A Temporal Logic for Multi-threaded Programs

Salvatore La Torre, Margherita Napoli

A Temporal Logic for Multi-threaded Programs*

Salvatore La Torre and Margherita Napoli

Dipartimento di Informatica, Università degli Studi di Salerno, Italy

Abstract. Temporal logics for nested words are a specification formalism for procedural programs, since they express requirements about matching calls and returns. We extend this formalism to multiply nested words, which are natural models of the computations of concurrent programs. We study both the satisfiability and the model-checking problems, when the multiply nested words are runs of multi-stack pushdown systems (MPDS). In particular, through a tableau-based construction, we define a Büchi MPDS for the models of a given formula. As expected both problems are undecidable, thus we consider some meaningful restrictions on the MPDS, and show decidability for the considered problems.

1 Introduction

Temporal logic is a standard specification language in program verification. Traditional linear time temporal logic (LTL) [14] allows to express ω-regular properties and recent research has enriched this formalism with temporal operators that allow to take into account the call-return structure of the control flow in sequential programs with recursive procedure calls, such as CARET [2] and NWTL [1]. In these *call-return* temporal logics, the key intuition is to look at the program computations not simply as a word but as a *nested word*, which is essentially a graph with two kinds of edges: the *linear* edges capturing the sequential structure of computations (total ordering among the program states), and the *call-return* edges, that connect a call to its matching return in the computation (defining a matching relation). The LTL operators refer only to the total ordering given by the linear edges. The mentioned call-return logics instead present versions of the standard LTL operators which refer to the call-return edges, and thus properties such as "a procedure A is always invoked through calls of a procedure B" and "a write to a variable x should be followed by a read of x in the same procedure invocation", become expressible.

In concurrent programs communicating through a shared memory and with recursive procedure calls, each thread has its own control flow structured into procedure calls. A suitable model for the computations of such programs are the *multiply nested words*, that is graphs with linear edges and different kinds of call-return edges (one for each thread and each kind defining a different matching relation). Along the line of the call-return temporal logics, we can define

* This work was partially funded by the MIUR grants FARB 2010-2011 Università degli Studi di Salerno (Italy).

extensions of temporal operators that refer to the different types of call-return edges.

In this paper, we consider the logic MultiCaRet that extends CaRet to multiply nested words, and study both the satisfiability and the model-checking problems. We model the programs as multi-stack pushdown systems (Mpds). The computations of Mpds naturally define multiply nested words, and thus, the model checking question we ask is: given a MultiCaRet formula φ and a Mpds M, do all the nested words generated by M satisfy φ?

We face these problems using the automata-theoretic approach. In particular, for a formula φ over n-nested words (i.e., multiply nested words with n matching relations), we give a tableau-based construction that defines a Büchi Mpds M_φ which generates all the n-nested words satisfying φ. We show that the size of M_φ is linear in n and exponential in the size of φ. Defining with $\mathcal{L}(M)$ the language of multiply nested words defined by M, satisfiability is reduced to checking the non-emptiness of $\mathcal{L}(M_\varphi)$ and model-checking reduces to checking the emptiness of $\mathcal{L}(M) \cap \mathcal{L}(M_{\neg\varphi})$. Observe that, since the push and pop transitions of a Mpds are visible on a generated multiply nested word, by a standard cross product synchronized on the labels, we can construct a Büchi Mpds that generates the intersection language. Therefore, both the considered problems reduce to solving the emptiness problems for Büchi Mpds.

Unfortunately, the emptiness problem for Mpds is known to be undecidable already with two stacks (two stacks are sufficient to encode the computations of a Turing machine), and this can be used to show that indeed also the MultiCaRet model-checking problem is undecidable. Moreover, we prove that also MultiCaRet satisfiability is undecidable with a simple reduction from PCP. We thus consider some known restrictions that have been recently studied to obtain decidable models of Mpds that are meaningful when dealing with non-terminating computations. In particular, we look at multiply nested words that correspond to Mpds computations where each symbol that is popped has been pushed into a stack during the last k contexts (*scope-bounded multiply nested words*) [11] or where a symbol is popped from stack i if all stacks $j < i$ are empty (*ordered multiply nested words*) [8]. Observe that, both the restrictions do not limit the number of execution contexts where we can do push or pop transitions of any stack and thus are suitable to take into account non-trivial infinite interactions among the different threads of a program.

It is known that the emptiness problem for Büchi Mpds restricted to ordered multiply nested words is decidable in time doubly exponential in the number of stacks and polynomial in the number of states, and is 2Etime-complete [3] (we recall that 2Etime denotes the class of all the decision problems which are solvable by a deterministic Turing machine in time $2^{2^{O(n)}}$). Combining this result with our construction we get that the MultiCaRet satisfiability and model checking problems on ordered multiply nested words are 2Etime-complete.

A further contribution of our paper is to solve the emptiness problem for Büchi Mpds restricted to scope-bounded multiply nested words. We reduce this problem to checking the emptiness of a standard Büchi automaton of size expo-

nential in the number of stacks and the bound k on the scope of the matching relations. The solution relies on defining for each stack a pushdown automaton whose reachable states correspond to *thread interfaces* of dimension at most k (tuples of pairs summarizing the control states when context-switching into and out of a thread along with the information if in a context an accepting state has been seen). The entire contribution of a thread to a k-scoped computation of the MPDS can be summarized with an infinite thread interface which is the composition of thread interfaces of dimension at most k (see [12]). Thus, we define an automaton that nondeterministically selects a thread interface of dimension k (k-*thread-interface*) for each thread and simulates a computation modifying the current state by applying the next pair of one such thread interface. Once a k-thread-interface is completely used, a new k-thread-interface is nondeterministically selected for that thread. In this simulation, we make accepting all the states that are introduced by a pair corresponding to a context where an accepting state has been visited. By the above described reduction, we thus have that the MULTICARET satisfiability and model-checking problems are both decidable in time exponential in the size of the formula, the number of stacks, and the bound k. Since this problems are already EXPTIME-hard for CARET [2], we get EXPTIME-completeness.

As a final contribution, we show that the logic MULTICARET can be expressed in FO, and since MSO is decidable on all classes of MSO-definable multiply nested words of bounded tree-width [13], we also get that MULTICARET is decidable for all such classes. We recall that the class of ordered multiply nested words defined by a MPDS have bounded tree-witdh and are MSO-definable [13], and so do the class of scope-bounded multiply nested words [12], therefore the decidability of MULTICARET satisfiability and model-checking for these classes can also be obtained using these arguments.

Related work. Besides the already cited research there are few other papers that are related to ours.

