Extending the WMSO+U Logic with Quantification over Tuples

Anita Badyl
Institute of Informatics, University of Warsaw, Poland

Paweł Parys
Institute of Informatics, University of Warsaw, Poland

Abstract

We study a new extension of the weak MSO logic, talking about boundedness. Instead of a previously considered quantifier U, expressing the fact that there exist arbitrarily large finite sets satisfying a given property, we consider a generalized quantifier U, expressing the fact that there exist tuples of arbitrarily large finite sets satisfying a given property. First, we prove that the new logic WMSO+$\mathsf{U}_{\mathsf{tup}}$ is strictly more expressive than WMSO+U. In particular, WMSO+$\mathsf{U}_{\mathsf{tup}}$ is able to express the so-called simultaneous unboundedness property, for which we prove that it is not expressible in WMSO+U. Second, we prove that it is decidable whether the tree generated by a given higher-order recursion scheme satisfies a given sentence of WMSO+$\mathsf{U}_{\mathsf{tup}}$.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Verification by model checking; Theory of computation → Rewrite systems

Keywords and phrases Boundedness, logic, decidability, expressivity, recursion schemes

Digital Object Identifier 10.4230/LIPIcs.CSL.2024.12

Related Version *Extended Version*: `http://arxiv.org/abs/2311.16607`

Funding Work supported by the National Science Centre, Poland (grant no. 2016/22/E/ST6/00041).

1 Introduction

In the field of logic in computer science, one of the goals is to find logics that, on the one hand, have decidable properties and, on the other hand, are as expressive as possible. An important example of such a logic is the monadic second-order logic, MSO, which defines exactly all regular properties of finite and infinite words [11, 18, 35] and trees [31], and is decidable over these structures.

A natural question that arises is whether MSO can be extended in a decidable way. Particular hopes were connected with expressing boundedness properties. Bojańczyk [3] introduced a logic called MSO+U, which extends MSO with a new quantifier U, with $\mathsf{U}X.\varphi$ saying that the subformula φ holds for arbitrarily large finite sets X. Originally, it was only shown that satisfiability over infinite trees is decidable for formulae where the U quantifier is only used once and not under the scope of set quantification. A significantly more powerful fragment of the logic, albeit for infinite words, was shown decidable by Bojańczyk and Colcombet [6] using automata with counters. These automata were further developed into the theory of cost functions initiated by Colcombet [15].

The difficulty of MSO+U comes from the interaction between the U quantifier and quantification over possibly infinite sets. This motivated the study of WMSO+U, which is a variant of MSO+U where set quantification is restricted to finite sets (the “W” in the name stands for *weak*). On infinite words, satisfiability of WMSO+U is decidable, and the logic has an automaton model [4]. Similar results hold for infinite trees [7], and have been used to decide properties of CTL* [12]. Currently, the strongest decidability result in this line is about WMSO+U over infinite trees extended with quantification over infinite paths [5]. The latter result entails decidability of problems such as the realisability problem for prompt LTL [26], deciding the winner in cost parity games [20], or deciding certain properties of energy games [8].

The results mentioned so far concern mostly the satisfiability problem (is there a model in which a given formula is true?), but arguably the problem more relevant in practice is the model-checking problem: is a given formula satisfied in a given model? In a typical setting, the model represents (possible computations of) some computer system, and the formula expresses some desired property of the system, to be verified. The model is thus usually infinite, although described in a finite way. In this paper, as the class of considered models we choose trees generated by higher-order recursion schemes, which is a very natural and highly expressive choice.

Higher-order recursion schemes (recursion schemes in short) are used to faithfully represent the control flow of programs in languages with higher-order functions [17, 23, 27, 24]. This formalism is equivalent via direct translations to simply-typed λY-calculus [34]. Collapsible pushdown systems [22] and ordered tree-pushdown systems [13] are other equivalent formalisms. Recursion schemes easily cover finite and pushdown systems, but also some other models such as indexed grammars [1] and ordered multi-pushdown automata [9].

A classic result, with several proofs and extensions, says that model-checking trees generated by recursion schemes against MSO formulae is decidable: given a recursion scheme $\mathcal{G}$ and a formula $\varphi \in$ MSO, one can say whether φ holds in the tree generated by $\mathcal{G}$ [27, 22, 25, 32, 10, 33]. When it comes to boundedness properties, one has to first mention decidability of the simultaneous unboundedness property (a.k.a. diagonal property) [21, 14, 30]. In this problem one asks whether, in the tree generated by a given recursion scheme $\mathcal{G}$, there exist branches containing arbitrarily many occurrences of each of the labels $a_1, \dots, a_k$ (i.e., whether for every $n \in \mathbb{N}$ there exists a branch on which every label from $\{a_1, \dots, a_k\}$ occurs at least n times). This result turns out to be interesting, because it entails other decidability results for recursion schemes, concerning in particular computability of the downward closure of recognized languages [36], and the problem of separability by piecewise testable languages [16]. Then, we also have decidability for logics talking about boundedness. Namely, it was shown recently that model-checking for recursion schemes is decidable against formulae from WMSO+U [28] (and even from a mixture of MSO and WMSO+U, where quantification over infinite sets is allowed but cannot be arbitrarily nested with the U quantifier [29]). Another paper [2] shows decidability of model-checking for a subclass of recursion schemes against alternating B-automata and against weak cost monadic second-order logic (WCMSO); these are other formalisms allowing to describe boundedness properties, but in a different style than the U quantifier.

Interestingly, the decidability of model-checking for WMSO+U is obtained by a reduction to (a variant of) the simultaneous unboundedness problem. On the other hand, it seems that the simultaneous unboundedness property cannot be expressed in WMSO+U (except for the case of a single distinguished letter a_1), which is very intriguing.

Our contribution. As a first contribution, we prove the fact that was previously only a hypothesis: WMSO+U is indeed unable to express the simultaneous unboundedness property. Then, we define a new logic, WMSO+$\mathsf{U}_{\mathsf{tup}}$; it is an extension of WMSO+U, where the U quantifier can be used with a tuple of set variables, instead of just one variable. A construct with the extended quantifier, $\mathsf{U}(X_1, \dots, X_k).\varphi$, says that the subformula φ holds for tuples of sets in which each of $X_1, \dots, X_k$ is arbitrarily large. This logic is capable of easily expressing properties in which multiple quantities are simultaneously required to be unbounded. In particular, it can express the simultaneous unboundedness property, and thus it is strictly more expressive than the standard WMSO+U logic:

▶ **Theorem 1.1.** *The* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *logic can express some properties of trees that are not expressible in* WMSO+U*; in particular, this is the case for the simultaneous unboundedness property.*

In fact, to separate the two logics it is enough to consider WMSO+$\mathsf{U}_{\mathsf{tup}}$ only with U quantifiers for pairs of variables (i.e., with $k = 2$). Actually, we are convinced that the proof of Theorem 1.1 contained in this paper can be modified for showing that, for every $k \geq 2$, WMSO+$\mathsf{U}_{\mathsf{tup}}$ without U quantifiers for tuples of length at least k is less expressive than WMSO+$\mathsf{U}_{\mathsf{tup}}$ with such quantifiers (cf.Remark 5.6).

Our main theorem says that the model-checking procedure for WMSO+U can be extended to the new logic:

▶ **Theorem 1.2.** *Given an* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *sentence φ and a recursion scheme $\mathcal{G}$ one can decide whether φ is true in the tree generated by $\mathcal{G}$.*

2 Preliminaries

The powerset of a set X is denoted $\mathcal{P}(X)$. For $i, j \in \mathbb{N}$ we define $[i, j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$. The domain of a function f is denoted $\mathrm{dom}(f)$. When f is a function, by $f[x \mapsto y]$ we mean the function that maps x to y and every other $z \in \mathrm{dom}(f)$ to $f(z)$.

Trees. We consider rooted, potentially infinite trees, where children are ordered. For simplicity of the presentation, we consider only binary trees, where every node has at most two children. This is not really a restriction. Indeed, it is easy to believe that our proofs can be generalized to trees of arbitrary bounded finite arity without any problem (except for notational complications). Alternatively, a tree of arbitrary bounded finite arity can be converted into a binary tree using the first child / next sibling encoding, and a logical formula can be translated as well to a formula talking about the encoding; this means that the WMSO+$\mathsf{U}_{\mathsf{tup}}$ model-checking problem over trees of arbitrary bounded finite arity can be reduced to such a problem over binary trees.

Formally, a *tree domain* (a set of tree nodes) is a set $D \subseteq \{\mathsf{L}, \mathsf{R}\}^*$ that is closed under taking prefixes (i.e., if $uv \in D$ then also $u \in D$). A *tree* over an alphabet $\mathbb{A}$ is a function $T : D \to \mathbb{A}$, for some tree domain D. The set of trees over $\mathbb{A}$ is denoted $\mathcal{T}(\mathbb{A})$. The *subtree* of T starting in a node v is denoted $T{\restriction}v$ and is defined by $(T{\restriction}v)(u) = T(vu)$ (with domain $\{u \in \{\mathsf{L}, \mathsf{R}\}^* \mid vu \in \mathrm{dom}(T)\}$). For nodes we employ the usual notions of child, parent, ancestor, descendant, etc. (where we assume that a node is also an ancestor and a descendant of itself).

For trees T_1, T_2, and for $a \in \mathbb{A}$ we write $a\langle T_1, T_2\rangle$ for the tree T such that $T{\restriction}\mathsf{L} = T_1$, $T{\restriction}\mathsf{R} = T_2$, and $T(\varepsilon) = a$. We also write $\perp$ for the tree with empty domain.

Recursion schemes. Recursion schemes are grammars used to describe some infinite trees in a finitary way. We introduce recursion schemes only by giving an example, rather than by defining them formally. This is enough, because this paper does not work with recursion schemes directly; it only uses some facts concerning them.

A recursion scheme is given by a set of rules, like this:

$$\mathsf{S} \to \mathsf{F}\,\mathsf{G}\,, \qquad \mathsf{D}\,\mathsf{g}\,\mathsf{x} \to \mathsf{g}\,(\mathsf{g}\,\mathsf{x})\,,$$
$$\mathsf{F}\,\mathsf{g} \to \mathsf{a}\langle \mathsf{g}\perp, \mathsf{F}\,(\mathsf{D}\,\mathsf{g})\rangle\,, \qquad \mathsf{G}\,\mathsf{x} \to \mathsf{b}\langle \mathsf{x}, \perp\rangle\,.$$

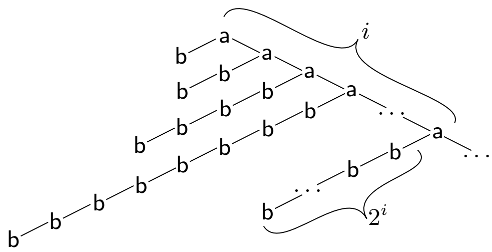

Figure 1 The tree generated by the example recursion scheme.

