# The Complexity of Model Checking Multi-Stack Systems

Benedikt Bollig[1], Dietrich Kuske[2], and Roy Mennicke[2]

[1] Laboratoire Spécification et Vérification, École Normale Supérieure de Cachan & Centre National de la Recherche Scientifique
[2] Technische Universität Ilmenau

**Abstract.** We study 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 local temporal logics as defined in the book by Gabbay, Hodkinson, and Reynolds (1994). The unifying feature of these local temporal logics is that their modalities are defined in monadic second-order (MSO) logic. In particular, this captures numerous local 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 local temporal logics is decidable in elementary time. More precisely, it is solvable in time exponential in the formula and $(n+2)$-fold exponential in the number of phases 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.

## 1 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 $\varphi$ is specified in a temporal logic such as LTL [31]. In the linear-time framework, model checking amounts to the question whether all executions of $\mathcal{A}$ satisfy $\varphi$.

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 behavior 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 [32]. 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* [21]. A phase does not constrain push operations, whereas pop actions are required to belong to some dedicated process. Orthogonal approaches are due to [24, 26], where the number of *scopes* is bounded (a call and its return are separated by a bounded number of contexts), and [11, 5], which assumes an ordering of the stacks and postulates that a pop operation is subject to the first nonempty stack. Furthermore, there is the notion of *split-width* which is similar to tree-width but better fits the setting of multiply nested words [12].

Model checking multi-stack systems has recently received a lot of attention [21, 4, 8, 12, 25, 6, 7]. In [21], 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 behavioral restriction described above, 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, [25] 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 [16], 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 nonrecursive programs and that are typically interpreted over partial orders such as Mazurkiewicz traces (cf. [17] and the references therein) or message sequence charts [29]. Note that we concentrate on local temporal logics only since global temporal logics do not admit model checking algorithms with a reasonable complexity [34].

In [8], it is shown that satisfiability and model checking for any MSO-definable local temporal logic are decidable in EXPTIME when restricting to phase bounded executions. The phase bound $\tau$ has to be fixed, though. It was left open if the problems are still elementary if $\tau$ is part of the input. This is an important issue, as an elementary procedure would allow for a gradual adjustment of $\tau$ 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 exponential in the size of the temporal formula and $(n+2)$-fold exponential in the number of phases where $n$ is the maximal level of the MSO modalities in the monadic quantifier alternation hierarchy. If the number of phases is fixed, the problem is therefore solvable in exponential time, i.e., the result from [8] follows.

Our result is in stark contrast to the non-elementary lower bounds of the branching-time model checking problem [18, 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 [21, 23]. 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 [21]. 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 labeled grids, and then reduced to the satisfiability problem for temporal logics over nested words.

*Outline.* Section 2 introduces crucial notions such as multiply nested words and MSO-definable temporal logics, Section 3 presents the upper bound of the satisfiability problem, Section 4 develops our lower bound, and Section 5 transfers these results to the model checking problem. We conclude in Section 6, giving directions for future work.

An extended abstract of this paper appeared as [10].

## 2 Preliminaries

Given $m, n \in \mathbb{N}$ with $m \leq n$, we denote by $[m, n]$ the set $\{m, m+1, \ldots, n\}$. If $m \in \mathbb{N}$, then we let $[m] = [1, m]$ and $[m]_0 = [0, m]$. Furthermore, the function $\mathrm{tower}\colon \mathbb{N}^2 \to \mathbb{N}$ is inductively defined by $\mathrm{tower}(0, m) = m$ and $\mathrm{tower}(\ell+1, m) = 2^{\mathrm{tower}(\ell, m)}$, for all $\ell, m \in \mathbb{N}$. We let $\mathrm{poly}(n)$ denote the set of polynomial functions in one argument. Finally, if $A$ and $B$ are sets, $f\colon A \to B$ is a mapping, and $b \in B$, then we write $f^{-1}(b)$ for the set $\{a \in A \mid f(a) = b\}$. Similarly, we define $f^{-1}(B') = \{a \in A \mid f(a) \in B'\}$ for all $B' \subseteq B$.

### 2.1 Multiply Nested Words

Let $\Gamma$ be an alphabet, i.e., a non-empty finite set. We define a *word over* $\Gamma$ to be a finite $\Gamma$-labeled linear order $w = (P, \leq, \lambda)$, i.e., $(P, \leq)$ is a finite, non-empty, linearly ordered set and $\lambda\colon P \to \Gamma$ is a mapping. By $\lessdot$, we denote the immediate successor relation with respect to $\leq$, 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 $P$.

A *nesting relation* $\curvearrowright$ over $(P, \leq)$ is a binary relation such that, for all $i, j, i', j' \in P$, the following conditions hold:

(i) if $i \curvearrowright j$, then $i < j$
(ii) if $i \curvearrowright j$, $i' \curvearrowright j'$, and $i \leq i'$, then $i < i' < j' < j$ or $i < j < i' < j'$ or ($i = i'$ and $j = j'$).

If $i \curvearrowright j$, 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 $i \curvearrowright j$ 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 (i) expresses that every return occurs after its matching call. Condition (ii) 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.

In the following, we will consider words with not only one, but with a fixed number of nesting relations $\sigma \geq 1$.

A *nested word over* $\Gamma$ is a tuple $\nu = (w, \curvearrowright_1, \curvearrowright_2, \ldots, \curvearrowright_\sigma)$ where $w = (P, \leq, \lambda)$ is a word over $\Gamma$ and, for all $s \in [\sigma]$, $\curvearrowright_s$ is a nesting relation over $(P, \leq)$ such that we have, for all $1 \leq s < s' \leq \sigma$ and $i, j, i', j' \in P$:

$$i \curvearrowright_s j \text{ and } i' \curvearrowright_{s'} j' \text{ imply } i \neq i' \text{ and } j \neq j' \,. \tag{1}$$

We 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 $i_1 \curvearrowright_s i_2 \curvearrowright_{s'} i_3$ is possible for $s \neq s'$, i.e., a position

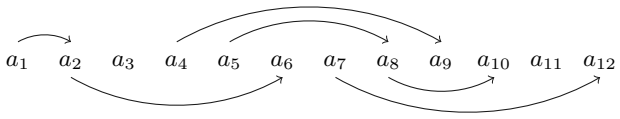


**Fig. 1.** The nested word $\nu = (w, \curvearrowright_1, \curvearrowright_2)$ from Example 2.1. Note that the relation $\curvearrowright_1$ ($\curvearrowright_2$, resp.) is represented by the edges above (below) the word $w$.

can be both, a return and a call, but only with respect to distinct nesting relations (the case $s = s'$ is excluded by condition (ii) of the definition of a nesting relation).

*Example 2.1.* Figure 1 is an illustration of the nested word $\nu = (w, \curvearrowright_1, \curvearrowright_2)$ where $w = ([12], \leq, \lambda)$, $\leq$ is the natural ordering on $[12]$, $\lambda(i) = a_i$ for all $i \in [12]$, $\curvearrowright_1 = \{(1,2), (4,9), (5,8)\}$, and $\curvearrowright_2 = \{(2,6), (7,12), (8,10)\}$.

In order to make the model checking 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. More formally, if $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ is a nested word, then a *phase* of $\nu$ is a subset $I \subseteq P$ for which the following conditions hold:

- there exist $i, j \in P$ such that $I = \{k \in P \mid i \leq k \leq j\}$
- for all $i, i' \in P$, $j, j' \in I$, and $s, s' \in [\sigma]$ with $i \curvearrowright_s j$ and $i' \curvearrowright_{s'} j'$, we have $s = s'$.

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

*Example 2.2.* Consider the nested word $\nu$ from Example 2.1. It can be divided into the four phases $\{1,2,3,4,5\}$, $\{6,7\}$, $\{8,9\}$, and $\{10,11,12\}$. Hence, $\nu$ 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, $\nu$ 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 $\tau$-phase nested word $\nu$ into different phases, the *greedy division* will serve as a canonical example: We define the mapping $\mathrm{ph}_\nu \colon P \to [\tau]$ where $\mathrm{ph}_\nu(i)$ is the minimal number $s \geq 1$ such that the restriction of $\nu$ to the positions $\{1, \ldots, i\}$ is an $s$-phase nested word. Then, for all $s \in [\tau]$, $I_s = \mathrm{ph}_\nu^{-1}(s)$ is a phase of $\nu$, and we have $P = I_1 \cup \ldots \cup I_\tau$. Note that $I_\tau = \emptyset$ if and only if $\nu$ is a $(\tau - 1)$-phase nested word.

### 2.2 Monadic Second-Order Logic

We fix the set $\{x, y, z, \ldots\}$ of individual variables and the set $\{X, Y, Z, \ldots\}$ of set variables. Furthermore, let $\Gamma$ be an alphabet. The class $\mathrm{MSO}(\Gamma)$ of *MSO*

*formulas over* $\Gamma$ 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:

$$\varphi ::= (\lambda(x) = a) \mid x \lessdot y \mid x \curvearrowright_s y \mid \mathrm{call}_s(x) \mid \mathrm{ret}_s(x) \mid \min(x) \mid \max(x) \\ \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

The set $\mathrm{FO}(\Gamma) \subseteq \mathrm{MSO}(\Gamma)$ contains all formulas without second order quantification $\exists X$. We use common abbreviations such as $\varphi_1 \wedge \varphi_2$ for $\neg(\neg\varphi_1 \vee \neg\varphi_2)$ and $\forall x\, \varphi$ for $\neg\exists x\, \neg\varphi$.

The semantics of $\mathrm{MSO}(\Gamma)$-formulas are as follows: Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word and $\delta$ be an assignment that maps individual variables to positions from $P$ and set variables to sets of positions from $P$. Then we define

$$\begin{aligned}
\nu, \delta \models (\lambda(x) = a) &\iff \lambda(\delta(x)) = a \\
\nu, \delta \models x \lessdot y &\iff \delta(x) \lessdot \delta(y) \\
\nu, \delta \models x \curvearrowright_s y &\iff \delta(x) \curvearrowright_s \delta(y) \\
\nu, \delta \models \mathrm{call}_s(x) &\iff \text{there exists } i \in P \text{ such that } \delta(x) \curvearrowright_s i \\
\nu, \delta \models \mathrm{ret}_s(x) &\iff \text{there exists } i \in P \text{ such that } i \curvearrowright_s \delta(x) \\
\nu, \delta \models \min(x) &\iff \delta(x) = \min(P, \leq) \\
\nu, \delta \models \max(x) &\iff \delta(x) = \max(P, \leq) \\
\nu, \delta \models (x = y) &\iff \delta(x) = \delta(y) \\
\nu, \delta \models x \in X &\iff \delta(x) \in \delta(X) \\
\nu, \delta \models \neg\varphi &\iff \nu, \delta \not\models \varphi \\
\nu, \delta \models \varphi_1 \vee \varphi_2 &\iff \nu, \delta \models \varphi_1 \text{ or } \nu, \delta \models \varphi_2 \\
\nu, \delta \models \exists x\, \varphi &\iff \text{there exists } i \in P \text{ such that } \nu, \delta[x \mapsto i] \models \varphi \\
\nu, \delta \models \exists X\, \varphi &\iff \text{there exists } I \subseteq P \text{ such that } \nu, \delta[X \mapsto P] \models \varphi
\end{aligned}$$

where the assignment $\delta[x \mapsto i]$ equals $\delta$ besides that $x$ is mapped to $i$ (similarly for $\delta[X \mapsto I]$). We often write $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ to stress the fact that $\varphi$ is 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 and only if $\nu, \delta \models \varphi$ where $\delta(x_m) = i_m$ and $\delta(X_n) = I_n$ for all $m \in [k]$ and $n \in [\ell]$.

*Example 2.3.* For every $s \in [\sigma]$, we define the following FO-formula $\mathrm{bij}_s(X)$ which expresses that $X$ can be partitioned into two sets $X_c$ and $X_r$ such that $\curvearrowright_s \cap X^2$ is a bijection from $X_c$ to $X_r$:

$$\mathrm{bij}_s(X) = \forall x \in X \exists y \in X\, (x \curvearrowright_s y \vee y \curvearrowright_s x)$$

Let $\nu = (P, \leq, \lambda, \nu_1, \ldots, \nu_\sigma)$ be a nested word and $I \subseteq P$. First suppose $\nu, I \models \mathrm{bij}_s$. Then, for every position from $I$, there is a matching call resp. return position in $I$. In particular, all positions from $I$ are either call or return positions with

respect to $\curvearrowright_s$. Let $I_c, I_r \subseteq I$ denote set of all call and return positions from $I$, resp. Then $\curvearrowright_s \cap I^2$ is a bijection from $I_c$ onto $I_r$.

Conversely, let $I_c, I_r \subseteq P$ be sets of call and return positions, resp., with respect to the nesting relation $\curvearrowright_s$ and let $I = I_c \cup I_r$. If $\curvearrowright_s \cap I^2$ is a bijection from $I_c$ onto $I_r$, then $\nu, I \models \mathrm{bij}_s$.

### 2.3 MSO-definable Temporal Logics

An MSO($\Gamma$)-formula is an *$m$-ary modality definition* if it has one free individual variable $x$ and $m$ free set variables $X_1, \ldots, X_m$. An *MSO($\Gamma$)-definable temporal logic* is a tuple $\mathrm{TL} = (B, \mathrm{ar}, [\![-]\!])$ where $B$ is a finite set of modalities, the mapping $\mathrm{ar}\colon B \to \mathbb{N}$ specifies the arity of every modality from $B$, and $[\![-]\!]\colon B \to \mathrm{MSO}(\Gamma)$ is a mapping such that $[\![M]\!]$ is an $m$-ary modality definition whenever $\mathrm{ar}(M) = m$ for $M \in B$. The set of all formulas from TL is the least set such that the following holds: if $M \in B$ and $F_1, F_2, \ldots, F_{\mathrm{ar}(M)}$ are formulas from TL, then $M(F_1, \ldots, F_{\mathrm{ar}(M)})$ is a formula from TL. The size $|F|$ of a temporal formula $F$ is its number of subformulas.

Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word and $F \in \mathrm{TL}$ be a formula. The semantics $F^{\nu,\mathrm{TL}}$ of $F$ in $\nu$ 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 $\nu, \min(P, \leq) \models_{\mathrm{TL}} F$.

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 $F^\nu$ instead of $F^{\nu,\mathrm{TL}}$ and $\nu \models F$ instead of $\nu \models_{\mathrm{TL}} F$.

*Example 2.4.* The Boolean connectives negation and conjunction can be expressed by $[\![\neg]\!](X_1, x) = \neg(x \in X_1)$ and $[\![\vee]\!](X_1, X_2, x) = (x \in X_1) \vee (x \in X_2)$. The modality $\mathsf{TRUE}$ with $[\![\mathsf{TRUE}]\!](x) = (x = x)$ is always evaluated to true.

Let us also consider the modality $\mathsf{RET}$. Intuitively, $\mathsf{RET}\, F$ expresses that the current position is a call position and that its matching return position fulfills the formula $F$. Formally, we set

$$[\![\mathsf{RET}]\!](X_1, x) = \exists y\, [y \in X_1 \wedge \textstyle\bigvee_{s \in [\sigma]} x \curvearrowright_s y]\,.$$

*Example 2.5.* A wide range of temporal logics has 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 [25]. As an example, we consider the *abstract until* $\mathsf{U}^{\mathrm{a}}_s$. Intuitively, $F_1 \,\mathsf{U}^{\mathrm{a}}_s\, F_2$ expresses that there exists an event $x$ fulfilling $F_2$ which can be reached from the current event by an abstract $s$-path such that every event on this path (but not necessarily $x$) fulfills $F_1$. 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. $\curvearrowright_s$). Formally, the abstract until is given by

$$[\![\mathsf{U}^{\mathrm{a}}_s]\!](X_1, X_2, x) = \exists Y \exists z \left[\begin{array}{l} z \in X_2 \wedge Y \subseteq X_1 \\ \wedge\, \forall y \, (y \in Y \vee y = z) \rightarrow \big(y = x \vee \exists y' \, (y' \in Y \wedge \varphi_s(y', y))\big) \end{array}\right]$$

where

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

Indeed, all the modalities considered in [1, 4, 6, 25] 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 [25]. Such extensions can be realized in our framework by giving their definition in $\mathrm{MSO}(\Gamma)$. 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 $\mathrm{MSO}(\Gamma)$ and the model checking problem of every non-trivial temporal logic is undecidable as well. La Torre et al. [21] proposed the restriction of these problems to $\tau$-phase words and showed that, under this restriction, the emptiness problem as well as the satisfiability problem of $\mathrm{MSO}(\Gamma)$ are decidable. Here, we define the satisfiability problem for temporal logics, restricted to a given number of phases $\tau$: Let TL be some $\mathrm{MSO}(\Gamma)$-definable temporal logic. The *satisfiability problem* of TL is the set of pairs $(F, \tau)$ where $F \in \mathrm{TL}$ is a formula and $\tau \in \mathbb{N}$ such that there exists some $\tau$-phase nested word $\nu$ with $\nu \models F$.

We now introduce a measure of the complexity of formulas: the *monadic quantifier alternation hierarchy*. An $\mathrm{MSO}(\Gamma)$-formula belongs to $M\Sigma_n(\Gamma)$ if it is of the form $\exists \overline{X}_1 \forall \overline{X}_2 \exists \overline{X}_3 \ldots \exists/\forall \overline{X}_n \, \psi$ where $\overline{X}_i$ are tuples of individual and set variables and $\psi$ is a first-order formula. In contrast, it belongs to $M\Pi_n(\Gamma)$ if it is of the form $\forall \overline{X}_1 \exists \overline{X}_2 \forall \overline{X}_3 \ldots \forall/\exists \overline{X}_n \, \psi$. Furthermore, $\mathit{Bool}M\Sigma_n(\Gamma)$ is the set of Boolean combinations of formulas from $M\Sigma_n(\Gamma)$.