A general temporal logic for concurrent programs which subsumes MULTICARET is introduced in [6]. However, the decidability results there are obtained by restricting the computations to a bounded number of *phases* [10], where in each phase pop transitions are all from the same stack. The phase restriction looks quite limiting when considering infinite computations: the last phase is infinite and there is no unbound alternation between popping different stacks, i.e., from some point there is only one stack that is really used. Besides, it is orthogonal to the scope-bounded restriction and does not allow to express more behaviors than placing an ordering on the matching relations [4]. It is not known a relation between these two last restrictions. We also note that the approach followed here is completely different from [6].

Another concurrent extension of CARET is considered in [7]. This logic differ from MULTICARET both in the syntax and the semantics. The model checking question for LTL formulas with respect to ordered MPDS is settled in [3]. The scope-bounded restriction strictly extends the notion of bounded-context switch that have been successfully used in finding bugs in concurrent programs [15].

The scope-bounded restriction on matching relation defined here is more permissive than that introduced in [11]. Here, we do not bind this definition to a round-based organization of computations (a round is a sequence of contexts where each stack is active exactly once). Therefore, while there the number of context-switches between a push and its matching pop is always bounded (nk where n is the number of stacks and k is the bound on the scope of the matching relations), here we can have unboundedly many context switches. As an example, consider a sequence $a_1(a_2a_3b_2b_3)^m b_1$ where a_i is a push and b_i is a pop of the i-th stack. For each m, this satisfies the 2-scoped restriction given in this paper, while it does not satisfy the $2m$-scoped restriction given in [11].

Simultaneously and independently of our research, in [5] the model-checking of linear-time temporal properties (LTL) for MPDS under the scope-bounded restriction has been shown to be EXPTIME-complete. This result can be obtained as a corollary of our results since our logic MULTICARET syntactically subsumes CARET and thus LTL.

2 A temporal logic over multi-nested words

Given two positive integers i and j, $i \leq j$, we denote with $[i, j]$ the set of integers k with $i \leq k \leq j$, and with $[j]$ the set $[1, j]$.

Multiply nested words. In this section, we recall the concept of multiply nested word which a natural formalism for expressing the computations of multi-stack pushdown systems, which in turns carefully capture the flow of control in concurrent programs with shared memory and recursive procedure calls.

Fix an alphabet Σ, an *infinite word* over Σ is a mapping that assigns to each *position* $i \in \mathbb{N}$ a symbol $\sigma_i \in \Sigma$, and is denoted as $\{\sigma_i\}_{i\in\mathbb{N}}$ or equivalently $\sigma_1, \ldots, \sigma_n \ldots$. Each infinite word defines a linear ordering among its positions which corresponds to the ordering induced by the relation $<$ over $\mathbb{N}$. In this paper, we make use only of the infinite words therefore we will refer to them also simply as words. A *multiply nested word* is a word equipped with one or more matching relations. For a word $\{\sigma_i\}_{i\in\mathbb{N}}$, a matching relation expresses a relation between two disjoint sets of its positions, the calls and the returns, such that each call i is matched to at most one return j that follows i in the linear ordering (i.e., $i < j$) and each return i is matched to at most one call j that precedes i in the linear ordering (i.e., $j < i$), and matched calls and returns are well-nested. Formally, a *matching relation* over $\mathbb{N}$ is a triple (μ, C, R) where $C, R \subseteq \mathbb{N}$ (respectively, the set of calls and the set of returns of the relation), $C \cap R = \emptyset$, and $\mu \subseteq C \times R$ is such that for all $i, j, h \in N$:

- $\mu(i, j)$ implies $i < j$ (respects the linear ordering of w);
- $\mu(i, j)$ and $\mu(i, h)$ implies $j = h$ (each call matches at most one return);
- $\mu(i, j)$ and $\mu(h, j)$ implies $i = h$ (each return matches at most one call);
- $i \leq j$, $i \in C$, and $j \in R$ implies that there is a $i \leq k \leq j$ such that either $\mu(i, k)$ or $\mu(k, j)$.

An *n-nested word* w is $(\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [n]\})$ where $C_1, R_1, \ldots, C_n, R_n$ are pairwise disjoint and for $i \in [n]$, μ_i is a matching relation with set of calls C_i and set of returns R_i. A *multiply nested word* is an n-nested word for some $n \in \mathbb{N}$. A 1-nested word is also called a *nested word* [1].

It is simple to see that a multiply nested word $w = (\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [n]\})$ can be graphically represented as a labeled directed graph $G_{nw} = (\mathbb{N}, E, \lambda)$ where: $\mathbb{N}$ is the set of vertices, E is the set of edges and is defined as the union of all μ_i along with the set $\{(i, i+1) \mid i \in \mathbb{N}\}$ denoting the linear ordering induced by $<$, and the labeling function $\lambda : \mathbb{N} \to \Sigma$ that maps each vertex i to σ_i.

In Fig. 1, we report the initial fragment of a 2-nested word where μ_1 matches each occurrence of a's with an occurrence of b, μ_2 matches each occurrence of c with an occurrence of d, and the occurrences of e stay unmatched. The calls and the returns are the positions of respectively the a'a and the b's for μ_1, and the c's and the d's for μ_2. We use subscripts to stress the position of each symbol. The relation μ_1 is denoted with the full curved edges and the relation μ_2 with dashed curved edges.

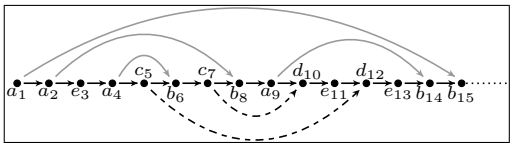

Fig. 1. A fragment of a 2-nested word.

Paths in multiply nested words. Different kinds of paths can be defined in multiply nested words depending on the notion of successor which is used. In this section we define the different notions of successors that will be used to give the semantics of the temporal logic we introduce in the next section. These successors are the adaptation to multiply nested words of those defined on nested words for defining the logic CaRet [2].

Fix a multiply nested word $w = (\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [n]\})$. The first kind of successor we consider is the *linear successor* that is defined by the linear order induced by $<$ on $\mathbb{N}$. The linear successor of a position $j \in \mathbb{N}$ is simply $j+1$.

The *abstract successor* over the i-th matching relation of a position j, denoted $next_i^a(j)$, is defined as the abstract successor in the nested word which is obtained by ignoring all the matching relations but the i-th one. In particular, it is the matching return for matched calls, is not defined on unmatched calls and on linear predecessors of matched returns, and is the linear successor in all the other cases (i.e. for positions that are not calls and whose linear successors are not returns). For each position $j \in \mathbb{N}$, $next_i^a(j)$ is: h if $\mu_i(j, h)$ holds; otherwise, $\perp$ (*undefined*) if either $j+1 \in R_i$ and $\mu_i(h, j+1)$ holds for some $h < j$, or $j \in C_i$ and $\mu_i(j, h)$ does not hold for each $h > j$; and $j+1$ in all the remaining cases.

