) that can be determinized. We use them to show how to determinize Muller automata over nested ω-words. None of these papers addresses alternating automata over nested words.

Letter-to-letter synchronization for defining regular relations is an old notion [15], and the concept of universal automatic structures [13,12] is based on it. Although such automatic structures exist for both words and trees [10,11], we show here that letter-to-letter synchronization is incompatible with nesting structure.

Organization Basic definitions are given in Section 2. We describe MSO unary queries via μ-calculus in Section 3. In Section 4 we study automata for nested words, define alternating automata, and describe determinization for Muller automata. In Section 5 we look at synchronization and regular relations for nested words.

2 Preliminaries

Words, ω-words, and automata Let Σ be a finite alphabet. A finite word $w = a_1 \ldots a_n$ in Σ^* is represented as a logical structure $\langle \{1, \ldots, n\}, (P_a)_{a \in \Sigma}, < \rangle$, where $<$ is the usual linear order on $\{1, \ldots, n\}$, and P_a is the set of i's such that $a_i = a$. We shall use w to refer to both the word and its logical representation. Infinite, or ω-words, are sequences $a_1 a_2 \cdots$ of symbols in Σ indexed by positive natural numbers, and are represented as structures $\langle \mathbb{N}^+, (P_a)_{a \in \Sigma}, < \rangle$. The length of w is denoted by $|w|$.

A (nondeterministic) *automaton* $\mathcal{A}$ over Σ is a tuple $(\Sigma, Q, Q_0, \delta, F)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states and $\delta : Q \times \Sigma \to 2^Q$ is a transition function. For automata over ω-words we shall use either a Büchi acceptance condition (given by $F \subseteq Q$) or a Muller acceptance condition (given by $\mathcal{F} \subseteq 2^Q$). A *run* of $\mathcal{A}$ over a word w is a map $\rho : \{1, \ldots\} \to Q$ such that $\rho(1) \in Q_0$ and $\rho(i + 1) \in \delta(\rho(i), a_i)$, for all i. A finite run is accepting if $\rho(|w| + 1) \in F$. We let $Inf(\rho)$ be the set of states that occurs infinitely often in an infinite run ρ. Then ρ is accepting for a Büchi condition F if $Inf(\rho) \cap F \neq \emptyset$, and it is accepting for a Muller condition $\mathcal{F}$ if $Inf(\rho) \in \mathcal{F}$. A word is accepted iff there exists an accepting run. Sets of (ω-)words accepted by automata are called *regular*.

$\mathcal{A}$ is *deterministic* if $|Q_0| = 1$, and $|\delta(q, a)| = 1$ for for every $a \in \Sigma$ and $q \in Q$. Nondeterministic automata over ω-words with Büchi and Muller conditions are equivalent, and automata with Muller acceptance condition can be determinized, cf. [25].

Nested words A finite *nested word* over Σ is a pair $\bar{w} = (w, \eta)$, where $w \in \Sigma^*$ and η is a binary *matching relation* on $\{1, \ldots, |w|\}$ that satisfies: (1) $\eta(i, j)$ implies $i < j$; (2) $\eta(i, j)$ and $\eta(i, j')$ imply $j = j'$ and $\eta(i, j)$ and $\eta(i', j)$ imply $i = i'$; and (3) if $\eta(i, j)$, $\eta(i', j')$, and $i < i'$ then either $j < i'$ or $j' < j$. A *nested ω-word* is a pair $\bar{w} = (w, \eta)$, where w is an ω-word and η is a matching on $\mathbb{N}^+$. We also refer to them as infinite nested words. We represent nested words as logical structures over the vocabulary $\{(P_a)_{a \in \Sigma}, <, \eta\}$, i.e. words expanded with a matching relation. For a nested word $\bar{w}$ and two positions $i < j$, we let $\bar{w}[i, j]$ be the substructure of $\bar{w}$ induced by positions ℓ such that $i \leq \ell \leq j$. A position i of a nested word $\bar{w}$ is: (1) a *call* position if there is j such that $\eta(i, j)$ holds; (2) a *return* position if there is j such that $\eta(j, i)$ holds; and (3) an *internal* position if it is neither a call nor a return. Whenever $\eta(i, j)$ holds we say that i is the call of j, and j is the return of i.

Nested word automata A *nested word automaton*, or NWA [6], $\mathcal{A}$ over Σ is defined as a usual automaton, except that δ is a triple $(\delta_c, \delta_\iota, \delta_r)$ of transition functions $\delta_c, \delta_\iota : Q \times \Sigma \to 2^Q$, and $\delta_r : Q \times Q \times \Sigma \to 2^Q$. A *run* of $\mathcal{A}$ over $\bar{w} = (a_1 \cdots, \eta)$ is a mapping $\rho : \{1, \ldots\} \to Q$ such that $\rho(1) \in Q_0$ and for every $i \in \mathbb{N}^+$ (or $i \in [1, |\bar{w}|]$ for finite nested words),

- if i is a call position, then $\rho(i + 1) \in \delta_c(\rho(i), a_i)$;
- if i is an internal position, then $\rho(i + 1) \in \delta_\iota(\rho(i), a_i)$;
- if i is a return position whose call is j, then $\rho(i + 1) \in \delta_r(\rho(i), \rho(j), a_i)$.