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

*Example 2.6.* We have $[\![\mathsf{TRUE}]\!], [\![\vee]\!], [\![\neg]\!], [\![\mathsf{RET}]\!] \in M\Sigma_0(\Gamma)$ and $[\![\mathsf{U}^{\mathrm{a}}_s]\!] \in M\Sigma_1(\Gamma)$ where $\mathsf{TRUE}$, $\vee$, $\neg$, $\mathsf{RET}$, and $\mathsf{U}^{\mathrm{a}}_s$ are the modalities from Examples 2.4 and 2.5, resp.

*Remark 2.7.* Let $\mathrm{TL} = (B, \mathrm{ar}, [\![-]\!])$ be some $Bool M\Sigma_n(\Gamma)$-definable temporal logic. Then there is a finite set $H$ of $M\Sigma_n(\Gamma)$-modalities such that, for every $M \in B$, $[\![M]\!]$ is a Boolean combination of formulas from $H$. Let $\mathrm{TL}_H$ be the temporal logic using the modalities from $H$ together with $\neg$ and $\vee$. Then we can express every modality from TL using the modalities from $\mathrm{TL}_H$. This allows us to reformulate an arbitrary formula from TL equivalently in $\mathrm{TL}_H$. In this process, the number of subformulas and therefore the size of the formula increases linearly. Hence, we polynomially reduced the uniform satisfiability problem of the $Bool M\Sigma_n(\Gamma)$-definable temporal logic TL to that of the $M\Sigma_n(\Gamma)$-definable temporal logic $\mathrm{TL}_H$.

As a consequence, it will suffice to prove the upper complexity bound for $M\Sigma_n(\Gamma)$-definable temporal logics. Dually, the lower bound will be proved for $M\Pi_n(\Gamma)$-definable temporal logics only. From the same reduction, we obtain that it holds for $M\Sigma_n(\Gamma)$-definable logics as well.

## 3 Upper Bound

The purpose of this section is to show that the satisfiability problem of every $Bool M\Sigma_n(\Gamma)$-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 $w = (P, \leq, \lambda)$. Furthermore, we fix the alphabet $\Gamma$ and we write MSO instead of $\mathrm{MSO}(\Gamma)$, $M\Sigma_n$ for $M\Sigma_n(\Gamma)$ etc.

### 3.1 From Temporal Logics to MSO

The first step in our solution of the satisfiability problem is a translation of the temporal formula into an MSO-formula of a particular form (see Prop. 3.5). For this, we start with a second measure for the complexity of formulas from MSO, the *full quantifier alternation hierarchy* (cf. the definition of the monadic quantifier alternation hierarchy on page 8): An MSO-formula belongs to $\Sigma_n^M$ if it is of the form $\exists \overline{X}_1 \, \forall \overline{X}_2 \, \exists \overline{X}_3 \, \ldots \, \exists/\forall \overline{X}_n \, \psi$ where $\overline{X}_i$ are tuples of individual and set variables and $\psi$ is quantifier-free. In contrast, it belongs to $\Pi_n^M$ if it is of the form $\forall \overline{X}_1 \, \exists \overline{X}_2 \, \forall \overline{X}_3 \, \ldots \, \forall/\exists \overline{X}_n \, \psi$.

*Example 3.1.* We have $[\![\mathsf{TRUE}]\!], [\![\vee]\!], [\![\neg]\!] \in \Sigma_0^M$, $[\![\mathsf{RET}]\!] \in \Sigma_1^M$, and $[\![\mathsf{U}_s^{\mathrm{a}}]\!] \in \Sigma_3^M$ where $\mathsf{TRUE}$, $\vee$, $\neg$, $\mathsf{RET}$, and $\mathsf{U}_s^{\mathrm{a}}$ are the modalities from Examples 2.4 and 2.5, respectively.

It is easily seen that $\Sigma_m^M \subseteq M\Sigma_m$ and $M\Sigma_m \not\subseteq \Sigma_n^M$ for any $m, n \in \mathbb{N}$. However, as the following theorem states, every $M\Sigma_n$-definable temporal logic is also $\Sigma_{n+1}^M$-definable without changing the semantics of temporal formulas. This result is achieved by applying Hanf's locality principle. The actual proof of Theorem 3.2 can be found on page 12.

**Theorem 3.2.** *Let $\varphi(x, X_1, \ldots, X_m)$ be an $M\Sigma_n$-modality definition. There exists a modality definition $\psi \in \Sigma^M_{n+1}$ such that, for any nested word $\nu$, any tuple $\overline{X} = (X_1, \ldots, X_m)$ of sets of positions, and any position $x$, we have*

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

Since our logic does not speak about the linear order $\leq$ directly, it handles nested words as structures of bounded degree. This allows us, using Hanf's theorem [19, 15, 14], to translate the first-order part of $\varphi$ into a Boolean combination of existential first-order formulas. For this, we need to introduce the following definitions.

A *nw-like structure* is a tuple $\mathcal{S} = (V, \lessdot, \lambda, (\curvearrowright_s, \mathrm{call}_s, \mathrm{ret}_s)_{s\in[\sigma]}, \min, \max)$ where $V$ is a finite set, $\lambda\colon V \to \Gamma$ is a function, $\mathrm{call}_s$, $\mathrm{ret}_s$, min, max $\subseteq V$ are unary, and $\lessdot, \curvearrowright_s \subseteq V^2$ are binary relations for all $s \in [\sigma]$.

Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word. For $s \in [\sigma]$, let $\mathrm{call}_s \subseteq P$ denote the set of positions $i \in P$ for which there exists a position $j \in P$ with $i \curvearrowright_s j$. The unary relation $\mathrm{ret}_s$ is defined analogously. Furthermore, min $\subseteq P$ is the singleton relation containing the minimal element of $P$ only. Again, max $\subseteq P$ is defined similarly. Then $\nu$ can be understood as the nw-like structure $(P, \lessdot, \lambda, (\curvearrowright_s, \mathrm{call}_s, \mathrm{ret}_s)_{s\in[\sigma]}, \min, \max)$. In the following, we will identify the nested word $\nu$ with this nw-like structure.

Let $\mathcal{S} = (V, \lessdot, \lambda, (\curvearrowright_s, \mathrm{call}_s, \mathrm{ret}_s)_{s\in[\sigma]}, \min, \max)$ be a nw-like structure. For $i, j \in P$, the *distance* $\mathrm{dist}(i, j)$ between $i$ and $j$ is the length of the shortest path from $i$ to $j$ in the undirected graph $(V, \lessdot \cup \lessdot^{-1} \cup \curvearrowright \cup \curvearrowright^{-1})$. In particular, $\mathrm{dist}(i, i) = 0$ and $\mathrm{dist}(i, j) = 1$ if $i \lessdot j$, $j \lessdot i$, $i \curvearrowright_s j$, or $j \curvearrowright_s i$ for some $s \in [\sigma]$. Next, let $\overline{Z} = (Z_0, \ldots, Z_m)$ be a tuple of subsets of $V$. For $r \in \mathbb{N}$, the *$r$-sphere* of $(\mathcal{S}, \overline{Z})$ around $v \in V$, denoted $S_{(\mathcal{S},\overline{Z}),r}(v)$, is the substructure of $(\mathcal{S}, \overline{Z}, v)$ induced by $\{i \in V \mid \mathrm{dist}(v, i) \leq r\}$.

It is a folklore result that, for every sphere $(\mathcal{S}, \overline{Z}, v)$ and $r \in \mathbb{N}$, there exists a first-order formula such that any nested word $\nu$, sets of positions $X_0, \ldots, X_m$, and postion $j$ fulfill this formula if and only if the $r$-sphere of $(\nu, \overline{X})$ around $j$ is isomorphic to $(\mathcal{S}, \overline{Z}, v)$. The obvious proof of this general fact results in a formula whose quantifier-prefix is of the form $\exists^*\forall$. The following lemma shows that in our case, we can avoid the universal quantifier.

**Lemma 3.3.** *Let $\mathcal{S} = (V, \lessdot, \lambda, (\curvearrowright_s, \mathit{call}_s, \mathit{ret}_s)_{s\in[\sigma]}, \mathit{min}, \mathit{max})$ be some nw-like structure, $Z_i \subseteq V$ for $i \in [m]_0$, $r \in \mathbb{N}$, and $v \in V$. Then there exists an existential first-order formula (i.e., a formula from $\Sigma^M_1 \cap FO$) $\mathit{sph}_{(\mathcal{S},\overline{Z}),r}(x_0, X_0, X_1, \ldots, X_m)$ such that, for all nested words $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$, all $X_i \subseteq P$, and all $j \in P$, we have*

$$\nu, X_0, \ldots, X_m, j \models \mathit{sph}_{(\mathcal{S},\overline{Z}),r} \iff S_{(\nu,\overline{X}),r}(j) \cong (\mathcal{S}, \overline{Z}, v)\,. \tag{2}$$

*Proof.* If there is no nested word $\nu$ with $S_{(\nu,\overline{X}),r}(j) \cong (\mathcal{S}, \overline{Z}, v)$ for any sets of positions $X_i$ and any position $j$, then set $\mathrm{sph}_{(\mathcal{S},\overline{Z}),r}(x_0, X_0, \ldots, X_m) = (x_0 \neq x_0)$.

From now on, assume that for some nested word $\nu$, some sets of positions $X_i$, and some position $j$, we have $S_{(\nu,\overline{X}),r}(j) \cong (\mathcal{S}, \overline{Z}, v)$. Suppose $V =$

$\{v_0, v_1, \ldots, v_n\}$ with $v = v_0$. Then let $\varphi$ denote the conjunction of the following formulas (for $a \in \Gamma$, $k, \ell \in [n]_0$, $i \in [m]_0$, and $s \in [\sigma]$):

- $\neg(x_k = x_\ell)$ for $k \neq \ell$
- $x_k \in X_i$ if $v_k \in Z_i$, and $x_k \notin X_i$ otherwise
- $\lambda(x_k) = a$ if $\lambda(v_k) = a$, and $\neg(\lambda(x_k) = a)$ otherwise
- $x_k \lessdot x_\ell$ if $v_k \lessdot v_\ell$, and $\neg(x_k \lessdot x_\ell)$ otherwise
- $x_k \curvearrowright_s x_\ell$ if $v_k \curvearrowright_s v_\ell$, and $\neg(x_k \curvearrowright_s x_\ell)$ otherwise
- $\mathrm{call}_s(x_k)$ if $v_k \in \mathrm{call}_s$, and $\neg\mathrm{call}_s(x_k)$ otherwise
- $\mathrm{ret}_s(x_k)$ if $v_k \in \mathrm{ret}_s$, and $\neg\mathrm{ret}_s(x_k)$ otherwise
- $\min(x_k)$ if $v_k \in \min$, and $\neg\min(x_k)$ otherwise
- $\max(x_k)$ if $v_k \in \max$, and $\neg\max(x_k)$ otherwise

Then set

$$\mathrm{sph}_{(\mathcal{S},\overline{Z}),r} = \exists x_1\, \exists x_2\, \ldots\, \exists x_n\, \varphi\,.$$

Let $\nu'$ be a nested word, $Y_i$ sets of positions, and let $x_0, \ldots, x_n$ be positions in $\nu'$. Then $\nu', \overline{Y}, x_0, \ldots, x_n \models \varphi$ if and only if the mapping $v_k \mapsto x_k$ is an embedding of $(\mathcal{S}, \overline{Z}, v)$ into $S_{(\nu',\overline{Y}),r}(x_0)$. It remains to be shown that this embedding is surjective. Towards a contradiction, assume it is not. Then there exist $k \in [n]_0$ and a position $x$ in $\nu'$ such that $\mathrm{dist}(x_0, x_k) < r$, $\mathrm{dist}(x_k, x) = 1$, and $x \notin \{x_0, \ldots, x_n\}$. Assume $x_k \lessdot x$. In this case, $x_k \notin \max$ and therefore $v_k \notin \max$. Since $(\mathcal{S}, \overline{Z}, v) \cong S_{(\nu,\overline{X}),r}(x_0)$, there is some $\ell \in [n]_0$ with $v_k \lessdot v_\ell$. Hence $x_k \lessdot x_\ell$. Since $\nu'$ is a nested word, $x_k \lessdot x$ and $x_k \lessdot x_\ell$ imply $x = x_\ell$ contradicting $x \notin \{x_0, \ldots, x_n\}$. The other cases, namely $x \lessdot x_k$, $x_k \curvearrowright_s x$ and $x \curvearrowright_s x_k$ for some $s \in [\sigma]$ can be handled similarly (using, instead of the relation max, the unary relations min, $\mathrm{call}_s$, and $\mathrm{ret}_s$, respectively). □

Towards the proof of Theorem 3.2, we state the following lemma.

**Lemma 3.4.** *Let $\varphi$ be some $M\Sigma_0$-modality definition. Then there exists a Boolean combination $\psi$ of existential first-order formulas such that*

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

*for any nested word $\nu$, any tuple $\overline{X}$ of sets of positions, and any position $x$.*

As an aside, we remark that one such Boolean combination $\psi$ can be computed in triply exponential time [9].

*Proof.* Note that every position in a nested word is related to at most 4 positions via the relations $\lessdot$ and $\curvearrowright_s$. By Hanf's theorem [19, 15, 14], $\varphi$ is therefore (on nested words) equivalent to a Boolean combination of statements of the form:

"There are at least $n$ positions whose $r$-sphere is isomorphic to $(\mathcal{S}, \overline{Z}, x)$."

where $n \in \mathbb{N}$, $\mathcal{S}$ is a nw-like structure, $Z_i$ are sets of elements of $\mathcal{S}$, and $x$ is an element of $\mathcal{S}$. By Lemma 3.3 this is expressible as a Boolean combination of existential first-order formulas. □

We are now able to prove Theorem 3.2.

*Proof (of Theorem 3.2).* Since $\varphi \in M\Sigma_n$, it is of the form

$$\exists \overline{Y}_1 \, \forall \overline{Y}_2 \ldots \, \exists / \forall \overline{Y}_n \, \varphi_0(x, \overline{X}, \overline{Y})$$

with $\varphi_0 \in \mathrm{FO} = M\Sigma_0$. Then, by Lemma 3.4, there exists a Boolean combination of $\Sigma_1^M \cap \mathrm{FO}$ formulas $\psi_0$ that is equivalent to $\varphi_0$ for all nested words. The formula $\exists \overline{Y}_1 \, \forall \overline{Y}_2 \ldots \, \exists / \forall \overline{Y}_n \, \psi_0(x, \overline{X}, \overline{Y})$ belongs to $\Sigma_{n+1}^M$ and is obviously equivalent to $\varphi$ for all nested words. □

We are now able to transform any temporal formula of an arbitrary but fixed MSO-definable temporal logic into an equivalent MSO-sentence in polynomial time.

**Proposition 3.5 (cf. [30]).** *Let $TL = (B, ar, [\![-]\!])$ be some fixed $M\Sigma_n$-definable temporal logic. From a temporal formula $F$ of $TL$, one can compute in time poly($|F|$) an MSO-sentence*

$$\psi = \exists \overline{X} \, \left[ \bigwedge_{i \in [|F|]} \left( \psi_{1,i}(\overline{X}) \wedge \forall y \, \psi_{2,i}(y, \overline{X}) \right) \right]$$

*(where $\overline{X}$ is a tuple of set variables) such that, for all $i \in [|F|]$, $\psi_{1,i} \in \Pi_{n+1}^M$, $\psi_{2,i} \in \Sigma_{n+1}^M$, and, for any nested word $\nu$, we have $\nu \models F$ if and only if $\nu \models \psi$. Furthermore, the sizes of the $\psi_{1,i}$'s and $\psi_{2,i}$'s are independent from $|F|$.*

Note that the above proposition differs the corresponding statemen [10, Proposition 1] in the conference version of this paper. There, we constructed formulas $\psi_1$ and $\psi_2$ sizes are linear in $|F|$. In contrast, the sizes of the above formulas $\psi_{1,i}$ and $\psi_{2,i}$ are independent from $|F|$. In Theorem 3.13, this allows us to construct a tree automaton for $F$ in space $|F| \cdot \mathrm{tower}_{n+1}(\mathrm{poly}(\tau))$ instead of $\mathrm{tower}_{n+1}(\mathrm{poly}(|F| + \tau))$ [10, Theorem 3], i.e., the influence of the size of $F$ on the space complexity is vastly reduced by a tower of $n+1$ exponents.

*Proof.* By Theorem 3.2, we can assume $[\![M]\!] \in \Sigma_{n+1}^M$ for all $M \in B$. Let $\{F_1, \ldots, F_m\}$ be the set of subformulas of the temporal formula $F$ with $F = F_1$. Then let $\overline{X} = (X_1, \ldots, X_m)$ and define MSO-formulas $\xi_i(y, \overline{X})$ as follows:

$$\xi_i(y, \overline{X}) = [\![M]\!](y, X_{i_1}, \ldots, X_{i_k}) \text{ if } F_i = M(F_{i_1}, \ldots, F_{i_k})$$

The idea is that $X_i$ is the set of positions in a nested word $\nu$ where $F_i$ holds. Having this idea in mind, the following is obvious: Let $\nu$ be a nested word. Then $\nu \models F$ if and only if $\nu$ satisfies the MSO-formula

$$\exists \overline{X} \, \left[ 1 \in X_1 \wedge \bigwedge_{i \in [m]} \forall y \, \left( y \in X_i \leftrightarrow \xi_i(y, \overline{X}) \right) \right]$$

where $1 \in X_1$ is a shortcut for $\forall x \, (\min(x) \to x \in X_1)$. Note that this formula is logically equivalent to

$$\exists \overline{X} \, \left[ \bigwedge_{i \in [m]} \left( 1 \in X_1 \wedge \forall y \, \left( y \in X_i \vee \neg \xi_i(y, \overline{Y}) \right) \wedge \forall y \, \left( y \notin X_i \vee \xi_i(y, \overline{Y}) \right) \right) \right] .$$

Now, for all $i \in [m]$, we set:

$$\psi_{1,i}(\overline{X}) = 1 \in X_1 \wedge \forall y \ \big(y \in X_i \vee \neg \xi_i(y, \overline{Y})\big)$$
$$\psi_{2,i}(y, \overline{X}) = y \notin X_i \vee \xi_i(y, \overline{Y})$$

Since $\xi_i \in \Sigma^M_{n+1}$ for all $i \in [m]$, this finishes the proof. □

It follows from Prop. 3.5 that, in order to solve the satisfiability problem of an MSO-definable temporal logic, it suffices to decide the satisfiability problem of an MSO-sentence of a certain form.