Analogously to the abstract successors, the *call successor* over the i-th matching relation of a position j, denoted $next_i^-(j)$, is defined as the call successor in the nested word which is obtained by ignoring all the matching relations but the i-th one. In particular, it is the largest call h that precedes j and is not matched up to j, if any, and is undefined otherwise. Formally, for each position $j \in \mathbb{N}$, $next_i^-(j)$ is the largest $h < j$ such that $h \in C_i$ and either $\{k \mid \mu_i(h, k)\} = \emptyset$ (h is unmatched) or $\mu_i(h, k)$ holds for some $k > j$ (the call is not matched yet), if any, and $\perp$ otherwise.

The temporal logic MultiCaRet. Multiply nested words naturally arise as models of the computations of concurrent threads communicating through shared memory. In this interpretation, the nesting relations capture the call-return relation of the recursive procedure calls within each thread. We will make this more precise in the next section where we will model such computations as multi-stack pushdown systems.

We use multiply nested words to interpret the formulas of the logic MultiCaRet which extends the logic CaRet [2] to express properties of multi-threaded programs. The logic MultiCaRet (CaRet for multiply nested words) has the usual linear temporal logic modalities according to the linear ordering and CaRet modalities indexed over the different matching relations. For each $i \in [n]$, we use the atomic propositions $call_i$ and ret_i to identify the respectively a call and a return of the i-th matching relation of an n-nested word.

Formally, we fix a finite sets of *atomic propositions* AP and $\{call_i, ret_i \mid i \in [n]\}$, for $n \in \mathbb{N}$. The formulas of MultiCaRet are inductively defined as follows:

$$\varphi ::= p \mid call_i \mid ret_i \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \bigcirc^b \varphi \mid \varphi \,\mathcal{U}^b\, \varphi$$

where $i \in [n]$, $b \in \{a_i, -_i \mid i \in [n]\}$ and $p \in AP$.

The constants true and false are defined as abbreviations: $\top \equiv \varphi \vee \neg\varphi$ and $\bot \equiv \varphi \wedge \neg\varphi$. Other common abbreviations are $\Diamond^b\varphi \equiv \top \mathcal{U}^b\, \varphi$ and $\Box^b\varphi \equiv \neg\Diamond^b\neg\varphi$. The semantics of the logic operators is given as usual. Each of the introduced temporal operators correspond to one of the given notions of successor. In particular, the *global* modalities $\bigcirc$ and $\mathcal{U}$ refer to the linear successor, the *abstract* modalities $\bigcirc^{a_i}$ and $\mathcal{U}^{a_i}$ to the abstract successor, and the *call* modalities $\bigcirc^{a_i}$ and $\mathcal{U}^{a_i}$ to the call successor. Formally, fix an n-nested word $w = (\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [n]\})$ over the alphabet 2^{AP}. The truth value of a formula w.r.t. a position $i \in \mathbb{N}$ in w is defined as follows:

- $(w,i) \models p$ iff $p \in \sigma_i$ (where $p \in AP$);
- $(w,i) \models call_j$ (resp. ret_j)iff $i \in C_j$ (resp. $i \in R_j$);
- $(w,i) \models \neg\varphi$ iff $(w,i) \models \varphi$ does not hold;
- $(w,i) \models \varphi_1 \vee \varphi_2$ iff either $(w,i) \models \varphi_1$ or $(w,i) \models \varphi_2$:
- $(w,i) \models \bigcirc\varphi$ iff $(w, i+1) \models \varphi$;
- $(w,i) \models \bigcirc^{b_j}\varphi$ (with $b \in \{a, -\}$) iff $next_j^b(i) \neq \bot$ and $(w, next_j^b(i)) \models \varphi$;
- $(w,i) \models \varphi_1 \mathcal{U} \varphi_2$ iff there exists a $h \geq i$ such that $(w,h) \models \varphi_2$ and $(w,k) \models \varphi_1$ for all $k \in [i, h-1]$;
- $(w,i) \models \varphi_1 \mathcal{U}^{b_j} \varphi_2$ (with $b \in \{a, -\}$) iff there exists a $h \in \mathbb{N}$ such that $(w,h) \models \varphi_2$, $i = x_1, x_2, \ldots, x_m = h$ where $x_{k+1} = next_j^b(x_k)$ for $k \in [m-1]$, and $(w, x_k) \models \varphi_1$ for $k \in [m-1]$.

We say that a multiply nested word w *satisfies* a formula φ, written $w \models \varphi$, if $(w, 1) \models \varphi$.

Satisfiability. The satisfiability problem for MultiCaRet formulas is defined as the problem of determining if given a MultiCaRet formula φ there exists a multiply nested word w such that $w \models \varphi$. This problem turns out to be undecidable already for formulas using only two matching relations. A proof

of this can be obtained by reducing the*Post's Correspondence Problem* (PCP). Given a set of pairs (u_i, v_i), $i \in [m]$, where $u_i, v_i \in \Sigma^*$ for a finite alphabet Σ, the PCP consists of determining if there is a sequence of indices $i_1, \ldots, i_h$ such that $u_{i_1} \ldots u_{i_h} = v_{i_1} \ldots v_{i_h}$.

The reduction consists of writing a formula that is satisfied only on multiply nested words $(\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [2]\})$ where denoting the word $\{\sigma_i\}_{i\in\mathbb{N}}$ as the concatenation $\alpha\beta\gamma$: $\alpha \in (\bigcup_{i\in[m]}(i.u_i.v_i))^*$ and $\beta \in \{\sigma.\sigma \mid \sigma \in \Sigma\}^*$; each position of the α part is in C_1, if it is labeled with a piece of a u_i, and is in C_2, if it is labeled with a piece of v_i; the positions in β are alternately in R_1 and R_2; none of the positions of the γ part is either a call or a return; there are no unmatched calls and returns; for $i \in [2]$, if $\mu_i(x, y)$ holds then $\sigma_x = \sigma_y$. It is simple to write a MultiCaRet formula φ_{PCP} that checks all the above properties. By a simple proof one can prove that the considered PCP instance admits a solution iff φ_{PCP} is satisfiable. Moreover, the formula is parameterized on the PCP instance, therefore, this construction reduces the PCP to MultiCaRet satisfiability using only two matching relations:

Theorem 1. *The* MultiCaRet *satisfiability problem is undecidable already with two matching relations.*

Expressing properties of multi-threaded programs. The main motivation for MultiCaRet is to introduce a suitable temporal logic for multi-threaded programs. MultiCaRet is the natural extension of CaRet with abstract and caller modalities over many matching relations, and thus can also capture the usual linear time temporal logic properties. A typical correctness requirement that can be expressed in MultiCaRet consists of the pairs of pre- and post-conditions that must be fulfilled by procedure invocations within each computation. For instance, we can require that a procedure A must satisfy a pre-condition p_A upon invocation and a post-condition q_A on returning from a call (note that as an additional correctness requirement this also implies that it must return from each call), and this has to hold for each thread. We can express this as: $\Box \bigwedge_i [(call_i \wedge p_A) \rightarrow \bigcirc^{a_i} q_A]$. Variations of such property requiring different pre/post-conditions for different threads, or limiting the request only to some threads, or admitting that some call may be not returned, can be easily designed.