Here $\mathsf{S}, \mathsf{D}, \mathsf{F}, \mathsf{G}$ are nonterminals, with S being the starting nonterminal, x, g are variables, and a, b are letters from $\mathbb{A}$. To generate a tree, we start with S, which reduces to $\mathsf{F}\,\mathsf{G}$ using the first rule. We now use the rule for F, where the parameter g is instantiated to be G; we obtain $\mathsf{a}\langle \mathsf{G}\,\bot, \mathsf{F}\,(\mathsf{D}\,\mathsf{G})\rangle$. This already defines the root of the tree, which should be a-labeled; its two subtrees should be generated from $\mathsf{G}\,\bot$ and $\mathsf{F}\,(\mathsf{D}\,\mathsf{G})$, respectively. We see that $\mathsf{G}\,\bot$ reduces to $\mathsf{b}\langle\bot,\bot\rangle$, which is a tree with a single b-labeled node. On the other hand, $\mathsf{F}\,(\mathsf{D}\,\mathsf{G})$ reduces to $\mathsf{a}\langle \mathsf{D}\,\mathsf{G}\,\bot, \mathsf{F}\,(\mathsf{D}\,(\mathsf{D}\,\mathsf{G}))\rangle$, which means that the right child of the root is a-labeled, and its left subtree generated from $\mathsf{D}\,\mathsf{G}\,\bot$ (which reduces to $\mathsf{G}\,(\mathsf{G}\,\bot)$, then to $\mathsf{b}\langle \mathsf{G}\bot, \bot\rangle$, and then to $\mathsf{b}\langle \mathsf{b}\langle\bot,\bot\rangle,\bot\rangle$) is a path consisting of two b-labeled nodes. Continuing like this, when going right we always obtain a next a-labeled node (we thus have an infinite a-labeled branch), and to the left of the i-th such node we have a tree generated from $\underbrace{\mathsf{D}\,(\mathsf{D}\,(\ldots(\mathsf{D}}_{i-1}\,\mathsf{G})\ldots))\,\bot$, which is a finite branch consisting of 2^{i-1} b-labeled nodes (note that every D applies its argument twice, and hence doubles the number of produced b-labeled nodes). The resulting tree is depicted on Figure 1.

For a formal definition of recursion schemes consult prior work (e.g., [23, 24, 32, 28]). Some of these papers use a lambda-calculus notation, where our rule for D would be rather written as $\mathsf{D} \to \lambda\mathsf{g}.\lambda\mathsf{x}.\mathsf{g}\,(\mathsf{g}\,\mathsf{x})$. Sometimes it is also allowed to have λ inside a rule, like $\mathsf{S} \to \mathsf{F}\,(\lambda\mathsf{x}.\mathsf{b}\langle\mathsf{x},\bot\rangle)$; this does not make the definition more general, because subterms starting with λ can be always extracted to separate nonterminals.

3 The WMSO+$\mathsf{U}_{\mathsf{tup}}$ logic

In this section we introduce the logic under consideration: the WMSO+$\mathsf{U}_{\mathsf{tup}}$ logic.

Definition. For technical convenience, we use a syntax in which there are no first-order variables. It is easy to translate a formula from a more standard syntax to ours: first-order variables may be simulated by set variables for which we check that they contain exactly one node (i.e., that they are nonempty and that every subset thereof is either empty or equal to the whole set).

We assume an infinite set $\mathcal{V}$ of variables, which can be used to quantify over finite sets of tree nodes. In order to distinguish variables from sets to which these variables are valuated, we denote variables using Sans Serif font (e.g., $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$). In the syntax of WMSO+$\mathsf{U}_{\mathsf{tup}}$ we have the following constructions:

$$\varphi ::= a(\mathsf{X}) \mid \mathsf{X} \curlywedge_d \mathsf{Y} \mid \mathsf{X} \subseteq \mathsf{Y} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi' \mid \exists_{\mathsf{fin}}\mathsf{X}.\varphi' \mid \mathsf{U}(\mathsf{X}_1,\ldots,\mathsf{X}_k).\varphi'\,,$$

where $a \in \mathbb{A}$, $d \in \{\mathsf{L},\mathsf{R}\}$, $k \in \mathbb{N}$, and $\mathsf{X}, \mathsf{Y}, \mathsf{X}_1, \ldots, \mathsf{X}_k \in \mathcal{V}$. Free variables of a formula are defined as usual; in particular $\mathsf{U}(\mathsf{X}_1,\ldots,\mathsf{X}_k)$ is a quantifier that bounds the variables $\mathsf{X}_1,\ldots,\mathsf{X}_k$.

We evaluate formulae of WMSO+$\mathsf{U_{tup}}$ in $\mathbb{A}$-labeled trees. In order to evaluate a formula φ in a tree T, we also need a *valuation*, that is, a function ν from $\mathcal{V}$ to finite sets of nodes of T (its values are meaningful only for free variables of φ). The semantics of formulae is defined as follows:

- $a(\mathsf{X})$ holds when every node in $\nu(\mathsf{X})$ is labeled with a,
- $\mathsf{X} \downarrow_d \mathsf{Y}$ holds when both $\nu(\mathsf{X})$ and $\nu(\mathsf{Y})$ are singletons, and the unique node in $\nu(\mathsf{Y})$ is the left (if $d = \mathsf{L}$) / right (if $d = \mathsf{R}$) child of the unique node in $\nu(\mathsf{X})$,
- $\mathsf{X} \subseteq \mathsf{Y}$ holds when $\nu(\mathsf{X}) \subseteq \nu(\mathsf{Y})$,
- $\varphi_1 \wedge \varphi_2$ holds when both φ_1 and φ_2 hold,
- $\neg\varphi'$ holds when φ' does not hold,
- $\exists_{\mathsf{fin}}\mathsf{X}.\varphi'$ holds if there exists a finite set X of nodes of T for which φ' holds under the valuation $\nu[\mathsf{X} \mapsto X]$, and
- $\mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\varphi'$ holds if for every $n \in \mathbb{N}$ there exist finite sets $X_1, \dots, X_k$ of nodes of T, each of cardinality at least n, such that φ' holds under the valuation $\nu[\mathsf{X}_1 \mapsto X_1, \dots, \mathsf{X}_k \mapsto X_k]$.

We write $T, \nu \models \varphi$ to denote that φ holds in T under the valuation ν.

Logical types. In proofs of both our results, Theorem 1.1 and Theorem 1.2, we use logical types, which we now define.

Let φ be a formula of WMSO+$\mathsf{U_{tup}}$, let T be a tree, and let ν be a valuation. We define the φ*-type* of T under valuation ν, denoted $[\![T]\!]^\nu_\varphi$, by induction on the size of φ as follows:

- if φ is of the form $a(\mathsf{X})$ (for some letter $a \in \mathbb{A}$) or $\mathsf{X} \subseteq \mathsf{Y}$ then $[\![T]\!]^\nu_\varphi$ is the logical value of φ in T, ν, that is, tt if $T, \nu \models \varphi$ and ff otherwise,
- if φ is of the form $\mathsf{X} \downarrow_d \mathsf{Y}$, then $[\![T]\!]^\nu_\varphi$ equals:
  - tt if $T, \nu \models \varphi$,
  - empty if $\nu(\mathsf{X}) = \nu(\mathsf{Y}) = \emptyset$,
  - root if $\nu(\mathsf{X}) = \emptyset$ and $\nu(Y) = \{\varepsilon\}$, and
  - ff otherwise,
- if $\varphi = (\psi_1 \wedge \psi_2)$, then $[\![T]\!]^\nu_\varphi = ([\![T]\!]^\nu_{\psi_1}, [\![T]\!]^\nu_{\psi_2})$,
- if $\varphi = (\neg\psi)$, then $[\![T]\!]^\nu_\varphi = [\![T]\!]^\nu_\psi$,
- if $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$, then

  $$[\![T]\!]^\nu_\varphi = \{\sigma \mid \exists X.\, [\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma\}\,,$$

  where X ranges over finite sets of nodes of T, and
- if $\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\psi$, then

  $$[\![T]\!]^\nu_\varphi = \big(\{\sigma \mid \forall n \in \mathbb{N}.\, \exists X_1.\cdots.\exists X_k.\, [\![T]\!]^{\nu[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]}_\psi = \sigma \\ \wedge \forall i \in I.\, |X_i| \geq n\}\big)_{I\subseteq[1,k]}\,,$$

  where $X_1, \dots, X_k$ range over finite sets of nodes of T (the above φ-type is a tuple of 2^k sets, indexed by subsets I of $[1, k]$).

For each φ, let Typ_φ denote the set of all potential φ-types. Namely, $\mathit{Typ}_\varphi = \{\mathsf{tt}, \mathsf{ff}\}$ if $\varphi = a(\mathsf{X})$ or $\varphi = (\mathsf{X} \subseteq \mathsf{Y})$, $\mathit{Typ}_\varphi = \{\mathsf{tt}, \mathsf{empty}, \mathsf{root}, \mathsf{ff}\}$ if $\varphi = \mathsf{X} \downarrow_d \mathsf{Y}$, $\mathit{Typ}_\varphi = \mathit{Typ}_{\psi_1} \times \mathit{Typ}_{\psi_2}$ if $\varphi = (\psi_1 \wedge \psi_2)$, $\mathit{Typ}_\varphi = \mathit{Typ}_\psi$ if $\varphi = (\neg\psi)$; $\mathit{Typ}_\varphi = \mathcal{P}(\mathit{Typ}_\psi)$ if $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$, and $\mathit{Typ}_\varphi = (\mathcal{P}(\mathit{Typ}_\psi))^{2^k}$ if $\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\psi$.

The following two facts can be shown by a straightforward induction on the structure of a considered formula:

▶ **Fact 3.1.** *For every* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula* φ *the set* Typ_φ *is finite.*

The second fact says that whether or not φ holds in T,ν is determined by $[\![T]\!]^\nu_\varphi$:

▶ **Fact 3.2.** *For every* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula* φ *there is a computable function* $\mathit{tv}_\varphi\colon \mathit{Typ}_\varphi \to \{\mathsf{tt},\mathsf{ff}\}$ *such that for every tree* $T \in \mathcal{T}(\mathbb{A})$ *and every valuation* ν *in* T*, it holds that* $\mathit{tv}_\varphi([\![T]\!]^\nu_\varphi) = \mathsf{tt}$ *if, and only if,* $T,\nu \models \varphi$.

Next, we observe that types behave in a compositional way, as formalized below. Here, for a node w we write $X{\restriction}w$ and $\nu{\restriction}w$ to denote the restriction of a set X and of a valuation ν to the subtree starting at w; formally, $X{\restriction}w = \{u \mid wu \in X\}$ and $\nu{\restriction}w$ maps every variable $\mathsf{X} \in \mathcal{V}$ to $\nu(\mathsf{X}){\restriction}w$.

