The Complexity of Model Checking Multi-Stack Systems

Benedikt Bollig
Laboratoire Spécification et Vérification,
École Normale Supérieure de Cachan
& Centre National de la Recherche Scientifique,
France

Dietrich Kuske
and Roy Mennicke
Ilmenau University of Technology,
Germany

***Abstract*—We consider the linear-time model checking problem for boolean concurrent programs with recursive procedure calls. While sequential recursive programs are usually modeled as pushdown automata, concurrent recursive programs involve several processes and can be naturally abstracted as pushdown automata with multiple stacks. Their behavior can be understood as words with multiple nesting relations, each relation connecting a procedure call with its corresponding return. To reason about multiply nested words, we consider the class of all temporal logics as defined in the book by Gabbay, Hodkinson, and Reynolds (1994). The unifying feature of these temporal logics is that their modalities are defined in monadic second-order (MSO) logic. In particular, this captures numerous temporal logics over concurrent and/or recursive programs that have been defined so far. Since the general model checking problem is undecidable, we restrict attention to phase bounded executions as proposed by La Torre, Madhusudan, and Parlato (LICS 2007). While the MSO model checking problem in this case is non-elementary, our main result states that the model checking (and satisfiability) problem for all MSO-definable temporal logics is decidable in elementary time. More precisely, it is solvable in $(n+2)$-EXPTIME where n is the maximal level of the MSO modalities in the monadic quantifier alternation hierarchy. We complement this result and provide, for each level n, a temporal logic whose model checking problem is n-EXPSPACE-hard.**

I. Introduction

The verification of finite-state (boolean) sequential programs is by now well understood. In its most classical form, the program is abstracted as a finite automaton $\mathcal{A}$, and a property φ is specified in a temporal logic such as LTL [25]. In the linear-time framework, model checking amounts to the question if all executions of $\mathcal{A}$ satisfy φ.

Nowadays, most programs are distributed in nature and involve recursive procedure calls. In order to model such programs more accurately, finite-state models have been extended with several pushdown stacks. When reasoning about the behaviour of such multi-stack systems, it is natural and convenient to consider an execution as a word with multiple nesting relations. The word itself reflects the order of atomic actions as observed during an execution. In addition, each nesting relation associates with a push operation (which corresponds to a procedure call) its corresponding return position. Over multiply nested words, one may then formalize properties such as "process p is not allowed to call a procedure while being in the scope of an active procedure call of process q", which do not have a natural interpretation over simple words without nesting relations.

However, it is folklore that even simple verification tasks such as reachability are undecidable for systems involving two or more stacks. Therefore, any model checking task can only cover an approximation of the system behavior. There have been several and, partially, orthogonal approaches to defining meaningful abstractions. The simplest one restricts to executions with a bounded number of *contexts*, each involving only actions of one particular stack [26]. This underapproximation does not consider any interaction of processes within a procedure call, which motivated La Torre et al. to define the more liberal notion of *phases* [17]. A phase does not constrain push operations, whereas pop actions are required to belong to some dedicated process. Orthogonal approaches are due to [20], [22], where the number of *scopes* is bounded (a call and its return are separated by a bounded number of contexts), and [9], [5], which assumes an ordering of the stacks and postulates that a pop operation is subject to the first non-empty stack.

Model checking multi-stack systems has recently received a lot of attention [17], [4], [8], [10], [21], [6], [7]. In [17], it was shown that model checking is decidable for monadic second-order (MSO) properties under the restriction of bounded phases. However, the problem is non-elementary (since it is already non-elementary without stacks). So, the focus has since moved to temporal logics. Previous works on temporal logic for multi-stack systems differ in the choice of the behavioural restriction described above (context-, phase-, scope-bounded, ordered), but also in the concrete temporal logic adopted for the model checking task. While [4], [6] consider properties over strings such as classical LTL rather than multiply nested words, [21] introduces a temporal logic that allows one to identify call and return positions of a given process and to distinguish between linear successors (referring to the word structure) and abstract successors (involving the nesting edges). As a matter of fact, there is so far no agreement on a canonical temporal logic for nested words, not even for those with one single nesting relation [1], [2]. Therefore, we consider the class of all temporal logics as defined in the book by Gabbay, Hodkinson, and Reynolds [12], which subsumes virtually all existing formalisms. The unifying feature

of these temporal logics is that their modalities are defined in MSO logic. Not only does this capture temporal logics over (multiply) nested words, but it also includes numerous temporal logics that have been designed for concurrent non-recursive programs and that are typically interpreted over partial orders such as Mazurkiewicz traces (cf. [13] and the references therein). In [8], it is shown that satisfiability and model checking for any MSO-definable temporal logic are decidable in EXPTIME when restricting to phase bounded executions. The phase bound τ has to be fixed, though. It was left open if the problems are still elementary if τ is part of the input. This is an important issue, as an elementary procedure would allow for a gradual adjustment of τ at the cost of only an elementary blow-up.

Contribution. In this paper, we show that the model checking problem for multi-stack systems wrt. phase bounded executions is indeed decidable in elementary time. More precisely, it is solvable in time $(n+2)$-fold exponential in the number of phases and the size of the temporal formula where n is the maximal level of the MSO modalities in the monadic quantifier alternation hierarchy.

Our result is in stark contrast to the non-elementary lower bounds of the branching-time model checking problem [14], [7] and of model checking against MSO logic. It is optimal for the first level $n = 0$, which contains the 2-EXPTIME-complete emptiness problem of multi-stack automata [17], [19]. For all other levels, we provide a temporal logic whose model checking problem is n-EXPSPACE-hard.

Two key ideas are pursued in the proof of the upper bound. First, we translate, in polynomial time, a temporal logic formula into an MSO formula in a certain normal form. The construction is based on Hanf's locality theorem and independent of the number of phases. Second, we show that an MSO formula in normal form can be transformed into a tree automaton in $(n+1)$-fold exponential space. The tree automaton works on tree encodings of multiply nested words and can then be checked for emptiness. One of its key ingredients is a tree automaton recovering the direct successor relation of the encoded multiply nested words. We show that such an automaton can be computed in polynomial space, avoiding the generic doubly exponential construction given in [17]. The use of this tree automaton for the direct successor relation is the main difference from [8]. There, the nested word is *interpreted* in its tree encoding and then the resulting formula is translated into a tree automaton. This results in a non-elementary blowup since the quantifier alternation rank of the interpretation increases linearly with the number of phases.

To prove the lower bound, we also proceed in two steps. It is first shown for a restricted version of the satisfiability problem of temporal logics over labelled grids, and then reduced to the satisfiability problem for temporal logics over nested words.

Outline. Section II introduces crucial notions such as multiply nested words and MSO-definable temporal logics, Section III presents the upper bound of the satisfiability problem, Section IV develops our lower bound, and Section V transfers these results to the model checking problem. We conclude in Section VI, giving directions for future work.

II. Preliminaries

Let Γ be an alphabet, i.e., a non-empty finite set. For an integer $n \in \mathbb{N}$, we let $[n] = \{1, 2, \ldots, n\}$ and $[n]_0 = \{0, 1, \ldots, n\}$. The function $\mathsf{tower} : \mathbb{N}^2 \to \mathbb{N}$ is inductively defined by $\mathsf{tower}(0, m) = m$ and $\mathsf{tower}(\ell+1, m) = 2^{\mathsf{tower}(\ell,m)}$, for all $\ell, m \in \mathbb{N}$. We let $\mathsf{poly}(n)$ denote the set of polynomial functions in one argument.

A. Multiply Nested Words

Definition 1 *We define a* word over Γ *to be a finite* Γ*-labelled linear order* $w = (P, \leq, \lambda)$*, i.e.,* $(P, \leq)$ *is a finite, non-empty, linearly ordered set and* $\lambda : P \to \Gamma$ *is a mapping. By* $\lessdot$*, we denote the immediate successor relation, i.e.,* $\lessdot = < \setminus <^2$*. Furthermore, the minimal and maximal elements of* P *with respect to* $\leq$ *are denoted by* $\min(P, \leq)$ *and* $\max(P, \leq)$*, resp.*

Most of the time one can think of P as an initial segment of $\mathbb{N}$ and of $\leq$ as the restriction of the natural ordering of $\mathbb{N}$ to the set P.

Definition 2 *Let* $(P, \leq)$ *be a finite linear order. A* nesting relation $\prec$ *over* $(P, \leq)$ *is a binary relation such that, for all* $i, j, i', j' \in P$*, the following conditions hold:*

1) *if* $i \prec j$*, then* $i < j$
2) *if* $i \prec j$*,* $i' \prec j'$*, and* $i \leq i'$*, then* $i < i' < j' < j$ *or* $i < j < i' < j'$ *or* $(i = i'$ *and* $j = j')$*.*

If $i \prec j$*, then we say that* i *is a* call *with* matching return j*.*

The idea is that the linear order $(P, \leq)$ describes the execution of some recursive program. Then $i \prec j$ shall mean that, at time i, the execution calls some procedure and, at time j, the control is returned to the calling program. Having this in mind, condition 1 expresses that every return occurs after its matching call. Condition 2 ensures that no position is both, a call and a return, every call has exactly one matching return and vice versa, and calls and matching returns are well nested. We will consider words with not only one, but with $\sigma \in \mathbb{N}$ many nesting relations.