Additional correctness properties can be required when two procedures are simultaneously invoked in different threads:

$$\Box \bigwedge_{i\neq j} [(\bigcirc^{-i} p_A \wedge \bigcirc^{-j} p_B) \rightarrow (\bigcirc^{-i} \bigcirc^{a_i} q_A \wedge \bigcirc^{-j} \bigcirc^{a_j} q_B)].$$

The temporal modalities based on the call successors allows to express properties on the contents of the stacks, which can be used to specify a variety of security properties. For instance, the requirement that a procedure A should be invoked only when in each thread i a call to a procedure B_i is still pending and no overriding call to procedure C_i is happening can be expressed by the formula $\Box[(\bigvee_i call_i \wedge p_A) \rightarrow \bigwedge_i (\neg p_{C_i} \, \mathcal{U}^{-i} p_{B_i})]$.

3 MultiCaRet model-checking

In this section we first recall the definition of multi-stack pushdown systems and show how their runs define multiply nested words. Then we consider the MultiCaRet model checking problem.

Multistack Pushdown Systems A multi-stack pushdown system consists of a finite control along with one or more stacks and is equipped with a labelling function of its states. The system can push a symbol on any of its stacks, or pop a symbol from any of them, or just change its control location by maintaining unchanged the stack contents. Thus there are several push/pop functions (one for each stack) and one internal action function. We also allow pop transitions on empty stack to take into account the unmatched returns. This is modeled with a bottom-of-the-stack symbol $\gamma^\perp$ which is never removed from the stack.

Let $n \in \mathbb{N}$. A *n-stack pushdown system* (n-Mpds) M is a tuple $(Q, Q_0, \Gamma \cup \{\gamma^\perp\}, \Sigma, \lambda, \delta^{int}, \{(\delta_i^{push}, \delta_i^{pop})\}_{i\in[n]})$ where Q is a finite set of control states, $Q_0 \subseteq Q$ is the set of initial states, Γ is a finite stack alphabet, $\gamma^\perp$ is the bottom-of-the-stack symbol, Σ is the alphabet of the state labels, $\lambda : Q \to \Sigma$ is a labelling function, $\delta_{int} \subseteq (Q \times Q)$ is a set of internal transitions and, for every $i \in [n]$, $\delta_i^{push}, \subseteq (Q \times \Gamma \times Q)$ and $\delta_i^{pop} \subseteq (Q \times \Gamma \cup \{\gamma^\perp\} \times Q)$ are respectively push and pop transitions involving the i'th stack. A Pds is a n-Mpds with $n = 1$.

A *configuration* of M is a tuple $\mathcal{C} = \langle q, \{w_i\}_{i\in[n]}\rangle$, where $q \in Q$ is the state of the configuration and each $w_i \in \Gamma^*.\{\gamma^\perp\}$ is the content of the i'th stack. Moreover, $\mathcal{C}$ is *initial* if $q \in Q_0$ and $w_i = \gamma^\perp$ for every $i \in [n]$. Let $Act = \bigcup_{i\in[n]}\{push_i, pop_i\} \cup \{int\}$ be the set of all *actions* of M. A transition between two configurations over an action $act \in Act$ is defined as follows:
$\langle q, \{w_i\}_{i\in[n]}\rangle \xrightarrow{act}_M \langle q', \{w'_i\}_{i\in[n]}\rangle$ if one of the following holds for some $i \in [n]$
[Internal] $act = int$, $(q, q') \in \delta_{int}$, and $w'_h = w_h$ for every $h \in [n]$.
[Push] $act = push_i$, $(q, \gamma, q') \in \delta_i^{push}$, $w'_i = \gamma.w_i$, and $w'_h = w_h$ for $h \in ([n]\backslash\{i\})$.
[Pop] $act = pop_i$, $(q, \gamma, q') \in \delta_i^{pop}$, $w'_h = w_h$ for $h \in ([n] \setminus \{i\})$, and either $w_i = \gamma.w'_i$ or $w_i = w'_i = \gamma = \gamma^\perp$.

A *run* ρ of M is a possibly empty sequence of transitions $\mathcal{C}_0 \xrightarrow{act_1} \mathcal{C}_1 \xrightarrow{act_2} \ldots$. Furthermore, ρ is a *computation* of M if $\mathcal{C}_0$ is initial.

MultiCaRet *model-checking* Each run $\rho = \mathcal{C}_0 \xrightarrow{act_1} \mathcal{C}_1 \xrightarrow{act_2} \ldots$ of M defines a multiply nested word $\langle\langle\rho\rangle\rangle = (\{\sigma_s\}_{s\in\mathbb{N}}, \{\langle\mu_i^\rho, C_i^\rho, R_i^\rho\rangle \mid i \in [n]\})$ with $\sigma_s = \lambda(q_s)$, where q_s is the state of the configuration $\mathcal{C}_s$ in ρ, $C_i = \{s \mid act_s = push_i\}$, $R_i = \{s \mid act_s = pop_i\}$ and $\mu_i^\rho(s, t)$ holding true if the t'th transition of ρ pops the symbol pushed on stack i at the s'th transition. With $\mathcal{L}(M)$ we denote the language $\{\langle\langle\rho\rangle\rangle \mid \rho \text{ is a computation of } M\}$. Moreover, we say that M satisfies a MultiCaRet formula φ, written $M \models \varphi$, if $w \models \varphi$ holds for each $w \in \mathcal{L}(M)$. Thus, the *model-checking* problem for Mpds and MultiCaRet formulas is:

Given a Mpds M *and a* MultiCaRet *formula* φ*, does* $M \models \varphi$*?*

From the undecidability of reachability of Turing machines and the fact that a 2-Mpds can simulate a Turing machine, the following theorem holds.

Theorem 2. *The model-checking problem for* 2-MPDS *and* MULTICARET *formulas is undecidable.*

Büchi MPDS. For a MPDS M, a *Büchi condition* is a subset $\mathcal{F}$ of the set of states of M. A Büchi MPDS is a MPDS along with a Büchi condition. Denoting with $\mathcal{F}^{\mathcal{C}}$ the set configurations of M of the form $\mathcal{C} = \langle q, \{w_i\}_{i\in[n]}\rangle$ where $q \in \mathcal{F}$, we say that a run $\rho = \mathcal{C}_0 \xrightarrow{act_1} \mathcal{C}_1 \xrightarrow{act_2} \dots$ is accepted by the Büchi MPDS $(M, \mathcal{F})$ (or equivalently satisfies a Büchi condition $\mathcal{F}$) if for infinitely many s, $\mathcal{C}_s \in \mathcal{F}^{\mathcal{C}}$. If M is a Büchi MPDS, we extend the notation $\mathcal{L}(M)$ by requiring that the nested words in $\mathcal{L}(M)$ also satisfy the Büchi condition.