### 3.2 The Encoding of Nested Words as Trees

We aim at reducing the satisfiability problem of some MSO-sentence expressing properties of nested words to the emptiness problem of tree automata. For this, we introduce the following definitions.

Let $\Lambda$ be an alphabet. A *$\Lambda$-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 $\ell \colon V \to \Lambda$ is the labeling 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 $\Lambda$-trees is denoted by $\mathcal{T}_\Lambda$.

A *(nondeterministic bottom-up) tree automaton over the alphabet $\Lambda$* is a triple $\mathcal{B} = (Q, \Delta, F)$ where $Q$ is the non-empty and finite set of states, $\Delta \subseteq Q \times \Lambda \times (Q \cup \{\bot\})^2$ is the transition relation (where $\bot \notin Q$), and $F \subseteq Q$ is the set of final states. Let $T = (V, E_0, E_1, \ell)$ be a $\Lambda$-tree. The mapping $\rho \colon V \to Q$ is a *run* of $\mathcal{B}$ on $T$ if, for all $v \in V$, there exists a transition $(\rho(v), \ell(v), q_0, q_1) \in \Delta$ such that for all $i \in \{0, 1\}$ the following holds: if there exists $v' \in V$ with $(v, v') \in E_i$, then $q_i = \rho(v')$ and $q_i = \bot$ otherwise. The run $\rho$ is *accepting* if $\rho(u) \in F$ where $u$ is the root of $T$. By $L(\mathcal{B})$, we denote the set of all $\Lambda$-trees for which there exists an accepting run of $\mathcal{B}$.

In order to translate not only MSO-sentences but also arbitrary MSO-formulas into tree automata, we need to extend node labels by additional bits in the usual way: If $T = (V, E_0, E_1, \ell)$ is a $\Lambda$-tree, $X_1, \ldots, X_m$ are sets of nodes of $T$, and $x_1, \ldots, x_n$ are nodes of $T$, then $(T, X_1, \ldots, X_m, x_1, \ldots, x_n)$ denotes the new tree $(V, E_0, E_1, \ell')$ over the alphabet $\Lambda \times \{0, 1\}^{m+n}$ where, for all $v \in V$, we have $\ell'(v) = (\ell(v), b_1, \ldots, b_m, c_1, \ldots, c_n)$ 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 the above 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 tree automaton with $2^n$ states.

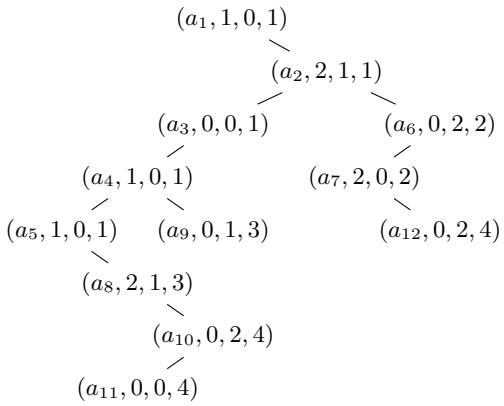


**Fig. 2.** The tree encoding of the nested word from Fig. 1.

Let $\nu$ be a nested word. Following [21], we will now describe its encoding as a tree $\mathrm{tree}(\nu) = (V, E_0, E_1, \ell)$. First, the nodes of the tree are the positions of $\nu$. 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 $\nu$ provided that this is no return position. It remains to describe the labeling 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 from $\Gamma$ at position $j$. 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 $\nu$, i.e., the number $\mathrm{ph}_\nu(j)$. More formally:

If $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ is a nested word, then $\mathrm{tree}(\nu)$ denotes the tree $(P, E_0, E_1, \ell)$ where

$$(i,j) \in E_1 \iff i \curvearrowright_s j \text{ for some } s \in [\sigma], \text{ and}$$
$$(i,j) \in E_0 \iff i \lessdot j \text{ and there do not exist } k \text{ and } s \text{ with } k \curvearrowright_s j\,.$$

and $\ell(j) = \big(\lambda(j), s, s', \mathrm{ph}_\nu(j)\big) \in \Gamma \times [\sigma]_0 \times [\sigma]_0 \times \mathbb{N}$ where

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

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

*Example 3.6.* The encoding of the nested word from Example 2.1 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, Madhusudan, and Parlato [21].

**Theorem 3.7 ([21]).** *From $\tau \in \mathbb{N}$, one can construct in time tower$_2$(poly($\tau$)) a tree automaton $\mathcal{B}_\tau$ with $L(\mathcal{B}_\tau) = tree(NW_\tau(\Gamma))$.*

Note that the proof of the above theorem only considers tree encodings whose labels belong to $\Gamma \times [\tau]$. However, the extension to our encodings tree($\nu$) can be handled in the same spirit.

### 3.3 Tree Automata for Possibly Negated Atomic Formulas

We plan to construct, from an MSO-formula $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$, a small tree automaton $\mathcal{B}_\varphi$ that accepts the tree $(\text{tree}(\nu), x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ if and only if $\nu, x_1, \ldots, x_k, X_1, \ldots, X_\ell \models \varphi$ for any $\tau$-phase nested word $\nu$. The simplest such formulas $\varphi$ are the atomic formulas $\lambda(x) = a$, $x \lessdot y$, $x \curvearrowright_s y$, $\min(x)$ etc. and their negations that we handle in this section.

**Proposition 3.8.** *Given a possibly negated atomic MSO-formula $\varphi$ not of the form $x \lessdot y$ or max(x)and $\tau \in \mathbb{N}$, one can construct in space poly($\tau$) a tree automaton $\mathcal{B}_\varphi$ with the following property: Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a $\tau$-phase nested word, $i, j \in P$, and $I \subseteq P$.[3] Then $(tree(\nu), i, j, I)$ is accepted by $\mathcal{B}_\varphi$ if and only if $\nu, i, j, I \models \varphi$.*

*Proof.* Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word, $i, j \in P$ and $I \subseteq P$. Then $\nu, i, j, I \models \min(x)$ if and only if $i$ is the root of the tree tree($\nu$). From the label of $i$ in $(\text{tree}(\nu), i, j, I)$, one can immediately tell whether $\nu, i, j, I \models (\lambda(x) = a)$ and similarly for the formulas $\text{call}_s(x)$, $\text{ret}_s(x)$, $x \in X$, $x = y$, and for their negations.

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

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 [21] 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 $\neg\max(x)$. □

The real difficulty comes with the remaining atomic formulas $\varphi$ of the form $x \lessdot y$ or $\max(x)$. We could, of course, simply complement the automaton $\mathcal{B}_{\neg\varphi}$ from Proposition 3.8. But this requires exponential space.

In a first step, we will construct a tree automaton accepting $(\text{tree}(\nu), X, x)$ if and only if $X$ is the set of positions of $\nu$ preceding $x$ in $\nu$. To this aim, Lemma 3.9 gives a new characterization of the order relation $\leq$ of $\nu$ in the tree tree($\nu$).

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 phase($v$) denote its phase, i.e., the number $t \in [\tau]$ with $\ell(v) = (a, s, s', t)$. Then the *phase word* $\text{pw}(v) \in [\tau]^+$ is defined by induction:

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

[3] Note that $\varphi$ has at most two free individual and one free set variable.

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 $\nu$ is a $\tau$-phase nested word, then any phase word from the tree $\mathrm{tree}(\nu)$ 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 3.9.** *Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a $\tau$-phase nested word, $x, y \in P$, and $\mathrm{tree}(\nu) = (P, E_0, E_1, \ell)$ be the tree encoding of $\nu$. Then $x < y$ if and only if*

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

$$(z, x') \in E_0 \text{ if and only if } \left(|pw(x)| - |pw(z)| \text{ even} \iff ph_\nu(x') = ph_\nu(y')\right).$$

*Proof.* First of all, note that, for every $x \in P$, we have two numbers that denote its phase: First, seen as a position in the nested word $\nu$, we have $\mathrm{ph}_\nu(x)$. Secondly, seen as a node in $\mathrm{tree}(\nu)$, we have $\mathrm{phase}(x)$. By the very definition of $\mathrm{tree}(\nu)$, we get $\mathrm{ph}_\nu(x) = \mathrm{phase}(x)$. If $x, y \in P$, then we write $x <_{\mathrm{pre}} y$ if and only if $x$ precedes $y$ in the preorder traversal of $\mathrm{tree}(\nu)$.

The lemma is shown by induction on $\max(\mathrm{ph}_\nu(x), \mathrm{ph}_\nu(y))$. First of all, let $\mathrm{ph}_\nu(x), \mathrm{ph}_\nu(y) \leq 1$, i.e., $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y) = 1$ and therefore $\mathrm{pw}(x) = \mathrm{pw}(y)$. Then, by [21, Lemma 2], $x < y$ if and only if $x <_{\mathrm{pre}} y$. But this is equivalent with

- $x$ is a predecessor of $y$, i.e., (2), or
- there exist $z, x', y'$ with $(z, x') \in E_0$, $(z, y') \in E_1$, $x'$ is a predecessor of $x$ and $y'$ is one of $y$. Note that $\mathrm{pw}(z) = 1 = \mathrm{pw}(x)$ and $\mathrm{ph}_\nu(x') = 1 = \mathrm{ph}_\nu(y')$, i.e., (3).

This proves the lemma in case $\mathrm{ph}_\nu(x), \mathrm{ph}_\nu(y) \leq 1$. Now, let $\mathrm{ph}_\nu(x), \mathrm{ph}_\nu(y) \leq t$ for some $t > 1$. Then, by [21, Lemma 2], we have $x < y$ if and only if

(a) $\mathrm{ph}_\nu(x) < \mathrm{ph}_\nu(y)$ or
(b) $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y)$ and there is $z \in P$ with $\mathrm{ph}_\nu(z) = \mathrm{ph}_\nu(x)$ that is a predecessor of both, $x$ and $y$, and $x <_{\mathrm{pre}} y$, or
(c) $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y)$ and there are positions $x_1, x_2, y_1, y_2 \in P$ with $x_2 \neq y_2$, $(x_1, x_2), (y_1, y_2) \in E_0 \cup E_1$, $x_2$ is a predecessor of $x$ and $y_2$ is one of $y$, $\mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2) = \mathrm{ph}_\nu(x)$, $\mathrm{ph}_\nu(y_1) < \mathrm{ph}_\nu(y_2) = \mathrm{ph}_\nu(y)$, and $x_1 > y_1$.

By the definition of $<_{\mathrm{pre}}$ and the induction hypothesis, we obtain that $x < y$ if and only if

(a) $\mathrm{ph}_\nu(x) < \mathrm{ph}_\nu(y)$ or

(b) $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y)$ and

(b.1) $x$ is a predecessor of $y$ or

(b.2) there are $x', y', z \in P$ with $\mathrm{ph}_\nu(z) = \mathrm{ph}_\nu(x)$, $(z, x') \in E_0$, $(z, y') \in E_1$, $x'$ is a predecessor of $x$, and $y'$ is a predecessor of $y$ or

(c) $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y)$ and there are positions $x_1, x_2, y_1, y_2 \in P$ with $x_2 \neq y_2$, $(x_1, x_2), (y_1, y_2) \in E_0 \cup E_1$, $x_2$ is a predecessor of $x$ and $y_2$ is one of $y$, $\mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2) = \mathrm{ph}_\nu(x)$, $\mathrm{ph}_\nu(y_1) < \mathrm{ph}_\nu(y_2) = \mathrm{ph}_\nu(y)$ such that

(c.1) $\mathrm{pw}(x_1) \sqsupset \mathrm{pw}(y_1)$ or

(c.2) $\mathrm{pw}(x_1) = \mathrm{pw}(y_1)$ and $y_1$ is a predecessor of $x_1$ or

(c.3) $\mathrm{pw}(x_1) = \mathrm{pw}(y_1)$ and there are positions $x', y', z \in P$ with $x' \neq y'$, $(z, x'), (z, y') \in E_0 \cup E_1$, $x'$ is a predecessor of $x_1$, $y'$ is a predecessor of $y_1$, and

$$(z, y') \in E_0 \text{ iff } \big(|\mathrm{pw}(y_1)| - |\mathrm{pw}(z)| \text{ even } \iff \mathrm{ph}_\nu(y') = \mathrm{ph}_\nu(x')\big)\,.$$

Note that the disjunction of (a) and (c.1) is equivalent to $\mathrm{pw}(x) \sqsubset \mathrm{pw}(y)$ since, in (c.1), we have $\mathrm{pw}(x) = \mathrm{pw}(x_1)\,\mathrm{ph}_\nu(x)$ and $\mathrm{pw}(y) = \mathrm{pw}(y_1)\,\mathrm{ph}_\nu(y)$, i.e., it is equivalent to (1). Furthermore, (b.1) holds if and only if $\mathrm{pw}(x) = \mathrm{pw}(y)$ and $x$ is a predecessor of $y$, i.e., it is equivalent to (2). It therefore remains to be shown that the disjunction of (b.2), (c.2), and (c.3) is equivalent to (3). We start with the implication "$\Rightarrow$" that naturally splits into three cases:

- Suppose (b.2) holds. Clearly, $\mathrm{pw}(z)$ is a prefix of $\mathrm{pw}(x)$ and ends with the same number. Since all phase words are increasing, this implies $\mathrm{pw}(z) = \mathrm{pw}(x)$. For analogous reasons, the phase words of $y$, $x'$, and $y'$ coincide with $\mathrm{pw}(z)$. Hence, we have $(z, x') \in E_0$, $|\mathrm{pw}(x)| - |\mathrm{pw}(z)| = 0$, and $\mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')$, i.e., we showed (3).
- Suppose (c.2) holds. From (c), we get $\mathrm{pw}(x) = \mathrm{pw}(x_1)\,\mathrm{ph}_\nu(x)$ and $\mathrm{pw}(y) = \mathrm{pw}(y_1)\,\mathrm{ph}_\nu(y)$. From (c.2), we know $\mathrm{pw}(x_1) = \mathrm{pw}(y_1)$ which, together with $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y)$ implies $\mathrm{pw}(x) = \mathrm{pw}(y)$. Recall that $\mathrm{tree}(\nu)$ was based on the greedy subdivision of the nested word $\nu$ into phases. This implies in particular that $y_1 \curvearrowright_s y_2$ for some $s \in [\sigma]$ and therefore $(y_1, y_2) \in E_1$. Similarly, $(x_1, x_2) \in E_1$ which, together with $x_2 \neq y_2$ implies $x_1 \neq y_1$. Let $x'$ be the first node on the path from $y_1$ to $x_1$, i.e., $(y_1, x') \in E_0 \cup E_1$ and $x'$ is a predecessor of $x_1$ (and therefore of $x_2$ and of $x$). If $(y_1, x') \in E_1$, then $x' = y_2$, i.e., $y_2$ is a predecessor of $x_1$. But this contradicts $\mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2) = \mathrm{ph}_\nu(y_2)$. Hence $(y_1, x') \in E_0$. Finally,

  $$|\mathrm{pw}(x)| - |\mathrm{pw}(y_1)| = |\mathrm{pw}(x)| - |\mathrm{pw}(x_1)| = 1$$

  is odd and $\mathrm{ph}_\nu(x') \leq \mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2) = \mathrm{ph}_\nu(y_2)$. Hence, also in this case, we have (3) with $z = y_1$ and $y' = y_2$.

– Finally, suppose (c.3). Then, as above, we have $\mathrm{pw}(x) = \mathrm{pw}(x_1)\,\mathrm{ph}_\nu(x)$. Hence we get

$$\begin{aligned}
&(z, x') \in E_0 \\
\text{iff } &(z, y') \notin E_0 \text{ since } x' \neq y' \\
\text{iff } &\big(|\mathrm{pw}(y_1)| - |\mathrm{pw}(z)| \text{ odd} \iff \mathrm{ph}_\nu(y') = \mathrm{ph}_\nu(x')\big) \\
&\qquad\qquad \text{by (c.3)} \\
\text{iff } &\big(|\mathrm{pw}(x_1)| - |\mathrm{pw}(z)| \text{ odd} \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')\big) \\
&\qquad\qquad \text{since } \mathrm{pw}(x_1) = \mathrm{pw}(y_1) \\
\text{iff } &\big(|\mathrm{pw}(x)| - |\mathrm{pw}(z)| \text{ even} \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')\big) \\
&\qquad\qquad \text{since } |\mathrm{pw}(x)| = |\mathrm{pw}(x_1)| + 1\,.
\end{aligned}$$

Hence, (3) follows from (c.3), too.

Finally, we have to show the implication "⇐", i.e., we have to show that (3) implies the disjunction of (b.2), (c.2), and (c.3). So assume (3).