Definition 3 *For* $\sigma \in \mathbb{N}$*, a* σ-nested word over Γ *is a tuple* $\nu = (w, \prec_1, \prec_2, \ldots, \prec_\sigma)$ *where* $w = (P, \leq, \lambda)$ *is a word over* Γ *and, for all* $s \in [\sigma]$*,* $\prec_s$ *is a nesting relation over* $(P, \leq)$ *such that we have, for all* $1 \leq s < s' \leq \sigma$ *and* $i, j, i', j' \in P$*:*

$$i \prec_s j \text{ and } i' \prec_{s'} j' \text{ imply } i \neq i' \text{ and } j \neq j' . \qquad (1)$$

We define $\prec = \bigcup_{s \in [\sigma]} \prec_s$ *and identify isomorphic nested words.*

The condition (1) restricts the interplay of the different nesting relations. It ensures that no position of the word can be call (return, resp.) of two distinct nesting relations. Note that $i_1 \prec_s i_2 \prec_{s'} i_3$ is possible for $s \neq s'$, i.e., a position can be both, a return and a call, but only with respect to distinct nesting relations (the case $s = s'$ is excluded by condition 2 of Definition 2).

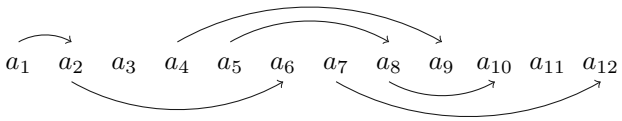

Fig. 1. The nested word $\nu = (w, \prec_1, \prec_2)$ from Example 1. Note that the relation $\prec_1$ ($\prec_2$, resp.) is represented by the edges above (below) the word w.

Example 1 Figure 1 is an illustration of the 2-nested word $\nu = ([12], \leq, \lambda, \prec_1, \prec_2)$ where $\leq$ is the natural ordering on $[12]$, $\lambda(i) = a_i$ for all $i \in [12]$, $\prec_1 = \{(1,2),(4,9),(5,8)\}$, and $\prec_2 = \{(2,6),(7,12),(8,10)\}$.

In order to make the satisfiability problem decidable, we adopt the notion of a phase. The latter is an interval in a nested word in which all returns refer to the same nesting relation.

Definition 4 *Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a nested word. A* phase *of ν is a subset $I \subseteq P$ for which the following conditions hold:*

1) *there exist $i, j \in P$ such that $I = \{k \in P \mid i \leq k \leq j\}$*
2) *for all $i, i' \in P$, $j, j' \in I$, and $s, s' \in [\sigma]$ with $i \prec_s j$ and $i' \prec_{s'} j'$, we have $s = s'$.*

Let $\tau \in \mathbb{N}$. We call ν a τ-phase nested word *if there exist phases $I_1, I_2, \ldots, I_\tau$ of ν with $P = I_1 \cup \ldots \cup I_\tau$. The set of τ-phase σ-nested words is denoted by $\mathsf{NW}_\tau(\Gamma, \sigma)$.*

Example 2 Consider the nested word ν from Example 1. It can be divided into the four phases $\{1,2,3,4,5\}$, $\{6,7\}$, $\{8,9\}$, and $\{10,11,12\}$. Hence, ν is a 4-phase nested word. Note that the phases $\{1,2\}$, $\{3,4,5,6\}$, $\{7,8,9\}$, and $\{10,11,12\}$ also witness this property. However, ν is no 3-phase nested word since no two of the positions 2, 6, 8, and 10 can belong to the same phase.

Among the possible divisions of the τ-phase σ-nested word ν into different phases, the *greedy division* will serve as a canonical example: We define the mapping $\mathsf{ph}_\nu : P \to [\tau]$ where $\mathsf{ph}_\nu(i)$ is the minimal number $s \geq 1$ such that the restriction of ν to the positions $\{1, \ldots, i\}$ is an s-phase nested word. Then, for all $s \in [\tau]$, $I_s = \mathsf{ph}_\nu^{-1}(s)$ is a phase of ν, and we have $P = I_1 \cup \ldots \cup I_\tau$. Note that $I_\tau = \emptyset$ if and only if ν is a $(\tau - 1)$-phase nested word.

B. Monadic Second-Order Logic and MSO-definable Temporal Logics

We fix an alphabet Γ and a natural number σ. Furthermore, we fix the set $\{x, y, z, \ldots\}$ of individual variables and the set $\{X, Y, Z, \ldots\}$ of set variables. We will define a logic that is capable of expressing properties of σ-nested words over Γ.

Definition 5 *The class $\mathrm{MSO}(\Gamma, \sigma)$ of MSO formulas is given by the following grammar, where $a \in \Gamma$, $s \in [\sigma]$, x, y are first-order variables, and X is a set variable:*

$$\begin{aligned} \varphi \quad ::= \quad & (\lambda(x) = a) \mid x \lessdot y \mid x \prec_s y \mid x = y \mid x \in X \\ & \mid \mathsf{call}_s(x) \mid \mathsf{ret}_s(x) \mid \mathsf{min}(x) \mid \mathsf{max}(x) \\ & \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\colon \varphi \mid \exists X\colon \varphi \end{aligned}$$

The set $\mathrm{FO}(\Gamma, \sigma) \subseteq \mathrm{MSO}(\Gamma, \sigma)$ contains all formulas without second order quantification $\exists X$. We use usual abbreviations such as $\forall x\colon \varphi$ for $\neg\exists x\colon \neg\varphi$.

Now, let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a σ-nested word and $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ be a formula with free variables from $\{x_1, \ldots, x_k, X_1, \ldots, X_\ell\}$. Furthermore, let $i_1, \ldots, i_k \in P$ and $I_1, \ldots, I_\ell \subseteq P$. Then we write $\nu, i_1, \ldots, i_k, I_1, \ldots, I_\ell \models \varphi$ if φ evaluates to true when interpreting the variables by $i_1, \ldots, i_k$ and $I_1, \ldots, I_\ell$, resp. In particular, we have $\nu, i \models (\lambda(x) = a)$ if $\lambda(i) = a$. Furthermore, $\nu, i \models \mathsf{call}_s(x)$ if there exists $j \in P$ with $i \prec_s j$ and $\nu, i \models \mathsf{min}(x)$ if $i = \min(P, \leq)$. The semantics of the atomic formulas $\mathsf{ret}_s(x)$ and $\mathsf{max}(x)$ is defined analogously. If φ is a sentence, then we denote by $L(\varphi)$ the set of σ-nested words ν with $\nu \models \varphi$.

Definition 6 *An $\mathrm{MSO}(\Gamma, \sigma)$-formula is an m-*ary modality *if it has one free individual variable x and m free set variables $X_1, \ldots, X_m$. An $\mathrm{MSO}(\Gamma, \sigma)$-*definable temporal logic *is a tuple $\mathrm{TL} = (B, \mathrm{arity}, [\![-]\!])$ where B is a finite set of modality names, the mapping $\mathrm{arity}\colon B \to \mathbb{N}$ specifies the arity of every modality name from B, and $[\![-]\!] : B \to \mathrm{MSO}(\Gamma, \sigma)$ is a mapping such that $[\![M]\!]$ is an m-ary modality whenever $\mathrm{arity}(M) = m$ for $M \in B$.*

The formulas F of the temporal logic TL *are defined by the grammar*

$$F ::= M(\underbrace{F, \ldots, F}_{\mathrm{arity}(M)})$$

where M ranges over B. The size $|F|$ of a temporal formula F is its number of subformulas.

Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a nested word and $F \in \mathrm{TL}$ be a formula. The semantics $F^{\nu,\mathrm{TL}}$ of F in ν is the set of positions from P where F holds. The inductive definition is as follows: If $F = M(F_1, \ldots, F_m)$ where $M \in B$ is of arity $m \geq 0$, then $F^{\nu,\mathrm{TL}} = \{i \in P \mid \nu, i, F_1^{\nu,\mathrm{TL}}, \ldots, F_m^{\nu,\mathrm{TL}} \models [\![M]\!]\}$. We write $\nu, i \models_{\mathrm{TL}} F$ for $i \in F^{\nu,\mathrm{TL}}$ and $\nu \models_{\mathrm{TL}} F$ for $\nu, \min(P, \leq) \models_{\mathrm{TL}} F$. By $L(F)$ we denote the set of all σ-nested words ν with $\nu \models_{\mathrm{TL}} F$.

Note that a temporal formula F can be part of different temporal logics and, therefore, can have several semantics. However, if we only deal with one temporal logic TL, then we often write F^ν instead of $F^{\nu,\mathrm{TL}}$ and $\nu \models F$ instead of $\nu \models_{\mathrm{TL}} F$.