The problem of determining the existence of an accepting run for a given Büchi MPDS (*emptiness problem*) is in general undecidable, again from the undecidability of reachability problem for Turing machines.

4 Büchi MPDS for MULTICARET formulas

In this section, we give a tableau-based construction of a Büchi MPDS which generates the multiply nested words satisfying a given MULTICARET formula. We fix a formula φ over the set of atomic propositions $AP \cup \{call_i, ret_i \mid i \in [n]\}$, for $n \in \mathbb{N}$, and denote with top_i, for $i \in [n]$, a new atomic proposition. The closure of φ, denoted cl_φ, is the smallest set of formulas that contains φ, $\neg\varphi$, top_i, $call_i$ and ret_i for $i \in [n]$, and satisfies the properties described in Fig. 2 (where $\neg\neg\psi$ is identified with ψ and $b \in \{a_i, -_i \mid i \in [n]\}$).

1. $\neg\psi \in cl_\varphi$ if and only if $\psi \in cl_\varphi$
2. if $\psi_1 \vee \psi_2 \in A$ if and only if $\psi_1 \in A$ or $\psi_2 \in A$
3. if $\bigcirc\psi \in cl_\varphi$ or $\bigcirc^b\psi \in cl_\varphi$ then $\psi \in cl_\varphi$
4. if $\psi_1 \,\mathcal{U}\, \psi_2 \in cl_\varphi$ then $\psi_1, \psi_2, \bigcirc(\psi_1 \,\mathcal{U}\, \psi_2) \in cl_\varphi$
5. if $\psi_1 \,\mathcal{U}^b\, \psi_2 \in cl_\varphi$ then $\psi_1, \psi 2, \bigcirc^b(\psi_1 \,\mathcal{U}^b\, \psi_2) \in cl_\varphi$

Fig. 2. Properties defining cl_φ

An atom A of φ is a maximal and logically consistent subset of cl_φ (see Fig. 3). We denote the set of all atoms of φ as $Atoms_\varphi$, and the set of all atoms that contain a formula of the form $\bigcirc^{a_i}\psi$ or $\psi_1 \mathcal{U}^{a_i} \psi_2$, for some $i \in [n]$, as $Atoms^a_\varphi$. Calls and returns of the i-th matching relation are identified by atoms containing respectively $call_i$ and ret_i.

1. for each $\psi \in cl_\varphi$, either $\neg\psi \in A$ or $\psi \in A$
2. A contains at most one among $\{call_i, ret_i \mid i \in [n]\}$
3. $\psi_1 \vee \psi_2 \in A$ iff $\psi_1 \in A$ or $\psi_2 \in A$
4. $\psi_1 \,\mathcal{U}\, \psi_2 \in A$ iff $\psi_2 \in A$ or $\psi_1, \bigcirc(\psi_1 \,\mathcal{U}\, \psi_2) \in A$
5. $\psi_1 \,\mathcal{U}^b\, \psi_2 \in A$ iff $\psi_2 \in A$ or $\psi_1, \bigcirc^b(\psi_1 \,\mathcal{U}^b\, \psi_2) \in A$
6. if $\bigcirc^{a_i}\psi \in A$ then $call_i \in A$.

Fig. 3. An atom A of φ

The main idea in the construction of an MPDS M_φ which defines the set of models of φ, is to mimic the labeling of each position of a multiply nested word with the atom of φ that contains exactly all the formulas of cl_φ that are fulfilled from there. Therefore, the states of M_φ are exactly the atoms of φ. The state labelling function of M_φ labels each atom A of φ with $A \cap AP$ (the set of atomic propositions contained in A).

The set of initial states $Atoms^0_\varphi$ contains all the atoms A of φ such that $\varphi \in A$, and for all $i \in [n]$, $top_i \in A$ and no formula of the form $\bigcirc^{-_i}\psi$ belongs to A. Actually, each atomic proposition top_i is used to mark the top positions for the i-th matching relation, i.e., the positions that are not in between a call of the i-th relation and its matching return. (Observe that all the top_i positions

form an infinite sequence of linearly ordered positions that are related by the abstract successor $next_i^a$, and among all the maximal such sequences -paths- this is the only one which is infinite.)

Besides the bottom-of-the-stack symbol $\gamma^\perp$, for the stack symbols, we take only the atoms that contain sub-formulas of the form $\bigcirc^{a_i}\psi$ or $\psi_1 \mathcal{U}^{a_i}\psi_2$ and a new symbol ∂. This symbol is never popped from a stack and is used as a placeholder for unmatched calls.

The transition functions are defined such that states and stack contents are consistently updated to ensure the correct propagation of the next modalities and the correct labeling with the top_i propositions.

- Each internal transition (A, A') is such that: $\bigcirc\psi \in A$ iff $\psi \in A'$ (global formulas propagation) and for each $i \in [n]$: $call_i, ret_i \notin A$ (internal moves are not from calls or returns), $top_i \in A$ iff $top_i \in A'$ (top_i status is preserved), $\bigcirc^{a_i}\psi \in A$ iff $\psi \in A'$ (abstract formulas propagation), and A and A' contain the same formulas of the form $\bigcirc^{-i}\psi$ (call formulas propagation). Moreover, if $ret_i \in A'$ and $top_i \notin A$, then A must not contain any formula of the form $\bigcirc^{a_i}\psi$ (undefined i-abstract successor of A).
- Each push transition (A, B, A') of stack $i \in [n]$ satisfies the following. $call_i \in A$ (push transitions are from calls), $\bigcirc\psi \in A$ iff $\psi \in A'$ holds (global formulas propagation) and for $j \neq i$: $top_j \in A$ iff $top_j \in A'$ (top_j status is preserved on i push transitions), A and A' contain the same formulas of the form $\bigcirc^{-j}\psi$ (same j-call successor), $\bigcirc^{a_j}\psi \in A$ iff $\psi \in A'$ (propagation of j-abstract formulas). Moreovoer, $\bigcirc^{-i}\psi \in A'$ iff $\psi \in A$ (i-call formulas update), and if $B \neq \partial$ (i.e. the call is matched), then: $top_i \in A$ iff $top_i \in B$ and $\neg top_i \in A'$, A and B contain the same formulas of the form $\bigcirc^{-i}\psi$, and $\bigcirc^{a_i}\psi \in A$ iff $\psi \in B$. Otherwise, i.e., $B = \partial$ (the call is not matched), $top_i \in A$ and $top_i \in A'$ (unmatched calls are all at top positions), and A does not contain formulas of the form $\bigcirc^{a_i}\psi$.
- Each pop transition (A, B, A') of stack $i \in [n]$ is such that: $ret_i \in A$ (pop transitions are from returns); $B \neq \partial$ (∂ cannot be popped out of any stack); $\bigcirc\psi \in A$ iff $\psi \in A'$ (global formulas are propagated from A);
  for $j \neq i$: $top_j \in A$ iff $top_j \in A'$ (top_j status is preserved on i pop transitions), A and A' contain the same formulas of the form $\bigcirc^{-j}\psi$ (same j-call successor), $\bigcirc^{a_j}\psi \in A$ iff $\psi \in A'$ (propagation of j-abstract formulas);
  if $B \neq \gamma^\perp$ (A is a matched return), then: $top_i \in B$ iff $top_i \in A'$ (A' gets the top_i status of its matching call), $\bigcirc^{a_i}\psi \in B$ iff $\psi \in A'$ (i-abstract formulas are propagated from matching call), and B and A' contain the same formulas of the form $\bigcirc^{-i}\psi$ (matching call and return have the same i-call successor);
  if $B = \gamma^\perp$ (A is an unmatched return), then: $top_i \in A$ and $top_i \in A'$ (unmatched returns are top), $\bigcirc^{a_i}\psi \in A$ iff $\psi \in A'$ (i-abstract formulas are propagated from A), and A and A' do not contain formulas of the form $\bigcirc^{-i}\psi$.