Büchi and Muller acceptance conditions can then be defined in exactly the same way as for the usual automata (and are easily shown to be equivalent over nested words, for nondeterministic automata). We refer to such automata as ω-NWAs. An NWA is deterministic if the values of all transition functions are singletons.

A class of nested (ω-)words accepted by an (ω-)NWA is called *regular*.

Monadic second-order logic and μ-calculus Monadic second-order logic(MSO) extends first-order logic with quantification over sets. Over nested words, its vocabulary contains predicates P_a ($a \in \Sigma$), $<$ and η. A class of nested (ω-)words is regular iff it is definable by an MSO sentence [5, 6].

The *μ-calculus* over nested words, denoted by L_μ, is defined by the grammar:

$$\varphi, \varphi' \; := \; a \mid X \mid \varphi \vee \varphi' \mid \varphi \wedge \varphi' \mid \neg \varphi \mid \Diamond \varphi \mid \Diamond_\eta \varphi \mid \mu X.\varphi(X)$$

with X occurring positively in $\varphi(X)$, and $a \in \Sigma \cup \{\texttt{call}, \texttt{int}, \texttt{ret}\}$. Given a nested ($\omega$-)word $\bar{w}$, a position i in $\bar{w}$, and a valuation v assigning each free variable X a set of positions of $\bar{w}$, the semantics is as follows (omitting the rules for Boolean connectives):

- $(\bar{w}, v, i) \models \texttt{int}$ iff i is an internal position; $(\bar{w}, v, i) \models \texttt{call}$ iff i is a call position; and $(\bar{w}, v, i) \models \texttt{ret}$ iff i is a return position.
- $(\bar{w}, v, i) \models a$, for $a \in \Sigma$, iff i is labeled a.
- $(\bar{w}, v, i) \models X$ iff $i \in v(X)$.
- $(\bar{w}, v, i) \models \Diamond \varphi$ iff $i + 1$ belongs to $\bar{w}$ and $(\bar{w}, v, i + 1) \models \varphi$.

- $(\bar{w}, v, i) \models \diamondsuit_\eta \varphi$ iff there is an ℓ such that $\eta(i, \ell)$ holds and $(\bar{w}, v, \ell) \models \varphi$.
- $(\bar{w}, v, i) \models \mu X. \varphi(X)$ iff i is in the least fixed point of the operator defined by φ; in other words, if $i \in \bigcap \{P \mid \{i' \mid (\bar{w}, v[P/X], i') \models \varphi\} \subseteq P\}$, where $v[P/X]$ extends the valuation v with $v(X) = P$.

The μ-calculus over words does not mention the modality $\diamondsuit_\eta \varphi$.

We shall also work with the *full* μ-calculus [28] (denoted by L_μ^{full}), which is an extension of L_μ with the *past* modalities $\diamondsuit^- \varphi$ and $\diamondsuit_\eta^- \varphi$:

- $(\bar{w}, v, i) \models \diamondsuit^- \varphi$ iff $i > 1$ and $(\bar{w}, v, i-1) \models \varphi$.
- $(\bar{w}, v, i) \models \diamondsuit_\eta^- \varphi$ iff there is an ℓ such that $\eta(\ell, i)$ holds and $(\bar{w}, v, \ell) \models \varphi$.

Greatest fixed-points $\nu X \varphi(X)$ are definable in L_μ as $\neg \mu X \neg \varphi(\neg X)$. Using greatest fixed-points and $\square \varphi$ (defined as $\neg \diamondsuit \neg \varphi$), one can push all negations to atoms in L_μ formulae. For resulting formulae, an important parameter is the alternation-depth of least and greatest fixed-points [7]. We refer to L_μ^k as the fragment of L_μ that consists of formulae of alternation depth at most k (e.g., L_μ^0 is the alternation-free fragment).

Languages and unary queries Formulae of L_μ (without free variables) are satisfied in positions of a nested word, and thus they give rise to classes of *unary queries* that return, for $\bar{w}$, the set $\{i \mid (\bar{w}, i) \models \varphi\}$. Every L_μ formula φ without free variables defines a language $\{\bar{w} \mid (\bar{w}, 1) \models \varphi\}$. Likewise, every MSO formula $\varphi(x)$ with one free first-order variable defines a unary query, and every MSO sentence defines a language. In the absence of nesting, it is known [7, 20] that a language (of words or ω-words) is definable by a L_μ formula iff it is definable by an MSO sentence (not using relation η).