– We first consider the case $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(z)$. Then we have $\mathrm{ph}_\nu(x) \geq \mathrm{ph}_\nu(x') \geq \mathrm{ph}_\nu(z) = \mathrm{ph}_\nu(x)$ as well as $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(y) \geq \mathrm{ph}_\nu(y') \geq \mathrm{ph}_\nu(z) = \mathrm{ph}_\nu(x)$ implying in particular $\mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')$. Since $\mathrm{ph}_\nu(x) = \mathrm{ph}_\nu(z)$, we get $\mathrm{pw}(x) = \mathrm{pw}(z)$ and therefore $|\mathrm{pw}(x)| - |\mathrm{pw}(z)| = 0$. From (3), we can now infer $(z, x') \in E_0$ which, together with $x' \neq y'$, implies $(z, y') \in E_1$. Hence we showed (b.2).
– Next suppose $\mathrm{ph}_\nu(x) > \mathrm{ph}_\nu(z)$. Then there are positions $x_1, x_2, y_1, y_2 \in P$ such that $(x_1, x_2), (y_1, y_2) \in E_0 \cup E_1$ are edges on the paths from $z$ to $x$ and $y$, resp., $\mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2) = \mathrm{ph}_\nu(x)$, and $\mathrm{ph}_\nu(y_1) < \mathrm{ph}_\nu(y_2) = \mathrm{ph}_\nu(y)$. Then

$$\mathrm{pw}(x_1)\,\mathrm{ph}_\nu(x) = \mathrm{pw}(x) = \mathrm{pw}(y) = \mathrm{pw}(y_1)\,\mathrm{ph}_\nu(y)$$

implies $\mathrm{pw}(x_1) = \mathrm{pw}(y_1)$. We consider three cases depending on the relation between $x_1$ and $y_1$:

- If $y_1$ is a predecessor of $x_1$, then we have (c.2).
- If $x_1$ is a proper predecessor of $y_1$, then $z = x_1$ and therefore $x_2 = x' \neq y'$. From $\mathrm{ph}_\nu(x_1) < \mathrm{ph}_\nu(x_2)$, we get $(z, x') = (x_1, x_2) \in E_1$ and therefore $(z, y') \in E_0$. This implies $\mathrm{ph}_\nu(z) = \mathrm{ph}_\nu(y')$. Because of (3), $(z, x') \in E_1$, and $\mathrm{ph}_\nu(x') \neq \mathrm{ph}_\nu(y')$, the difference $|\mathrm{pw}(y)| - |\mathrm{pw}(z)| = |\mathrm{pw}(x)| - |\mathrm{pw}(z)|$ is even. Consequently, the difference $|\mathrm{pw}(y_1)| - |\mathrm{pw}(z)|$ is odd.

  In summary, we showed $(z, y') \in E_0$, $|\mathrm{pw}(y_1)| - |\mathrm{pw}(z)|$ is odd, and $\mathrm{ph}_\nu(x') \neq \mathrm{ph}_\nu(y')$, i.e., (c.3) holds.

- If $x_1$ and $y_1$ are not predecessors of each other, then $x'$ is a predecessor of $x_1$ and $y'$ one of $y_1$. Furthermore, we have

$$\begin{aligned}
&(z, y') \in E_0 \\
\text{iff } &(z, x') \notin E_0 \text{ since } x' \neq y' \\
\text{iff } &(|\mathrm{pw}(x)| - |\mathrm{pw}(z)| \text{ odd } \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')) \\
&\qquad\qquad \text{by (3)} \\
\text{iff } &(|\mathrm{pw}(y)| - |\mathrm{pw}(z)| \text{ odd } \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')) \\
&\qquad\qquad \text{since } \mathrm{pw}(y) = \mathrm{pw}(x) \\
\text{iff } &(|\mathrm{pw}(y_1)| - |\mathrm{pw}(z)| \text{ even } \iff \mathrm{ph}_\nu(y') = \mathrm{ph}_\nu(x')) \\
&\qquad\qquad \text{since } |\mathrm{pw}(y)| = \mathrm{pw}(y_1)| + 1\,.
\end{aligned}$$

  Hence, also in this case, (c.3) holds.

This concludes the proof of Lemma 3.9. □

Based on Lemma 3.9, one can prove the following.

**Lemma 3.10.** *From $\tau \in \mathbb{N}$, one can construct in space poly($\tau$) a tree automaton $\mathcal{B}$ satisfying the following property: Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a $\tau$-phase nested word, $x \in P$, and $X \subseteq P$. Then $(\mathrm{tree}(\nu), x, X)$ is accepted by $\mathcal{B}$ if and only if $X = \{y \in P \mid y \leq x\}$.*

*Proof.* The tree automaton $\mathcal{B}$ is based on the following observation: For a $\tau$-phase nested word $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ and $x \in P$, let $H(\nu, x) \subseteq P$ denote the set of all positions $y \in P$ incomparable with $x$ such that

$$(z, x') \in E_0 \quad \text{iff} \quad \big(|\mathrm{pw}(x)| - |\mathrm{pw}(z)| \text{ even } \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y')\big)$$

where $z$ is the largest common prefix of $x$ and $y$ in $\mathrm{tree}(\nu)$ and $x'$ and $y'$ are the direct successors of $z$ on the path to $x$ and $y$, resp. Then $y \in H(\nu, x)$ if and only if $y$ is incomparable with $x$ and

- the direct predecessor of $y$ belongs to $H(\nu, x)$ or
- the direct predecessor $z$ of $y$ is a predecessor of $x$ and

$$y \text{ is a return} \quad \text{iff} \quad \big(|\mathrm{pw}(x)| - |\mathrm{pw}(z)| \text{ is even } \iff \mathrm{ph}_\nu(x') = \mathrm{ph}_\nu(y)\big)$$

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

We first construct an intermediate automaton $\mathcal{B}_1$ whose set of states equals

$$\mathrm{PW}(\tau)^2 \times [\tau]_0^2 \times \{0,1\}^4\,.$$

The behavior of $\mathcal{B}_1$ can be described as follows: For all $\tau$-phase nested words $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$, $x \in P$, and $H \subseteq P$, the tree $(\mathrm{tree}(\nu), x, H)$ is accepted by $\mathcal{B}_1$. Furthermore, every accepting run $\rho$ of $\mathcal{B}_1$ on $(\mathrm{tree}(\nu), x, H)$ satisfies the following conditions (for all $z \in P$ where $\rho(z) = (u, v, t_0, t_1, b, c, d_0, d_1)$):

– $u = \mathrm{pw}(x)$ and $v = \mathrm{pw}(z)$
– if there is $y \in P$ with $(z, y) \in E_0$, then $t_0 = \mathrm{ph}_\nu(y)$ and $d_0 = 1 \iff y \in H$
– if there is $y \in P$ with $(z, y) \in E_1$, then $t_1 = \mathrm{ph}_\nu(y)$ and $d_1 = 1 \iff y \in H$
– $b = 1$ if and only if $z$ is a predecessor of $x$ in $\mathrm{tree}(\nu)$
– $c = 1$ if and only if $z$ is a successor of $x$ in $\mathrm{tree}(\nu)$

By restricting the set of transitions of $\mathcal{B}_1$, we get a tree automaton $\mathcal{B}_2$ that only accepts $(\mathrm{tree}(\nu), x, H)$ if $H = H(\nu, x)$. Now $\mathcal{B}$ is obtained from $\mathcal{B}_2$ as follows (recall that $\mathcal{B}$ runs on trees of the form $(\mathrm{tree}(\nu), x, X)$):

– It guesses a set $H \subseteq P$ and verifies that $(\mathrm{tree}(\nu), x, H)$ is accepted by $\mathcal{B}_2$, i.e., that $H = H(\nu, x)$.
– It verifies that $X$ contains precisely those positions $y$ that satisfy
  (1) $\mathrm{pw}(x) \sqsubset \mathrm{pw}(y)$ or
  (2) $\mathrm{pw}(x) = \mathrm{pw}(y)$ and $x$ is a predecessor of $y$ or
  (3) $\mathrm{pw}(x) = \mathrm{pw}(y)$ and $y \in H$ .

By Lemma 3.9, $\mathcal{B}$ accepts the correct trees $(\mathrm{tree}(\nu), x, X)$. Since the intermediate tree automata $\mathcal{B}_1$ and $\mathcal{B}_2$ need not be constructed explicitly, we have that $\mathcal{B}$ can be constructed in space $\mathrm{poly}(\tau)$. □

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

**Proposition 3.11.** *Given a formula $\varphi$ of the form $x \lessdot y$ or $max(x)$ and $\tau \in \mathbb{N}$, one can construct in space $poly(\tau)$ a tree automaton $\mathcal{B}_\varphi$ satisfying the following property: Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a $\tau$-phase nested word and $i, j \in P$. Then $(tree(\nu), i, j)$ is accepted by $\mathcal{B}_\varphi$ iff $\nu, i, j \models \varphi$.*

*Proof.* Running on $(\mathrm{tree}(\nu), i, j)$, 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 $(\mathrm{tree}(\nu), i)$, the tree automaton $\mathcal{B}_{\max(x)}$ guesses a set $X \subseteq P$ and verifies $X = \{x \in P \mid x \leq i\}$ and $X = P$. By Lemma 3.10, these automata can be constructed in space $\mathrm{poly}(\tau)$ and accept the correct encodings of $\tau$-phase nested words. □

### 3.4 The Decision Procedure

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

**Lemma 3.12.** *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 $poly(|Q| + |\Lambda|)$.*

Note that, using standard constructions, one would produce $\mathcal{B}'$ from $\mathcal{B}$ by complementation, projection, and complementation, again. This construction can be carried out in space in the size of $\mathcal{B}$. Thus, the above lemma yields in an exponentially more efficient construction compared to the standard one.

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

- $M_1$ is the set of states that $\mathcal{B}$ can reach on $(T, \emptyset)$, and,
- for all positions $v$ of $T$, there is a state contained in $M_2$ which can be reached by $\mathcal{B}$ on $(T, \{v\})$.

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

More formally, let $\mathcal{B} = (Q, \Delta, F)$. For a tree $T$ and a position $v$ of $T$, let $T_v$ denote the subtree of $T$ rooted at $v$. The states of the tree automaton $\mathcal{B}'$ are pairs of sets of states of $\mathcal{B}$, i.e., $Q' = 2^Q \times 2^Q$ is the set of states of $\mathcal{B}'$. A state $(M_1, M_2) \in Q'$ is accepting if and only if $M_2 \subseteq F$. The transition relation $\Delta$ is the least relation fulfilling the following properties:

- There is a transition $\big((\{\iota\}, \{\iota\}), a, \bot, \bot\big) \in \Delta$ for every $a \in \Lambda$ which is meant to be executed at the $a$-labeled leaves of a tree.
- We have $((K_1, K_2), a, (L_1, L_2), (M_1, M_2)) \in \Delta'$ (for $K_1, K_2, L_1, L_2, M_1, M_2 \subseteq Q$ and $a \in \Lambda$) if and only if
  - $K_1 = \{p \in Q \mid \text{there exist } q \in L_1, r \in M_1 \text{ such that } (p, (a, 0), q, r) \in \Delta\}$,
  - for all $q \in L_2$ there exist $r \in M_1, p \in K_2$ such that $(p, (a, 0), q, r) \in \Delta$,
  - for all $q \in M_2$ there exist $r \in L_1, p \in K_2$ such that $(p, (a, 0), q, r) \in \Delta$, and
  - there exist $q \in L_1, r \in M_1, p \in K_2$ such that $(p, (a, 1), q, r) \in \Delta$.
- We have $((K_1, K_2), a, (L_1, L_2), \bot) \in \Delta'$ (for $K_1, K_2, L_1, L_2 \subseteq Q$ and $a \in \Lambda$) if and only if
  - $K_1 = \{p \in Q \mid \text{there exist } q \in L_1 \text{ such that } (p, (a, 0), q, \bot) \in \Delta\}$,
  - for all $q \in L_2$ there exist $p \in K_2$ such that $(p, (a, 0), q, \bot) \in \Delta$, and
  - there exist $q \in L_1$ and $p \in K_2$ such that $(p, (a, 1), q, \bot) \in \Delta$.
- Similarly for transitions of the form $((K_1, K_2), a, \bot, (M_1, M_2)) \in \Delta'$.

This finishes the definition of $\mathcal{B}'$ (note that $\mathcal{B}'$ can indeed be computed in space $\mathrm{poly}(|Q| + |\Lambda|)$. By induction on the number of nodes of the tree $|T_v|$, one can show the following: Let $T = (V, E_0, E_1, \ell) \in \mathcal{T}_\Lambda$ and $\rho' \colon V \to Q'$. Then $\rho'$ is a run of $\mathcal{B}'$ on $T$ if and only if, for all nodes $v \in V$ with $\rho'(v) = (M_1, M_2)$, the following holds:

(i) $M_1 = \{p \in Q \mid (T_v, \emptyset) \in L(Q, \Delta, \{p\})\}$
(ii) For all $w \in T_v$, there is a state $p \in M_2$ such that $(T_v, \{w\}) \in L(Q, \Delta, \{p\})$.

Now the claim about the tree language $L(\mathcal{B}')$ follows immediately from (ii) and the definition of the set of accepting states $F' = 2^Q \times 2^F$. □

The main difficulties of the translation of a temporal formula expressing properties of nested words into a tree automaton have been mastered in the above Propositions 3.5, 3.8, and 3.11. It remains to assemble these ingredients using quite standard arguments:

**Theorem 3.13.** *Let TL be a fixed $M\Sigma_n$-definable temporal logic. From a formula $F$ from TL and $\tau \in \mathbb{N}$, one can construct in space $|F| \cdot tower_{n+1}(poly(\tau))$ a tree automaton $\mathcal{B}_F$ over the alphabet $\Gamma$ with the following property: For any $\tau$-phase nested word $\nu$, we have*

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

*Proof.* By Prop. 3.5, we can construct an MSO-formula

$$\psi = \exists \overline{X} \left[ \bigwedge\nolimits_{i \in [|F|]} \left( \psi_{1,i}(\overline{X}) \wedge \forall y\, \psi_{2,i}(y, \overline{X}) \right) \right]$$

that is (over nested words) equivalent to $F$. Recall that $\psi_{1,i} \in \Pi^M_{n+1}$ and $\psi_{2,i} \in \Sigma^M_{n+1}$ for all $i \in [|F|]$. In particular, if $i \in [|F|]$, then $\psi_{2,i}$ is of the form $\exists \overline{X}_1\, \neg \exists \overline{X}_2 \ldots \neg \exists \overline{X}_{n+1}\, \varphi$ where $\varphi$ is quantifier-free and $\overline{X}_i$ are tuples of individual and set variables. Even more, we can assume that $\varphi$ is a positive Boolean combination of possibly negated atomic formulas. Also recall that the size of $\psi_{2,i}$ is independent from the size of $|F|$ and that TL is fixed. Therefore, the size of $\psi_{2,i}$ is a constant. Using Propositions 3.8 and 3.11 and standard constructions (for union and intersection) from automata theory, we can transform $\varphi$ into a tree automaton $\mathcal{B}_\varphi$ in space $\mathrm{poly}(\tau)$. The desired tree automaton $\mathcal{B}_{\psi_{2,i}}$ is obtained from $\mathcal{B}_\varphi$ by a sequence of $n$ complementations and $n+1$ projections. Hence this construction can be carried out in space $\mathrm{tower}_n(\mathrm{poly}(\tau))$. Then, by Lemma 3.12, we can also translate $\forall y\, \psi_{2,i}(y, \overline{X})$ into a tree automaton in space $\mathrm{tower}_{n+1}(\mathrm{poly}(\tau))$. Since a formula belongs to $\Pi^M_{n+1}$ if and only if its negation belongs to $\Sigma^M_{n+1}$, we similarly obtain a tree automaton $\mathcal{B}_{\psi_{1,i}}$ for $\psi_{1,i}$ in space $\mathrm{tower}_{n+1}(\mathrm{poly}(\tau))$. Finally, by standard automata techniques for intersection and projection, we obtain the tree automaton for $\psi$ in space $|F| \cdot \mathrm{tower}_{n+1}(\mathrm{poly}(\tau))$. □

Now, our main theorem states that the satisfiability problem of some fixed $M\Sigma_n$-definable temporal logic can be solved in time exponential in the size of the temporal formula and $(n+2)$-fold exponential in the phase bound $\tau$ (which is a vast improvement over the conference version of this paper [10] where the space was also $(n+2)$-fold exponential in the size of the temporal formula)

**Theorem 3.14.** *Let $n \geq 0$ and TL be some fixed $Bool M\Sigma_n(\Gamma)$-definable temporal logic. The satisfiability problem of TL is solvable in time*

$$tower_1(|F|^2 \cdot tower_{n+1}(poly(\tau)))$$

*(where $\tau$ is encoded in unary).*

*Proof.* By Remark 2.7, we can assume that TL is $M\Sigma_n$-definable. Let $\tau \in \mathbb{N}$ and $F$ be a temporal formula from TL. First construct the tree automaton $\mathcal{B}_F$ from Theorem 3.13. Furthermore, let $\mathcal{B}_\tau$ be the tree automaton from Theorem 3.7. The number of states of both these automata is bounded by $\mathrm{tower}_1(|F| \cdot \mathrm{tower}_{n+1}(\mathrm{poly}(\tau)))$.

Note that we have $L(\mathcal{B}_F) \cap L(\mathcal{B}_\tau) = \emptyset$ if and only if $F$ is not satisfiable by any $\tau$-phase nested word. This can be decided in time polynomial in $|\mathcal{B}_F| \cdot |\mathcal{B}_\tau|$, i.e., in time $\mathrm{tower}_1(|F|^2 \cdot \mathrm{tower}_{n+1}(\mathrm{poly}(\tau)))$. □

As an immediate consequence, we get the main result from [8].