Example 3 The Boolean connectives negation and conjunction can be expressed by

$$[\![\neg]\!](X_1, x) = \neg(x \in X_1)$$

and

$$[\![\vee]\!](X_1, X_2, x) = (x \in X_1) \vee (x \in X_2)\,.$$

Let us also consider the modality N_1. Intuitively, $\mathsf{N}_1\, F$ means that the formula F holds for the immediate successor of the current position with respect to the nesting relation $\prec_1$. Formally, we can set

$$[\![\mathsf{N}_1]\!](X_1, x) = \exists y\colon (x \prec_1 y \wedge y \in X_1)\,.$$

Example 4 A wide range of temporal logics have been defined for nested words and concurrent systems. Their *until* operator usually depends on what is considered a path between two word positions and, more specifically, on a notion of *successor*. In the classical setting of words without nesting relations, one naturally considers the direct successor following the linear order. The situation is less clear in the presence of one or more nesting relations. In [1], Alur et al. identify three different kinds of successors in *singly* nested words, namely the linear, call, and abstract successor. Each of them comes with a separate until operator.

Towards nested words with multiple nesting relations, Atig et al. consider only the linear successor [4], [6], while La Torre and Napoli also define modalities that correspond to linear, call, and abstract successors [21]. As an example, we consider the *abstract until*. An abstract s-path in a nested word is a path that does not choose a linear direct successor *from* a call position or *to* a return position (wrt. $\prec_s$). The modality U_s^a for the abstract until is then given by

$$\begin{aligned}&[\![\mathsf{U}_s^a]\!](X_1, X_2, x) = \\ &\exists Y \exists z\colon \Big(z \in X_2 \wedge Y \subseteq X_1 \wedge \\ &\forall y\colon (y \in Y \vee y = z) \to \big(y = x \vee \exists y'\,(y' \in Y \wedge \varphi_s(y', y))\big)\Big)\end{aligned}$$

where

$$\varphi_s(y', y) = y' \prec_s y \vee (\neg \mathsf{call}_s(y') \wedge \neg \mathsf{ret}_s(y) \wedge y' \lessdot y)\,.$$

Indeed, all the modalities considered in [1], [4], [6], [21] are MSO-definable. However, they appear to be just a few of many other possibilities. For example, one may define an abstract path including two or more nesting relations, or include past-time counterparts of until modalities, which are not present in [21]. Such extensions can be realized in our framework by giving their definition in MSO. An elementary upper bound of the satisfiability and model checking problem follows immediately from our result, without changing anything in the decidability proof.

Note that the emptiness problem of a 2-stack automaton is undecidable (since a Turing machine can be simulated using two stacks). It follows that the satisfiability problem of $\mathrm{MSO}(\Gamma, \sigma)$ and the model checking problem of every non-trivial temporal logic is undecidable as well. La Torre et al. [17] proposed the restriction of these problems to τ-phase words and showed that, under this restriction, the emptiness problem as well as the satisfiability problem of $\mathrm{MSO}(\Gamma, \sigma)$ are decidable. Here, we define the satisfiability problem for temporal logics, restricted to a given number of phases τ:

Definition 7 *Let* TL *be some* $\mathrm{MSO}(\Gamma, \sigma)$*-definable temporal logic. The* satisfiability problem *of* TL *is the set of pairs* (F, τ) *where* $F \in$ TL *is a formula and* $\tau \in \mathbb{N}$ *such that there exists some* τ*-phase* σ*-nested word* ν *with* $\nu \models F$*.*

We now introduce a measure of the complexity of monadic formulas: the *monadic quantifier alternation hierarchy*. In this definition, we use the following convention: We say "φ can be transformed into ψ" if, using the usual rules for renaming of bound variables and regrouping of quantifiers (e.g., $\alpha \vee \exists x\colon \beta$ can be transformed into $\exists x\colon (\alpha \vee \beta)$ if x does not occur freely in α), we can obtain ψ from φ.

Definition 8 *We introduce the following two definitions:*

1) *An* $\mathrm{MSO}(\Gamma, \sigma)$*-formula* φ *belongs to* $\mathrm{M}\Sigma_n(\Gamma, \sigma)$ *if it can be transformed into the form*

$$\exists \overline{X}_1\, \forall \overline{X}_2\, \ldots\, \exists / \forall \overline{X}_n\colon \psi_0$$

where $\overline{X}_i$ *are tuples of individual and set variables and* ψ_0 *is a first-order formula. It belongs to* $\mathrm{M}\Pi_n(\Gamma, \sigma)$ *if it can be brought into the form*

$$\forall \overline{X}_1\, \exists \overline{X}_2\, \ldots\, \forall / \exists \overline{X}_n\colon \psi_0\,.$$

2) *Let* $\mathcal{L}$ *be some fragment of* $\mathrm{MSO}(\Gamma, \sigma)$ *such as* $\mathrm{FO}(\Gamma, \sigma)$ *or* $\mathrm{M}\Sigma_n(\Gamma, \sigma)$ *etc. An* $\mathrm{MSO}(\Gamma, \sigma)$*-definable temporal logic* $\mathrm{TL} = (B, \mathrm{arity}, [\![-]\!])$ *is* $\mathcal{L}$-definable *if* $[\![M]\!] \in \mathcal{L}$ *for all modality names* $M \in B$*.*

It is easily seen that any formula from $\mathrm{M}\Sigma_n(\Gamma, \sigma)$ or from $\mathrm{M}\Pi_n(\Gamma, \sigma)$ can be brought into the required form in polynomial time.

Examples 3 and 4 (continued): We have $[\![\vee]\!], [\![\neg]\!], [\![\mathsf{N}_1]\!] \in \mathrm{M}\Sigma_0(\Gamma, \sigma)$ and $[\![\mathsf{U}_s^a]\!] \in \mathrm{M}\Sigma_1(\Gamma, \sigma)$.

III. Upper Bound

In this section, we show that the satisfiability problem of every $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic belongs to $(n+2)$-EXPTIME. In the remainder of the section, we assume that P is an initial segment of $\{1, 2, \ldots\}$ and that $\leq$ is the natural ordering over P for every word $w = (P, \leq, \lambda)$.

A. From Temporal Logics to MSO

The first step in our solution of the satisfiability problem is a translation of the temporal formula into an $\mathrm{MSO}(\Gamma, \sigma)$-formula of a particular form (see Prop. 1). We start with a second measure for the complexity of formulas from $\mathrm{MSO}(\Gamma, \sigma)$, the *full quantifier alternation hierarchy* (cf. Definition 8):

Definition 9 *An* $\mathrm{MSO}(\Gamma, \sigma)$*-formula* φ *belongs to* Σ_n^M *if it can be transformed into the form*

$$\exists \overline{X}_1\, \forall \overline{X}_2\, \ldots\, \exists / \forall \overline{X}_n\colon \varphi_0$$

where $\overline{X}_i$ are tuples of individual and set variables and φ_0 is quantifier-free. It belongs to Π_n^M if it can be brought into the form

$$\forall \overline{X}_1 \, \exists \overline{X}_2 \, \ldots \, \forall / \exists \overline{X}_n \colon \varphi_0 \, .$$

Examples 3 and 4 (continued): We have $[\![\vee]\!], [\![\neg]\!] \in \Sigma_0^M$, $[\![\mathsf{N}_1]\!] \in \Sigma_1^M$, and $[\![\mathsf{U}_s^a]\!] \in \Sigma_3^M$.

It is easily seen that $\Sigma_m^M \subseteq \mathrm{M}\Sigma_m(\Gamma, \sigma)$ and $\mathrm{M}\Sigma_m(\Gamma, \sigma) \not\subseteq \Sigma_n^M$ for any $m, n \in \mathbb{N}$. However, as the following theorem states, every $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic is also Σ_{n+1}^M-definable without changing the semantics of temporal formulas. This result is achieved by applying Hanf's locality principle.

Theorem 1 *Let $\varphi = \varphi(x, X_1, \ldots, X_m)$ be an $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-modality. Then there exists a modality $\psi \in \Sigma_{n+1}^M$ such that, for any σ-nested word ν, any tuple $\overline{X} = (X_1, \ldots, X_m)$ of sets of positions, and any position x, we have*

$$\nu, x, \overline{X} \models \varphi \iff \nu, x, \overline{X} \models \psi \, .$$

Proof: Let φ be a first-order formula. Since our logic does not speak about the linear order $\leq$ directly, it handles σ-nested words as "structures of bounded degree". This allows us, using Hanf's theorem [15], [11], to translate φ into a Boolean combination of first-order formulas of the form

$$\exists x_1, \ldots, x_n \, \alpha$$

where α is a Boolean combination of atomic formulas involving the variables $x_1, \ldots, x_n$ and formulas of the form

$$\forall y \, (\beta \to \bigvee_{1 \leq j \leq n} y = x_j) \text{ and } \forall y \, \neg \beta \tag{2}$$

where β can be any of the formulas $x_i \lessdot y$, $y \lessdot x_i$, $y \prec_s x_i$, and $x_i \prec_s y$.