The fulfillment of formulas of the form $\psi_1 \mathcal{U} \psi_2$ and, only on the top_i positions, of formulas of the form $\psi_1 \mathcal{U}^{a_i} \psi_2$ is ensured with the addition of a Büchi condition.

In particular, for each formula of the form $\psi_1 \mathcal{U} \psi_2 \in cl_\varphi$, we define an acceptance set with all the atoms containing either ψ_2 or $\neg(\psi_1 \mathcal{U} \psi_2)$, and for each formula of the form $\psi_1 \mathcal{U}^{a_i} \psi_2$ we define an acceptance set with all the atoms containing top_i along with either ψ_2 or $\neg(\psi_1 \mathcal{U}^{a_i} \psi_2)$.

For each $i \in [n]$, a Büchi acceptance condition with all the atoms containing top_i is also needed if there are no formulas of the form $\psi_1 \mathcal{U}^{a_i} \psi_2$ in cl_φ. This ensures that each accepting run visits infinitely often top_i-atoms, and thus by the transition rules, each call that is declared matched (by pushing a $B \neq \partial$ onto a stack) is effectively matched in any accepting run.

Note that in this construction we actually use a generalized Büchi acceptance condition, that is a set of Büchi acceptance conditions that have to be all fulfilled in order to accept. Moving from a generalized Büchi acceptance condition with m acceptance sets to a standard Büchi condition by using a modulo $m+1$ counter is a well known technique and thus we omit further details on this.

The size of cl_φ is linear in the size of φ, thus the number of states, stack symbols and transitions of M_φ is $2^{O(|\varphi|)}$. The translation from generalized to standard Büchi conditions increases the size only by a $O(|\varphi|)$ factor (number of until formulas in cl_φ plus n). Therefore, we get:

Theorem 3. *Given a* MultiCaRet *formula φ over n-nested words, it is possible to construct a Büchi n-*Mpds *M_φ such that for each w: $w \models \varphi$ iff $\langle\langle \rho, \lambda_\varphi \rangle\rangle = w$ for some computation ρ of M_φ. Moreover, the size of M_φ is $2^{O(|\varphi|)}$.*

5 Büchi Mpds with scope-bounded matching relations

In this section, we show that emptiness of Büchi Mpds restricted to computations with scope-bounded matching relations is decidable in exponential time.

Scoped Runs. We restrict Mpds to runs where a symbol can be popped from a stack i only if it has been pushed within one of the last k execution contexts of this stack, where a *context* is a run such that all the pop and push transitions are over the same stack. We formally define this restriction on the multiply nested words and then export it to corresponding runs.

A multiply nested word $w = (\{\sigma_i\}_{i \in \mathbb{N}}, \{\langle \mu_i, C_i, R_i \rangle \mid i \in [n]\})$ is *k-scoped* if for each $i, j \in \mathbb{N}$ for which $\mu_h(i,j)$, $h \in [n]$, holds then there are at most $2d-3$ positions $x_1, \ldots, x_{d-1} \in \bigcup_{h' \neq h} (C_{h'} \cup R_{h'})$ and $y_1, \ldots, y_{d-2} \in C_h \cup R_h$ such that $i < x_1 < y_1 < \ldots < x_{d-2} < y_{d-2} < x_{d-1} < j$ and $d \leq k$. A run ρ is *k-scoped* if $\langle\langle \rho \rangle\rangle$ is k-scoped.

A context where the only active stack is the h-th is also called a h-context. For a finite h-context from $\mathcal{C}_0$ to $\mathcal{C}_r$, we write $(q, w) \rightsquigarrow_h (q', w')$ if $\mathcal{C}_0 = \langle q, \{w_i\}_{i \in [n]} \rangle$ with $w_h = w$ and $\mathcal{C}_r = \langle q', \{w'_i\}_{i \in [n]} \rangle$ with $w'_h = w'$.

Decision procedure. We reduce the emptiness problem for Büchi Mpds to the same problem for standard Büchi automata.

Given M, we first define a PDS M_h, for $h \in [n]$, obtained by ignoring all the actions $push_i$ and pop_i, for $i \neq h$. M_h collects, in its states, pairs of states

of M, which are the beginning and the end of a context involving stack h, along with a bit storing the information whether a state from $\mathcal{F}$ has been entered in that context or not. The idea exploited here is similar to that in [12] where the concept of *thread-interface* is used to summarize the executions of a thread in consecutive rounds within a fixed-point algorithm to solve the scope-bounded reachability problem for MPDS.

Fix $M = (Q, q_0, \Gamma \cup \{\gamma^\perp\}, \Sigma, \lambda, \delta^{int}, \{(\delta_i^{push}, \delta_i^{pop})\}_{i\in[n]})$ be an n-MPDS, $h \in [n]$ and $\mathcal{F} \subseteq Q$.