**Corollary 3.15 (cf. [8]).** *Let $n \geq 0$, TL be some $\mathrm{Bool}M\Sigma_n(\Gamma)$-definable temporal logic, and $\tau \geq 1$. The satisfiability problem of TL for $\tau$-phase-bounded nested words is solvable in exponential time.*

## 4 Lower Bound

In this section, we show that there exists an $M\Sigma_n$-definable temporal logic 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 for the satisfiability problem of temporal logics over labeled grids. Secondly, we polynomially reduce this satisfiability problem to the satisfiability problem for $M\Sigma_n$-definable temporal logics over nested words.

### 4.1 The Lower Bound for Labeled Grids

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

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

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

$$\varphi ::= (\mu(x) = a) \mid x\ S_h\ y \mid x\ S_v\ y \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

The semantics of the above formulas are as expected. In particular, for a grid $G$ and cells $(r_1, c_1), (r_2, c_2) \in \mathrm{dom}(G)$ we have $G, (r_1, c_1), (r_2, c_2) \models x\ S_h\ y$ if and only if $(r_1, c_1)\ S_h\ (r_2, c_2)$. The sets $\mathrm{FO}^G(\Gamma)$, $M\Sigma_n^G(\Gamma)$, and $M\Pi_n^G(\Gamma)$ are defined like their corresponding fragments of the logic $\mathrm{MSO}(\Gamma)$.

*Example 4.1.* Consider the two $\mathrm{MSO}^G(\Gamma)$-formulas $\mathrm{lc}(x) = \neg\exists y\,(y\ S_h\ x)$ and $\mathrm{rc}(x) = \neg\exists y\,(x\ S_h\ y)$. If $G = (k, m, \mu)$ is a grid and $(r, c) \in \mathrm{dom}(G)$, then

$G, (r, c) \models \mathrm{lc}$ if and only if $(r, c)$ is contained in the leftmost column of $G$. Analogously, $\mathrm{rc}(x)$ expresses that $x$ is a cell in the rightmost column. Now, consider the following two $\mathrm{MSO}^G(\Gamma)$-formulas:

$$\mathrm{ul}(x) = \mathrm{lc}(x) \wedge \neg\exists y\,(y\ S_v\ x)$$
$$\mathrm{lr}(x) = \mathrm{rc}(x) \wedge \neg\exists y\,(x\ S_v\ y)$$

We have $G, (r, c) \models \mathrm{ul}$ if and only if $(r, c) = (1, 1)$. Analogously, $\mathrm{lr}(x)$ expresses that $x$ is the cell $(k, m)$ in the lower right of $G$.

An *$\mathrm{MSO}^G(\Gamma)$-definable temporal logic* expressing properties of labeled grids is defined analogous to an $\mathrm{MSO}(\Gamma)$-definable temporal logic talking about nested words. If $\mathrm{TL}^G$ is an $\mathrm{MSO}^G(\Gamma)$-definable temporal logic and $F \in \mathrm{TL}^G$, then we write $G \models F$ if $G, (1, 1) \models F$.

*Example 4.2.* Consider the modality $\mathsf{S_h}$ defined by $[\![\mathsf{S_h}]\!](X_1, x) = \exists y\,(x\ S_h\ y \wedge y \in X_1)$. Intuitively, $\mathsf{S_h}\,F$ holds at a cell if $F$ holds at its right neighbor.

If $\mathrm{TL}^G$ is some $\mathrm{MSO}^G(\Gamma)$-definable temporal logic, then the *satisfiability problem* of $\mathrm{TL}^G$ 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 $G \models F$.

Our intermediate goal is to prove the following lower bound on the satisfiability problem of $\mathrm{MSO}^G(\Gamma)$-definable temporal logics.

**Theorem 4.3.** *Let $n \geq 1$. There exists an $\mathit{MII}_n^G(\Gamma)$-definable temporal logic $\mathit{TL}^G$ with an $n$-EXPSPACE-hard satisfiability problem.*

Our proof adapts the techniques from [28, 33, 17] (the actual proof can be found at the end of this section on page 31).

For all $\ell, m \in \mathbb{N}$, the function $F_\ell \colon \mathbb{N} \to \mathbb{N}$ is defined by $F_0(m) = m$ and $F_{\ell+1}(m) = F_\ell(m) \cdot 2^{F_\ell(m)}$. Clearly, we have $\mathrm{tower}(\ell, m) \leq F_\ell(m)$ for all $\ell \geq 0$ and $m \geq 1$. Hence, there exists a Turing machine $M$ fulfilling the following conditions:

- $M$ runs in space $F_n(m) - 3$ on an input of length $m - 2$.
- $M$ accepts some $n$-EXPSPACE-hard problem.
- No state of $M$ is initial and accepting at the same time.
- All accepting states of $M$ are halting.

Let $\Gamma_T$ be the tape alphabet (which includes the blank symbol $\square$ and the end-of-tape markers $\triangleright$ and $\triangleleft$) and $Q$ the set of states of $M$. Furthermore, let $\Gamma = \Gamma_T \uplus Q$. An *$m$-configuration of $M$* is a word $\triangleright\alpha q\beta\triangleleft \in \Gamma^*$ of length $F_n(m)$ where $\alpha\beta \in (\Gamma_T \setminus \{\triangleright, \triangleleft\})^*$ is the tape content, $q \in Q$ is the current state, and the head is on the first letter of $\beta\triangleleft$[4]. It is *initial* if $q$ is an initial state and $\alpha = \varepsilon$. It is *accepting* if $q$ is a final state. An *$m$-computation* of $M$ of length $\ell \geq 1$ is a sequence $\zeta = \zeta_1\zeta_2 \ldots \zeta_\ell$ where $\zeta_i$ are $m$-configurations of $M$ such that $\zeta_{i+1}$

[4] Note that an $m$-configuration uses space $|\alpha\beta| = F_n(m) - 3$, whence the name $m$-configuration.

can be reached from $\zeta_i$ in one step for all $1 \leq i < \ell$. We say that $\zeta$ is initial (accepting) if $\zeta_1$ ($\zeta_\ell$) is initial (accepting).

Now let $G = (k, m, \mu)$ be a grid over $\Gamma$. The *concatenation of the rows of* $G$ is denoted by

$$\mathrm{seq}(G) = \mu(1,1)\mu(1,2)\dots\mu(1,m)\mu(2,1)\dots\mu(2,m)\mu(3,1)\dots\mu(k,m) \in \Gamma^{k\cdot m}\,.$$

Let $v$ be some input word of length $m-2$. Suppose $v$ is accepted by $M$. Since $M$ uses space $F_n(m)-3$, there is an initial and accepting $m$-computation $\zeta$ with input $v$. Then $|\zeta|$ (the length of the word $\zeta$ and *not* the number of configurations in $\zeta$) is a multiple of $F_n(m)$ and therefore of $m$. Consequently, there exists a grid $G$ with $m$ columns and $\mathrm{seq}(G) = \zeta$, i.e.,

$$\begin{minipage}{0.85\textwidth}$G$ has $m$ columns and $\mathrm{seq}(G)$ is an initial and accepting $m$-computation of $M$ with input $v$.\end{minipage} \qquad (*)$$

Consequently, a word $v$ of length $m-2$ is accepted by $M$ if and only if there is a grid $G$ satisfying (*). Since we want to reduce the language of $M$ to the satisfiability problem of the (not yet defined) temporal logic $\mathrm{TL}^G$, the logic has to be able express this property.

For a grid $G = (k, m, \mu)$, the function $p\colon \mathrm{dom}(G) \to [k \cdot m]$ maps every cell $(r, c) \in \mathrm{dom}(G)$ to the corresponding position within $\mathrm{seq}(G)$, i.e., $p(r, c) = (r-1)\cdot m + c$.

The first lemma states that there exists a simple formula $\mathrm{interval}(x, y, Z)$ which ensures that $Z$ contains exactly those cells which make up the interval between the cells $x$ and $y$ in the sequence $\mathrm{seq}(G)$ (the existence of an $\mathrm{MSO}^G(\Gamma)$-formula expressing this fact is rather straightforward, the crucial point is to get a formula from $M\Sigma_1^G(\Gamma)$).

**Lemma 4.4.** *There exists a formula* $\mathit{interval}(x, y, Z) \in M\Sigma_1^G(\Gamma)$ *such that, for all* $G = (k, m, \mu)$*, cells* $(r_1, c_1), (r_2, c_2) \in \mathit{dom}(G)$*, and* $I \subseteq \mathit{dom}(G)$*, we have*

$$G, (r_1, c_1), (r_2, c_2), I \models \mathit{interval} \iff \begin{pmatrix} p(r_1, c_1) \leq p(r_2, c_2) \text{ and} \\ I = \{(r, c) \in \mathit{dom}(G) \mid p(r_1, c_1) \leq p(r, c) \leq p(r_2, c_2)\} \end{pmatrix}$$

*Proof.* Let $\mathrm{after}(x, X)$ denote the following formula:

$$x \in X \wedge \forall y\,(y\ S_h\ x \to y \notin X) \wedge \forall y\,(y\ S_v\ x \to y \notin X) \tag{3a}$$

$$\wedge\ \forall y\,[y \in X \leftrightarrow (x = y \vee \exists z \in X\,(z\ S_h\ y \vee z\ S_v\ y \vee (y\ S_h\ z \wedge z \neq x)))] \tag{3b}$$

Let $G$ be a grid, $(r, c) \in \mathrm{dom}(G)$ and $A \subseteq \mathrm{dom}(G)$. It can be shown that $G, (r, c), A \models \mathrm{after}$ if and only if $A = \{(r', c') \in \mathrm{dom}(G) \mid p(r, c) \leq p(r', c')\}$. Suppose that $G, (r, c), A \models \mathrm{after}$. The first line expresses that the cell $(r, c)$ is in $A$ and that its left neighbor $(r, c-1)$ (if existent) and upper neighbor $(r-1, c)$

(if existent) are not contained in $A$. From $(r,c) \in A$ and the implication $\rightarrow$ in (3b), we obtain that all cells of the form $(r, c+j)$, $(r+i, c+j)$, and finally $(r+i+1, c-j)$ belong to $A$ for all $i, j \in \mathbb{N}$. In other words, $\{(r',c') \in \mathrm{dom}(G) \mid p(r,c) \leq p(r',c')\} \subseteq A$. Now assume $(r',c') \in A$ with $p(r',c') < p(r,c)$. Then the implication $\leftarrow$ from (3b) allows to show that $(r-1,c) \in A$ (if $r' < r$) or $(r,c-1) \in A$ (if $c' < c$), contradicting (3a). Hence, we have $A = \{(r',c') \in \mathrm{dom}(G) \mid p(r,c) \leq p(r',c')\}$. Conversely, if $A = \{(r',c') \in \mathrm{dom}(G) \mid p(r,c) \leq p(r',c')\}$, then it can be easily checked that $G, (r,c), A \models \mathrm{after}$.

Analogously, we can define a formula $\mathrm{before}(x,X)$ such that $G, (r,c), B \models \mathrm{before}$ if and only if we have $B = \{(r',c') \in \mathrm{dom}(G) \mid p(r',c') \leq p(r,c)\}$ for all grids $G$, $(r,c) \in \mathrm{dom}(G)$, and $B \subseteq \mathrm{dom}(G)$. Now, let $\mathrm{interval}(x,y,Z)$ be the following formula:

$$\exists X, Y \left[\mathrm{after}(x,X) \wedge \mathrm{before}(y,Y) \wedge Z = X \cap Y \wedge Z \neq \emptyset\right]$$

Because of the presence of $Z \neq \emptyset$, we must have $p(x) \leq p(y)$. □

The next lemma presents a formula $\mathrm{succ}(X,Y)$ which ensures that $Y$ contains exactly the direct successors (wrt. $\mathrm{seq}(G)$) of the cells from $X$. In particular, if $X$ contains a cell $(r,m)$ from the rightmost column of a grid $G = (k,m,\mu)$, then $Y$ contains the cell $(r+1,1)$ (provided that $r < k$) because $p(r,m)+1 = p(r+1,1)$.

**Lemma 4.5.** *There exists a formula $\mathit{succ}(X,Y) \in M\Sigma_1^G(\Gamma)$ such that, for all grids $G = (k,m,\mu)$ and sets $H, S \subseteq \mathit{dom}(G)$, we have*

$$\begin{aligned} &G, H, S \models \mathit{succ} \\ \iff\ &S = \{(r,c) \in \mathit{dom}(G) \mid \mathit{there\ exists}\ (r',c') \in H\ \mathit{s.t.}\ p(r,c) = p(r',c')+1\}\,. \end{aligned}$$

Assuming this lemma, we write $\mathrm{succ}(x,y)$ as an abbreviation for the $M\Sigma_1^G(\Gamma)$-formula

$$\exists X, Y \left[\forall z\,(z \in X \leftrightarrow z = x) \wedge \forall z\,(z \in Y \leftrightarrow z = y) \wedge \mathrm{succ}(X,Y)\right].$$

Clearly, $G, x, y \models \mathrm{succ}(x,y)$ if and only if $p(y) = p(x)+1$.

*Proof.* Let $\mathrm{succ}(X,Y)$ be the following $M\Sigma_1^G(\Gamma)$-formula (where $\mathrm{lc}(x)$ and $\mathrm{rc}(x)$ are the formulas from Example 4.1):

$$\begin{aligned} &\exists Z \\ &\quad \forall x\,[\mathrm{rc}(x) \rightarrow (x \in Z \leftrightarrow x \in X)] && \text{(4a)} \\ &\wedge\ \forall x, y\,[x\ S_h\ y \rightarrow (x \in Z \leftrightarrow y \in Z)] && \text{(4b)} \\ &\wedge\ \forall x\ \big[x \in Y \leftrightarrow \exists y\,\big((y\ S_h\ x \wedge y \in X) \vee (\mathrm{lc}(x) \wedge \exists y \in Z\colon y\ S_v\ x)\big)\big] && \text{(4c)} \end{aligned}$$

Let $G$ be a grid and $X, Y, Z \subseteq \mathrm{dom}(G)$ such that the lines (4a)–(4c) hold. The lines (4a) and (4b) say that an arbitrary cell of $G$ is contained in $Z$ if and only if the cell in the last column of the same row is contained in $X$. Therefore, we have $Z = \{(r,c) \in \mathrm{dom}(G) \mid (r,m) \in X\}$. Line (4c) expresses that a cell is contained

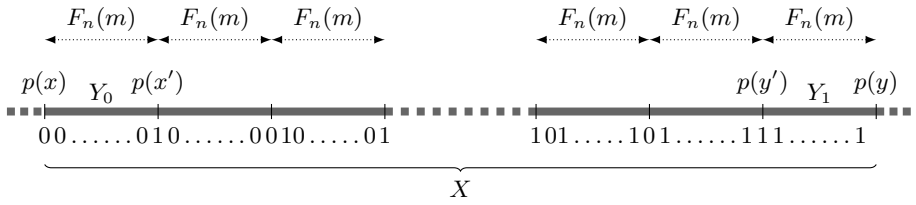


**Fig. 3.** This picture visualizes the situation ensured by the formula $\varphi_{n+1}$ of the proof of Lemma 4.6. The thick line illustrates $\mathrm{seq}(G)$. The vertical lines drawn on the thick line mark the positions of the elements from $Z$. The bits below the positions show whether they represent a 0 or 1 through the set $B$. The arrows indicate that the positions of successive elements of $Z$ have distance $F_n(m)$.

in $Y$ if and only if (i) its left neighbor is in $X$ or (ii) it is in the leftmost column and its upper neighbor is contained in $Z$ (i.e., the rightmost cell of the row directly above belongs to $X$). Hence, we have

$$Y = \{(r,c) \in \mathrm{dom}(G) \mid \text{there exists } (r',c') \in X \text{ s.t. } p(r,c) = p(r',c') + 1\}\,.$$

If $G$ is a grid, $H \subseteq \mathrm{dom}(G)$, and $S$ is as above, then it can be easily checked that $G, H, S \models \mathrm{succ}$. □

Recall that we want to express that $\mathrm{seq}(G)$ is an $m$-computation of $M$. To do this, we have in particular to relate the positions $i$ and $i + F_n(m)$ (where $m$ is the number of columns of $G$) in $\mathrm{seq}(G)$. Since our formulas are interpreted over the grid $G$, we actually need a formula $\varphi(x,y)$ expressing $p(y) = p(x) + F_n(m)$. This is the purpose of the following lemma, its proof uses the technique introduced in [33] by Klaus Reinhardt.

**Lemma 4.6.** *There exists a formula $\varphi_n(x,y) \in M\Sigma_n^G(\Gamma)$ such that, for all grids $G = (k,m,\mu)$ and cells $(r_1,c_1),(r_2,c_2) \in dom(G)$, we have*

$$G, (r_1,c_1), (r_2,c_2) \models \varphi_n \iff p(r_2,c_2) = p(r_1,c_1) + F_n(m)\,.$$

*Proof.* The lemma is shown by induction on $n$. Let $G = (k,m,\mu)$ be a grid. Since $F_0(m) = m$ and since every row of $G$ contains precisely $m$ elements, we can set $\varphi_0(x,y) = (x\ S_v\ y)$, a formula from $\mathrm{FO}^G(\Gamma) = M\Sigma_0^G(\Gamma)$.