If, e.g., $\beta = (x_i \lessdot y)$, then the former of the two formulas above can be replaced by $\max(x_i) \vee \bigvee_{1 \leq j \leq n} x_i \lessdot x_j$ and $\forall y \, \neg \beta$ by $\max(x_i)$. This allows us to eliminate all occurrences of formulas of the form (2) in α. Hence the first-order formula φ is equivalent to an existential first-order formula. ∎

It follows that, in order to solve the satisfiability problem of an $\mathrm{MSO}(\Gamma, \sigma)$-definable temporal logic, it suffices to decide the satisfiability problem of $\mathrm{MSO}(\Gamma, \sigma)$-formulas of a certain form.

Proposition 1 *Let $\mathrm{TL} = (B, \mathrm{arity}, [\![-]\!])$ be some $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic. From a temporal formula F of TL, one can compute in time $\mathsf{poly}(|F|)$ an $\mathrm{MSO}(\Gamma, \sigma)$-formula without free variables*

$$\psi = \exists \overline{X} \, (\psi_1(\overline{X}) \wedge \forall y \, \psi_2(y, \overline{X}))$$

(where $\overline{X}$ is a tuple of set variables) such that $\psi_1 \in \Pi_{n+1}^M$, $\psi_2 \in \Sigma_{n+1}^M$, and, for any nested word ν, we have $\nu \models^{\mathrm{TL}} F$ if and only if $\nu \models \psi$.

B. The Encoding of Nested Words as Trees

Definition 10 *Let Λ be an alphabet. A* Λ-tree *is a structure $T = (V, E_0, E_1, \ell)$ where $V \neq \emptyset$ is the finite set of* nodes*, $E_0, E_1 \subseteq V \times V$ are sets of edges (E_0 is the left-successor relation, and E_1 the right-successor relation), and $\ell : V \to \Lambda$ is the labelling function. Furthermore, there is a node $u \in V$ (the* root*) such that, for every node v, there is a unique path in $(V, E_0 \cup E_1)$ from the root u to v. Finally, every node has at most one successor wrt. E_0, and at most one successor wrt. E_1. The set of all Λ-trees is denoted by $\mathcal{T}_\Lambda$.*

If $T = (V, E_0, E_1, \ell)$ is a Λ-tree, $X_1, \ldots, X_m \subseteq V$, and $x_1, \ldots, x_n \in V$, then we define a new tree $(T, X_1, \ldots, X_m, x_1, \ldots, x_n) = (V, E_0, E_1, \ell')$ over the alphabet $\Lambda \times \{0,1\}^{m+n}$ as usual: For $v \in V$, we set $\ell'(v) = (\ell(v), b_1, \ldots, b_m, c_1, \ldots, c_n)$ with

- $b_i = 1$ if and only if $v \in X_i$ for $i \in [m]$ and
- $c_i = 1$ if and only if $v = x_i$ for $i \in [n]$.

Note that a tree over $\Lambda \times \{0,1\}^{m+n}$ is of this form if and only if, for all $1 \leq i \leq n$, there is a unique node whose bit c_i is set to 1. Hence the set of these trees forms a regular tree language that can be accepted by a (nondeterministic bottom-up) tree automaton with 2^n states. Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a σ-nested word. Following [17], we will now describe its encoding as a tree $\mathsf{tree}(\nu) = (V, E_0, E_1, \ell)$. First, the nodes of the tree are the positions of ν. The position j is the right-successor of the position i if i and j are matching call and return positions. Moreover, the left-successor of i is its immediate successor in ν provided that this is no return position. Formally, we have $V = P$,

$$(i,j) \in E_1 \iff i \prec j \, , \text{ and}$$
$$(i,j) \in E_0 \iff i \lessdot j \text{ and there does not exist } k \text{ with } k \prec j \, .$$

It remains to describe the labelling of the nodes j of the tree. Any such label $\ell(j)$ will be a tuple. The first component of $\ell(j)$ is the letter $\lambda(j) \in \Gamma$. For calls, the second component is the number s of the nesting relation, otherwise it is 0. Similarly, for returns, the third component is the number s' of the nesting relation, otherwise it is 0. Finally, in the fourth component of $\ell(j)$, we want to store the phase to which j belongs in the greedy division of ν, i.e., the number $\mathsf{ph}_\nu(j)$. In summary, we have $\ell(j) = \big(\lambda(j), s, s', \mathsf{ph}_\nu(j)\big) \in \Gamma \times [\sigma]_0 \times [\sigma]_0 \times \mathbb{N}$ where

- $s > 0$ if and only if there exists $k \in P$ with $j \prec_s k$ and
- $s' > 0$ if and only if there exists $i \in P$ with $i \prec_{s'} j$.

If ν is a τ-phase nested word and j is a position in ν, then $\mathsf{ph}_\nu(j) \in [\tau]$. Hence, in this case, $\mathsf{tree}(\nu) = (V, E_0, E_1, \ell)$ is a tree over the alphabet $\Gamma \times [\sigma]_0^2 \times [\tau]$.

Example 1 (continued): The encoding of the 2-nested word from Example 1 is depicted in Fig. 2.

Our decision procedure will work with these tree encodings and not with nested words. It is therefore important to describe those trees that are actually encodings of nested words. Such a description was obtained by La Torre et al. [17]. (Note that

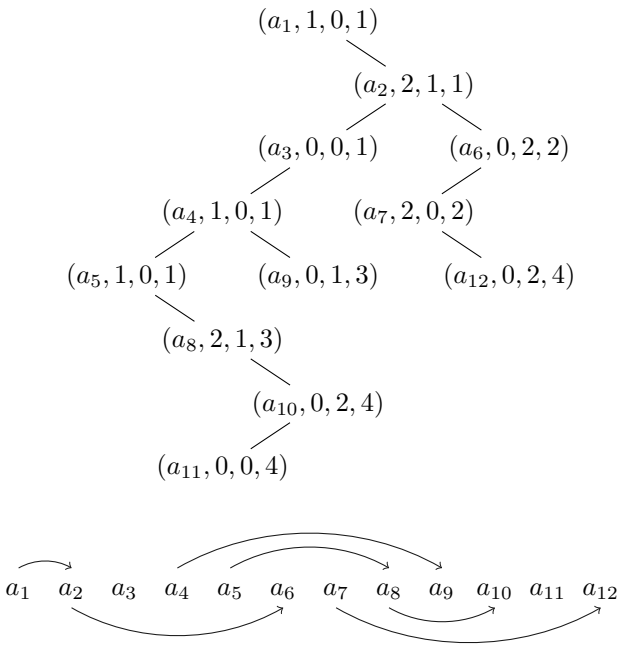

Fig. 2. The tree encoding of the nested word from Figure 1

the proof in [17] only considers tree encodings whose labels belong to $\Gamma \times [\tau]$, but the extension to our encodings $\mathsf{tree}(\nu)$ can be handled in the same spirit.)

Theorem 2 ([17]) *From $\tau \in \mathbb{N}$, one can construct in time $\mathsf{tower}_2(\mathsf{poly}(\tau))$ a tree automaton $\mathcal{B}_\tau$ with $L(\mathcal{B}_\tau) = \mathsf{tree}(\mathsf{NW}_\tau(\Gamma, \sigma))$.*

C. Tree Automata for Possibly Negated Atomic Formulas

The ultimate goal is to construct, from an $\mathrm{MSO}(\Gamma, \sigma)$-formula $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$, a "small" tree automaton $\mathcal{B}_\varphi$ that accepts a tree of the form $(\mathsf{tree}(\nu), x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ (where ν is a τ-phase σ-nested word) if and only if $\nu, x_1, \ldots, x_k, X_1, \ldots, X_\ell \models \varphi$. The simplest such formulas φ are the atomic formulas $\lambda(x) = a$, $x \lessdot y$, $x \prec_s y$, $\mathsf{min}(x)$ etc. and their negations that we handle in this section.

Proposition 2 *Given a possibly negated atomic $\mathrm{MSO}(\Gamma, \sigma)$-formula φ not of the form $x \lessdot y$ or $\mathsf{max}(x)$[1] and $\tau \in \mathbb{N}$, one can construct in space $\mathsf{poly}(\tau)$ a tree automaton $\mathcal{B}_\varphi$ with the following property: Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a τ-phase nested word, $i, j \in P$, and $I \subseteq P$. Then $(\mathsf{tree}(\nu), i, j, I)$ is accepted by $\mathcal{B}_\varphi$ if and only if $\nu, i, j, I \models \varphi$.*

Proof: Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a σ-nested word, $i, j \in P$ and $I \subseteq P$. Then $\nu, i, j, I \models \mathsf{min}(x)$ if and only if i is the root of the tree $\mathsf{tree}(\nu)$. From the label of i in $(\mathsf{tree}(\nu), i, j, I)$, one can immediately tell whether $\nu, i, j, I \models (\lambda(x) = a)$ and similarly for the formulas $\mathsf{call}_s(x)$, $\mathsf{ret}_s(x)$, $x \in X$, $x = y$, and for their negations.

[1]Note that φ has at most two free individual and one free set variable.