Formally, the PDS M_h is $(Q', Q_0, \Gamma, \Sigma, \lambda, \delta'^{int}, (\delta'^{push}, \delta'^{pop}))$ where:

- the set of states is $Q' = \bigcup_{m\in[k]}(Q \times Q \times \{0,1\})^m$;
- the set of iniatial states is $Q_0 = \{(q, q, 0)|q \in Q \setminus \mathcal{F}\} \cup \{(q, q, 1)|q \in \mathcal{F}\}$;
- the transition functions are as follows (set X be either a state of Q', with length $m < k$, or the empty word)
  - $(X(q, p, f), \gamma, X(q, p', f')) \in \delta'^{push}$ if $(p, \gamma, p') \in \delta_h^{push}$, $f' = 1$ if $p' \in \mathcal{F}$, and $f' = f$, otherwise.
  - $(X(q, p, f), \gamma, X(q, p', f')) \in \delta'^{pop}$, if $(p, \gamma, p') \in \delta_h^{pop}$, $f' = 1$ if $p' \in \mathcal{F}$, and $f' = f$, otherwise.
  - $(X(q, p, f), X(q, p', f')) \in \delta'^{int}$, if $(p, p') \in \delta^{int}$, $f' = 1$ if $p' \in \mathcal{F}$, and $f' = f$, otherwise.
  - $(X, X(q, q, f)) \in \delta'^{int}$ (a *jump*), for every state $q \in Q$ and $f = 1$ if $q \in \mathcal{F}$, and $f = 0$, otherwise.

A triple $(q, p, f) \in Q \times Q \times \{0, 1\}$ is called a *summary*. The PDS M_h collects summaries in its states: it starts from a state (q, q, f) and modifies the second and the third component, following a run of M, then, by nondeterministic jumps, it adds a new summary, forming lists of at most k summaries. To obtain longer, possibly infinite, lists of summaries, we use a sequential composition which simply appends a list of summary after another list. Let $\mathcal{R}_h$ be the set of reachable states of M_h and let $Closure(\mathcal{R}_h)$ contain the finite and infinite lists of summaries obtained by sequential composition, starting from elements of $\mathcal{R}_h$. The the following lemma follows, using induction, from the definition of M_h.

Lemma 1. *Let $X = \{(q_i, p_i, f_i)\}_{i\in\mathbb{N}}$. If $X \in Closure(\mathcal{R}_h)$ then there exist h-contexts ρ_i and words $w_i \in \Gamma^*$, for $i \in \mathbb{N}$, such that:*

1. *$w_1 = \epsilon$ and $(q_i, w_i) \rightsquigarrow_h (p_i, w_{i+1})$,*
2. *$f_i = 1$ if and only if a state from $\mathcal{F}$ occurs in ρ_i.*

To obtain also a reverse implication of Lemma 1 for k-scoped runs, we show that a list X of summaries can be associated to a k-scope bounded run ρ in such a way that the summaries in X are associated to the h-contexts, and X is obtained just by sequentially composing sequences, each having at most k summaries. The idea is similar to that used in [12], by taking into account the non terminating computations and the Büchi condition.

Lemma 2. *Let ρ be a k-scoped run of M. There exist h-contexts $\{\rho_i\}_{i\in\mathbb{N}}$ in ρ such that, called q_i and p_i the first and the last states of ρ_i, $\{(q_i, p_i, f_i)\}_{i\in\mathbb{N}} \in Closure(\mathcal{R}_h)$, where $f_i = 1$ if and only if a state from $\mathcal{F}$ occurs in ρ_i.*

Now, we define a Büchi automaton $B_M = (Q_B, \{\sigma\}, \delta_B, Q_0^M, \mathcal{F}_\mathcal{B})$ which puts together summaries to simulate a run of M. For this, it saves in its states the current state of M, a reachable state X_h of M_h, for $h \in [n]$, and a bit for the acceptance condition. At each step, B_M consumes a summary. When the summaries of an X_h has been exhausted then it chooses another X'_h in the same set $\mathcal{R}_h$. Thus the states of B_M are $(p, X_1, \cdots, X_n, f)$, where $p \in Q$, $X_h \in \mathcal{R}_h$ and $f \in \{0, 1\}$ and the initial states are $(q, X_1, \cdots, X_n, f)$ such that $q \in Q_0$ and $f = 1$ if and only if $q \in \mathcal{F}$. The transition function δ_B contains $((p, X_1, \cdots, X_n, f), \sigma, (p', X'_1, \cdots, X'_n, f'))$ if there exists $h \in [n]$ such that $X'_i = X_i$, for $i \neq h$, $X_h = (p, p', f_h)Y$, $f' = f_h$, and $X'_h = Y$ if $Y \neq \epsilon$ and $X'_h \in \mathcal{R}_h$, otherwise. Finally, the acceptance condition $\mathcal{F}_\mathcal{B}$ is $\{(p, X_1, \cdots, X_n, f) \mid f = 1\}$.

It is easy to see that B_M accepts a word if and only if $(M, \mathcal{F})$ accepts a k-scoped computation. Moreover, the size of B_M is exponential in the number of the stacks and in the bound k. Thus, since the reachable states of a pushdown system can be efficiently computed (e.g., [9]), we can state the theorem:

Theorem 4. *The problem of deciding whether there exists a k-scope run of an n-*MPDS *M satisfying a Büchi condition $\mathcal{F}$ is decidable in $|M|^{O(nk)}$ time.*

6 Decidability results for MULTICARET

In this section, we show that the satisfiability and model-checking problems for MULTICARET become decidable by restricting the models to meaningful subclasses of multiply nested words. Let us first show that these problems reduce to the emptiness problem for Büchi MPDS.

Automata-theoretic approach to MULTICARET *model-checking.* For $i \in [2]$, let $M_i = (Q_i, Q_i^0, \Gamma_i \cup \{\gamma^\perp\}, \Sigma, \lambda_i, \delta_i^{int}, \{(\delta_i^{push_j}, \delta_i^{pop_j})\}_{j\in[n]}, F_i)$ be a n-MPDS. The synchronized cross product $M_1 \otimes M_2$, is the n-MPDS M such that:
$M = (Q, Q_0, (\Gamma_1 \times \Gamma_2) \cup \{\gamma^\perp\}, \Sigma, \lambda, \delta^{int}, \{(\delta_j^{push}, \delta_j^{pop})\}_{j\in[n]}, F)$ where $Q = \{(q_1, q_2) \mid \lambda_1(q_1) = \lambda_2(q_2)\}$, $Q_0 = Q \cap (Q_1^0 \times Q_2^0)$, λ is such that $\lambda(q_1, q_2) = \lambda_1(q_1)$, the transition functions are such that: $((q_1, q_2), (q'_1, q'_2)) \in \delta^{int}$ iff $(q_i, q'_i) \in \delta_i^{int}$ for $i \in [2]$, and for $j \in [n]$: $((q_1, q_2), (\gamma_1, \gamma_2), (q'_1, q'_2)) \in \delta_j^{push}$ iff $(q_i, \gamma_i, q'_i) \in \delta_i^{push_j}$ for $i \in [2]$, and $((q_1, q_2), (\gamma_1, \gamma_2), (q'_1, q'_2)) \in \delta_j^{pop}$ iff $(q_i, \gamma_i, q'_i) \in \delta_i^{pop_j}$ for $i \in [2]$ (where we have identified $(\gamma^\perp, \gamma^\perp)$ with $\gamma^\perp$).