3 Mu-calculus over nested words

Since NWA generalize finite state automata, the translation from MSO to NWAs is nonelementary. But just as for finite words or trees, one can find equally expressive logical formalisms with better model-checking complexity. We show that the equivalence MSO $= L_\mu$ extends from words and trees to nested words. It applies not only in sentences evaluated in the first position of a nested word, but more generally to unary queries that select a set of positions. This is relevant for finite nested words viewed as streaming XML documents: while theoretical investigations have mostly looked at the case of sentences [23, 8], in practical application one typically needs to evaluate unary queries (e.g. XPath) over such streams [21]. To deal with unary queries, we look at L_μ with the past, i.e. L_μ^{full}, and prove that it is equivalent to MSO unary queries. That is:

Theorem 1. *For finite nested words and nested ω-words, MSO and L_μ^{full} define the same classes of unary queries.*

As a corollary to the proof, we get

Corollary 1. *The languages of nested words definable in MSO and L_μ are the same.*

We can tighten this for finite nested words. Let $(L_\mu^{\text{full}})^+$ be the negation-free (and thus alternation-free) fragment of L_μ^{full} that has two additional constants "first" and "last" with their intuitive meanings: "first" holds only at the first position of a nested word, and "last" holds at the last position. Likewise we define $(L_\mu)^+$ from L_μ.

Corollary 2. *For unary queries over finite nested words,* $\mathrm{MSO} = L_\mu^{\mathrm{full}} = (L_\mu^{\mathrm{full}})^+$. *Furthermore,* MSO, L_μ, *and* $(L_\mu)^+$ *define the same languages of finite nested words.*

From [14], we conclude that for every $(L_\mu^{\mathrm{full}})^+$ formula φ and every finite nested word $\bar{w}$, the set $\{i \mid (\bar{w}, i) \models \varphi\}$ can be computed in time $O(|\varphi| \cdot |\bar{w}|)$.

We make a couple of remarks about the proof of Theorem 1. Nested words are naturally translated into trees, and there is a closely related result in the literature, Niwinski's theorem, showing that over the full infinite binary tree, MSO and L_μ, evaluated in the root of the tree, are equally expressive [20]. Despite this, there does not seem to be any easy adaptation of proof techniques in [20] that yields a proof of Theorem 1. Not only do we need a stronger result for unary queries and an extension with the past modalities, but in addition translations of infinite nested words are not complete binary trees.

Another natural attempt at a proof is to use the expressive-completeness result of Janin and Walukiewicz: every bisimulation-invariant MSO property is definable in L_μ [16]. Then we could express runs of tree automata on tree encodings of nested words by bisimulation-invariant MSO sentences, apply [16] to get an equivalent L_μ formula for trees, and translation it into an L_μ formula over nested words. This sketch indeed can be turned into a proof of $\mathrm{MSO} = L_\mu$ for languages of nested words, but it breaks already for unary queries over finite nested words, where one needs to encode a more complicated run of a query automaton [19], and it is even harder to adapt this argument to infinite nested words for which we do not have an automaton model capturing unary queries. Thus, our proof is a direct argument based on the *composition method*.

4   Automata models for nested ω-words

Nested ω-word automata Visibly pushdown automata (VPA), with both Büchi and Muller acceptance conditions, were introduced in [5], and shown to be equivalent to MSO, but not necessarily determinizable. The example of a VPL that cannot be accepted by a deterministic automaton [5] can use arbitrarily many calls without matching returns, something that cannot happen in nested words. Then [18] introduced a notion of *stair visibly pushdown automata* (stair VPA) to control such unmatched calls and showed that stair VPAs are determinizable. These models were defined for VPLs, so we first specialize a particular class of stair VPAs [18] to nested words, thereby obtaining a notion of combined nested word automata, that admit determinization. We then use such automata to show that over nested words, for every ω-NWA (with a Büchi or a Muller acceptance condition), there exists an equivalent deterministic Muller ω-NWA.

A *combined nested word automaton* (*CNWA*) puts together an ω-word automaton $\mathcal{A}_1$ with a Muller acceptance condition and a finite NWA $\mathcal{A}_2$. It runs $\mathcal{A}_1$ over all positions that are not inside a call. Every time $\mathcal{A}_1$ finds a call position i, it invokes $\mathcal{A}_2$ to process the finite nested word formed by the elements between i and its matching return j, and then it uses its final state to determine what state to assign to $j + 1$, and continues its run from position $j + 1$. Formally, a CNWA $\mathcal{A}$ over Σ is a pair $(\mathcal{A}_1, \mathcal{A}_2)$, where:

- $\mathcal{A}_2 = (\Sigma, Q_2, Q_2^0, \delta_2 = (\delta_c^2, \delta_\iota^2, \delta_r^2))$ is an NWA without accepting states;
- $\mathcal{A}_1 = (\Sigma \cup Q_2, Q_1, Q_1^0, \delta_1, \mathcal{F}_1)$ is an ω-word automaton over alphabet $\Sigma \cup Q_2$ (we assume, of course, that Σ and Q_2 are disjoint).