Note that $i \prec_s j$ is equivalent to $(i, j) \in E_1$ and $i \in \mathsf{call}_s$. Hence the above automata for $\mathsf{call}_s(x)$ and for $\neg\mathsf{call}_s(x)$ together with standard automata techniques allow us to handle the formulas $x \prec_s y$ and $\neg(x \prec_s y)$.

Note that i is *no* immediate predecessor of j if and only if $j \leq i$ or there exists a position k with $i < k < j$. Since [17] proves the claim for the formula $\varphi = (x \leq y)$ (which is not part of our logic), standard automata techniques allow us to handle $\neg(x \lessdot j)$. Similar arguments apply to the formula $\neg\mathsf{max}(x)$. ■

The real difficulty comes with the remaining atomic formulas φ of the form $x \lessdot y$ or $\mathsf{max}(x)$. We could, of course, simply complement the automaton $\mathcal{B}_{\neg\varphi}$ from Proposition 2. But this requires exponential space.

In a first step, we will construct a tree automaton accepting $(\mathsf{tree}(\nu), X, x)$ if and only if X is the set of positions of ν preceding x in ν. To this aim, we use the new characterisation of the order relation $\leq$ of ν in the tree $\mathsf{tree}(\nu)$ from Lemma 1.

Definition 11 *Let $T = (V, E_0, E_1, \ell)$ be a tree over the alphabet $\Gamma \times [\sigma]_0^2 \times [\tau]$. For $v \in V$, we let $\mathsf{phase}(v)$ denote its phase, i.e., the number $t \in [\tau]$ with $\ell(v) = (a, s, s', t)$. Then the* phase word $\mathsf{pw}(v) \in [\tau]^+$ *is defined by induction:*

$$\mathsf{pw}(v) = \begin{cases} \mathsf{phase}(v) & \text{if } v \text{ is the root of } T \\ \mathsf{pw}(u) & \text{if } (u,v) \in E_0 \cup E_1 \text{ and} \\ & \quad \mathsf{phase}(v) = \mathsf{phase}(u) \\ \mathsf{pw}(u)\,\mathsf{phase}(v) & \text{if } (u,v) \in E_0 \cup E_1 \text{ and} \\ & \quad \mathsf{phase}(v) \neq \mathsf{phase}(u) \end{cases}$$

Intuitively, the phase word of v recalls the sequence of the phases on the path from the root to the node v where stuttering is deleted.

If ν is a τ-phase nested word, then any phase word from the tree $\mathsf{tree}(\nu)$ begins with 1 and its entries increase properly. On the set of these phase words, we define a strict linear order: $(s_1, \ldots, s_m) \sqsubset (t_1, \ldots, t_n)$ if and only if

- $s_m < t_n$ or
- $s_m = t_n$ and $(s_1, \ldots, s_{m-1}) \sqsupset (t_1, \ldots, t_{n-1})$.

For instance, $(1, 2, 4) \sqsubset (1, 5)$ and therefore $(1, 2, 4, 6) \sqsupset (1, 5, 6)$.

Lemma 1 *Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a τ-phase σ-nested word, $x, y \in P$, and $\mathsf{tree}(\nu) = (P, E_0, E_1, \ell)$ be the tree encoding of ν. Then $x < y$ if and only if*

(1) $\mathsf{pw}(x) \sqsubset \mathsf{pw}(y)$ *or*

(2) $\mathsf{pw}(x) = \mathsf{pw}(y)$ *and x is a predecessor of y in $\mathsf{tree}(\nu)$ or*

(3) $\mathsf{pw}(x) = \mathsf{pw}(y)$ *and there exist positions $z, x', y' \in P$ such that $x' \neq y'$, $(z, x'), (z, y') \in E_0 \cup E_1$, x' is a predecessor of x and y' one of y and*

$$(z, x') \in E_0 \text{ if and only if}$$
$$\Big(|\mathsf{pw}(x)| - |\mathsf{pw}(z)| \text{ even } \iff \mathsf{ph}_\nu(x') = \mathsf{ph}_\nu(y')\Big).$$

Based on this lemma, one can prove the following.

Lemma 2 *From $\tau \in \mathbb{N}$, one can construct in space $\mathsf{poly}(\tau)$ a tree automaton $\mathcal{B}$ satisfying the following property: Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a τ-phase nested word, $x \in P$, and $X \subseteq P$. Then $(\mathsf{tree}(\nu), x, X)$ is accepted by $\mathcal{B}$ iff $X = \{y \in P \mid y \leq x\}$.*

Proof: One first builds a tree automaton $\mathcal{B}'$ that accepts $(\mathsf{tree}(\nu), x, H)$ if H is the set of positions y of $\mathsf{tree}(\nu)$ that are incomparable with x and satisfy

- the direct predecessor of y belongs to H or
- the direct predecessor z of y is a predecessor of x and

$$y \text{ is a return} \iff \begin{pmatrix} |\mathsf{pw}(x)| - |\mathsf{pw}(z)| \text{ is even} \\ \iff \\ \mathsf{ph}_\nu(x') = \mathsf{ph}_\nu(y) \end{pmatrix}$$

where x' is the direct successor of z on the path to x.

Now $\mathcal{B}$ is obtained from $\mathcal{B}'$ as follows (recall that $\mathcal{B}$ runs on trees of the form $(\mathsf{tree}(\nu), x, X)$):

- It guesses a set $H \subseteq P$ and verifies that $(\mathsf{tree}(\nu), x, H)$ is accepted by $\mathcal{B}'$.
- It verifies that X contains precisely those positions y that satisfy

(1) $\mathsf{pw}(x) \sqsubset \mathsf{pw}(y)$ or
(2) $\mathsf{pw}(x) = \mathsf{pw}(y)$ and x is a predecessor of y or
(3) $\mathsf{pw}(x) = \mathsf{pw}(y)$ and $y \in H$.

By Lemma 1, $\mathcal{B}$ accepts the correct trees $(\mathsf{tree}(\nu), x, X)$. ∎

Based on this automaton, one can easily construct tree automata for the formulas $x \lessdot y$ and $\mathsf{max}(x)$.

Proposition 3 *Given a formula φ of the form $x \lessdot y$ or $\mathsf{max}(x)$ and $\tau \in \mathbb{N}$, one can construct in space $\mathsf{poly}(\tau)$ a tree automaton $\mathcal{B}_\varphi$ satisfying the following property: Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a τ-phase nested word and $i, j \in P$. Then $(\mathsf{tree}(\nu), i, j)$ is accepted by $\mathcal{B}_\varphi$ iff $\nu, i, j \models \varphi$.*

Proof: Running on $(\mathsf{tree}(\nu), i, j)$, the tree automaton $\mathcal{B}_{x \lessdot y}$ proceeds as follows:

(1) It guesses sets $X, Y \subseteq P$ and verifies that $X = \{x \in P \mid x \leq i\}$ and $Y = \{x \in P \mid x \leq j\}$.
(2) It verifies that $Y \setminus X = \{j\}$.

Similarly, running on $(\mathsf{tree}(\nu), i)$, the tree automaton $\mathcal{B}_{\mathsf{max}(x)}$ guesses a set $X \subseteq P$ and verifies $X = \{x \in P \mid x \leq i\}$ and $X = P$.

By Lemma 2, these automata can be constructed in space $\mathsf{poly}(\tau)$ and accept the correct encodings of τ-phase nested words. ∎

D. The Decision Procedure

We need one final lemma for the translation of temporal formulas into tree automata:

Lemma 3 *From a tree automaton $\mathcal{B}$ over $\Lambda \times \{0,1\}$, one can construct a tree automaton $\mathcal{B}'$ with $L(\mathcal{B}') = \{T \in \mathcal{T}_\Lambda \mid (T, v) \in L(\mathcal{B})$ for all $v \in T\}$ in space $\mathsf{poly}(|Q| + |\Lambda|)$.*

Proof: Let Q be the set of states of $\mathcal{B}$. The states of $\mathcal{B}'$ are pairs (M_1, M_2) of subsets of Q. Its transition relation is defined in such a way that, on the tree T, the tree automaton $\mathcal{B}'$ can reach the state (M_1, M_2) if and only if

(i) M_1 is the set of states $p \in Q$ that $\mathcal{B}$ can reach on $(T, \emptyset)$.
(ii) For all positions v of T, there is a state $p \in M_2$ such that $\mathcal{B}$ can reach p on $(T, \{v\})$.