For $n \geq 0$, let $\varphi_{n+1}(x,y)$ denote the following formula (we advise to read the explanations below and look at Fig. 3 simultaneously with each line of the

formula):

$$
\begin{aligned}
&\exists X, Y_0, Y_1, Z, B, S, x', y' \\
&\qquad \operatorname{interval}(x, y, X) \wedge \operatorname{interval}(x, x', Y_0) \wedge \operatorname{interval}(y', y, Y_1) && \text{(5a)}\\
&\quad \wedge\ x', y' \in X && \text{(5b)}\\
&\quad \wedge\ \varphi_n(x, x') \wedge \varphi_n(y', y) && \text{(5c)}\\
&\quad \wedge\ Z \cap Y_0 = \{x, x'\} \wedge \forall z_1, z_2 \in X\, [\varphi_n(z_1, z_2) \to (z_1 \in Z \leftrightarrow z_2 \in Z)] && \text{(5d)}\\
&\quad \wedge\ y \in Z && \text{(5e)}\\
&\quad \wedge\ \operatorname{succ}(B, S) && \text{(5f)}\\
&\quad \wedge\ B \cap (Y_0 \setminus \{x'\}) = \emptyset && \text{(5g)}\\
&\quad \wedge\ Y_1 \setminus \{y\} \subseteq B && \text{(5h)}\\
&\quad \wedge\ \forall z_1, z_2 \in Z\, [\varphi_n(z_1, z_2) \to (z_1 \in B \leftrightarrow z_2 \notin B)] && \text{(5i)}\\
&\quad \wedge\ \forall z_1, z_2 \in X\, [(\varphi_n(z_1, z_2) \wedge z_1 \notin Z) && \text{(5j)}\\
&\qquad \to ((z_1 \in S \wedge z_2 \notin S) \leftrightarrow (z_1 \notin B \leftrightarrow z_2 \in B))] \\
&\quad \wedge\ \forall z_1, z_2 \in X\, [(\varphi_n(z_1, z_2) \wedge z_1 \in Z \wedge z_1 \in S) \to z_2 \in S] && \text{(5k)}
\end{aligned}
$$

Let $G = (k, m, \mu)$ be a grid, $X, Y_0, Y_1, Z, B, S \subseteq \operatorname{dom}(G)$ and $x, y, x', y' \in \operatorname{dom}(G)$ such that the lines (5a)–(5k) hold. Line (5a) expresses that $x \leq y$, $x \leq x'$, $y' \leq y$, and:

$$X = \{(r, c) \in \operatorname{dom}(G) \mid p(x) \leq p(r, c) \leq p(y)\} \tag{6a}$$
$$Y_0 = \{(r, c) \in \operatorname{dom}(G) \mid p(x) \leq p(r, c) \leq p(x')\} \tag{6b}$$
$$Y_1 = \{(r, c) \in \operatorname{dom}(G) \mid p(y') \leq p(r, c) \leq p(y)\} \tag{6c}$$

Together with (5b), we obtain $x \leq y'$ and $x' \leq y$. By the induction hypothesis for $\varphi_n$, line (5c) says that $p(x') = p(x) + F_n(m)$ and $p(y) = p(y') + F_n(m)$.

From (5d), we obtain $Z \cap X = \{(r, c) \mid p(r, c) = p(x) + k \cdot F_n(m), k \in \mathbb{N}\} \cap X$. Together with (5e), we obtain $p(y) = p(x) + k \cdot F_n(m)$ for some $k > 0$. In other words, the set $Z$ divides the interval $\{p(x), p(x) + 1, \ldots, p(y)\}$ represented by $X$ into blocks of length $F_n(m)$ each. The first block starts at position $p(x)$ and the last one at position $p(y')$. With any such block, we associate a natural number depending on the set $B$: if the block starts at position $p(z)$ with $z \in Z$ and

$$C = \{i < F_n(m) \mid \text{there exists } (r, c) \in B \text{ such that } p(r, c) = p(z) + i\},$$

the the associated number is $\sum_{i \in C} 2^i$. In other words, each block is interpreted as a binary number (least significant bit first) where $B$ contains those bits set to 1. Line (5f) says that

$$S = \{(r, c) \in \operatorname{dom}(G) \mid \text{there exists } (r', c') \in B \text{ s.t. } p(r, c) = p(r', c') + 1\}. \tag{7}$$

Recalling that $Y_0 \setminus \{x'\}$ represents the first block, (5g) expresses that its associated number is 0. Dually, using (5h), we deduce that $\sum_{0 \leq i < F_n(m)} 2^i = 2^{F_n(m)} - 1$ is the number associated with the final block represented by $Y_1 \setminus \{y\}$. We show

that the blocks "count" from 0 to $2^{F_n(m)} - 1$. By (5i), the least significant bits of consecutive blocks alternate. Considering line (5j), the premise expresses that $z_1$ and $z_2$ mark the same position $i$ in consecutive blocks and that $z_1$ (and therefore $z_2$) is not the first position of a block. The conclusion says that the $(i-1)$-th bit drops from 1 to 0 if and only if the $i$-th bit changes. Hence, line (5j) expresses that the number associated with the following block is obtained by adding one modulo $2^{F_n(m)}$. The final formula (5k) ensures that the last (most significant) bit never drops from 1 to 0. Hence, the number of blocks must be $2^{F_n(m)}$. Since each of them is of length $F_n(m)$, we obtain $p(y) = p(x) + F_n(m) \cdot 2^{F_n(m)} = p(x) + F_{n+1}(m)$.

For the converse direction, let $G = (k, m, \mu)$ be a grid, $x, x', y', y \in \mathrm{dom}(G)$ such that $p(x) = p(y) + F_{n+1}(m)$, $p(x') = p(x) + F_n(m)$, and $p(y) = p(y) + F_n(m)$. Let $X, Y_0, Y_1 \subseteq \mathrm{dom}(G)$ be as in the lines (6a) – (6c). Furthermore, let

$$Z = X \cap \{(r, c) \in \mathrm{dom}(G) \mid p(r, c) = p(x) + k \cdot F_n(m), k \in \mathbb{N}\},$$

i.e., $Z$ divides $X$ into $2^{F_n(m)}$ many blocks of length $F_n(m)$. Let $B \subseteq \mathrm{dom}(G)$ such that, for every $i \in [2^{F_n(m)}]$, the $i$-th block represents the number $i-1$ in the above sense. Finally, let $S$ be as in line (7). It can be checked by easy inspection that the lines (5a)–(5k) hold.

By induction, $\varphi_n \in M\Sigma_n^G(\Gamma)$. Note that this formula occurs in the lines (5b), (5c), (5d), (5i), (5j), and (5k). At all these places, it occurs either positively under the existential quantification in the very first line, or negatively under an additional universal quantification. Hence, $\varphi_{n+1} \in M\Sigma_{n+1}^G(\Gamma)$ as required. □

We are now ready to express that a (sufficiently large) grid $G$ encodes an initial and accepting $m$-computation (where $m$ is the number of columns of $G$):

**Proposition 4.7.** *There exists a sentence $\psi_n \in M\Pi_n^G(\Gamma)$ such that, for all grids $G = (k, m, \mu)$ with $|seq(G)| > F_n(m)$, we have*

$$G \models \psi_n \iff \textit{seq}(G) \textit{ is an initial and accepting } m\textit{-computation of } M_n\,.$$

*Proof.* Let $G = (k, m, \mu)$ be a grid with $|\mathrm{seq}(G)| > F_n(m)$. Consider the following formula:

$$\begin{aligned}\alpha_1 = {} & \exists x\,(\mathrm{ul}(x) \wedge \mu(x) = \triangleright) \wedge \exists x\,(\mathrm{lr}(y) \wedge \mu(y) = \triangleleft) \\ & \wedge \forall x, y\,[\mathrm{succ}(x, y) \to (\mu(x) = \triangleleft \leftrightarrow \mu(y) = \triangleright)]\,.\end{aligned}$$

Note that $\mathrm{ul}(x)$ and $\mathrm{lr}(x)$ are the formulas from Example 4.1. If $G \models \alpha_1$, then, by the first line of $\alpha_1$, the cell $(1, 1)$ is labeled by $\triangleright$ and $(k, m)$ is labeled by $\triangleleft$. The second line expresses that each cell from $\mathrm{dom}(G) \setminus \{(1, 1), (k, m)\}$ labeled by $\triangleright$ is directly preceded by a $\triangleleft$-labeled cell in $\mathrm{seq}(G)$ and that each cell from $\mathrm{dom}(G) \setminus \{(1, 1), (k, m)\}$ labeled by $\triangleleft$ is followed by a $\triangleright$-cell. It can be shown that the grid $G$ satisfies the formula $\alpha_1$ if and only if $\mathrm{seq}(G)$ is in $(\triangleright(\Gamma \setminus \{\triangleright, \triangleleft\})^*\triangleleft)^+$.

Recall that seq($G$) is assumed to be at least of length $F_n(m)+1$. Therefore, $G$ satisfies the conjunction of $\alpha_1$ and

$$\alpha_2 = \forall x,y,X \left[ \begin{pmatrix} \varphi_n(x,y) \wedge \mu(x) = \triangleright \\ \wedge\ \mathrm{interval}(x,y,X) \end{pmatrix} \rightarrow X \cap \mu^{-1}(\triangleright) = \{x,y\} \right] \\ \wedge \forall x,y,X \left[ \begin{pmatrix} \varphi_n(x,y) \wedge \mu(y) = \triangleleft \\ \wedge\ \mathrm{interval}(x,y,X) \end{pmatrix} \rightarrow X \cap \mu^{-1}(\triangleleft) = \{x,y\} \right]$$

if and only if seq($G$) $\in (\Gamma^{F_n(m)} \cap \triangleright(\Gamma \setminus \{\triangleright,\triangleleft\})^*\triangleleft)^+$, i.e., each factor of seq($G$) from $\triangleright(\Gamma \setminus \{\triangleright,\triangleleft\})^*\triangleleft$ is of length $F_n(m)$. Similarly, the formula

$$\alpha_3 = \forall x,y,X \left[ \begin{pmatrix} \varphi_n(x,y) \wedge \mu(x) = \triangleright \\ \wedge\ \mathrm{interval}(x,y,X) \end{pmatrix} \rightarrow |X \cap \mu^{-1}(Q)| = 1 \right].$$

ensures that each factor of seq($G$) from $\triangleright(\Gamma \setminus \{\triangleright,\triangleleft\})^*\triangleleft$ contains exactly one state from the set of states $Q$ of $M_n$. Hence, we have $G \models \alpha_1 \wedge \alpha_2 \wedge \alpha_3$ if and only if

$$\mathrm{seq}(G) \in C = \left(\Gamma^{F_n(m)} \cap \triangleright(\Gamma_T \setminus \{\triangleright,\triangleleft\})^* Q (\Gamma_T \setminus \{\triangleright,\triangleleft\})^* \triangleleft\right)^+.$$

Note that there is a relation $R \subseteq \Gamma^6$ such that $\zeta \in C$ is an $m$-computation of the Turing machine $M_n$ if and only if, for all $\gamma_1,\gamma_2,\gamma_3,\delta_1,\delta_2,\delta_3 \in \Gamma$ with $\zeta \in \Gamma_T^* \gamma_1\gamma_2\gamma_3 \Gamma_T^{F_n(m)-3} \delta_1\delta_2\delta_3\Gamma_T^*$, we have $(\gamma_1,\gamma_2,\gamma_3,\delta_1,\delta_2,\delta_3) \in R$. Let

$$\alpha_4 = \forall x_1,x_2,x_3,y_1,y_2,y_3 \\ \left[ \begin{pmatrix} \varphi_n(x_1,y_1) \wedge \mathrm{succ}(x_1,x_2) \wedge \mathrm{succ}(x_2,x_3) \\ \wedge\ \mathrm{succ}(y_1,y_2) \wedge \mathrm{succ}(y_2,y_3) \end{pmatrix} \right. \\ \left. \rightarrow \bigvee_{(\gamma_1,\gamma_2,\gamma_3,\delta_1,\delta_2,\delta_3)\in R} \begin{pmatrix} \mu(x_1) = \gamma_1 \wedge \mu(x_2) = \gamma_2 \wedge \mu(x_3) = \gamma_3 \\ \wedge\ \mu(y_1) = \delta_1 \wedge \mu(y_2) = \delta_2 \wedge \mu(y_3) = \delta_3 \end{pmatrix} \right]$$

The grid $G$ satisfies $\psi = \bigwedge_{i\in[4]} \alpha_i$ if and only if seq($G$) is an $m$-computation of $M_n$. Let $q_0$ be the initial and $q_1$ be the final state of the Turing machine $M_n$. Consider the following formula (where ul($x$) is the formula from Example 4.1):

$$\alpha_5 = \forall x,y\,[(\mathrm{ul}(x) \wedge x\ S_h\ y) \rightarrow \mu(y) = q_0] \\ \wedge \forall x,y,X \left[ \begin{array}{l} (\mathrm{ul}(x) \wedge \varphi_n(x,y) \wedge \mathrm{interval}(x,y,X)) \\ \qquad \rightarrow \forall z \in X \setminus \{y\}\,(\mu(z) \in \{\triangleleft,\square\} \leftrightarrow \exists z'\,(z'\ S_v\ z)) \end{array} \right]$$

The grid $G$ satisfies the formula $\psi = \bigwedge_{i\in[5]} \alpha_i$ if and only if seq($G$) is an initial $m$-computation of $M_n$, i.e., seq($G$) is an $m$-computation and the first configuration of seq($G$) is an initial configuration of $M_n$ on an input word of length $m-2$. The last formula to consider is the following:

$$\alpha_6 = \exists z\,(\mu(z) = q_1)$$

Recall that all accepting states of the Turing machine $M_n$ are halting. Hence, $G$ satisfies $\psi_n = \bigwedge_{i\in[6]} \alpha_i$ if and only if seq($G$) is an initial and accepting $m$-computation of $M_n$.

Moreover, we have $\psi_n \in M\Pi_n^G(\Gamma)$ since it is a conjunction of $M\Pi_n^G(\Gamma)$-formulas. The formula $\alpha_1$ is an $M\Pi_1$-formula because the $M\Sigma_1$-formula $\mathrm{succ}(x,y)$ only occurs negatively under universal quantification. In the formulas $\alpha_2 - \alpha_6$, the $M\Sigma_n^G(\Gamma)$-formula $\varphi_n$ and the $M\Sigma_1$-formulas $\mathrm{succ}(x,y)$ and $\mathrm{interval}(x,y,X)$ only occur negatively under universal quantification. Therefore, $\alpha_2, \ldots, \alpha_6 \in M\Pi_n^G(\Gamma)$. □

Consider the $M\Pi_n^G(\Gamma)$-definable temporal logic $\mathrm{TL}_n^G$ based on the usual boolean connectives and the constants $\mathsf{COMPUTATION}$ and $\mathsf{TOOSHORT}$ where

$$[\![\mathsf{COMPUTATION}]\!] = \psi_n\,,$$
$$[\![\mathsf{TOOSHORT}]\!] = \forall x, y\, \neg\varphi_n(x,y)\,,$$

$\psi_n$ is the formula from Prop. 4.7, and $\varphi_n$ is the formula from Lemma 4.6. Recall that every initial and accepting computation of $M_n$ has at least length 2 since no state is initial and accepting at the same time. If $G$ is a grid with $m$ columns, then the following holds:

$$\begin{aligned} & G \models \neg\mathsf{TOOSHORT} \wedge \mathsf{COMPUTATION} \\ & \iff |\mathrm{seq}(G)| > F_n(m) \text{ and } G \models \mathsf{COMPUTATION} \\ & \iff \mathrm{seq}(G) \text{ is an initial and accepting } m\text{-computation of } M_n \text{ of length} \geq 2 \end{aligned}$$

Now, we can prove the desired lower bound for the satisfiability problem of $\mathrm{MSO}^G(\Gamma)$-definable temporal logics:

*Proof (of Theorem 4.3).* Recall that the Turing machine $M$ works in space $F_n(m) - 3$ where $m - 2$ is the length of the input. Consider the $M\Pi_n^G(\Gamma)$ definable temporal logic $\mathrm{TL}_n^G$ based on the modality $\mathsf{S_h}$ (see Example 4.2), the usual boolean connectives, and the constants $\mathsf{COMPUTATION}$, $\mathsf{TOOSHORT}$, and $\mathsf{TRUE}$ where $[\![\mathsf{TRUE}]\!] = \exists x\,(x = x)$ is always evaluated to true. Let $v = v_1 \ldots v_{m-2}$ be an input word of the Turing machine $M$ and consider the formula

$$\mathsf{INIT}_v = \mathsf{S_h}\,(\triangleright \wedge \mathsf{S_h}\,(q_0 \wedge \mathsf{S_h}\,(v_1 \wedge \mathsf{S_h}\,(v_2 \wedge \mathsf{S_h}\,(\ldots v_{m-2} \wedge \neg\,\mathsf{S_h}\,\mathsf{TRUE})\ldots))))$$

which intuitively expresses the fact that the first configuration is actually the initial configuration of $M$ on the input word $v$ and that the grid has exactly $m$ columns. Then the input $v$ is accepted by $M$ if and only if there is a grid $G$ with $m$ columns satisfying the formula $\neg\mathsf{TOOSHORT} \wedge \mathsf{COMPUTATION} \wedge \mathsf{INIT}_v$. Note that this formula can be constructed from $v$ in linear time. Since the language of $M$ is $n$-EXPSPACE-hard, the satisfiability problem for $\mathrm{TL}_n^G$ is $n$-EXPSPACE-hard. □

### 4.2 The Reduction from Labeled Grids to Nested Words

Our goal is a polynomial reduction of the satisfiability problem of a $M\Pi_n^G(\Gamma)$-definable temporal logic expressing properties of labeled grids to the satisfiability problem of an $M\Pi_n(\Gamma)$-definable temporal logic which is evaluated on nested