Given a nested ω-word $\bar{w}$ and $i \geq 1$, we define the set of *external* positions $E(\bar{w})$ as positions i such that there are no $j, k \geq 1$ such that $j < i \leq k$ and $\eta(j, k)$ holds. Note that $1 \in E(\bar{w})$ and $E(\bar{w})$ is infinite. If $i \in E(\bar{w})$ is not a call, then $i + 1 \in E(\bar{w})$. If $i \in E(\bar{w})$ is a call with j being its matching return, then the next, after i, element of $E(\bar{w})$ is $j + 1$. With this, we define a *run* of $\mathcal{A}$ over a nested ω-word $\bar{w} = (a_1 a_2 \cdots, \eta)$ as a mapping $\rho : E(\bar{w}) \to Q_1$ such that $\rho(1) \in Q_1^0$ and for every $i \in E(\bar{w})$:

- if i is not a call (and $i + 1 \in E(\bar{w})$), then $\rho(i + 1) \in \delta_1(\rho(i), a_i)$;
- if i is a call with return j (and the successor of i in $E(\bar{w})$ is $j + 1$), then $\rho(j + 1) \in \delta_1(\rho(i), q)$, where q is a state in Q_2 such that there exists a run ρ_2 of $\mathcal{A}_2$ over $\bar{w}[i, j]$ having q as the last state.

A CNWA $\mathcal{A}$ accepts $\bar{w}$ if there is a run ρ of $\mathcal{A}$ over $\bar{w}$ such that $Inf(\rho) \in \mathcal{F}_1$. We say that CNWA $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is deterministic if both $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic. Then results in [18] can be restated in this terminology as:

Proposition 1 ([18]). *Over nested ω-words, CNWAs and deterministic CNWAs are equivalent.*

We show, by using standard techniques, that CNWA and MSO are equivalent, from which the main result of this section follows:

Theorem 2. *Over nested ω-words, MSO, ω-NWA and deterministic ω-NWA with Muller acceptance condition, define precisely the regular languages. Moreover, translations between these formalisms are effective.*

Determinization of ω-NWAs is done by translating them into CNWAs, determinizing those (which involves a $2^{O(n \log n)}$ Safra construction [22] and a $2^{O(n^2)}$ determinization procedure from [5]) and then translating back into ω-NWAs with Muller acceptance condition. Putting these three components together, we get (note that the bound is the same as for determinization of stair VPAs for VPLs [18]):

Corollary 3. *For every ω-NWA with n states, there exists an equivalent deterministic ω-NWA with a Muller acceptance condition and with $2^{O(n^2)}$ states.*

It is well-known that a language of ω-words is regular (accepted by a Büchi or a Muller automaton) iff it is a finite union of languages of the form UV^ω, where U, V are regular languages. Automata characterizations imply a similar result for nested ω-words.

Corollary 4. *A language of nested ω-words is regular iff it is a finite union of languages of the form UV^ω where U and V are regular languages of finite nested words.*

A basic problem in automata theory is the nonemptiness problem: is the language accepted by an automaton nonempty? It was shown in [5], that nonemptiness, and more generally reachability problem for visibly pushdown ω-automata, is polynomial. Combining this with a NLOGSPACE algorithm for nonemptiness of ω-word automata, we get polynomial nonemptiness algorithms for ω-NWA and CNWA. Further, a modification of PTIME-hardness reduction for emptiness for context-free grammars gives us:

Corollary 5. *The nonemptiness problem for ω-NWA and CNWA is PTIME-complete.*

It is easy to code a deterministic automaton by an L_μ^1 formula. Thus,

Corollary 6. *Over nested ω-words, L_μ collapses to L_μ^1.*

Alternating automata for nested ω-words In the context of formal verification, alternating automata have proved to be the key to a comprehensive automata-theoretic framework for temporal logics [27]. With the development of temporal logics for nested words [4, 2, 1], it is natural to develop alternating automata for nested words, with the hope that they can simplify the process of translating temporal logics into automata.

We now define alternating automata for both finite and infinite nested words, and show that they are equivalent to NWAs. We note that this is in sharp contrast with the theory of alternating automata for nested trees, where alternating automata are known to be more expressive than nondeterministic automata [3].

First recall the definition of alternating automata for finite and infinite words. Given a set of states Q, let $\mathcal{B}^+(Q)$ be the set of positive Boolean combinations of elements from Q. Given $X \subseteq Q$ and $\varphi \in \mathcal{B}^+(Q)$, we say that X *satisfies* φ if the truth assignment σ_X satisfies φ, where σ_X is defined as $\sigma_X(q) = 1$ iff $q \in X$. Then an *alternating (ω-)word automaton* $\mathcal{A}$ is a tuple $(\Sigma, Q, Q_0, \delta, F)$, where Q, Q_0 and F are defined as for the case of word automata, and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function.