The state (M_1, M_2) of $\mathcal{B}'$ is accepting iff M_2 is a set of accepting states of $\mathcal{B}$. ∎

The main difficulties of the translation of a temporal formula into a tree automaton have been mastered in the above Propositions 1, 2, and 3. It remains to assemble these ingredients using quite standard arguments:

Theorem 3 *Let TL_n be an $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic. From a formula F from TL_n and $\tau \in \mathbb{N}$, one can construct in space $\mathsf{tower}_{n+1}(\mathsf{poly}(|F| + \tau))$ a tree automaton $\mathcal{B}_F$ over the alphabet Γ with the following property: For any τ-phase σ-nested word ν, we have*

$$\nu \models F \iff \mathsf{tree}(\nu) \in L(\mathcal{B}_F)\,.$$

Proof: By Prop. 1, we can construct a formula $\psi = \exists \overline{X}\,(\psi_1(\overline{X}) \wedge \forall y\, \psi_2(y, \overline{X}))$ that is (over σ-nested words) equivalent to F. Recall that $\psi_1 \in \Pi^M_{n+1}$ and $\psi_2 \in \Sigma^M_{n+1}$. We can compute in polynomial time a formula $\psi_2' = \exists \overline{X}_1\, \neg \exists \overline{X}_2 \ldots \neg \exists \overline{X}_{n+1} \colon \psi_0$, where ψ_0 is quantifier-free and $\overline{X}_i$ are tuples of individual and set variables, which is logically equivalent to ψ_2. Even more, we can assume that ψ_0 is a positive Boolean combination of possibly negated atomic formulas. Note that $|\psi_2'|$ (and therefore $|\psi_0|$) is polynomial in $|\psi|$. Using Propositions 2 and 3 and standard constructions (for union and intersection) from automata theory, we can transform ψ_0 into a tree automaton $\mathcal{B}_{\psi_0}$ in space $\mathsf{poly}(|\psi| \cdot \tau)$. The desired tree automaton $\mathcal{B}_{\psi_2}$ is obtained from $\mathcal{B}_{\psi_0}$ by a sequence of n complementations and $n+1$ projections. Hence this construction can be carried out in space $\mathsf{tower}_n(\mathsf{poly}(|\psi| \cdot \tau))$ and the size of $\mathcal{B}_{\psi_2}$ is bounded by $\mathsf{tower}_{n+1}(\mathsf{poly}(|\psi| \cdot \tau))$. Then, by Lemma 3, we can also translate $\forall y\, \psi_2(y, \overline{X})$ into a tree automaton of size $\mathsf{tower}_{n+2}(\mathsf{poly}(|\psi| \cdot \tau))$. Since a formula belongs to Π^M_{n+1} iff its negation belongs to Σ^M_{n+1}, we similarly obtain a tree automaton $\mathcal{B}_{\psi_1}$ for ψ_1 of size $\mathsf{tower}_{n+2}(\mathsf{poly}(|\psi| \cdot \tau))$ in space $\mathsf{tower}_{n+1}(\mathsf{poly}(|\psi| \cdot \tau))$. Finally, by standard automata techniques for intersection and projection, we obtain the tree automaton for ψ. ∎

Theorem 4 *Let $n \geq 0$ and TL_n be some $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic. Then the satisfiability problem of TL_n is in $(n+2)$-EXPTIME (where τ is encoded in unary).*

Proof: Let $\tau \in \mathbb{N}$ and F be a temporal formula from TL_n. First construct the tree automaton $\mathcal{B}_F$ from Theorem 3. Furthermore, let $\mathcal{B}_\tau$ be the tree automaton from Theorem 2. Since $n \geq 0$, the number of states of both these automata is bounded by $\mathsf{tower}_{n+2}(\mathsf{poly}(|F| + \tau))$.

Note that we have $L(\mathcal{B}_F) \cap L(\mathcal{B}_\tau) = \emptyset$ if and only if F is not satisfiable by any τ-phase nested word. This can be decided in time polynomial in $|\mathcal{B}_F| \cdot |\mathcal{B}_\tau|$, i.e., in time $\mathsf{tower}_{n+2}(\mathsf{poly}(|F| + \tau))$. ∎

IV. Lower Bound

In this section, we show that there exist $\mathrm{M}\Sigma_n(\Gamma,\sigma)$-definable temporal logics whose satisfiability problem is n-EXPSPACE-hard. In order to present a plain and modular proof, we proceed in two steps. First of all, we show a lower bound on a restricted version of the satisfiability problem of temporal logics over labelled grids. Secondly, we reduce this satisfiability problem to the satisfiability problem for $\mathrm{M}\Sigma_n(\Gamma,\sigma)$-definable temporal logics over nested words.

A. Labelled Grids

In this section, we introduce the notion of labelled grids. We also define MSO and temporal logics which are evaluated over labelled grids.

Definition 12 *A* labelled grid *over Γ is a tuple $G=(k,m,\mu)$ where $k,m\geq 1$ specify the number of rows and columns, resp., and $\mu:[k]\times[m]\to\Gamma$ is a mapping. The elements of $\mathsf{dom}(G)=[k]\times[m]$ are called* cells*. Furthermore, we define the horizontal and the vertical successor relations:*

$$S_h=\{((r,c_1),(r,c_2)) \mid c_2=c_1+1\}\subseteq \mathsf{dom}(G)^2$$
$$S_v=\{((r_1,c),(r_2,c)) \mid r_2=r_1+1\}\subseteq \mathsf{dom}(G)^2$$

Definition 13 *The set $\mathrm{MSO}^G(\Gamma)$ of MSO formulas φ over labelled grids is given by the following grammar, where $a\in\Gamma$:*

$$\begin{aligned}\varphi \quad ::= \quad & \mu(x)=a \mid x\ S_h\ y \mid x\ S_v\ y \mid x=y\\ & \mid x\in X \mid \neg\varphi \mid \varphi\vee\varphi \mid \exists x\,\varphi \mid \exists X\,\varphi\end{aligned}$$

For a grid G, $(r_1,c_1),(r_2,c_2)\in\mathsf{dom}(G)$, and $a\in\Gamma$, we have $G,(r_1,c_1)\models\mu(x)=a$ if $\mu(r_1,c_1)=a$. Furthermore, we have $G,(r_1,c_1),(r_2,c_2)\models x\ S_h\ y$ if $(r_1,c_1)\ S_h\ (r_2,c_2)$; the other connectives are as expected. If φ is a sentence, then we denote by $L(\varphi)$ the set of grids G with $G\models\varphi$.

Definition 14 *An $\mathrm{MSO}^G(\Gamma)$-definable temporal logic is defined similar to an $\mathrm{MSO}(\Gamma,\sigma)$-definable temporal logic (see Definition 6). If TL^G is an $\mathrm{MSO}^G(\Gamma)$-definable temporal logic and $F\in\mathrm{TL}^G$, then we write $G\models F$ if $G,(1,1)\models F$. The sets $\mathrm{FO}^G(\Gamma)$, $\mathrm{M}\Sigma_n^G(\Gamma)$, and $\mathrm{M}\Pi_n^G(\Gamma)$ are defined like the corresponding fragments of the logic $\mathrm{MSO}(\Gamma,\sigma)$ (see Definitions 5 and 8). If TL^G is some $\mathrm{MSO}^G(\Gamma)$-definable temporal logic, then the* satisfiability problem *of TL^G is the set of pairs (F,m) where $F\in\mathrm{TL}^G$ is a formula and $m\in\mathbb{N}$ such that there exists some grid G with m columns and $G\models F$.*

Let M be a Turing machine whose language is n-EXPSPACE-complete. We can assume that M works in space $F_n(m)$ where $F_0(m)=m$ and $F_{\ell+1}(m)=F_\ell(m)\cdot 2^{F_\ell(m)}$. Now let w be an input of M of length $m-3$. We then consider grids with m columns whose cells carry tape symbols or states or endmarkers. For the hardness of the satisfiability problem, one then expresses about a grid G with m columns

- that the initial configuration with input w is the inscription of the first row and
- that the inscription of the grid, read *row by row*, is a successful computation of M.

The first statement can easily be expressed using the modality $[\![S_h]\!](x,X_1)=\exists y\colon x\ S_h\ y\wedge y\in X_1$. The main problem with the second statement is that one has to relate positions x and $x+F_n(m)$ (in the sequence of labels of the grid read row by row). This is easy for $n=0$ since then, $x+F_0(m)$ is the position directly below x: $y=x+F_0(m)$ if and only if $G,x,y\models(x\ S_v\ y)$. Adopting the technique from [24], [27], [13], one can then show by induction (with $\varphi_0=(x\ S_v\ y)$):

Lemma 4 *For all $n\geq 0$, there exists a formula $\varphi_n(x,y)\in\mathrm{M}\Sigma_n^G(\Gamma)$ such that, for all grids $G=(k,m,\mu)$ and cells $(r_1,c_1),(r_2,c_2)\in\mathsf{dom}(G)$, we have*

$$\begin{aligned}&G,(r_1,c_1),(r_2,c_2)\models\varphi_n\\ \iff\ &(r_2-1)m+c_2=(r_1-1)m+c_1+F_n(m)\,.\end{aligned}$$

Using this formula $\varphi_n\in\mathrm{M}\Sigma_n^G(\Gamma)$, one can build a 0-ary modality from $\mathrm{M}\Pi_n^G(\Gamma)$ expressing that the sequence of labels of the grid G is an accepting computation of the Turing machine M. Together with the modalities for Boolean connectives and S_h (see above), one obtains

Theorem 5 *For every $n\geq 1$ and alphabet Γ with $|\Gamma|\geq 2$, there exists an $\mathrm{M}\Pi_n^G(\Gamma)$-definable temporal logic TL^G with an n-EXPSPACE-hard satisfiability problem.*