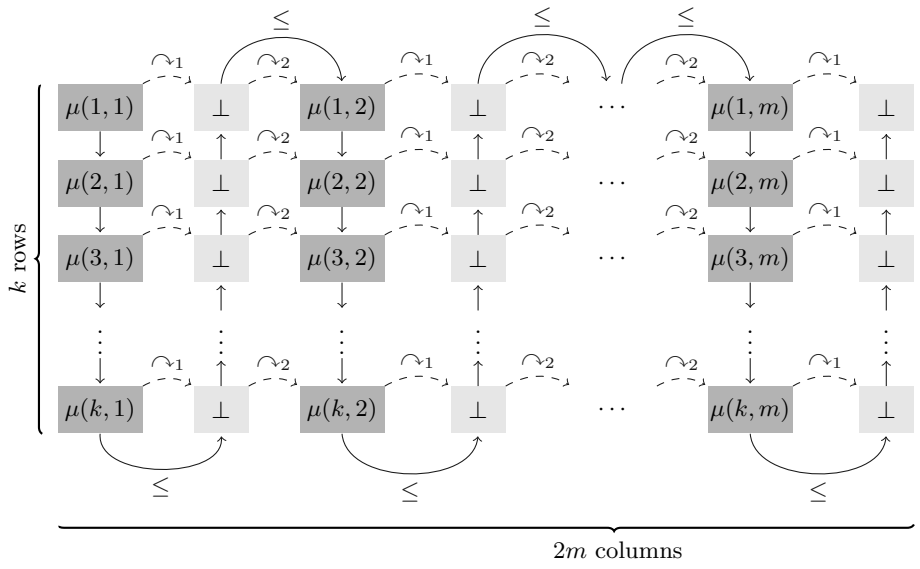


**Fig. 4.** The nested word $\nu_G$ obtained from a grid $G = (k, m, \mu)$.

words. For this purpose, we represent a grid $G = (k, m, \mu)$ over $\Gamma$ by a $2m$-phase nested word $\nu_G = (P, \leq, \lambda, \curvearrowright_1, \curvearrowright_2)$ over the alphabet $\Gamma \uplus \{\perp\}$ like it is shown in Fig. 4. First, we insert after every column of $G$ an artificial column whose cells are labeled by $\perp$'s. Hence, we obtain an intermediate grid with $2m$ columns whose cells become the positions of $\nu_G$. The direct successor relation of the linear ordering $\leq$ of $\nu_G$ is depicted by the continuous edges $\rightarrow$ in Fig. 4. The cell $(1, 1)$ is the minimal element wrt. $\leq$ followed by the remaining cells of the first column top down. That means, we have $(1, 1) \lessdot (2, 1) \lessdot \ldots \lessdot (k, 1)$. The cell $(k, 1)$ is then followed by the cells of the second column (labeled by the $\perp$'s) bottom up, i.e., $(k, 1) \lessdot (k, 2) \lessdot (k - 1, 2) \lessdot \ldots \lessdot (1, 2)$. Then the cells of the third column (which is the second column in the original grid) follow top down and so on. We insert a nesting edge $\curvearrowright_1$ from every cell of an odd column to its right neighbor. Similarly, we introduce a nesting edge $\curvearrowright_2$ pointing from every cell of an even column to its right neighbor.

Recall that we insert artificial rows into the grid which become intervals of $\perp$'s in the corresponding nested word. Indeed, they allow an easy navigation within $\nu_G$. More precisely, moving from a grid cell to its lower neighbor corresponds always to going to the direct successor with respect to $\leq$. Whereas moving to the right neighbor of a grid cell is accomplished by following a $\curvearrowright_1$ and a $\curvearrowright_2$ edge.

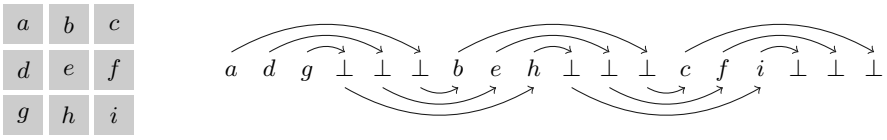


**Fig. 5.** The grid $G$ and the nested word $\nu_G$ from Example 4.8.

*Example 4.8.* Consider the grid $G = (3, 3, \mu)$ from Fig. 5. The corresponding nested word $\nu_G$ is also depicted in Fig. 5. The upper and lower edges visualize the nesting relation $\curvearrowright_1$ and $\curvearrowright_2$, resp.

More formally, if $G = (k, m, \mu)$ is a grid over the alphabet $\Gamma$, then the $2m$-phase nested word $\nu_G = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ 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 $\lambda(i, j) = \mu(i, {}^{j+1}/_2)$ if $j$ is odd and $\lambda(i, j) = \bot$ otherwise. For all $(i_1, j_1), (i_2, j_2) \in P$, we define $(i_1, j_1) \leq (i_2, j_2)$ if and only if

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

The nesting relations $\curvearrowright_1$ and $\curvearrowright_2$ are defined as follows:

$$\begin{aligned} \curvearrowright_1 &= \{\big((i_1, j_1), (i_2, j_2)\big) \in P^2 \mid i_1 = i_2,\ j_2 = j_1 + 1 \text{ even}\} \\ \curvearrowright_2 &= \{\big((i_1, j_1), (i_2, j_2)\big) \in P^2 \mid i_1 = i_2,\ j_2 = j_1 + 1 \text{ odd}\} \end{aligned}$$

Whereas we set $\curvearrowright_s = \emptyset$ for all $3 \leq s \leq \sigma$.

The next lemma states that there is an FO-formula $\varphi_{\Gamma\bot}(X)$ which expresses that $X$ is an interval within a nested word of the form $(\Gamma^+\bot^+)^+$ such that the position directly left (resp. right) of $X$ (if existent) is not labeled by $\Gamma$ ($\bot$).

**Lemma 4.9.** *There exists a formula $\varphi_{\Gamma\bot}(X) \in FO(\Gamma \uplus \{\bot\})$ such that, for all nested words $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ and $I \subseteq P$, we have $\nu, I \models \varphi_{\Gamma\bot}$ if and only if there exist $i, j \in P$ with*

*(1) $I = [i, j]$,*
*(2) $\lambda(i) \in \Gamma$ and ($i = \min(P, \leq)$ or $\lambda(k) = \bot$ with $k \lessdot i$), and*
*(3) $\lambda(j) = \bot$ and ($j = \max(P, \leq)$ or $\lambda(k) \in \Gamma$ with $j \lessdot k$).*

*Proof.* Consider the following FO-formula:

$$\begin{aligned} \varphi_{\Gamma\bot}(X) = \exists x, y \\ & x, y \in X && \text{(8a)} \\ & \wedge \forall z\,[z \in X \to (z = x \vee \exists z' \in X\, z' \lessdot z)] && \text{(8b)} \\ & \wedge (\min(x) \vee \exists z\,[\min(z) \to z \notin X]) && \text{(8c)} \\ & \wedge \forall z\,[y \lessdot z \to z \notin X] && \text{(8d)} \\ & \wedge \lambda(x) \in \Gamma \wedge \forall z\,(z \lessdot x \to \lambda(z) = \bot) && \text{(8e)} \\ & \wedge \lambda(y) = \bot \wedge \forall z\,(y \lessdot z \to \lambda(z) \in \Gamma) && \text{(8f)} \end{aligned}$$

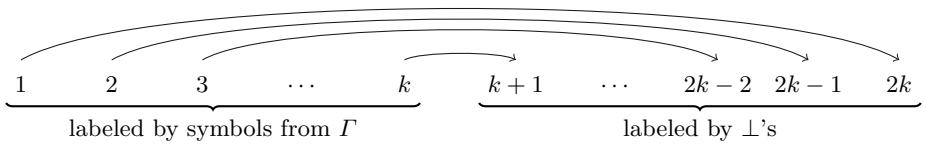


**Fig. 6.** The nested word $\nu$ from the base clause of the proof of Prop. 4.11. The edges visualize the nesting relation $\curvearrowright_1$.

Let $\nu = (P, \leq, \lambda, \curvearrowright_1, \curvearrowright_2)$ be a nested word, $X \subseteq P$, and $x, y \in P$ such that the lines (8a) – (8f) hold. Without loss of generality, we can assume that $P = [n]$ for some $n \in \mathbb{N}$. By (8a), we obtain $x, y \in X$. Let us assume that there exists $k \in X$ with $k < x$. From line 8b, it follows that $k - 1 \in X$, $k - 2 \in X$, ..., $1 \in X$. Since $k < x$, we have $x \neq 1$. This is a contradiction to line 8c. That means that $[x - 1] \cap X = \emptyset$ and, in particular, $x \leq y$. From (8b), we obtain $[x, y] \subseteq X$. Let us assume that $k \in X$ with $y < k$. By (8b), we have $k - 1 \in X$, $k - 2 \in X$, ..., $y + 1 \in X$. This contradicts (8d). Hence, $X = [x, y]$ and condition 1 from Lemma 4.9 holds, respectively. It can be easily checked that the conditions 2 and 3 are ensured by the lines (8e) and (8f). The converse direction follows by simple inspection. □

**Lemma 4.10.** *There exists a formula $\varphi_{\perp\Gamma}(X) \in FO(\Gamma \uplus \{\perp\})$ such that, for all nested words $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ and $I \subseteq P$, we have $\nu, I \models \varphi_{\perp\Gamma}$ if and only if there exist $i, j \in P$ with*

*(1) $I = [i, j]$,*
*(2) $\lambda(i) = \perp$ and ($i = \min(P, \leq)$ or $\lambda(k) \in \Gamma$ with $k \lessdot i$), and*
*(3) $\lambda(j) \in \Gamma$ and ($j = \max(P, \leq)$ or $\lambda(k) = \perp$ with $j \lessdot k$)*

*Proof.* The formula $\varphi_{\perp\Gamma}$ can be obtained from the formula $\varphi_{\Gamma\perp}$ in the proof of Lemma 4.9 by exchanging $\lambda(v) \in \Gamma$ and $\lambda(v) = \perp$ in lines (8e) and (8f) for all $v \in \{x, y, z\}$. □

Using the Lemmas 4.9 and 4.10, we can now specify an $MI\Pi_1(\Gamma \uplus \{\perp\})$-formula defining the set of nested words representing grids.

**Proposition 4.11.** *There exists a sentence* grid $\in MI\Pi_1(\Gamma \uplus \{\perp\})$ *such that, for all nested words $\nu$, we have $\nu \models$* grid *if and only if there exists a grid $G$ over $\Gamma$ with $\nu_G = \nu$.*

*Proof.* Consider the following formula $\psi$:

$$\exists x\,(\min(x) \wedge \lambda(x) \in \Gamma) \;\wedge\; \exists x\,(\max(x) \wedge \lambda(x) = \perp) \tag{9a}$$

$$\wedge \curvearrowright_1 \subseteq \lambda^{-1}(\Gamma) \times \lambda^{-1}(\perp) \;\wedge\; \forall X\, \varphi_{\Gamma\perp}(X) \to \mathrm{bij}_1(X) \tag{9b}$$

$$\wedge \curvearrowright_2 \subseteq \lambda^{-1}(\perp) \times \lambda^{-1}(\Gamma) \;\wedge\; \forall X\, \varphi_{\perp\Gamma}(X) \to \mathrm{bij}_2(X) \tag{9c}$$

$$\wedge \curvearrowright_3 = \emptyset \;\wedge\; \curvearrowright_4 = \emptyset \;\wedge\; \ldots \;\wedge\; \curvearrowright_\sigma = \emptyset \tag{9d}$$

Recall that $\mathrm{bij}_1$ and $\mathrm{bij}_2$ are the formulas from Example 2.3. It suffices to consider nested words $\nu = (P, \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ where $P = [n]$ for some $n \in \mathbb{N}$ and $\leq$ is the natural ordering over $[n]$. It can be easily verified that $\nu_G \models \psi$ holds for every grid $G$. Now, let us prove the direction from left to right. By $c(\nu)$ we denote the number of $\bot$-labeled positions from $[n]$ who do *not* have a direct successor which is labeled by $\bot$ (i.e., either there is no successor or the direct successor is labeled by a symbol from $\Gamma$). More formally, we define

$$c(\nu) = \left|\{i \in \lambda^{-1}(\bot) \mid i = n \text{ or } (i+1 \leq n \text{ and } \lambda(i+1) \in \Gamma)\}\right|.$$

Let $\nu = ([n], \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word satisfying $\psi$. Because of line (9a), we have $\lambda(n) = \bot$. Hence, $c(\nu) \geq 1$. We show that there exists a grid $G$ with $\nu = \nu_G$ by induction over $c(\nu)$. Assume that $c(\nu) = 1$. By line (9a), we have $\lambda(1) \in \Gamma$ and $\lambda(n) = \bot$. Therefore, there exists a $k \in [n-1]$ such that $\lambda(k) \in \Gamma$ and $\lambda(k+1) = \bot$. Let $I_1 = [k]$ and $I_2 = [k+1, n]$. Since $c(\nu) = 1$, we have $\lambda(I_1) \subseteq \Gamma$ and $\lambda(I_2) \subseteq \{\bot\}$. By (9b) and Lemma 4.9, we obtain that $\curvearrowright_1$ is a bijection from $I_1$ to $I_2$. Hence, we have $n = 2k$ and $\curvearrowright_1 = \{(1, 2k), (2, 2k-1), \ldots, (k, k+1)\}$. From $\lambda^{-1}(\bot) \times \lambda^{-1}(\Gamma) \cap \leq \; = \emptyset$ and the left part of the conjunction in line (9c) it follows that $\curvearrowright_2 = \emptyset$. Because of (9d), we have $\curvearrowright_s = \emptyset$ for all $3 \leq s \leq \sigma$. That means that $\nu$ is of the form shown in Fig. 6. We have $\nu_G = \nu$ for the one-column grid $G = (k, 1, \mu)$ with $\mu(r, 1) = \lambda(r)$ for all $r \in [k]$.

Assume that there exists a grid $G = (k, c(\nu), \nu)$ with $\nu = \nu_G$ and $k = {}^{n}/_{2c(\nu)}$ for every nested word $\nu$ with $n$ positions, $1 \leq c(\nu) \leq m$, and $\nu \models \psi$. For the induction step, let $\nu = ([n], \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word with $c(\nu) = m+1$ and $\nu \models \psi$. Furthermore, let $n' \in [n]$ be the largest position with $\lambda(n') = \bot$ and $\lambda(n'+1) \in \Gamma$. Note that such an $n'$ exists since $c(\nu) > 1$. Furthermore, let $\nu'$ be the restriction of $\nu$ to the positions $[n']$. It can be easily checked that $c(\nu') = m$ and $\nu' \models \psi$. By induction, there exists a grid $G' = (k, m, \mu')$ with $\nu_{G'} = \nu'$. Because of $\lambda(n'+1) \in \Gamma$ and $\lambda(n) = \bot$ (due to the right part of the conjunction in line (9a)), there exists $i \in [n'+1, n]$ such that $\lambda(i) \in \Gamma$ and $\lambda(i+1) = \bot$. Since $c(\nu) = c(\nu') + 1$, we must have $\lambda([n'+1, i]) \subseteq \Gamma$ and $\lambda([i+1, n]) \subseteq \{\bot\}$. We set $I_1 = [n'+1, i]$ and $I_2 = [i+1, n]$. Furthermore, let $J = [n'-k+1, n']$ (i.e., $J$ represents the last interval of $\bot$'s in $\nu'$). From $\lambda(J) \subseteq \{\bot\}$, $\lambda(I_1) \subseteq \Gamma$, and line (9c), we obtain that $\curvearrowright_2 \cap (J \cup I_1)^2$ is a bijection from $J$ to $I_1$ and $|I_1| = |J| = k$, respectively. Analogously, from (9b), it follows that $\curvearrowright_1 \cap (I_1 \cup I_2)^2$ is a bijection from $I_1$ to $I_2$. Therefore, we have $|I_2| = k$. Consider the grid $G = (k, m+1, \mu)$ with

$$\mu(r, c) = \begin{cases} \mu'(r, c) & \text{if } c \in [m] \\ \lambda(c \cdot 2k + r) & \text{if } c = m+1 \end{cases}$$

for all $(r, c) \in \mathrm{dom}(G)$. It holds that $\nu_G = \nu$. □

We now translate $M\Pi_n^G(\Gamma)$-modalities into suitable $M\Pi_n$-modalities.