▶ **Proposition 3.3.** *For every letter* $a \in \mathbb{A}$ *and every formula* φ*, one can compute a function* $\mathit{Comp}_{a,\varphi}\colon \mathcal{P}(\mathcal{V}) \times \mathit{Typ}_\varphi \times \mathit{Typ}_\varphi \to \mathit{Typ}_\varphi$ *such that for every tree* T *whose root has label* a *and for every valuation* ν*,*

$$[\![T]\!]^\nu_\varphi = \mathit{Comp}_{a,\varphi}(\{\mathsf{X} \mid \varepsilon \in \nu(\mathsf{X})\}, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)\,. \tag{1}$$

We remark that *a priori* the first argument of $\mathit{Comp}_{a,\varphi}$ is an arbitrary subset of $\mathcal{V}$, but in fact we only need to know which free variables of φ it contains; in consequence, $\mathit{Comp}_{a,\varphi}$ can be seen as a finite object.

Proof of Proposition 3.3. We proceed by induction on the size of φ.

When φ is of the form $b(\mathsf{X})$ or $\mathsf{X} \subseteq \mathsf{Y}$, then we see that φ holds in T,ν if, and only if, it holds in the subtrees $T{\restriction}\mathsf{L}, \nu{\restriction}\mathsf{L}$ and $T{\restriction}\mathsf{R}, \nu{\restriction}\mathsf{R}$, and in the root of T. Thus for $\varphi = b(\mathsf{X})$ as $\mathit{Comp}_{a,\varphi}(S,\tau_\mathsf{L},\tau_\mathsf{R})$ we take tt when $\tau_\mathsf{L} = \tau_\mathsf{R} = \mathsf{tt}$ and either $a = b$ or $\mathsf{X} \notin S$. For $\varphi = (\mathsf{X} \subseteq \mathsf{Y})$ the last part of the condition is replaced by "if $\mathsf{X} \in S$ then $\mathsf{Y} \in S$".

Next, suppose that $\varphi = (\mathsf{X} \swarrow_d \mathsf{Y})$. Then as $\mathit{Comp}_{a,\varphi}(S,\tau_\mathsf{L},\tau_\mathsf{R})$ we take

- tt if $\mathsf{X} \notin S$, $\mathsf{Y} \notin S$, and either $\tau_\mathsf{L} = \mathsf{tt}$ and $\tau_\mathsf{R} = \mathsf{empty}$ or $\tau_\mathsf{L} = \mathsf{empty}$ and $\tau_\mathsf{R} = \mathsf{tt}$,
- tt also if $\mathsf{X} \in S$, $\mathsf{Y} \notin S$, $\tau_d = \mathsf{root}$, and $\tau_i = \mathsf{empty}$ for the direction i other than d,
- empty if $\mathsf{X} \notin S$, $\mathsf{Y} \notin S$, and $\tau_L = \tau_S = \mathsf{empty}$,
- root if $\mathsf{X} \notin S$, $\mathsf{Y} \in S$, and $\tau_L = \tau_S = \mathsf{empty}$, and
- ff otherwise.

By comparing this definition with the definition of the type we immediately see that Equality (1) is satisfied.

When $\varphi = (\neg\psi)$, we simply take $\mathit{Comp}_{a,\varphi} = \mathit{Comp}_{a,\psi}$, and when $\varphi = (\psi_1 \wedge \psi_2)$, as $\mathit{Comp}_{a,\varphi}(S,(\tau^1_\mathsf{L},\tau^2_\mathsf{L}),(\tau^1_\mathsf{R},\tau^2_\mathsf{R}))$ we take the pair of $\mathit{Comp}_{a,\psi_i}(S,\tau^i_\mathsf{L},\tau^i_\mathsf{R})$ for $i \in \{1,2\}$.

Suppose now that $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$. We define $\mathit{Comp}_{a,\varphi}(S,\tau_\mathsf{L},\tau_\mathsf{R})$ to be

$$\{\mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R}) \mid S \setminus \{\mathsf{X}\} \subseteq S' \subseteq S \cup \{\mathsf{X}\}, \sigma_\mathsf{L} \in \tau_\mathsf{L}, \sigma_\mathsf{R} \in \tau_\mathsf{R}\}\,.$$

Let us check Equality (1) in details. Denote $S = \{\mathsf{Y} \mid \varepsilon \in \nu(\mathsf{Y})\}$. In order to show the left-to-right inclusion recall that, by definition, $[\![T]\!]^\nu_\varphi$ is a set of ψ-types, whose every element is of the form $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi$ for some finite set of nodes X. For every such X by the induction hypothesis we have $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi = \mathit{Comp}_{a,\psi}(S', [\![T{\restriction}\mathsf{L}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{R}}_\varphi)$, where $S' = S \cup \{\mathsf{X}\}$ if $\varepsilon \in X$ and $S' = S \setminus \{\mathsf{X}\}$ if $\varepsilon \notin X$; moreover $[\![T{\restriction}\mathsf{L}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{L}}_\psi \in [\![T{\restriction}L]\!]^{\nu{\restriction}\mathsf{L}}_\varphi$ and $[\![T{\restriction}\mathsf{R}]\!]^{\nu[\mathsf{X}\mapsto X]{\restriction}\mathsf{R}}_\psi \in [\![T{\restriction}L]\!]^{\nu{\restriction}\mathsf{R}}_\varphi$, which implies that $[\![T]\!]^{\nu[\mathsf{X}\mapsto X]}_\psi \in \mathit{Comp}_{a,\varphi}(S, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)$, as required. For the opposite inclusion take some $\sigma \in \mathit{Comp}_{a,\varphi}(S, [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi, [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi)$; it is of the form $\mathit{Comp}_{a,\psi}(S',\sigma_\mathsf{L},\sigma_\mathsf{R})$ for some $\sigma_\mathsf{L} \in [\![T{\restriction}\mathsf{L}]\!]^{\nu{\restriction}\mathsf{L}}_\varphi$ and $\sigma_\mathsf{R} \in [\![T{\restriction}\mathsf{R}]\!]^{\nu{\restriction}\mathsf{R}}_\varphi$, where S' is either $S \cup \{\mathsf{X}\}$ or $S \setminus \{\mathsf{X}\}$. Then, by definition, σ_L and σ_R are of the form $[\![T{\restriction}\mathsf{L}]\!]^{(\nu{\restriction}\mathsf{L})[\mathsf{X}\mapsto X_\mathsf{L}]}_\psi$

and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{(\nu\restriction\mathsf{R})[\mathsf{X}\mapsto X_\mathsf{R}]}$, respectively, for some finite sets of nodes X_L and X_R. We now take X such that $X{\restriction}\mathsf{L} = X_\mathsf{L}$ and $X{\restriction}\mathsf{R} = X_\mathsf{R}$, and $\varepsilon \in X$ if, and only if, $S' = S \cup \{\mathsf{X}\}$; we have $(\nu{\restriction}\mathsf{L})[\mathsf{X} \mapsto X_\mathsf{L}] = \nu[\mathsf{X} \mapsto X]{\restriction}\mathsf{L}$ and $(\nu{\restriction}\mathsf{R})[\mathsf{X} \mapsto X_\mathsf{R}] = \nu[\mathsf{X} \mapsto X]{\restriction}\mathsf{R}$. By the induction hypothesis we then have $\sigma = [\![T]\!]_\psi^{\nu[\mathsf{X}\mapsto X]}$, which by definition is an element of $[\![T]\!]_\varphi^\nu$, as required.

Finally, suppose that $\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\psi$. For $\tau_\mathsf{L} = (\rho_{\mathsf{L},I})_{I\subseteq[1,k]}$ and $\tau_\mathsf{R} = (\rho_{\mathsf{R},I})_{I\subseteq[1,k]}$ we define $\mathit{Comp}_{a,\varphi}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ to be $(\rho_I)_{I\subseteq[1,k]}$, where

$$\rho_I = \{\mathit{Comp}_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R}) \mid S \setminus \{\mathsf{X}_1, \dots, \mathsf{X}_k\} \subseteq S' \subseteq S \cup \{\mathsf{X}_1, \dots, \mathsf{X}_k\}, \\ \sigma_\mathsf{L} \in \rho_{\mathsf{L},I_\mathsf{L}}, \sigma_\mathsf{R} \in \rho_{\mathsf{R},I_\mathsf{R}}, I_\mathsf{L} \cup I_\mathsf{R} = I\}\,.$$

In order to check Equality (1), denote $[\![T]\!]_\varphi^\nu = (\rho'_I)_{I\subseteq[1,k]}$, $[\![T{\restriction}\mathsf{L}]\!]_\varphi^{\nu\restriction\mathsf{L}} = (\rho_{\mathsf{L},I})_{I\subseteq[1,k]}$, $[\![T{\restriction}\mathsf{R}]\!]_\varphi^{\nu\restriction\mathsf{R}} = (\rho_{\mathsf{R},I})_{I\subseteq[1,k]}$, and $S = \{\mathsf{Y} \mid \varepsilon \in \nu(\mathsf{Y})\}$; we then have to prove that $\rho'_I = \rho_I$ for all $I \subseteq [1,k]$ (where ρ_I is as defined above).

For the left-to-right inclusion, take some $\sigma \in \rho'_I$. By definition, it is a ψ-type such that for every $n \in \mathbb{N}$ there exist finite sets $X_{n,1}, \dots, X_{n,k}$ for which $[\![T]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]} = \sigma$, where the cardinality of the sets $X_{n,i}$ with $i \in I$ is at least n. To every n let us assign the following information, called *characteristic*, and consisting of $2k$ bits and 2 ψ-types:

- for every $i \in [1,k]$, does the root ε belong to $X_{n,i}$?
- for every $i \in [1,k]$, is $X_{n,i}{\restriction}\mathsf{L}$ larger than $X_{n,i}{\restriction}\mathsf{R}$?
- the ψ-types $[\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]\restriction\mathsf{L}}$ and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]\restriction\mathsf{R}}$.