A run of such an automaton is a labeled tree. A Σ-labeled tree T is a pair (D, λ), where $\lambda : D \to \Sigma$ and D is a prefix-closed subset of $\mathbb{N}^*$ such that (1) if $x \cdot i \in D$ and $0 \le j < i$, then $x \cdot j \in D$, and (2) for every $x \in D$, there exists a finite number of strings of the form $x \cdot i$ in D (finite branching). For $x \in \mathbb{N}^*$, its length is denoted by $|x|$. The depth of a tree is $\max_{x \in D} |x|$.

A *run* of an alternating word automaton $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ over $w = a_1 \cdots a_n$ is a finite Q-labeled tree $T = (D, \lambda)$ of depth n such that $\lambda(\varepsilon) \in Q_0$ and for every $x \in D$ that has children $x \cdot 0, \ldots, x \cdot \ell$ of length i, we have that $\{\lambda(x \cdot 0), \ldots, \lambda(x \cdot \ell)\}$ satisfies $\delta(\lambda(x), a_i)$. An alternating word automaton $\mathcal{A}$ accepts a word w if there is a run $T = (D, \lambda)$ of $\mathcal{A}$ over w such that $\lambda(x) \in F$ for every node x in T of length n. The run of an alternating ω-word automaton $\mathcal{A} = (\Sigma, Q, Q_0, \delta, F)$ over an ω-word $w = a_1 a_2 \cdots$ is defined in exactly the same way as an infinite Q-labeled tree $T = (D, \lambda)$. Then $\mathcal{A}$ accepts ω-word w if there is an accepting run $T = (D, \lambda)$ of $\mathcal{A}$ over w, i.e. every infinite branch ρ of T includes infinitely many labels in F.

An *alternating nested word automaton* (or alternating NWA, or ANWA) is an NWA that admits alternation in call, return, and internal transitions. Formally, an ANWA $\mathcal{A}$ is a tuple $(\Sigma, Q, Q_0, \delta, F)$, where Q, Q_0 and F are defined as for the case of alternating word automata, and δ is a triple $(\delta_c, \delta_\iota, \delta_r)$ of transition functions $\delta_c, \delta_\iota : Q \times \Sigma \to \mathcal{B}^+(Q)$, and $\delta_r : Q \times Q \times \Sigma \to \mathcal{B}^+(Q)$. A *run* of $\mathcal{A}$ over $\bar{w} = (a_1 \cdots a_n, \eta)$ is a Q-labeled finite tree $T = (D, \lambda)$ of depth n such that $\lambda(\varepsilon) \in Q_0$ and for every $x \in D$ with children $x \cdot 0, \ldots, x \cdot \ell$, of length $i \le n$:

- if $|x|$ (i.e. $i - 1$) is a call position, then $\{\lambda(x \cdot 0), \ldots, \lambda(x \cdot \ell)\}$ satisfies $\delta_c(\lambda(x), a_i)$;
- if $|x|$ is an internal position, then $\{\lambda(x \cdot 0), \ldots, \lambda(x \cdot \ell)\}$ satisfies $\delta_\iota(\lambda(x), a_i)$;
- if $|x|$ is a return position with matching call j and y is the prefix of x with $|y| = j - 1$, then $\{\lambda(x \cdot 0), \ldots, \lambda(x \cdot \ell)\}$ satisfies $\delta_r(\lambda(x), \lambda(y), a_i)$.

Note that if $i - 1$ is an internal position and x does not have children, then $\delta_\iota(\lambda(x), a_i)$ has to be a tautology, and likewise for call and return positions. An alternating nested

word automaton $\mathcal{A}$ accepts a nested word $\bar{w}$ if there is a run $T = (D, \lambda)$ of $\mathcal{A}$ over $\bar{w}$ such that $\lambda(x) \in F$ for every node x in T of length n.

Proposition 2. *For every alternating NWA, there exists an equivalent NWA.*

This can be extended to nested ω-words. An alternating nested ω-word automaton (ω-ANWA) $\mathcal{A}$ is a tuple $(\Sigma, Q, Q_0, \delta, F)$, where Q, Q_0, δ and F are defined exactly as for ANWA. A *run* is defined in the same way as above, and the acceptance condition again states that along each infinite branch, states from F are seen infinitely often.

Theorem 3. *For every ω-ANWA with n states, there exists (and can be effectively constructed) an equivalent ω-NWA with a Büchi acceptance condition and $2^{O(n^4)}$ states.*

For the proof, we introduce a notion of *alternating combined nested word automaton* (ACNWA) and provide a direct translation from ω-ANWA into ACNWA. Then by using Proposition 2 and the fact that alternating ω-word automata can be translated into ω-word automata [27], we give a translation from ACNWA into CNWA. Theorem 3 then follows from Proposition 1. We note that Theorem 3 provides an exponential-time algorithm for checking nonemptiness of ANWAs and ω-ANWAs. Since even in the finite case the problem is as hard as universality for finite tree automata [24], we get:

Corollary 7. *The nonemptiness problem for both ANWAs and ω-ANWAs is EXPTIME-complete.*