B. Representing Labelled Grids by Nested Words

Our goal is a polynomial reduction of the satisfiability problem of a $\mathrm{M}\Pi_n^G(\Gamma)$-definable temporal logic expressing properties of labelled grids to the satisfiability problem of an $\mathrm{M}\Pi_n(\Gamma,\sigma)$-definable temporal logic which is evaluated on nested words. For this purpose, we represent a grid $G=(k,m,\mu)$ over Γ by a $2m$-phase 2-nested word ν_G over the alphabet $\Gamma\uplus\{\bot\}$. Note that we insert artificial blocks of $\bot$ since, by this means, it is technically easier to navigate within ν_G. More precisely, the $2m$-phase 2-nested word $\nu_G=(P,\leq,\lambda,\prec_1,\prec_2)$ over $\Gamma\uplus\{\bot\}$ is defined as follows: The set of positions is given by $P=[k]\times[2m]$ and, for all $(i,j)\in P$, we have $\lambda(i,j)=\mu(i,j/2)$ if j is even and $\lambda(i,j)=\bot$ otherwise. For all $(i_1,j_1),(i_2,j_2)\in P$, we have $(i_1,j_1)\leq(i_2,j_2)$ if

$$\begin{aligned}& j_1<j_2\\ \text{or}\quad & (j_1=j_2 \text{ is odd and } i_1\leq i_2)\\ \text{or}\quad & (j_1=j_2 \text{ is even and } i_2\leq i_1)\,.\end{aligned}$$

The nesting relations $\prec_1$ and $\prec_2$ are defined as follows:

$$\prec_1=\{\big((i_1,j_1),(i_2,j_2)\big)\in P^2 \mid i_1=i_2,\ j_2=j_1+1 \text{ even}\}$$
$$\prec_2=\{\big((i_1,j_1),(i_2,j_2)\big)\in P^2 \mid i_1=i_2,\ j_2=j_1+1 \text{ odd}\}$$

Example 5 Consider the grid $G=(3,3,\mu)$ from Fig. 3. The corresponding nested word ν_G is also depicted in Fig. 3. The upper and lower edges visualize the nesting relation $\prec_1$ and $\prec_2$, resp.

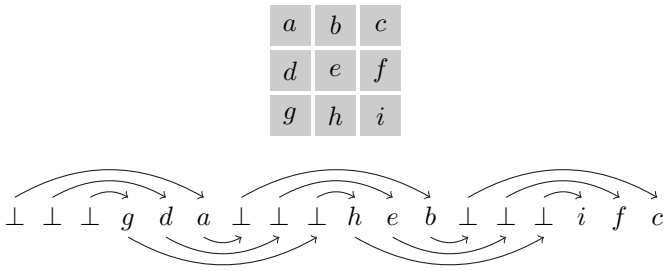

Fig. 3. The grid G and the nested word ν_G from Example 5.

Proposition 4 *There exists a sentence* $\mathrm{grid} \in \mathrm{M}\Pi_1(\Gamma, \sigma)$ *such that, for all nested words* ν *over the alphabet* $\Gamma \uplus \bot$*, we have* $\nu \models \mathrm{grid}$ *if and only if there exists a grid* G *over* Γ *with* $\nu_G = \nu$*.*

Note that the horizontal successor of a node of the grid is the $\prec_2 \circ \prec_1$-successor in the nested word and the vertical successor is the direct predecessor in the nested word. From this interpretation, we get

Lemma 5 *From a modality* $\varphi(x, X_1, \ldots, X_n) \in \mathrm{M}\Pi_n^G(\Gamma)$*, one can compute a modality* $\varphi^{\#} \in \mathrm{M}\Pi_n(\Gamma, 2)$ *such that the following holds for all grids* $G = (k, m, \mu)$*, all positions* $(r, c) \in \mathsf{dom}(G)$*, and all sets* $I_1, \ldots, I_n \subseteq \mathsf{dom}(G)$*:*

$$G, (r, c), I_1, \ldots, I_n \models \varphi \iff \nu_G, (r, 2c), I_1', \ldots, I_n' \models \varphi^{\#}$$

where $I_i' = \{(r, 2c) \mid (r, c) \in I_i\}$ *for all* $i \in [n]$*.*

Now let $n \geq 1$ and let $\mathrm{TL} = (B, \mathrm{arity}, [\![-]\!])$ be some $\mathrm{M}\Pi_n^G(\Gamma)$-definable temporal logic. Then define a $\mathrm{M}\Pi_n(\Gamma, 2)$-definable temporal logic $\mathrm{TL}^{\#} = (B^{\#}, \mathrm{arity}^{\#}, [\![-]\!]^{\#})$ as follows:

- $B^{\#} = B \uplus \{\mathrm{GRID}, \mathsf{N}_1\}$
- $\mathrm{arity}^{\#} \restriction B = \mathrm{arity}$, $\mathrm{arity}^{\#}(\mathrm{GRID}) = 0$, $\mathrm{arity}^{\#}(\mathsf{N}_1) = 1$
- $[\![M]\!]^{\#} = (\varphi^{\#} \wedge (\lambda(x) \neq \bot)$ for $M \in B$ with $[\![M]\!] = \varphi(x, X_1, \ldots, X_n)$, $[\![\mathrm{GRID}]\!]^{\#} = \mathrm{grid}$, and $[\![\mathsf{N}_1]\!]^{\#} = \exists y \colon (x \prec_1 y \wedge y \in X_1)$.

Then the following holds:

Lemma 6 *If* TL *is an* $\mathrm{MSO}^G(\Gamma)$*-definable temporal logic,* $F \in \mathrm{TL}$*, and* G *is a grid, then* $G \models_{\mathrm{TL}} F$ *if and only if* $\nu_G \models_{\mathrm{TL}^{\#}} \mathsf{N}_1 F$*.*

Now, we are able to prove the main theorem of this section:

Theorem 6 *For all* $n \geq 1$*, alphabets* Γ *with* $|\Gamma| \geq 3$*, and* $\sigma \geq 2$*, there is an* $\mathrm{M}\Pi_n(\Gamma, \sigma)$*-definable temporal logic whose satisfiability problem is* n-EXPSPACE*-hard.*

Proof: It suffices to only consider the case $\sigma = 2$. By Theorem 5, there exists an $\mathrm{M}\Pi_n^G(\Gamma)$-definable temporal logic TL whose satisfiability problem is n-EXPSPACE-hard. It follows from Prop. 4 and Lemma 6 that, for every $F \in \mathrm{TL}$ and $m \geq 1$, there exists an m-column grid G with $G \models_{\mathrm{TL}} F$ if and only if there exists a $2m$-phase 2-nested word ν with $\nu \models_{\mathrm{TL}^{\#}} \mathrm{GRID} \wedge \mathsf{N}_1 F$. The temporal logic $\mathrm{TL}^{\#}$ and the formula $\mathsf{N}_1 F$ can be constructed in linear time. Hence, we polynomially reduced the satisfiability problem of TL to the satisfiability problem of the $\mathrm{M}\Pi_n(\Gamma, 2)$-definable temporal logic $\mathrm{TL}^{\#}$. ∎

V. Model Checking

This section deals with the model checking problem: do all runs of a system satisfy a given temporal-logic formula? As a system model, we consider σ-stack automata. There are essentially two approaches of presenting such automata. In [17], stacks are included explicitly. Here, following [3], we let automata run directly on nested words. Note that we extend the model straightforwardly to also handle nested words where a position can be both a call and a return (from different stacks).

Definition 15 *A* σ-stack automaton *over* Γ *is a tuple* $\mathcal{A} = (Q, \Delta, \iota, F)$ *where*

- Q *is the finite set of* states*,*
- $\iota \in Q$ *is the* initial state*,*
- $F \subseteq Q$ *is the set of* final states*, and*
- $\Delta \subseteq Q \times \big(\{\#\} \cup ([\sigma] \times Q)\big) \times \Gamma \times Q$ *is the* transition relation*.*

Reading a position i of a nested word, transition $(q, C, a, q') \in \Delta$ lets the automaton move on from the current state q to the target state q' if i is labeled with letter a. In addition, the transition is guarded by $C \in \{\#\} \cup ([\sigma] \times Q)$. This allows $\mathcal{A}$ to retrieve, from a return position, the state reached after executing the corresponding call. In a sense, this is equivalent to reading a stack symbol previously pushed. More precisely, if $C = (s, q) \in [\sigma] \times Q$, then we require that i is a return from stack s and that q is the state reached at position j with $j \prec_s i$. If, on the other hand, $C = \#$, then i should *not* be a return at all.

Let $\nu = (P, \leq, \lambda, \prec_1, \ldots, \prec_\sigma)$ be a σ-nested word where we suppose $P = \{1, \ldots, n\}$. A *run* of $\mathcal{A}$ on ν is a mapping $\rho : P \to Q$ such that $\big(\iota, \#, \lambda(1), \rho(1)\big) \in \Delta$ and, for every $i \in \{2, \ldots, n\}$,

$$\big(\rho(i-1), C_i, \lambda(i), \rho(i)\big) \in \Delta$$

where

$$C_i = \begin{cases} (s, \rho(j)) & \text{if } j \prec_s i \\ \# & \text{if there are no } s, j \text{ such that } j \prec_s i \end{cases}$$

(note that C_i is well-defined by Definitions 2 and 3). The run ρ is accepting if $\rho(n) \in F$.