Because there are only finitely many possible characteristics, by the pigeonhole principle we may find an infinite set $G \subseteq \mathbb{N}$ of indices n such that the same characteristic is assigned to every $n \in G$. We then take

$$S' = S \setminus \{\mathsf{X}_1, \dots, \mathsf{X}_k\} \cup \{\mathsf{X}_i \mid \varepsilon \in X_{n,i} \text{ for } n \in G\}\,,$$
$$I_\mathsf{L} = \{i \in I \mid |X_{n,i}{\restriction}\mathsf{L}| > |X_{n,i}{\restriction}\mathsf{R}| \text{ for } n \in G\}\,, \qquad I_\mathsf{R} = I \setminus I_\mathsf{L}\,,$$
$$\sigma_\mathsf{L} = [\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]\restriction\mathsf{L}}\,, \qquad \sigma_\mathsf{R} = [\![T{\restriction}\mathsf{R}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]\restriction\mathsf{R}} \qquad \text{for } n \in G.$$

The induction hypothesis (used with the valuation $\nu[\mathsf{X}_1 \mapsto X_{n,1}, \dots, \mathsf{X}_k \mapsto X_{n,k}]$ for any $n \in G$) gives us $\sigma = \mathit{Comp}_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R})$. For every $m \in \mathbb{N}$ we can find $n \in G$ such that $n \geq 2m+1$; then $[\![T{\restriction}\mathsf{L}]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_{n,1},\dots,\mathsf{X}_k\mapsto X_{n,k}]\restriction\mathsf{L}} = \sigma_\mathsf{L}$ and $|X_{n,i}{\restriction}\mathsf{L}| \geq m$ for all $i \in I_\mathsf{L}$ ($X_{n,i}$ has at least $2m+1$ elements because $i \in I$, one of them may be the root, and at least half of the other elements is in the left subtree by definition of I_L). This implies that $\sigma_\mathsf{L} \in \rho_{\mathsf{L},I}$, by definition of the φ-type. Likewise $\sigma_\mathsf{R} \in \rho_{\mathsf{R},I}$. By definition of ρ_I this gives us $\sigma \in \rho_I$ as required.

The right-to-left inclusion is completely straightforward. Indeed, take some $\sigma \in \rho_I$. The definition of ρ_I gives us a set S' such that $S \setminus \{\mathsf{X}_1, \dots, \mathsf{X}_k\} \subseteq S' \subseteq S \cup \{\mathsf{X}_1, \dots, \mathsf{X}_k\}$, sets $I_\mathsf{L}, I_\mathsf{R} \subseteq [1,k]$ such that $I_\mathsf{L} \cup I_\mathsf{R} = I$, and types $\sigma_\mathsf{L} \in \rho_{\mathsf{L},I_\mathsf{L}}$ and $\sigma_\mathsf{R} \in \rho_{\mathsf{R},I_\mathsf{R}}$ such that $\sigma = \mathit{Comp}_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R})$. By definition of the two ψ-types, σ_L and σ_R, for every n there are sets $X_{\mathsf{L},1}, \dots, X_{\mathsf{L},k}$ and $X_{\mathsf{R},1}, \dots, X_{\mathsf{R},k}$ such that $[\![T{\restriction}\mathsf{L}]\!]_\psi^{(\nu\restriction\mathsf{L})[\mathsf{X}_1\mapsto X_{\mathsf{L},1},\dots,\mathsf{X}_k\mapsto X_{\mathsf{L},k}]} = \sigma_\mathsf{L}$, and $[\![T{\restriction}\mathsf{R}]\!]_\psi^{(\nu\restriction\mathsf{R})[\mathsf{X}_1\mapsto X_{\mathsf{R},1},\dots,\mathsf{X}_k\mapsto X_{\mathsf{R},k}]} = \sigma_\mathsf{R}$, and $|X_{\mathsf{L},i}| \geq n$ for all $i \in I_\mathsf{L}$, and $|X_{\mathsf{R},i}| \geq n$ for all $i \in I_\mathsf{R}$. We now take $X_1, \dots, X_k$ such that $X_i{\restriction}\mathsf{L} = X_{\mathsf{L},i}$, and $X_i{\restriction}\mathsf{R} = X_{\mathsf{R},i}$, and $\varepsilon \in X_i$ if, and only if, $\mathsf{X}_i \in S'$, for all $i \in [1,k]$. By the induction hypothesis we then have $[\![T]\!]_\psi^{\nu[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]} = \mathit{Comp}_{a,\psi}(S', \sigma_\mathsf{L}, \sigma_\mathsf{R}) = \sigma$. Because additionally $|X_i| \geq n$ for all $i \in I = I_\mathsf{L} \cup I_\mathsf{R}$, we obtain that $\sigma \in \rho'_I$, as required. ◀

Figure 2 An example tree T (left), and the corresponding tree $\mathsf{refl}_\varphi(T)$ (right) obtained for an MSO sentence φ saying "the right child of the root has label a".

The next fact says that one can find a type of the empty tree. In the empty tree, a valuation has to map every variable to the empty set; we denote such a valuation by $\varnothing$. This fact is trivial: we simply follow the definition of $[\![\bot]\!]^{\varnothing}_{\varphi}$.

▶ **Fact 3.4.** *For ever formula φ, one can compute $[\![\bot]\!]^{\varnothing}_{\varphi}$.*

4 Decidability of model-checking

In this section we show how to evaluate WMSO+U$_{\mathsf{tup}}$ formulae over trees generated by recursion schemes, that is, we prove Theorem 1.2. To this end, we first introduce three kinds of operations on recursion schemes, known to be computable. Then, we show how a sequence of these operations can be used to evaluate our formulae.

MSO reflection. The property of logical reflection for recursion schemes comes from Broadbent, Carayol, Ong, and Serre [10]. They state it for sentences of μ-calculus, but μ-calculus and MSO are equivalent over infinite trees [19].

Consider a tree T, and an MSO sentence φ (we skip a formal definition of MSO, assuming that it is standard). We define $\mathsf{refl}_\varphi(T)$ to be the tree having the same domain as T, and such that every node u thereof is labeled by the pair (a_u, b_u), where a_u is the label of u in T, and b_u is tt if φ is satisfied in $T{\restriction}u$ and ff otherwise. In other words, $\mathsf{refl}_\varphi(T)$ adds, in every node of T, a mark saying whether φ holds in the subtree starting in that node. Consult Figure 2 for an example.

▶ **Theorem 4.1** (MSO reflection [10, Theorem 7.3(2)])**.** *Given a recursion scheme $\mathcal{G}$ generating a tree T, and an* MSO *sentence φ, one can construct a recursion scheme $\mathcal{G}_\varphi$ generating the tree $\mathsf{refl}_\varphi(T)$.*

SUP reflection. The SUP reflection is the heart of our proof (where "SUP" stands for simultaneous unboundedness property). In order to talk about this property, we need a few more definitions. By $\#_a(V)$ we denote the number of a-labeled nodes in a (finite) tree V. For a set of (finite) trees $\mathcal{L}$ and a set of letters A, we define a predicate $SUP_A(\mathcal{L})$, which holds if for every $n \in \mathbb{N}$ there is some $V_n \in \mathcal{L}$ such that for all $a \in A$ it holds that $\#_a(V_n) \geq n$.

Originally, in the simultaneous unboundedness property we consider devices recognizing a set of finite trees, unlike recursion schemes, which generate a single infinite tree. We use here an equivalent formulation, in which the set of finite trees is encoded in a single infinite tree. To this end, we use two special letters: nd, denoting a nondeterministic choice (disjunction between two children), and $\mathsf{nd}_\bot$, denoting that there is no choice (empty disjunction). We write $T \to_{\mathsf{nd}} V$ if V is obtained from T by choosing some nd-labeled node u and some its child v, and attaching $T{\restriction}v$ in place of $T{\restriction}u$. In other words, $\to_{\mathsf{nd}}$ is the smallest relation such that $\mathsf{nd}\langle T_{\mathsf{L}}, T_{\mathsf{R}}\rangle \to_{\mathsf{nd}} T_d$ for $d \in \{\mathsf{L}, \mathsf{R}\}$, and $a\langle T_{\mathsf{L}}, T_{\mathsf{R}}\rangle \to_{\mathsf{nd}} a\langle T'_{\mathsf{L}}, T_{\mathsf{R}}\rangle$ if $T_{\mathsf{L}} \to_{\mathsf{nd}} T'_{\mathsf{L}}$, and $a\langle T_{\mathsf{L}}, T_{\mathsf{R}}\rangle \to_{\mathsf{nd}} a\langle T_{\mathsf{L}}, T'_{\mathsf{R}}\rangle$ if $T_{\mathsf{R}} \to_{\mathsf{nd}} T'_{\mathsf{R}}$. For a tree T, $\mathcal{L}(T)$ is the set of all finite trees V such

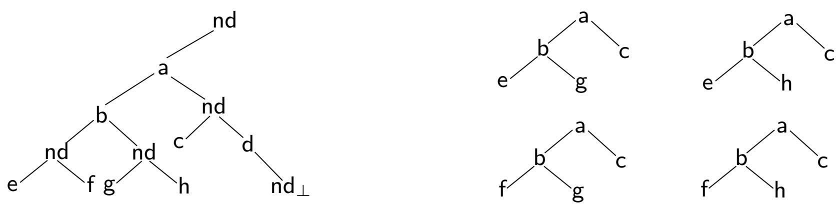

Figure 3 An example tree T, and four trees in $\mathcal{L}(T)$ (right). Additionally, $\mathcal{L}(T)$ contains $\perp$, the tree with empty domain, obtained by choosing the right child in the topmost nd-labeled node. Note that no tree in $\mathcal{L}(T)$ contains d, because d in T is followed by $\mathsf{nd}_\perp$, which is forbidden in trees in $\mathcal{L}(T)$.

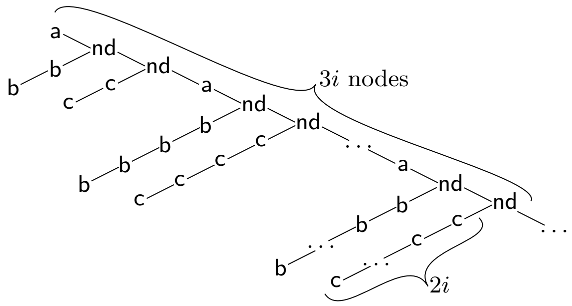

Figure 4 A tree T illustrating SUP reflection.

that $\#_{\mathsf{nd}}(V) = \#_{\mathsf{nd}_\perp}(V) = 0$ and $T \to^*_{\mathsf{nd}} V$. See Figure 3 for an example. We then say that T satisfies the *simultaneous unboundedness property with respect to* a set of letters A if $SUP_A(\mathcal{L}(T))$ holds, that is, if for every $n \in \mathbb{N}$ there are trees in $\mathcal{L}(T)$ having at least n occurrences of every letter from A.

Let T be a tree over an alphabet $\mathbb{A}$. We define $\mathsf{refl}_{SUP}(T)$ to be the tree having the same domain as T, and such that every node u thereof, having in T label a_u, is labeled by

- the pair $(a_u, \{A \subseteq \mathbb{A} \mid SUP_A(\mathcal{L}(T{\restriction}u))\})$, if $a_u \not\in \{\mathsf{nd}, \mathsf{nd}_\perp\}$, and
- the original letter a_u, if $a_u \in \{\mathsf{nd}, \mathsf{nd}_\perp\}$.

In other words, $\mathsf{refl}_{SUP}(T)$ adds, in every node u of T (except for nd- and $\mathsf{nd}_\perp$-labeled nodes) and for every set A of letters, a mark saying whether $T{\restriction}u$ has the simultaneous unboundedness property with respect to A.