5 Synchronization of nested words

Synchronization of words leads to a concept of *regular relations*. It ties together (synchronizes) positions in several words, and then runs an automaton over them [15]. To be concrete, let $w_1, \ldots, w_k$ be words from Σ^*. Assume that $\#$ is a letter that is not in Σ. Let $n = \max_i |w_i|$, and let $[(w_1, \ldots, w_k)]$ be a word of length n constructed as follows. It is over the alphabet $(\Sigma \cup \{\#\})^k$, and its ith letter is a k-tuple $\boldsymbol{a}_i = (a_1^i, \ldots, a_k^i)$, where each a_j^i is the ith letter of w_j if $i \leq |w_j|$, and $\#$ if $i > |w_j|$. That is, we pad words shorter than n with $\#$'s to make them all of length n, and then take the ith letter of $[(w_1, \ldots, w_k)]$ to be the tuple of the ith letters of these padded words.

Then *regular k-ary relations* over Σ are defined as sets $R \subseteq (\Sigma^*)^k$ such that the set $\{[(w_1, \ldots, w_k)] \mid (w_1, \ldots, w_k) \in R\}$ is accepted by an automaton over the alphabet $(\Sigma \cup \{\#\})^k$ [13, 12]. Such automata are called *letter-to-letter automata*. Regular relations are closed under Boolean combinations, product, and projection. This makes it possible to find infinite structures over Σ^* with decidable first-order theories whose definable sets are precisely the regular relations (these are universal *automatic structures*, cf. [13, 12]). The most commonly used such structure is $\langle \Sigma^*, \prec, (P_a)_{a \in \Sigma}, \text{el} \rangle$, where $\prec$ is the prefix relation, $P_a(w)$ is true iff the last letter of w is a, and $\text{el}(w, w')$ (the equal-length predicate) holds iff $|w| = |w'|$ [13, 12, 10].

We now study synchronization for nested words. We show that the usual letter-to-letter synchronization for words is completely incompatible with the nesting structure because even the simplest nested extension of letter-to-letter automata is undecidable.

We propose a different decidable synchronization scheme for nested words that gives us regular relations with all the expected properties.

Letter-to-letter nested word automata Assume that we have k nested words $\bar{w}_1, \ldots, \bar{w}_k$, and we again pad the shorter words with a special symbol $\#$ so that all of them are of the same length n. Let $[(\bar{w}_1, \ldots, \bar{w}_k)]$ be such a nested word over the alphabet $(\Sigma \cup \{\#\})^k$, and let $\boldsymbol{a}_i$ be its ith letter. The letter-to-letter automaton runs from left to right on $[(\bar{w}_1, \ldots, \bar{w}_k)]$, as an NWA. The main difference with NWAs is that each position i may now be a return position in *several* of the $\bar{w}_j$'s, and thus states in several call positions determine the next state.

That is, in a *k-letter-to-letter NWA* over k-tuples of nested words, we have multiple return transitions $\delta_r^X : Q \times Q^{|X|} \times (\Sigma \cup \{\#\})^k \to 2^Q$, indexed by nonempty $X \subseteq \{1, \ldots, k\}$. Suppose i is a return position in $\bar{w}_{l_1}, \ldots, \bar{w}_{l_m}$, where $1 \leq l_1 < \ldots < l_m \leq k$ and $m > 0$. If $j_1, \ldots, j_m$ are the matching calls, i.e. $\eta_{l_1}(j_1, i), \ldots, \eta_{l_m}(j_m, i)$ hold, then $\rho(i+1)$ depends on $\rho(i)$, $\boldsymbol{a}_i$, and the states in positions $j_1, \ldots, j_m$:

$$\rho(i+1) \in \delta_r^{\{l_1, \ldots, l_m\}}(\rho(i), \rho(j_1), \ldots, \rho(j_m), \boldsymbol{a}_i).$$

For positions without returns, we have one transition $\delta : Q \times (\Sigma \cup \{\#\})^k \to 2^Q$.

We show that even a much simpler automaton is undecidable. Define a *simplified k-letter-to-letter NWA* as a k-letter-to-letter NWA with one return transition is $\delta_r : Q \times Q \times (\Sigma \cup \{\#\})^k \to 2^Q$, just as in the usual NWA. The condition on the run is as follows: if i is a return position in words $\bar{w}_{l_1}, \ldots, \bar{w}_{l_m}$, for $1 \leq l_1 < \ldots < l_m \leq k$, then $\rho(i+1) \in \delta_r(\rho(i), \rho(j_1), \boldsymbol{a}_i)$, where j_1 is the call of i in $\bar{w}_{l_1}$. In other words, we look at the state of only one call position, corresponding to the return with the smallest index. For all other positions we have a single transition $\delta : Q \times (\Sigma \cup \{\#\})^k \to 2^Q$.

If $k = 1$, these are the usual NWAs. But if $k = 2$, they are undecidable.