**Lemma 4.12.** *Let $\varphi(x, X_1, \ldots, X_n) \in \mathit{MII}_n^G(\Gamma)$ be a modality definition. There exists a modality definition $\varphi^\# \in \mathit{MII}_n(\Gamma \uplus \{\bot\})$ such that the following holds for all grids $G = (k, m, \mu)$, all positions $(r, c) \in \mathit{dom}(G)$, and all sets $I_1, \ldots, I_n \subseteq \mathit{dom}(G)$:*

$$G, (r, c), I_1, \ldots, I_n \models \varphi \iff \nu_G, (r, 2c-1), I_1', \ldots, I_n' \models \varphi^\#$$

*where $I_i' = \{(r, 2c-1) \mid (r, c) \in I_i\}$ for all $i \in [n]$.*

*Proof.* A cell $(r, c)$ from $\mathrm{dom}(G)$ corresponds to the position $(r, 2c-1)$ from $\mathrm{dom}(\nu_G)$. For cells $(r_1, c_1), (r_2, c_2) \in \mathrm{dom}(G)$, we have $(r_1, c_1)\ S_h\ (r_2, c_2)$ if and only if there exists a position $(i, j) \in \mathrm{dom}(\nu_G)$ such that $(r_1, 2c_1 - 1) \curvearrowright_1 (i, j)$ and $(i, j) \curvearrowright_2 (r_2, 2c_2 - 1)$. Whereas we have $(r_1, c_1)\ S_v\ (r_2, c_2)$ if and only if $(r_1, 2c_1 - 1) \lessdot (r_2, 2c_2 - 1)$. Furthermore, the cells from $\mathrm{dom}(G)$ are precisely the positions from $\mathrm{dom}(\nu_G) \setminus \lambda^{-1}(\bot)$. With this in mind, it can be easily seen that the following definition is sound. If $\varphi$ is an $\mathrm{MSO}^G(\Gamma)$-formula, then the MSO-formula $\varphi^\#$ is inductively defined as follows:

$$\varphi^\# = \begin{cases} \lambda(x) = a & \text{if } \varphi = (\mu(x) = a) \\ \exists y\,[x \curvearrowright_1 y \wedge y \curvearrowright_2 z] & \text{if } \varphi = (x\ S_h\ z) \\ x \lessdot y & \text{if } \varphi = (x\ S_v\ y) \\ x = y & \text{if } \varphi = (x = y) \\ x \in X & \text{if } \varphi = x \in X \\ \neg\psi^\# & \text{if } \varphi = \neg\psi \\ \psi_1^\# \vee \psi_2^\# & \text{if } \varphi = \psi_1 \vee \psi_2 \\ \exists X\,(X \cap \lambda^{-1}(\bot) = \emptyset \wedge \psi^\#) & \text{if } \varphi = \exists X\,\psi \\ \exists x\,(\lambda(x) \neq \bot \wedge \psi^\#) & \text{if } \varphi = \exists x\,\psi \end{cases}$$

Note that $\varphi^\# \in \mathit{MII}_n(\Gamma)$ whenever $\varphi \in \mathit{MII}_n^G(\Gamma)$ for all $n \in \mathbb{N}$. □

Let $\mathrm{TL} = (B, \mathrm{ar}, [\![-]\!])$ be an $\mathrm{MSO}^G(\Gamma)$-definable temporal logic. Then $\mathrm{TL}^\#$ denotes the $\mathrm{MSO}(\Gamma \uplus \{\bot\})$-definable logic $(B^\#, \mathrm{ar}^\#, [\![-]\!]^\#)$ where

- $B^\# = B \cup \{\neg, \wedge, \mathsf{TRUE}, \mathsf{RET}, \mathsf{GRID}\}$,
- $\mathrm{ar}^\# \upharpoonright B = \mathrm{ar}$,
- $\mathrm{ar}^\#(\wedge) = 2$, $\mathrm{ar}^\#(\mathsf{RET}) = \mathrm{ar}^\#(\neg) = 1$, $\mathrm{ar}^\#(\mathsf{TRUE}) = \mathrm{ar}^\#(\mathsf{GRID}) = 0$,
- $[\![M]\!]^\# = \varphi^\# \wedge (\lambda(x) \neq \bot)$ for all $M \in B$ with $[\![M]\!] = \varphi(x, X_1, \ldots, X_n)$, $[\![\mathsf{GRID}]\!]^\#$ is defined as in Prop. 4.11, $[\![\neg]\!]^\#$, $[\![\wedge]\!]^\#$, $[\![\mathsf{TRUE}]\!]^\#$, and $[\![\mathsf{RET}]\!]^\#$ are defined as in Example 2.4.

Clearly, every formula of TL is also a formula of $\mathrm{TL}^\#$. Vice versa, this is not the case since $\mathrm{TL}^\#$ contains additional modalities. If TL is $\mathit{MII}_n^G(\Gamma)$-definable for some $n \in \mathbb{N}$, then $\mathrm{TL}^\#$ is $\mathit{MII}_n(\Gamma)$-definable.

**Lemma 4.13.** *If TL is an $\mathit{MSO}^G(\Gamma)$-definable temporal logic, $F \in \mathit{TL}$, and $G$ is a grid, then $G \models_{\mathit{TL}} F$ if and only if $\nu_G \models_{\mathit{TL}^\#} F$.*

*Proof.* Let $\mathrm{TL} = (B, \mathrm{ar}, [\![-]\!])$. Firstly, we show that the following holds:

$$F^{\nu_G, \mathrm{TL}^\#} = \{(r, 2c-1) \mid (r,c) \in F^{G,\mathrm{TL}}\} \tag{10}$$

Let $F = M(F_1, \ldots, F_n)$ with $M \in B$, $\mathrm{ar}(M) = n$, and $F_1, \ldots, F_n \in \mathrm{TL}$. By induction, we have $F_i^{\nu_G, \mathrm{TL}^\#} = \{(r, 2c-1) \mid (r,c) \in F_i^{G,\mathrm{TL}}\}$. Let $(r,c) \in \mathrm{dom}(G)$. If $(r,c) \in F^{G,\mathrm{TL}}$, then, by definition, we have $G, F_1^{G,\mathrm{TL}}, \ldots, F_n^{G,\mathrm{TL}}, (r,c) \models [\![M]\!]$. From Lemma 4.12, we obtain $\nu_G, F_1^{\nu_G,\mathrm{TL}^\#}, \ldots, F_n^{\nu_G,\mathrm{TL}^\#}, (r, 2c-1) \models [\![M]\!]^\#$. Hence, $(r, 2c-1) \in F^{\nu_G,\mathrm{TL}^\#}$.

For the converse direction, let $(i,j) \in F^{\nu_G,\mathrm{TL}^\#}$. Because of $\lambda(i,j) \neq \bot$ (cf. the definition of $\mathrm{TL}^\#$), there exists $(r,c) \in \mathrm{dom}(G)$ with $(r, 2c-1) = (i,j)$. Therefore, $\nu_G, F_1^{\nu_G,\mathrm{TL}^\#}, \ldots, F_n^{\nu_G,\mathrm{TL}^\#}, (r, 2c-1) \models [\![M]\!]^\#$ holds. By Lemma 4.12, we obtain $G, F_1^{G,\mathrm{TL}}, \ldots, F_n^{G,\mathrm{TL}}, (r,c) \models [\![M]\!]$ and $(r,c) \in F^{G,\mathrm{TL}}$, respectively. Hence, (10) holds.

Finally, let $G = (k, m, \mu)$. If $G \models_{\mathrm{TL}} F$, then, by definition, $G, (1,1) \models_{\mathrm{TL}} F$. By (10), we obtain $\nu_G, (1,1) \models_{\mathrm{TL}^\#} F$ and $\nu_G \models_{\mathrm{TL}^\#} F$, respectively. The converse direction can be shown analogously. □

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

**Theorem 4.14.** *Let $n \geq 1$ and $\sigma \geq 2$. There is an $M\Sigma_n(\Gamma)$-definable temporal logic whose satisfiability problem is $n$-EXPSPACE-hard.*

Using standard coding tricks, one can assume $|\Gamma| = 2$.

*Proof.* By Theorem 4.3, there exists an $M\Pi_n^G(\Gamma)$-definable temporal logic TL whose satisfiability problem is $n$-EXPSPACE-hard. It follows from Prop. 4.11 and Lemma 4.13 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 nested word $\nu$ with $\nu \models_{\mathrm{TL}^\#} \mathsf{GRID} \wedge F \wedge \mathsf{WIDTH}$ where

$$\mathsf{WIDTH} = \underbrace{\mathsf{RET}(\mathsf{RET}(\ldots(\mathsf{RET}(}_{2m-1 \text{ times}} \neg \mathsf{RET}(\mathsf{TRUE}))\ldots)\,.$$

Note that the formula $\mathsf{WIDTH}$ ensures that $\nu$ represents a grid of width exactly $m$. The temporal logic $\mathrm{TL}^\#$ can be constructed in linear time. Hence, we polynomially reduced the satisfiability problem of TL to the satisfiability problem of the $M\Pi_n(\Gamma)$-definable temporal logic $\mathrm{TL}^\#$. Theorem 4.14 follows from Remark 2.7. □

## 5 Model Checking

This section deals with the model checking problem: do all runs of a system satisfy a given formula of a fixed temporal logic? As a system model, we consider

$\sigma$-stack automata. There are essentially two approaches of presenting such automata. In [21], 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 with respect to different stacks.

A *$\sigma$-stack automaton* over $\Gamma$ 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, the 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 $j \curvearrowright_s i$. If, on the other hand, $C = \#$, then $i$ should *not* be a return at all.

Let $\nu = ([n], \leq, \lambda, \curvearrowright_1, \ldots, \curvearrowright_\sigma)$ be a nested word. A *run* of $\mathcal{A}$ on $\nu$ is a mapping $\rho\colon P \to Q$ 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

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

(note that $C_i$ is well-defined by the definitions of a nesting relation and a nested word). The run $\rho$ is accepting if $\rho(n) \in F$. The set of nested words for which there is an accepting run is denoted by $L(\mathcal{A})$. The restriction of $L(\mathcal{A})$ to $\tau$-phase words is denoted by $L_\tau(\mathcal{A})$.

Let TL be some MSO($\Gamma$)-definable temporal logic. The *model checking problem* of TL is the set of all triples $(\mathcal{A}, F, \tau)$ where $\mathcal{A}$ is a $\sigma$-stack automaton, $F \in \mathrm{TL}$ is a temporal formula, and $\tau \in \mathbb{N}$ such that every $\tau$-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 $\sigma$-stack automata into tree automata due to La Torre, Madhusudan, and Parlato [21]:

**Theorem 5.1 ([21]).** *From a $\sigma$-stack automaton $\mathcal{A}$ and $\tau \in \mathbb{N}$, one can construct in time $\mathit{tower}_1(|\mathcal{A}| \cdot \mathit{tower}_1(\mathit{poly}(\tau)))$ a tree automaton $\mathcal{B}_{\mathcal{A},\tau}$ such that $L(\mathcal{B}_{\mathcal{A},\tau}) = \mathit{tree}(L_\tau(\mathcal{A}))$.*

Let TL be some fixed $M\Sigma_n$-definable temporal logic. Given a $\sigma$-stack automaton $\mathcal{A}$, a formula $F$ from TL, and $\tau \geq 1$, one can construct the automata $\mathcal{B}_{\neg F}$ and $\mathcal{B}_{\mathcal{A},\tau}$ from the Theorems 3.13 and 5.1, respectively. Note that the negation $\neg$ can be easily defined by an FO-modality (see Example 2.4). Every $\tau$-phase nested

word accepted by $\mathcal{A}$ satisfies $F$ if and only if $L(\mathcal{B}_{\mathcal{A},\tau}) \cap L(\mathcal{B}_{\neg F}) = \emptyset$. The size of both automata is bounded by $\text{tower}_1(|F| \cdot |\mathcal{A}| \cdot \text{tower}_{n+1}(\text{poly}(\tau)))$. Since the emptiness problem of tree automata is solvable in polynomial time, we obtain together with Remark 2.7:

**Theorem 5.2.** *Let $n \geq 0$ and TL be some $BoolM\Sigma_n(\Gamma)$-definable temporal logic. Then the model checking problem of TL is in $(n+2)$-EXPTIME (where $\tau$ is encoded in unary).*

More precisely, it is exponential in the size of the formula $F$ and $(n+2)$-fold exponential in the number of phases $\tau$.

A temporal formula $F$ from TL is not satisfiable if and only if all nested words satisfy $\neg F$. One can easily construct a $\sigma$-stack automaton accepting the set of all nested words in polynomial time. Hence, the next result follows from Theorem 4.14:

**Theorem 5.3.** *Let $n \geq 1$ and $\sigma = 2$. There is an $M\Sigma_n(\Gamma)$-definable temporal logic whose model checking problem is hard for $n$-EXPSPACE.*

## 6 Conclusion

In this paper, we showed that the satisfiability and the model checking problem of bounded-phase multi-stack systems are decidable in time exponential in the size of the formula and $(n+2)$-fold exponential in the number of phases for all $BoolM\Sigma_n$-definable temporal logics. This in particular implies the main result of [8] where the problem is considered for a fixed number of phases. Moreover, we identified for every $n \geq 1$ an $M\Sigma_n$-definable temporal logic for which the problems are $n$-EXPSPACE-hard.

It was shown in [6, 25] for very specific temporal logics (cf. Example 2.5) 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. In [12], it was shown that the model checking problem for multi-stack systems restricted to bounded split-width executions and MSO logic can be solved in non-elementary time. The authors of [12] asked whether temporal logics could be used to gain a reasonable complexity. Recall that the notions of bounded phases, bounded scopes, ordered stacks and split-width are orthogonal. [30] proves that our techniques can be used to show tight upper bounds for *all* MSO-definable temporal logics when restricting to bounded scopes and bounded split-width, the case of ordered stacks remains to be considered.

We find it also interesting to check whether this technique can be adapted to different types of concurrent systems; [17] shows this for Mazurkiewicz traces and [29] for communicating finite-state machines and MSO-definable temporal logics.

In the literature, even pushdown automata communicating via first-in first-out channels were considered. 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) [22, 27, 20]. Also the notion of split-width has been extended to such systems: in [13], optimal decision procedures for the model checking problem of an extension of linear time logic and various variants of propositional dynamic logic were presented. Still it is actually very unclear how to define "canonical" temporal operators such as an until in the setting of recursive message-passing systems. 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*, 4(11):1–44, 2008.
2. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proceedings of TACAS 2004*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer, 2004.
3. R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56:16:1–16:43, 2009.
4. M. F. Atig. Global Model Checking of Ordered Multi-Pushdown Systems. In *Proceedings of FSTTCS 2010*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 216–227, 2010.
5. M. F. Atig, B. Bollig, and P. Habermehl. Emptiness of multi-pushdown automata is 2ETIME-complete. In *Proceedings of DLT 2008*, volume 5257 of *Lecture Notes in Computer Science*, pages 121–133. Springer, 2008.
6. M. F. Atig, A. Bouajjani, K. Narayan Kumar, and P. Saivasan. Linear-time model-checking for multithreaded programs under scope-bounding. In *Proceedings of ATVA'12*, volume 7561, pages 152–166. Springer, October 2012.
7. M. F. Atig, A. Bouajjani, K. Narayan Kumar, and P. Saivasan. Model checking branching-time properties of multi-pushdown systems is hard. *CoRR*, 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*, volume 6907 of *Lecture Notes in Computer Science*, pages 132–144. Springer, 2011.
9. B. Bollig and D. Kuske. An optimal construction of Hanf sentences. *Journal of Applied Logic*, 10(2):179–186, 2012.
10. B. Bollig, D. Kuske, and R. Mennicke. The complexity of model checking multi-stack systems. In *LICS*, pages 163–172. IEEE Computer Society, 2013.
11. L. Breveglieri, A. Cherubini, C. Citrini, and S. Crespi Reghizzi. Multi-push-down languages and grammars. *International Journal of Foundations of Computer Science*, 7(3):253–292, 1996.
12. A. Cyriac, P. Gastin, and K. Narayan Kumar. MSO decidability of multi-pushdown systems via split-width. In *Proceedings of CONCUR 2012*, volume 7454 of *Lecture Notes in Computer Science*, pages 547–561. Springer, 2012.
13. A. Cyriac, P. Gastin, and K. Narayan Kumar. Verifying Communicating Multi-pushdown Systems. Technical report, Laboratoire Spécification et Vérification, ENS Cachan, January 2014.

14. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.
15. R. Fagin, L.J. Stockmeyer, and M. Vardi. On monadic NP vs. monadic co-NP (extended abstract). In *Structure in Complexity Theory Conference*, pages 19–30, 1993.
16. D.M. Gabbay, I. Hodkinson, and M.A. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, vol. 1.* Oxford University Press, 1994.
17. P. Gastin and D. Kuske. Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces. *Information and Computation*, 208(7):797–816, 2010.
18. S. Göller and A. W. Lin. Concurrency Makes Simple Theories Hard. In *Proceedings of STACS'12*, volume 14 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 148–159, 2012.
19. W. Hanf. Model-theoretic methods in the study of elementary logic. In *The Theory of Models*, pages 132–145. North Holland, 1965.
20. A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3:23):1–20, 2012.
21. S. La Torre, P. Madhusudan, and G. Parlato. A robust class of context-sensitive languages. In *Proceedings of LICS'07*, pages 161–170. IEEE Computer Society Press, 2007.
22. S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *Proceedings of TACAS 2008*, Lecture Notes in Computer Science, pages 299–314. Springer, 2008.
23. S. La Torre, P. Madhusudan, and G. Parlato. An infinite automaton characterization of double exponential time. In *Proceedings of CSL'08*, volume 5213 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.
24. S. La Torre and M. Napoli. Reachability of multistack pushdown systems with scope-bounded matching relations. In *Proceedings of CONCUR 2011*, volume 6901 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2011.
25. S. La Torre and M. Napoli. A temporal logic for multi-threaded programs. In *Proceedings of IFIP-TCS'12*, volume 7604 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2012.
26. S. La Torre and G. Parlato. Scope-bounded Multistack Pushdown Systems: Fixed-Point, Sequentialization, and Tree-Width. In *Proceedings of FSTTCS 2012*, volume 18 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 173–184, 2012.
27. P. Madhusudan and Gennaro Parlato. The tree width of auxiliary storage. In *Proceedings of POPL 2011*, pages 283–294. ACM, 2011.
28. O. Matz and W. Thomas. The monadic quantifier alternation hierarchy over graphs is infinite. In *LICS'97*, pages 236–244. IEEE Computer Society Press, 1997.
29. R. Mennicke. Model checking communicating finite-state machines using local temporal logics. Submitted, 2014.
30. R. Mennicke. Model checking concurrent recursive programs using local temporal logics. Submitted, 2014.
31. A. Pnueli. The temporal logic of programs. In *Proceedings of FOCS 1977*, pages 46–57. IEEE, 1977.
32. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *Proceedings of TACAS 2005*, volume 3440 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005.
33. K. Reinhardt. The complexity of translating logic to finite automata. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 231–238. Springer, 2002.

34. I. Walukiewicz. Difficult configurations–on the complexity of LTrL. *Formal Methods in System Design*, 26(1):27–43, 2005.