Consider, for example, the tree T from Figure 4. The tree $\mathsf{refl}_{SUP}(T)$ has the same shape as T. Every node u having label a in T gets label $(\mathsf{a}, \{\emptyset, \{\mathsf{a}\}, \{\mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{a}, \mathsf{c}\}\})$. Note that the set does not contain $\{\mathsf{b}, \mathsf{c}\}$ nor $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$: in $\mathcal{L}(T{\restriction}u)$ there are no trees having simultaneously many occurrences of b and many occurrences of c. Nodes u having in T label b or c are simply relabeled to $(\mathsf{b}, \{\emptyset\})$ or $(\mathsf{c}, \{\emptyset\})$, respectively, because $\mathcal{L}(T{\restriction}u)$ contains only a single tree, with a fixed number of nodes.

▶ **Theorem 4.2** (SUP reflection [30, Theorem 10.1])**.** *Given a recursion scheme $\mathcal{G}$ generating a tree T, one can construct a recursion scheme $\mathcal{G}_{SUP}$ generating the tree* $\mathsf{refl}_{SUP}(T)$*.*

▶ Remark 4.3. In the introduction we have described an easier variant of the simultaneous unboundedness property, called a *word variant*. In this variant, every node with label other than nd has at most one child; then choosing a tree in $\mathcal{L}(T)$ corresponds to choosing a branch of T (and trees in $\mathcal{L}(T)$ consist of single branches, hence they can be seen as words). Although the word variant of SUP is more commonly known than the *tree variant* described in this section, Theorem 4.2 holds also for the more general tree variant, as presented above.

Transducers. A *(deterministic, top-down) finite tree transducer* is a tuple $\mathcal{F} = (\mathbb{A}, \mathbb{B}, Q, q_0, \delta)$, where $\mathbb{A}$ is a finite input alphabet, $\mathbb{B}$ is a finite output alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, and δ is a transition function mapping $Q \times (\mathbb{A} \cup \{\perp\})$ to finite trees over the alphabet $\mathbb{B} \cup (Q \times \{\mathsf{L}, \mathsf{R}\})$. Letters from $Q \times \{\mathsf{L}, \mathsf{R}\}$ are allowed to occur only in leaves of trees $\delta(q, a)$ with $a \in \mathbb{A}$ (internal nodes of these trees, and all nodes of trees $\delta(q, \perp)$ are labeled by letters from $\mathbb{B}$). Moreover, it is assumed that that there is no sequence $(q_1, a_1, d_1), (q_2, a_2, d_2), \ldots, (q_n, a_n, d_n)$ such that $\delta(q_i, a_i) = (q_{(i \bmod n)+1}, d_i)\langle \perp, \perp \rangle$ for all $i \in [1, n]$.

For an input tree T over $\mathbb{A}$ and a state $q \in Q$, we define an output tree $\mathcal{F}_q(T)$ over $\mathbb{B}$. Namely $\mathcal{F}_q(a\langle T_\mathsf{L}, T_\mathsf{R}\rangle)$ is the tree obtained from $\delta(q, a)$ by substituting $\mathcal{F}_r(T_d)$ for every leaf labeled with $(r, d) \in Q \times \{\mathsf{L}, \mathsf{R}\}$; additionally, $\mathcal{F}_q(\perp)$ simply equals $\delta(q, \perp)$ (recall that this tree has no labels from $Q \times \{\mathsf{L}, \mathsf{R}\}$). In other words, while being in state q over an a-labeled node of the input tree, the transducer produces a tree prefix specified by $\delta(q, a)$, where instead of outputting an (r, L)-labeled (or (r, R)-labeled) leaf, it rather continues by going to the left (respectively, right) child in the input tree, in state r; when $\mathcal{F}$ leaves the domain of the input tree, it still has a chance to output something, namely $\delta(q, \perp)$, and then it stops. In the root we start from the initial state, that is, we define $\mathcal{F}(T) = \mathcal{F}_{q_0}(T)$. To make the above definition formal, we can define $\mathcal{F}_q(T)(v)$, the label of $\mathcal{F}_q(T)$ in a node $v \in \{\mathsf{L}, \mathsf{R}\}^k$, by induction on the depth k, simultaneously for all input trees T and states $q \in Q$. Transitions $\delta(q, a)$ with (r, d) immediately in the root are a bit problematic, because we go down along the input tree without producing anything in the output tree; we have assumed, however, that such transitions do not form a cycle, so after a few (at most $|Q|$) steps we necessarily advance in the output tree.

Note that transducers need not be linear. For example, we may have $\delta(q, a) = a\langle a\langle (q, \mathsf{L}), (q, \mathsf{L})\rangle, a\langle (q, \mathsf{R}), (q, \mathsf{R})\rangle\rangle$, which creates two copies of the tree produced out of the left subtree, and two copies of the tree produced out of the right subtree.

We have the following theorem:

▶ **Theorem 4.4.** *Given a finite tree transducer $\mathcal{F} = (\mathbb{A}, \mathbb{B}, Q, q_0, \delta)$ and a recursion scheme $\mathcal{G}$ generating a tree T over the alphabet $\mathbb{A}$, one can construct a recursion scheme $\mathcal{G}_\mathcal{F}$ generating the tree $\mathcal{F}(T)$.*

This theorem follows from the equivalence between recursion schemes and collapsible pushdown systems [22], as it is straightforward to compose a collapsible pushdown system with $\mathcal{F}$. A formal proof can be found for instance in Parys [30, Appendix A].

Sequences of operations. We consider sequences of operations of the form $O_1, O_2, \ldots, O_n$, where every O_i is either an MSO sentence φ, or the string "*SUP*", or a finite tree transducer $\mathcal{F}$. Having a tree T, we can apply such a sequence of operations to it. Namely, we take $T_0 = T$, and for every $i \in [1, n]$, as T_i we take

- $\mathsf{refl}_\varphi(T_{i-1})$ if $O_i = \varphi$ is an MSO sentence,
- $\mathsf{refl}_{SUP}(T_{i-1})$ if $O_i = SUP$, and
- $\mathcal{F}(T_{i-1})$ if $O_i = \mathcal{F}$ is a finite tree transducer.

As the result we take T_n. We implicitly assume that whenever we apply a finite tree transducer to some tree, then the tree is over the input alphabet of the transducer; likewise, we assume that while computing $\mathsf{refl}_\varphi(T_{i-1})$, the formula uses letters from the alphabet of T_{i-1}.

Using the aforementioned closure properties (Theorems 4.1, 4.2, and 4.4) we can apply the operations on the level of recursion schemes generating our tree:

▶ **Proposition 4.5.** *Given a recursion scheme $\mathcal{G}$ generating a tree T, and a sequence of operations $O_1, O_2, \ldots, O_n$ as above, one can construct a recursion scheme $\mathcal{G}'$ generating the result of applying $O_1, O_2, \ldots, O_n$ to T.*

Main theorem. Let $\mathbb{A}$ be the alphabet used by WMSO+$\mathsf{U}_{\mathsf{tup}}$ formulae under consideration. We prove the following theorem:

▶ **Theorem 4.6.** *Given a* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *sentence φ, one can compute a sequence of operations $O_1, O_2, \ldots, O_n$, such that for every tree T over $\mathbb{A}$, by applying $O_1, O_2, \ldots, O_n$ to T we obtain $\mathsf{tt}\langle\perp,\perp\rangle$ if φ is true in T, and $\mathsf{ff}\langle\perp,\perp\rangle$ otherwise.*

Having a recursion scheme generating either $\mathsf{tt}\langle\perp,\perp\rangle$ or $\mathsf{ff}\langle\perp,\perp\rangle$, we can easily check what is generated: we just repeatedly apply rules of the recursion scheme. Thus Theorem 1.2 is an immediate consequence of Theorem 4.6 and Proposition 4.5.

▶ Remark 4.7. Note that in Theorem 4.6 we do not assume that T is generated by a recursion scheme; the theorem holds for any tree T. Thus our decidability result, Theorem 1.2, can be immediately generalized from the class of trees generated by recursion schemes to any class of trees that is effectively closed under the considered three types of operations (i.e., any class for which Theorems 4.1, 4.2, and 4.4 remain true).

We now formulate a variant of Theorem 4.6 suitable for induction. On the input side, we have to deal with formulae with free variables (subformulae of our original sentence). On the output side, it is not enough to produce the truth value; we rather need to produce trees decorated by logical types. While logical types in general depend on the valuation of free variables, we consider here only a very special valuation mapping all variables to the empty set; recall that we denote such a valuation by $\varnothing$. Additionally, in the input tree we have to allow presence of some additional labels (used to store types with respect to other subformulae): we suppose that we have a tree T over an alphabet $\mathbb{A}\times\mathbb{B}$, where $\mathbb{A}$ is our fixed alphabet used by WMSO+$\mathsf{U}_{\mathsf{tup}}$ formulae, and $\mathbb{B}$ is some other auxiliary alphabet. Then by $\pi_{\mathbb{A}}(T)$ we denote the tree over $\mathbb{A}$ having the same domain as T, with every node thereof relabeled from $(a,b)\in\mathbb{A}\times\mathbb{B}$ to a.

▶ **Lemma 4.8.** *Given a* WMSO+$\mathsf{U}_{\mathsf{tup}}$ *formula φ and an auxiliary alphabet $\mathbb{B}$, one can compute a sequence of operations $O_1, O_2, \ldots, O_n$, such that for every tree T over $\mathbb{A}\times\mathbb{B}$, by applying $O_1, O_2, \ldots, O_n$ to T we obtain a tree having the same domain as T, such that every node u thereof is labeled by the pair $(\ell_u, [\![\pi_{\mathbb{A}}(T)\restriction u]\!]^{\varnothing}_{\varphi})$, where $\ell_u\in\mathbb{A}\times\mathbb{B}$ is the label of u in T.*

Theorem 4.6 is an immediate consequence of Lemma 4.8. Indeed, let us use Lemma 4.8 with a singleton alphabet $\mathbb{B}$; for such an alphabet we identify $\mathbb{A}$ with $\mathbb{A}\times\mathbb{B}$. By applying operations $O_1, \ldots, O_n$ obtained from Lemma 4.8 we obtain a tree with the root labeled by (a,τ) for $\tau = [\![T]\!]^{\varnothing}_{\varphi}$. Recall that, by Fact 3.2, we have a function tv_φ such that $tv_\varphi([\![T]\!]^{\varnothing}_{\varphi}) = \mathsf{tt}$ if, and only if, $T,\varnothing\models\varphi$. Thus, after all the operations $O_1, \ldots, O_n$, we can simply apply a transducer $\mathcal{F}$ that reads the root's label (a,τ) and returns the tree $\mathsf{tt}\langle\perp,\perp\rangle$ if $tv_\varphi(\tau)=\mathsf{tt}$, and the tree $\mathsf{ff}\langle\perp,\perp\rangle$ otherwise. There is a small exception if the original tree T has empty domain: then there is no root at all, in particular no root from which we can read the φ-type τ. Thus, if the transducer $\mathcal{F}$ sees an empty tree, it should rather use $\tau = [\![\perp]\!]^{\varnothing}_{\varphi}$, which is known by Fact 3.4.