Theorem 4. *The nonemptiness problem is undecidable for simplified 2-letter-to-letter NWAs (and thus for k-letter-to-letter NWAs for $k > 1$).*

Thus, there is no hope to use even the simplest possible form of letter-to-letter synchronization in nested words. As another example of such incompatibility, we show that there are no natural decidable extensions of universal automatic structures on words to nested words. We look at structures $\mathfrak{M} = \langle \Sigma_{\text{nw}}^*, \Theta \rangle$ (where Σ_{nw}^* is the set of all finite nested words over Σ) of a vocabulary Θ. We assume that Θ includes some basic relations. One is a prefix relation $\bar{w} \preceq_{\text{nw}} \bar{w}'$ iff $\bar{w} = \bar{w}'[1, m]$ for some $m \leq |\bar{w}'|$ (so we can refer to the linear structure of nested words). The other allows us to refer to the nesting structure: we relate a prefix $\bar{w}$ of $\bar{w}'$ so that in $\bar{w}'$, there is a call-return edge from the last position of $\bar{w}$ to the last position of $\bar{w}'$. That is, $\bar{w} \preceq_\eta \bar{w}'$ iff $\bar{w} = \bar{w}'[1, m]$, and $\eta(m, |\bar{w}'|)$ holds in $\bar{w}'$. We say that $\mathfrak{M}$ *defines all regular languages of nested words* if for each such language L, there is a formula $\varphi_L(x)$ such that $L = \{\bar{w} \in \Sigma_{\text{nw}}^* \mid \mathfrak{M} \models \varphi(\bar{w})\}$. We say that $\mathfrak{M}$ *defines all regular relations over words* if for each regular relation $R \subseteq (\Sigma^*)^k$, there is a formula $\psi_R(x_1, \ldots, x_k)$ such that $\mathfrak{M} \models \psi_R(\bar{w}_1, \ldots, \bar{w}_k)\}$ iff $(w_1, \ldots, w_k) \in R$ (recall that w_i is a word from Σ^* obtained by removing the nesting structure from $\bar{w}_i$).

Proposition 3. *There is no structure $\mathfrak{M} = \langle \Sigma_{\text{nw}}^*, \preceq_{\text{nw}}, \preceq_\eta, \ldots \rangle$ that defines all regular languages of nested words, all regular relations over words, and has a decidable theory.*

Call-return synchronization As the usual letter-to-letter synchronization is incompatible with nested words, we propose a different, *call-return* synchronization. Intuitively, instead of synchronizing positions with the same index i in different words, we synchronize positions for which the shortest paths to them (from the first position) are the same. To formalize this, we use a notion of a *summary path* introduced recently in connection with the study of LTL-like logics on nested ω-words [1]. A summary path to a position i in a nested word $\bar{w}$ is the shortest path from 1 to i that combines both successor and matching edges. That is, it is a sequence $1 = i_0 < i_1 < \ldots < i_k = i$ such that, if i_l is a call with $\eta(i_l, j)$ and $i \geq j$, then $\eta(i_l, i_{l+1})$ holds, and otherwise $i_{l+1} = i_l + 1$. We represent this summary path as a word $a_1 \ldots a_k$ over the alphabet $\Lambda = \{\mathsf{i}, \mathsf{c}, \mathsf{m}\}$:

1. if $i_l = i_{l-1} + 1$ and i_{l-1} is not a call, then $a_l = \mathsf{i}$ (path goes via an internal edge);
2. if $i_l = i_{l-1} + 1$ and i_{l-1} is a call, then $a_l = \mathsf{c}$ (path goes via a call edge);
3. if $\eta(i_{l-1}, i_l)$ holds, then $a_l = \mathsf{m}$ (path goes via a matching edge).

If both $i_1 = i_{l-1} + 1$ and $\eta(i_{l-1}, i_l)$ hold, we let a_l be m. The unique summary path to position i will be denoted by $\pi_{\bar{w}}^{\eta}(i) \in \Lambda^*$, and the set of all summary paths by $\Pi^{\eta}(\bar{w})$. The label of $\pi_{\bar{w}}^{\eta}(i)$ is the label of i in $\bar{w}$. Note that $\Pi^{\eta}(\bar{w})$ is closed under prefix.

The idea of the *call-return synchronization* is that now with each position i, we keep its summary paths $\pi_{\bar{w}}^{\eta}(i)$, to remember how it was reached in different nested words. That is, a call-return synchronization of nested words $\bar{w}_1, \ldots, \bar{w}_k$ is a pair $(\Pi^{\eta}(\bar{w}_1, \ldots, \bar{w}_k), \lambda)$ where $\Pi^{\eta}(\bar{w}_1, \ldots, \bar{w}_k) = \bigcup_l \Pi^{\eta}(\bar{w}_l)$, and $\lambda : \Pi^{\eta}(\bar{w}_1, \ldots, \bar{w}_k) \to (\Sigma \cup \{\#\})^k$ is a labeling function that labels each summary path with its label in $\bar{w}_i$ if it occurs in $\bar{w}_i$, and with $\#$ otherwise, for each $i \leq k$. This synchronization can naturally be viewed as a tree.