The set of σ-nested words for which there is an accepting run is denoted by $L(\mathcal{A})$. The restriction of $L(\mathcal{A})$ to τ-phase words is denoted by $L_\tau(\mathcal{A})$.

Let TL be some $\mathrm{MSO}(\Gamma, \sigma)$-definable temporal logic. The *model checking problem* of TL is the set of all triples $(\mathcal{A}, F, \tau)$ where $\mathcal{A}$ is a σ-stack automaton, $F \in \mathrm{TL}$ is a temporal formula, and $\tau \in \mathbb{N}$ such that every τ-phase nested word accepted by $\mathcal{A}$ satisfies F. In order to use our techniques from

the satisfiability problem, we need the following translation of σ-stack automata into tree automata:

Theorem 7 ([17]) *From a σ-stack automaton $\mathcal{A}$ and $\tau \in \mathbb{N}$, one can construct in time $\mathsf{tower}_1(|\mathcal{A}| \cdot \mathsf{tower}_1(\mathsf{poly}(\tau)))$ a tree automaton $\mathcal{B}_\mathcal{A}$ such that $L(\mathcal{B}_\mathcal{A}) = \mathsf{tree}(L_\tau(\mathcal{A}))$.*

Combining this result with Theorem 3, we obtain:

Theorem 8 *Let $n \geq 0$ and TL_n be some $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic. Then the model checking problem of TL_n is in $(n+2)$-EXPTIME (where τ is encoded in unary).*

From Theorem 6, we can infer that, for $n \geq 1$, there is some $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logic for which the model checking problem is n-EXPSPACE-hard.

VI. Conclusion

In this paper, we showed that the satisfiability and the model checking problem of bounded-phase multi-stack systems are decidable in $(n+2)$-EXPTIME for all $\mathrm{M}\Sigma_n(\Gamma, \sigma)$-definable temporal logics. Moreover, we identified temporal logics, for which the problems are n-EXPSPACE-hard.

It was shown in [6], [21] for very specific temporal logics (cf. Example 4) that model checking *bounded-scope* multi-stack systems is in EXPTIME. Note that such an upper bound cannot be achieved under the phase-bound restriction, since the corresponding emptiness problem of multi-stack automata is already 2-EXPTIME-hard. Ordered multi-stack systems were considered in [4], establishing a 2-EXPTIME upper bound for linear-time properties that do not allow one to reason about nesting edges. Recall that the notions of bounded phases, bounded scopes, and ordered stacks are orthogonal. It will be worthwhile to study if our techniques can be used to show tight upper bounds for *all* MSO-definable temporal logics when restricting to bounded scopes or ordered stacks.

Extensions of the shared-memory model were considered in terms of pushdown automata communicating via first-in first-out channels. It was shown that natural restrictions, partly based on the notion of bounded phases, allow for decidable reachability and model checking problems (against MSO properties) [18], [23], [16]. However, temporal logics for this kind of systems have, to our knowledge, not been explored. It is actually very unclear in that case how to define "canonical" temporal operators such as an until. Therefore, our generic approach may serve here as a starting point as well.

References

[1] R. Alur, M. Arenas, P. Barceló, K. Etessami, N. Immerman, and L. Libkin, "First-order and temporal logics for nested words," *Logical Methods in Computer Science*, vol. 4, no. 11, pp. 1–44, 2008.

[2] R. Alur, K. Etessami, and P. Madhusudan, "A temporal logic of nested calls and returns," in *Proceedings of TACAS 2004*, LNCS, vol. 2988. Springer, 2004, pp. 467–481.

[3] R. Alur and P. Madhusudan, "Adding nesting structure to words," *Journal of the ACM*, vol. 56, pp. 16:1–16:43, 2009.

[4] M. F. Atig, "Global Model Checking of Ordered Multi-Pushdown Systems," in *Proceedings of FSTTCS 2010*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 8, 2010, pp. 216–227.

[5] M. F. Atig, B. Bollig, and P. Habermehl, "Emptiness of multi-pushdown automata is 2ETIME-complete," in *Proceedings of DLT 2008*, LNCS, vol. 5257. Springer, 2008, pp. 121–133.

[6] M. F. Atig, A. Bouajjani, K. N. Kumar, and P. Saivasan, "Linear-time model-checking for multithreaded programs under scope-bounding," in *Proceedings of ATVA'12*, M. Mukund and S. Chakraborty, Eds., vol. 7561. Springer, Oct. 2012, pp. 152–166.

[7] ——, "Model checking branching-time properties of multi-pushdown systems is hard," *CoRR*, vol. abs/1205.6928, 2012.

[8] B. Bollig, A. Cyriac, P. Gastin, and M. Zeitoun, "Temporal logics for concurrent recursive programs: Satisfiability and model checking," in *Proceedings of MFCS'11*, LNCS, vol. 6907. Springer, 2011, pp. 132–144.

[9] L. Breveglieri, A. Cherubini, C. Citrini, and S. Crespi Reghizzi, "Multi-push-down languages and grammars," *International Journal of Foundations of Computer Science*, vol. 7, no. 3, pp. 253–292, 1996.

[10] A. Cyriac, P. Gastin, and K. Narayan Kumar, "MSO decidability of multi-pushdown systems via split-width," in *Proceedings of CONCUR 2012*, LNCS, vol. 7454. Springer, 2012, pp. 547–561.

[11] R. Fagin, L. Stockmeyer, and M. Vardi, "On monadic NP vs. monadic co-NP (extended abstract)," in *Structure in Complexity Theory Conference*, 1993, pp. 19–30.

[12] D. Gabbay, I. Hodkinson, and M. Reynolds, *Temporal Logic: Mathematical Foundations and Computational Aspects, vol. 1*. Oxford University Press, 1994.

[13] P. Gastin and D. Kuske, "Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces," *Information and Computation*, vol. 208, no. 7, pp. 797–816, 2010.

[14] S. Göller and A. W. Lin, "Concurrency Makes Simple Theories Hard," in *Proceedings of STACS'12*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 14, 2012, pp. 148–159.

[15] W. Hanf, "Model-theoretic methods in the study of elementary logic," in *The Theory of Models*, J. Addison, L. Henkin, and A. Tarski, Eds. North Holland, 1965, pp. 132–145.

[16] A. Heußner, J. Leroux, A. Muscholl, and G. Sutre, "Reachability analysis of communicating pushdown systems," *Logical Methods in Computer Science*, vol. 8, no. 3:23, pp. 1–20, 2012.

[17] S. La Torre, P. Madhusudan, and G. Parlato, "A robust class of context-sensitive languages," in *Proceedings of LICS'07*. IEEE Computer Society Press, 2007, pp. 161–170.

[18] ——, "Context-bounded analysis of concurrent queue systems," in *Proceedings of TACAS 2008*, LNCS. Springer, 2008, pp. 299–314.

[19] ——, "An infinite automaton characterization of double exponential time," in *Proceedings of CSL'08*, LNCS, vol. 5213. Springer-Verlag, 2008.

[20] S. La Torre and M. Napoli, "Reachability of multistack pushdown systems with scope-bounded matching relations," in *Proceedings of CONCUR 2011*, LNCS, vol. 6901. Springer, 2011, pp. 203–218.

[21] ——, "A temporal logic for multi-threaded programs," in *Proceedings of IFIP-TCS'12*, LNCS. Springer, 2012, vol. 7604, pp. 225–239.

[22] S. La Torre and G. Parlato, "Scope-bounded Multistack Pushdown Systems: Fixed-Point, Sequentialization, and Tree-Width," in *Proceedings of FSTTCS 2012*, ser. Leibniz International Proceedings in Informatics (LIPIcs), D. D'Souza, T. Kavitha, and J. Radhakrishnan, Eds., vol. 18, 2012, pp. 173–184.

[23] P. Madhusudan and G. Parlato, "The tree width of auxiliary storage," in *Proceedings of POPL 2011*. ACM, 2011, pp. 283–294.

[24] O. Matz and W. Thomas, "The monadic quantifier alternation hierarchy over graphs is infinite," in *LICS'97*. IEEE Computer Society Press, 1997, pp. 236–244.

[25] A. Pnueli, "The temporal logic of programs," in *Proceedings of FOCS 1977*. IEEE, 1977, pp. 46–57.

[26] S. Qadeer and J. Rehof, "Context-bounded model checking of concurrent software," in *Proceedings of TACAS 2005*, LNCS, vol. 3440. Springer, 2005, pp. 93–107.

[27] K. Reinhardt, "The complexity of translating logic to finite automata," in *Automata Logics, and Infinite Games*, LNCS, E. Grädel, W. Thomas, and T. Wilke, Eds. Springer, 2002, vol. 2500, pp. 231–238.