Proof of Lemma 4.8. The proof is by induction on the structure of φ. We have several cases depending on the shape of φ.

Recall that in this lemma we only consider the valuation $\varnothing$ mapping all variables to the empty set. Because of that, if φ is of the form $a(X)$ or $X \subseteq Y$, then the φ-type $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\varphi}$ is tt for every tree T and node u thereof. It is thus enough to return (as the only operation O_1) a transducer that appends tt to the label of every node of T. Similarly, if $\varphi = (X \curlywedge_d Y)$, then the φ-type $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\varphi}$ is always empty. For $\varphi = (\neg\psi)$ the situation is also trivial: we can directly use the induction hypothesis since $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\varphi} = [\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\psi}$.

Suppose that $\varphi = (\psi_1 \wedge \psi_2)$. The induction hypothesis for ψ_1 gives us a sequence of operations $O_1, O_2, \dots, O_n$ that appends $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\psi_1}$ to the label of every node u of T. The resulting tree T' is over the alphabet $\mathbb{A} \times \mathbb{B} \times \mathit{Typ}_{\psi_1}$, which can be seen as $\mathbb{A} \times \mathbb{B}'$ for $\mathbb{B}' = \mathbb{B} \times \mathit{Typ}_{\psi_1}$; we have $\pi_{\mathbb{A}}(T') = \pi_{\mathbb{A}}(T)$. We can thus apply the induction hypothesis for ψ_2 to the resulting tree T'; it gives us a sequence of operations $O_{n+1}, O_{n+2}, \dots, O_{n+m}$ that appends $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\psi_2}$ to the label of every node u of T'. The tree obtained after applying all the $n+m$ operations is as needed: in every node thereof we have appended the pair containing the ψ_1-type and the ψ_2-type, and such a pair is precisely the φ-type.

The case of $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$ is handled by a reduction to the case of $\varphi' = \mathsf{U}\mathsf{X}.\psi$. Indeed, recall that the type for $\mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k)$ is a tuple of 2^k coordinates indexed by sets $I \subseteq [1,k]$; in the case of a single variable $\mathsf{X}_1 = \mathsf{X}$, there are only two coordinates, one for $I = \emptyset$, and the other for $I = \{1\}$. The coordinate for $I = \emptyset$ in $[\![T']\!]^{\varnothing}_{\mathsf{UX}.\psi}$ is simply $\{\sigma \mid \exists X.[\![T']\!]^{\nu[\mathsf{X}\mapsto X]}_{\psi} = \sigma\}$, that is, the φ-type $[\![T']\!]^{\nu}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}$. Thus, we can take the sequence of operations $O_1, O_2, \dots, O_n$ from the forthcoming case of $\varphi' = \mathsf{UX}.\psi$, which appends the φ'-type, and then add a simple transducer that removes the second coordinate of this type.

Finally, suppose that $\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\psi$. By the induction hypothesis we have a sequence of operations $O_1, O_2, \dots, O_n$ that appends the ψ-type $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]^{\varnothing}_{\psi}$ to the label of every node u of T. Let T^1 be the tree obtained from T by applying these operations.

As a first step, to T^1 we apply a transducer $\mathcal{F}$ defined as follows. Its input alphabet is $\mathbb{A}' = \mathbb{A} \times \mathbb{B} \times \mathit{Typ}_{\psi}$, the alphabet of T^1, its output alphabet is $\mathbb{A}' \cup \{?, \#, \mathsf{nd}, \mathsf{nd}_{\perp}, \mathsf{X}_1, \dots, \mathsf{X}_k\}$, and its set of states is $\{q_0\} \cup \mathit{Typ}_{\psi}$. Having a letter $\ell = (a, b, \tau) \in \mathbb{A}'$, let $\pi_{\mathbb{A}}(\ell) = a$ and $\pi_{\mathit{Typ}_{\psi}}(\ell) = \tau$. Coming to transitions, first for every triple $(S, \tau_{\mathsf{L}}, \tau_{\mathsf{R}})$, where $S = \{\mathsf{X}_{i_1}, \dots, \mathsf{X}_{i_m}\} \subseteq \{\mathsf{X}_1, \dots, \mathsf{X}_k\}$ and $\tau_{\mathsf{L}}, \tau_{\mathsf{R}} \in \mathit{Typ}_{\psi}$ we define

$$\mathit{sub}(S, \tau_{\mathsf{L}}, \tau_{\mathsf{R}}) = \mathsf{X}_{i_1}\langle \perp, \mathsf{X}_{i_2}\langle \perp, \dots \mathsf{X}_{i_m}\langle \perp, \#\langle (\tau_{\mathsf{L}}, \mathsf{L}), (\tau_{\mathsf{R}}, \mathsf{R})\rangle\rangle \dots\rangle\rangle\,.$$

Moreover, for every $\ell \in \mathbb{A}'$ and $\tau \in \mathit{Typ}_{\psi}$, let $\mathit{here}(\ell, \tau) = \perp$ if $\tau = \pi_{\mathit{Typ}_{\psi}}(\ell)$ and $\mathit{here}(\ell, \tau) = \mathsf{nd}_{\perp}\langle \perp, \perp\rangle$ otherwise. In order to define $\delta(\tau, \ell)$, we consider all triples $(S_1, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1}), \dots, (S_s, \tau_{\mathsf{L},s}, \tau_{\mathsf{R},s})$ for which $\mathit{Comp}_{\pi_{\mathbb{A}}(\ell),\psi}(S_i, \tau_{\mathsf{L},i}, \tau_{\mathsf{R},i}) = \tau$ (assuming some fixed order in which these triples are listed). Then, we take

$$\delta(\tau, \ell) = ?\langle \perp, \mathsf{nd}\langle \mathit{sub}(S_1, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1}), \mathsf{nd}\langle \mathit{sub}(S_2, \tau_{\mathsf{L},2}, \tau_{\mathsf{R},2}), \dots \\ \mathsf{nd}\langle \mathit{sub}(S_s, \tau_{\mathsf{L},s}, \tau_{\mathsf{R},s}), \mathit{here}(\ell, \tau)\rangle \dots\rangle\rangle\rangle\,.$$

Additionally, we consider the list $\tau_1, \dots, \tau_r$ of all ψ-types from Typ_{ψ} (listed in some fixed order), and we define

$$\delta(q_0, \ell) = \ell\langle (q_0, \mathsf{L}), \#\langle (q_0, \mathsf{R}), \#\langle \delta(\tau_1, \ell), \#\langle \delta(\tau_2, \ell), \dots \#\langle \delta(\tau_r, \ell), \perp\rangle \dots\rangle\rangle\rangle\rangle\,.$$

For the empty tree we define

$$\delta(q_0, \perp) = \perp, \qquad \delta([\![\perp]\!]^{\varnothing}_{\psi}, \perp) = \perp, \qquad \text{and} \qquad \delta(\tau, \perp) = \mathsf{nd}_{\perp} \qquad \text{for } \tau \neq [\![\perp]\!]^{\varnothing}_{\psi}\,.$$

The "main part" of the result $\mathcal{F}(T^1)$, produced using the state q_0 is an almost unchanged copy of T^1; there is only a technical change, that a new $\#$-labeled node is inserted between every node and its right child, so that the right child is moved to the left child of this new

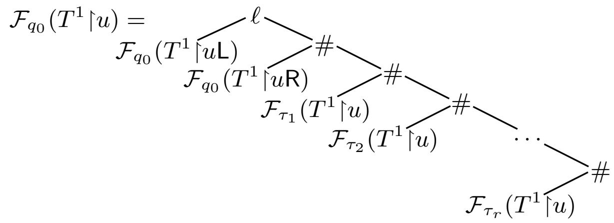

Figure 5 An illustration of $\mathcal{F}_{q_0}(T^1{\restriction}u)$. Here, ℓ is the label of u in T^1, and $\tau_1, \dots, \tau_r$ are all possible ψ-types.

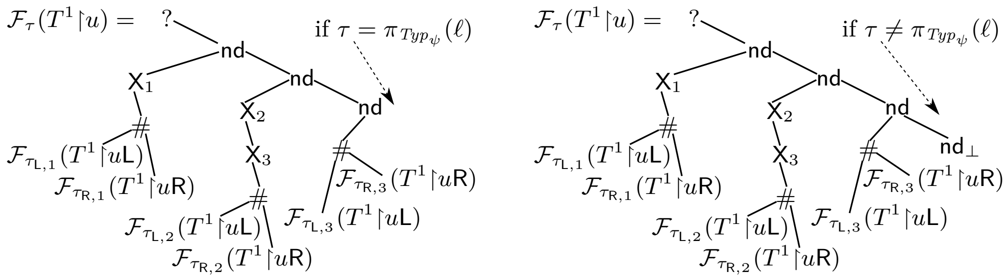

Figure 6 An illustration of $\mathcal{F}_\tau(T^1{\restriction}u)$. We assume that there are exactly three triples $(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ such that $Comp_{\pi_\mathbb{A}(\ell),\psi}(S, \tau_\mathsf{L}, \tau_\mathsf{R}) = \tau$, namely $(\{\mathsf{X}_1\}, \tau_{\mathsf{L},1}, \tau_{\mathsf{R},1})$, $(\{\mathsf{X}_2, \mathsf{X}_3\}, \tau_{\mathsf{L},2}, \tau_{\mathsf{R},2})$, and $(\emptyset, \tau_{\mathsf{L},3}, \tau_{\mathsf{R},3})$, for ℓ being the label of u in T^1. We have two cases depending on whether the ψ-type written in ℓ is τ or not.

right child. But additionally, below the new #-labeled right child of every node u of T^1, there are $|Typ_\psi|$ modified copies of $T^1{\restriction}u$, attached below a branch of #-labeled nodes (cf. Figure 5). For each ψ-type τ we have such a copy, namely $\mathcal{F}_\tau(T^1{\restriction}u)$, responsible for checking whether the type of $\pi_\mathbb{A}(T){\restriction}u$ can be τ. The tree $\mathcal{F}_\tau(T^1{\restriction}u)$ is a disjunction (formed by nd-labeled nodes) of all possible triples $(S, \tau_\mathsf{L}, \tau_\mathsf{R})$ such that types τ_L and τ_R in children of u, together with S being the set of those variables among $\mathsf{X}_1, \dots, \mathsf{X}_k$ that contain u, result in type τ in u (cf. Figure 6). We output the variables from S in the resulting tree, so that they can be counted, and then we have subtrees $\mathcal{F}_{\tau_\mathsf{L}}(T^1{\restriction}u\mathsf{L})$ and $\mathcal{F}_{\tau_\mathsf{R}}(T^1{\restriction}u\mathsf{R})$, responsible for checking whether the type in the children of u can be τ_L and τ_R. Additionally, the *here* subtree allows to finish immediately if τ is the ψ-type of $T^1{\restriction}u$ under the empty valuation. Formally, we have the following claim:

▷ Claim 4.9. For every ψ-type τ, numbers $n_1, \dots, n_k \in \mathbb{N}$, and node u, the following two statements are equivalent:

- there exist sets $X_1, \dots, X_k$ of nodes of $T{\restriction}u$ such that $[\![\pi_\mathbb{A}(T){\restriction}u]\!]_\psi^{\varnothing[\mathsf{X}_1\mapsto X_1,\dots,\mathsf{X}_k\mapsto X_k]} = \tau$ and $|X_i| = n_i$ for $i \in [1, k]$, and
- there exists a tree $V \in \mathcal{L}(\mathcal{F}_\tau(T^1{\restriction}u))$ such that $\#_{\mathsf{X}_i}(V) = n_i$ for $i \in [1, k]$.