If one or both the MPDS are Büchi MPDS the above construction can be adapted in the usual way to suit the Büchi condition(s). A salient property of this construction is that the resulting MPDS defines a language of multiply nested words that is the intersection of the languages of the starting MPDS.

Lemma 3. *For (Büchi)* MPDS *M_i with $i \in [2]$, $M = M_1 \otimes M_2$ is a (Büchi)* MPDS *and $\mathcal{L}(M) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ holds.*

In section 4, we have shown that given a formula φ over n-nested words, we can construct a Büchi MPDS M_φ that captures all the n-nested words that

satisfy φ. Thus, a given model checking instance formed by a MPDS M and a formula φ, reduces to checking that $\mathcal{L}(M) \cap \mathcal{L}(M_{\neg\varphi})$ is empty. Therefore, by the above lemma and Theorem 3, we get:

Theorem 5. *For a* MPDS M *and a* MULTICARET *formula* φ, $M \models \varphi$ *iff* $\mathcal{L}(M) \cap \mathcal{L}(M_{\neg\varphi}) = \emptyset$.

6.1 Scope-bounded multiply nested words

Let us restrict to k-scoped multiply nested words. To capture the set of all k-scoped multiply nested words satisfying a MULTICARET formula, it suffices to place the same limitation on the runs of the Büchi MPDS M_φ from Section 4. Therefore, following the automata-theoretic approach described above, by Theorems 3, 4 and 5, we have:

Theorem 6. *The* MULTICARET *satisfiability and model-checking problems restricted to k-scoped multiply nested works are* EXPTIME*-complete.*

6.2 Multiply nested words with ordered matching relations

We recall that in ordered MPDS a symbol can be popped out from a stack h provided that all stacks of lower indices (from 1 through $h-1$) are empty [8]. We define ordered multiply nested words by imposing the same restriction. A multiply nested word $(\{\sigma_i\}_{i\in\mathbb{N}}, \{\langle \mu_i, C_i, R_i\rangle \mid i \in [n]\})$ is *ordered* if for every $i, j \in \mathbb{N}$ for which $\mu_h(i,j)$ holds for some $h \in [n]$: if there is a $x < j$ such that $x \in C_{h'}$, $h' < h$, then there is a $y < j$ such that $\mu_{h'}(x,y)$ holds (all calls of lower-index relations preceding j are already matched at j).

Checking the emptiness of Büchi ordered MPDS is known to be 2ETIME-complete [3, 4], and can be solved in time $|M|^{2^{O(n)}}$ where $|M|$ denotes the size of the input MPDS [3]. Therefore, by Theorems 3 and 5, we have:

Theorem 7. *The* MULTICARET *satisfiability and model-checking problems restricted to ordered multiply nested works are* 2ETIME*-complete.*

6.3 Multiply nested words of bounded tree-width

The expressiveness of MULTICARET does not go beyond first-order logic interpreted over multiply nested words. We define FO_μ as the first-order logic over multiply nested words which has in its signature relations that capture the matching relations. Namely, the logic contains the usual binary predicate $<$ (the ordering relation over integers) along with a binary predicate μ_i and unary predicates C_i, R_i for $i \in [n]$ such that (μ_i, C_i, R_i) define a matching relation. Also, we use the unary predicates $P_\sigma(x)$ meaning that x is labeled with symbol σ, and fix a countable set of first-order variables $x, y, \ldots$. The logic FO_μ is defined as:

$$\varphi := P_\sigma(x) | x < y | \mu_i(x,y) | \neg\varphi | \varphi \vee \varphi | \exists x \varphi \quad \text{(where } i \in [n]\text{)}$$

With similar constructions as those used in [1] to show that CARET formulas are FO definable, we can show the following theorem.

Theorem 8. *Given a* MultiCaRet *formula φ it is possible to construct effectively a sentence ψ of FO_μ such that $|\psi| = O(|\varphi|)$ and $w \models \varphi$ if and only if w satisfies ψ.*

This result allows us to extend the decidability of MultiCaRet satisfiability and model-checking to all the MSO-definable classes of multiply nested words of bounded tree-width. In fact, for each class of MSO-definable graphs of bounded tree width, the satisfiability of MSO sentences is decidable [13]. Thus, by Theorem 8 we get:

Theorem 9. *Restricting the models to any MSO-definable class of multiply nested words, the satisfiability and model-checking problems of* MultiCaRet *formulas are decidable.*

Acknowledgments. We thank Gennaro Parlato for helpful discussions.

References

1. Alur, R., Arenas, M., Barceló, P., Etessami, K., Immerman, N., Libkin, L.: First-order and temporal logics for nested words. Log. Meth. in Comp. Sci. 4(4) (2008)
2. Alur, R., Etessami, K., Madhusudan, P.: A temporal logic of nested calls and returns. In: TACAS. pp. 467–481 (2004)
3. Atig, M.F.: Global model checking of ordered multi-pushdown systems. In: FSTTCS. pp. 216–227 (2010)
4. Atig, M.F., Bollig, B., Habermehl, P.: Emptiness of multi-pushdown automata is 2ETIME-complete. In: DLT. pp. 121–133 (2008)
5. Atig, M.F., Bouajjani, A., Kumar, K.N., Saivasan, P.: Linear-time model-checking for multithreaded programs under scope-bounding. In: ATVA. To appear (2012)
6. Bollig, B., Cyriac, A., Gastin, P., Zeitoun, M.: Temporal logics for concurrent recursive programs: Satisfiability and model checking. In: MFCS. pp. 132–144 (2011)
7. Bozzelli, L., La Torre, S., Peron, A.: Verification of well-formed communicating recursive state machines. Theor. Comput. Sci. 403(2-3), 382–405 (2008)
8. Breveglieri, L., Cherubini, A., Citrini, C., Crespi-Reghizzi, S.: Multi-push-down languages and grammars. Int. J. Found. Comput. Sci. 7(3), 253–292 (1996)
9. Esparza, J., Schwoon, S.: A BDD-based model checker for recursive programs. In: CAV. pp. 324–336 (2001)
10. La Torre, S., Madhusudan, P., Parlato, G.: A robust class of context-sensitive languages. In: LICS. pp. 161–170. IEEE Computer Society (2007)
11. La Torre, S., Napoli, M.: Reachability of multistack pushdown systems with scope-bounded matching relations. In: CONCUR. pp. 203–218 (2011)
12. La Torre, S., Parlato, G.: Scope-bounded multistack pushdown systems: Fixed-point, sequentialization, and tree-width. Available at http://users.ecs.soton.ac.uk/gp4/papers/scoped2012.pdf (2012)
13. Madhusudan, P., Parlato, G.: The tree width of auxiliary storage. In: POPL. pp. 283–294 (2011)
14. Pnueli, A.: The temporal logic of programs. In: FOCS. pp. 46–57. IEEE Computer Society (1977)
15. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: TACAS. pp. 93–107 (2005)