As an example, consider two nested words below, $\bar{w}_1$ (on the left) and $\bar{w}_2$ (on the right), with summary paths shown above positions.
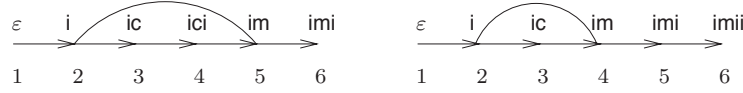
The synchronization occurs in the first and the second position, and we recursively synchronize the calls (from i) and what follows their returns (from im). Intuitively, this results in adding a dummy internal node ici inside the call for $\bar{w}_2$, and adding a dummy last internal position imii for $\bar{w}_2$. Note that position 4 (i.e. ici) in $\bar{w}_1$ is in no way related to position 4 (im) in $\bar{w}_2$, as it would have been in letter-to-letter synchronization.

We now say that $R \subseteq (\Sigma_{\mathrm{nw}}^*)^k$ is a *regular k-ary relation of nested words* iff there is a tree automaton on ternary trees that accepts precisely $(\Pi^{\eta}(\bar{w}_1, \ldots, \bar{w}_k), \lambda)$, for $(\bar{w}_1, \ldots, \bar{w}_k) \in R$. The following is an immediate consequence of coding tree representations in MSO, and of the work on automatic structures over trees [11]:

Proposition 4. – *Regular relations of nested words are closed under union, intersection, complementation, product, and projection.*
 – *Regular 1-ary relations of nested words are precisely the regular nested languages.*
 – *There is a finite collection Θ of unary and binary predicates on Σ_{nw}^* such that $\langle \Sigma_{\mathrm{nw}}^*, \Theta \rangle$ is a universal automatic structure for nested words, i.e. its definable relations are precisely the regular relations of nested words, and its theory is decidable.*

References

1. R. Alur, M. Arenas, P. Barceló, K. Etessami, N. Immerman, L. Libkin. First-order and temporal logics for nested words. In *LICS 2007*.
2. R. Alur, S. Chaudhuri, P. Madhusudan. A fixpoint calculus for local and global program flows. In *POPL 2006*, pages 153–165.
3. R. Alur, S. Chaudhuri, P. Madhusudan. Languages of nested trees. In *CAV 2006*, 329–342.
4. R. Alur, K. Etessami and P. Madhusudan. A temporal logic of nested calls and returns. In *TACAS'04*, 467-481.
5. R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC'04*, 202-211.
6. R. Alur and P. Madhusudan. Adding nesting structure to words. In *DLT'06*, pages 1–13.
7. A. Arnold and D. Niwinski. *Rudiments of μ-calculus*. North-Holland, 2001.
8. V. Bárány, C. Löding, O. Serre. Regularity problems for visibly pushdown languages. *STACS'06*, 420–431.
9. P. Barceló and L. Libkin. Temporal logics over unranked trees. In *LICS'05*, 31-40.
10. M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. Definable relations and first-order query languages over strings. *J. ACM* 50(5): 694–751, 2003.
11. M. Benedikt, L. Libkin, F. Neven. Logical definability and query languages over ranked and unranked trees. *ACM TOCL*, 8(2), 2007. Extended abstract in LICS'02 and LICS'03.
12. A. Blumensath and E. Grädel. Automatic structures. In *LICS'00*, pages 51–62.
13. V. Bruyère, G. Hansel, C. Michaux, R. Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.* 1 (1994), 191–238.
14. R. Cleaveland, B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. *CAV'91*, pages 48–58.
15. C. Elgot and J. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Develop.* 9 (1965), 47–68.
16. D. Janin, I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. *CONCUR 1996*, pages 263–277.
17. C. Lautemann, T. Schwentick, D. Thérien. Logics for context-free languages. *CSL'94*, 205–216.
18. C. Löding, P. Madhusudan, O. Serre. Visibly pushdown games. In *FSTTCS 2004*, 408–420.
19. F. Neven, Th. Schwentick. Query automata over finite trees. *TCS* 275 (2002), 633–674.
20. D. Niwinski. Fixed points vs. infinite generation. In *LICS 1988*, pages 402–409.
21. F. Peng and S. Chawathe. Xpath queries on streaming data. In *SIGMOD'03*, pages 431–442.
22. S. Safra. On the complexity of omega-automata. In *FOCS 1988*, pages 319–327.
23. L. Segoufin, V. Vianu. Validating streaming XML documents. In *PODS'02*, pages 53–64.
24. H. Seidl. Deciding equivalence of finite tree automata. *SICOMP* 19(3): 424-437 (1990).
25. W. Thomas. Languages, automata, and logic. *Handbook of Formal Languages, Vol. 3*, 1997.
26. W. Thomas. Infinite trees and automaton-definable relations over ω-words. *TCS* 103 (1992), 143–159.
27. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. Banff Higher Order Workshop 1995, pages 238-266.
28. M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP 1998*, 628–641.