Proof. Let us concentrate on the left-to-right implication. The proof is by induction on the maximal depth of nodes in the X_i sets. We have three cases. First, it is possible that u is not a node of T. Then, all the sets X_i have to be empty, so we have $\tau = [\![\perp]\!]_\psi^\varnothing$, and hence $\mathcal{F}_\tau(T^1{\restriction}u) = \delta(\tau, \perp) = \perp$ (recall that T and T^1 have the same domain). The set $\mathcal{L}(\perp)$ contains the tree $\perp$ which indeed has no X_i labeled nodes, as needed.

Second, it is possible that u is a node of T, but all the sets X_i are empty. Let ℓ be the label of u in T^1. By construction of T^1, we have $\pi_{Typ_\psi}(\ell) = [\![\pi_\mathbb{A}(T){\restriction}u]\!]_\psi^\varnothing = \tau$. On the rightmost branch of $\mathcal{F}_\tau(T^1{\restriction}u)$, after a ?-labeled node and a few nd-labeled nodes, we have the subtree $here(\ell, \tau)$, which is $\perp$ by the above equality. We can return the tree $?\langle\perp, \perp\rangle$, which belongs to $\mathcal{L}(\mathcal{F}_\tau(T^1{\restriction}u))$.

Finally, suppose that our sets are not all empty. Then necessarily u is inside T (and T^1); let ℓ be the label of u in T^1 (by construction of T^1, the label of u in T consists of the first two coordinates of ℓ). Consider $S = \{\mathsf{X}_i \mid \varepsilon \in X_i\}$ and $\tau_d = [\![\pi_{\mathbb{A}}(T){\restriction}ud]\!]_\psi^{\varnothing[\mathsf{X}_1 \mapsto X_1, \ldots, \mathsf{X}_k \mapsto X_k]{\restriction}d}$ for $d \in \{\mathsf{L}, \mathsf{R}\}$. By the induction hypothesis, there are trees $V_d \in \mathcal{L}(\mathcal{F}_{\tau_d}(T^1{\restriction}ud))$ such that $\#_{\mathsf{X}_i}(V_d) = |X_i{\restriction}d|$ for $i \in [1,k]$. Due to Equality (1) we have $\tau = \mathit{Comp}_{\pi_{\mathbb{A}}(\ell),\psi}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$. This means that $\delta(\tau, \ell)$, below a ?-labeled node and a few nd-labeled, produces a subtree using $\mathit{sub}(S, \tau_\mathsf{L}, \tau_\mathsf{R})$. We define V by choosing this subtree. Then, there are some X_i-labeled nodes, for all $\mathsf{X}_i \in S$ (that is, for those sets X_i that contain the root of $T{\restriction}u$). Below them, we have the tree $\#\langle \mathcal{F}_{\tau_\mathsf{L}}(T^1{\restriction}u\mathsf{L}), \mathcal{F}_{\tau_\mathsf{R}}(T^1{\restriction}u\mathsf{R})\rangle$; in its left subtree we choose V_L, and in its right subtree we choose V_R. This way, we obtain a tree $V \in \mathcal{L}(\mathcal{F}_\tau(T^1{\restriction}u))$, where the number of X_i-labeled nodes is indeed $|X_i| = n_i$, for all $i \in [1,k]$.

We skip the proof of the right-to-left implication, as it is analogous (this time, the induction is on the height of the tree V). ◁

Let $T^2 = \mathcal{F}(T^1)$. As the next operation after $\mathcal{F}$, we use SUP. Let $T^3 = \mathsf{refl}_{\mathit{SUP}}(T^2)$. The SUP operation attaches a label to every node of T^3 (except for nd-labeled nodes), but we are interested in these labels only in nodes originally (i.e., in T_2) labeled by "?". Every such node is the root of a subtree $\mathsf{refl}_{\mathit{SUP}}(\mathcal{F}_\tau(T^1{\restriction}u))$ for some node u of T^1; it becomes labeled by $(?, \mathcal{U})$, where $\mathcal{U} = \{A \subseteq \mathbb{A}' \mid \mathit{SUP}_A(\mathcal{L}(\mathcal{F}_\tau(T^1{\restriction}u)))\}$. Recall that $\varphi = \mathsf{U}(\mathsf{X}_1, \ldots, \mathsf{X}_k).\psi$ and that the φ-type is a tuple of 2^k coordinates, indexed by sets $I \subseteq [1,k]$. Consider such a set I, and take $A_I = \{\mathsf{X}_i \mid i \in I\}$. By definition of SUP_{A_I}, the label $\mathcal{U}$ contains A_I if, and only if, for every $n \in \mathbb{N}$ the language $\mathcal{L}(\mathcal{F}_\tau(T^1{\restriction}u))$ contains trees with at least n occurrences of every element of A_I. By Claim 4.9 this is the case if, and only if, for every $n \in \mathbb{N}$ there exist sets $X_1, \ldots, X_k$ of nodes of $T{\restriction}u$ such that $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]_\psi^{\varnothing[\mathsf{X}_1 \mapsto X_1, \ldots, \mathsf{X}_k \mapsto X_k]} = \tau$ and $|X_i| \geq n$ for all $i \in I$. This, in turn, holds if, and only if, the I-coordinate of the φ-type $[\![\pi_{\mathbb{A}}(T){\restriction}u]\!]_\varphi^\varnothing$ contains τ. (The case of $I = \emptyset$ is a bit delicate, but one can see that the proof works without any change also in this case.)

The above means that all the φ-types we wished to compute are already present in T^3, we only have to move them to correct places. To this end, for every ψ-type τ_i, and for every set $I \subseteq [1,k]$ we append to our sequence of operations a formula $\theta_{i,I}$ saying that the node $\mathsf{R}^{i+1}\mathsf{L}$ has label of the form $(?, \mathcal{U}, \ldots)$ with $A_I \in \mathcal{U}$ (note that this node in $\mathcal{F}_{q_0}(T^1{\restriction}u)$ is the ?-labeled root of $\mathcal{F}_{\tau_i}(T^1{\restriction}u)$; the operation SUP appends a set $\mathcal{U}$ to this label, and operations $\theta_{i',I'}$ applied so far append some additional coordinates that we ignore).

After that, we already have all φ-types in correct nodes, but in a wrong format; we also have additional nodes not present in the original tree T. To deal with this, at the end we apply a transduction $\mathcal{F}'$, which

- removes all nodes labeled by $(\#, \ldots)$ and their right subtrees, hence leaving only nodes present in the original tree T;
- the remaining nodes have labels of the form $(a, b, \tau, \mathcal{U}, v_{i_1,I_1}, \ldots, v_{i_s,I_s})$; we relabel them to $(a, b, (\{\tau_i \mid v_{i,I} = \mathsf{tt}\})_{I \subseteq [1,k]})$.

This last transduction produces a tree exactly as needed. ◀

5 Expressivity

In this section we prove our second main result, Theorem 1.1, saying that the simultaneous unboundedness property can be expressed in WMSO+U_tup, but not in WMSO+U. The positive part of this statement is easy:

▶ **Proposition 5.1.** *For every set of letters A there exists a* WMSO+U_tup *sentence φ which holds in a tree T if, and only if, $\mathit{SUP}_A(\mathcal{L}(T))$ holds.*

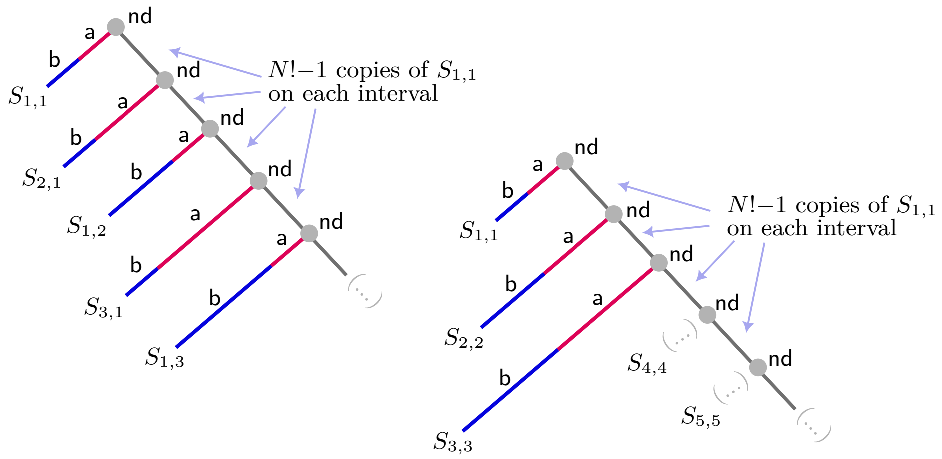

Figure 7 T_1 (left) and T_2 (right).

Proof. Let $A = \{a_1, \dots, a_k\}$. We take

$$\varphi = \mathsf{U}(\mathsf{X}_1, \dots, \mathsf{X}_k).\exists_{\mathsf{fin}}\mathsf{Y}.(a_1(\mathsf{X}_1) \wedge \dots \wedge a_k(\mathsf{X}_k) \wedge \mathsf{X}_1 \subseteq \mathsf{Y} \wedge \dots \wedge \mathsf{X}_k \subseteq \mathsf{Y} \wedge \psi(\mathsf{Y})),$$

where $\psi(\mathsf{Y})$ expresses the fact that Y contains nodes of a single tree from $\mathcal{L}(T)$, together with their nd-labeled ancestors, that is, that for every node v of Y,

- the parent of v, if exists, belongs to Y;
- if v has label nd, then exactly one child of v belongs to Y (strictly speaking: there is a direction $d \in \{\mathsf{L}, \mathsf{R}\}$ such that a child in this direction, if exists, belongs to Y, and the child in the opposite direction, if exists, does not belong to Y);
- v does not have label $\mathsf{nd}_\perp$; and
- if v has label other than nd, then all children of v belong to Y.

It is easy to write the above properties in WMSO+$\mathsf{U}_{\mathsf{tup}}$. Then φ expresses that for every $n \in \mathbb{N}$ there exist sets $X_1, \dots, X_k$ of nodes of some $V \in \mathcal{L}(T)$ such that $|X_i| \geq n$ and nodes of X_i have label a_i, for all $i \in [1, k]$; this is precisely the simultaneous unboundedness property with respect to the set $A = \{a_1, \dots, a_k\}$. ◀

In the remaining part of this section we prove that SUP with respect to $\{\mathsf{a}, \mathsf{b}\}$ cannot be expressed in WMSO+U (i.e., without using the U quantifier for tuples of variables). We prove this already for the word variant of SUP (cf. Remark 4.3), which is potentially easier to be expressed than SUP in its full generality.

Our proof is by contradiction. Assume thus that there is a sentence φ_{SUP} of WMSO+U that holds exactly in those trees T for which $SUP_{\{\mathsf{a},\mathsf{b}\}}(T)$ is true. Having φ_{SUP} fixed, we take a number N such that $|Typ_\varphi| \leq N$ and $|Typ_{\exists_{\mathsf{fin}}\mathsf{X}.\varphi}| \leq N$ for all subformulae φ of φ_{SUP} (recall that Typ_φ is a set containing all possible φ-types).

Based on N, we now define two trees, T_1 and T_2, such that $SUP_{\{\mathsf{a},\mathsf{b}\}}(T_2)$ but not $SUP_{\{\mathsf{a},\mathsf{b}\}}(T_1)$, and we show that they are indistinguishable by φ_{SUP}. We achieve that by demonstrating their type equality as stated in Lemma 5.5, which by Fact 3.2 gives their indistinguishability by the WMSO+U sentence φ_{SUP}.

▶ **Definition 5.2** (T_1 and T_2). *We define T_1 as a tree with an infinite rightmost path (that we call its* trunk*), containing* nd*-labeled nodes. For each integer $k \geq 0$, there is a leftward path called* vault *attached to the $(kN!+1)$-th node of the trunk. If k is even, we denote the* *vault as $S_{1,\frac{k}{2}+1}$, and otherwise as $S_{\frac{k+1}{2}+1,1}$. Each vault $S_{m,n}$ consists of two parts: the upper sub-path of length $mN!$, where every node has label a, and the lower sub-path of length $nN!$, where every node has label b (cf. Figure 7).*

To each node of the trunk that does not have a vault attached we attach a copy of $S_{1,1}$. Note that we do not call these copies vaults; only the original $S_{1,1}$ starting at the root of T_1 is a vault.

The definition of T_2 is similar to that of T_1, except that this time the vault associated with each k is $S_{k,k}$, still starting at depth $kN!+1$ and having $kN!$ nodes with label a followed by $kN!$ nodes with label b.

The technical core of our proof lies in the following two lemmata:

▶ **Lemma 5.3.** *Let ψ be such that $|\mathit{Typ}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$. If for all $k', \ell' \in \mathbb{N}$ we have $[\![T_1{\restriction}\mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2{\restriction}\mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$, then for all $k, \ell \in \mathbb{N}$ and $\tau \in \mathit{Typ}_{\psi}$ there exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to\infty} f(n) = \infty$ and for all $n \in \mathbb{N}$,*

$$\exists X_1 \subseteq \mathrm{dom}(T_1{\restriction}\mathsf{R}^{kN!}).\, |X_1| = n \wedge [\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi} = \tau$$
$$\implies \qquad \exists X_2 \subseteq \mathrm{dom}(T_2{\restriction}\mathsf{R}^{\ell N!}).\, f(n) \leq |X_2| < \infty \wedge [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing[\mathsf{X}\mapsto X_2]}_{\psi} = \tau.$$

▶ **Lemma 5.4.** *Let ψ be such that $|\mathit{Typ}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$. If for all $k', \ell' \in \mathbb{N}$ we have $[\![T_1{\restriction}\mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2{\restriction}\mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$, then for all $k, \ell \in \mathbb{N}$ and $\tau \in \mathit{Typ}_{\psi}$ there exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to\infty} f(n) = \infty$ and for all $n \in \mathbb{N}$,*

$$\exists X_1 \subseteq \mathrm{dom}(T_1{\restriction}\mathsf{R}^{kN!}).\, f(n) \leq |X_1| < \infty \wedge [\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi} = \tau$$
$$\impliedby \qquad \exists X_2 \subseteq \mathrm{dom}(T_2{\restriction}\mathsf{R}^{\ell N!}).\, |X_2| = n \wedge [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing[\mathsf{X}\mapsto X_2]}_{\psi} = \tau.$$

Note that the function f in Lemmata 5.3 and 5.4 may depend on k and ℓ. We only sketch here the proof of the above lemmata; a full proof can be found in Appendix A of the extended version.

Lemma 5.3 is slightly easier. Indeed, suppose first that $k = \ell = 0$. By assumption, in T_1 we have a finite set of nodes X_1 resulting in a ψ-type τ; based on X_1, we have to produce a finite set of nodes X_2 in T_2, producing the same ψ-type τ. The non-vault nodes of X_1 are transferred to X_2 without any change; note that the trees T_1, T_2 are identical outside of vaults. When in T_1 we have some vault $S_{1,i}$ (or $S_{i,1}$, handled in the same way), then in the analogous place of T_2 we have a vault $S_{j,j}$ with $j \geq i$. We use a form of pumping to convert $S_{1,i}$ with some nodes marked as elements of X_1 into $S_{j,j}$ with marked nodes, which we take to X_2; this is done so that the ψ-type does not change. Namely, we concentrate on ψ-types of subtrees of $S_{1,i}$ starting on different levels. Already in the bottom, b-labeled part of $S_{1,i}$ we can find two levels in distance at most N, where the ψ-type repeats (by the pigeonhole principle; recall that the number of possible ψ-types is at most N). We then repeat the fragment of $S_{1,i}$ between these two places (together with the set elements marked in it), so that $(j-i)N!$ new nodes are created, and we obtain $S_{1,j}$. Note that the repeated length, being at most N, necessarily divides $N!$. Because of Proposition 3.3, such a pumping does not change the ψ-type. In a similar way, we can pump the upper, a-labeled part of $S_{1,j}$, and obtain $S_{j,j}$. In this way, we convert a finite top part of T_1 (with a set X_1) into T_2 (with a set X_2) without changing the ψ-type; the infinite parts located below (where the sets X_1, X_2 do not contain any elements) have the same ψ-type by the assumption $[\![T_1{\restriction}\mathsf{R}^{k'N!}]\!]^{\varnothing}_{\psi} = [\![T_2{\restriction}\mathsf{R}^{\ell'N!}]\!]^{\varnothing}_{\psi}$. All nodes originally in X_1 remained in X_2 (possibly shifted), so we have $|X_2| \geq |X_1|$; the lemma holds with $f(n) = n$ in this case.

When k, ℓ are arbitrary (and we want to change $T_1{\restriction}\mathsf{R}^{kN!}$ into $T_2{\restriction}\mathsf{R}^{\ell N!}$), we proceed in a similar way, but there is a potential problem that a vault $S_{1,i}$ (or $S_{i,1}$) should be mapped to $S_{j,j}$ with $j < i$; then we should not stretch the vault, but rather contract it. But contracting is also possible: this time we look on $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-types (instead of ψ-types) on the shorter target vault $S_{1,j}$; we can pump the vault as previously, so $S_{1,i}$ and $S_{1,j}$ have the same $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type. Because $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type is a set of all possible ψ-types, we can choose elements of $S_{1,j}$ (and later of $S_{j,j}$) to X_2, so that the ψ-type is the same as originally in $S_{1,i}$, although without any guarantees on the size of the new set. Anyway, the length of vaults in T_2 grows two times faster than in T_1, so the above problem concerns only the first $\max(0, k-2\ell)$ vaults, where the number of elements of X_1 is bounded by a constant $c_{k,\ell}$ (depending on k and ℓ). All further elements of X_1 contribute to the size of X_2; the lemma holds with $f(n) = n - c_{k,\ell}$.

Consider now Lemma 5.4, where we have to create a set X_1 in T_1 based on a set X_2 in T_2. There are two cases. Suppose first that at least half of elements of X_2 lie outside of the vaults. In this case we proceed as previously, appropriately stretching and/or contracting the vaults. While there is no size guarantee for vault elements, already by counting elements outside of the vaults we obtain $|X_2| \geq \frac{|X_1|}{2}$.

In the opposite case, we check which label is more frequent among the (at least $\frac{|X_2|}{2}$) vault elements of X_2. Suppose this is a (the case of b is analogous), and that $k = \ell = 0$. We then map every vault $S_{i,i}$ into $S_{i,1}$, contracting only the b-labeled part; all the a-labeled vault elements of X_2 remain in X_1. Because the distance between $S_{i,i}$ and $S_{i+1,i+1}$ in T_2 is $N!$, while the distance between $S_{i,1}$ and $S_{i+1,1}$ in T_1 is $2N!$, we also need to stretch the trunk, which is possible using a similar pumping argument (and we stretch some $S_{1,1}$ into the vault $S_{1,i}$ that should be in the middle between $S_{i,1}$ and $S_{i+1,1}$).

This is almost the end, except that we need to handle arbitrary k, ℓ. To this end, we either stretch the initial fragment of the trunk of length $N!$ into multiple such fragments, or we contract the initial fragment of appropriate length into a fragment of length $N!$, so that the vault lengths become synchronized.

Having Lemmata 5.3 and 5.4 we can conclude that the trees T_1 and T_2 (cf. Definition 5.2) have the same types:

▶ **Lemma 5.5.** *Let φ be a subformula of φ_{SUP}. Then for all $k, \ell \in \mathbb{N}$ we have $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\varphi} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\varphi}$.*

Proof. We proceed by induction on φ, considering all possible forms of the formula. First, note that we only consider the valuation $\varnothing$, mapping all variables to the empty set, so for atomic formulae of the form $a(\mathsf{X})$ or $\mathsf{X} \subseteq \mathsf{Y}$ the φ-type is always tt, and for $\mathsf{X} \curlywedge_d \mathsf{Y}$ the φ-type is always empty. For $\varphi = \psi_1 \wedge \psi_2$ the φ-type is just the pair containing the ψ_1-type and the ψ_2-type; for them we have the equality $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\psi_i} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\psi_i}$ by the induction hypothesis. Likewise, for $\varphi = \neg\psi$ the φ-type equals the ψ-type, and we immediately conclude by the induction hypothesis $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing}_{\psi} = [\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]^{\varnothing}_{\psi}$.

Suppose now that $\varphi = \exists_{\mathsf{fin}}\mathsf{X}.\psi$. Then the φ-type of $T_1{\restriction}\mathsf{R}^{kN!}$ is the set of ψ-types $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]^{\varnothing[\mathsf{X}\mapsto X_1]}_{\psi}$ over all possible finite sets $X_1 \subseteq \mathrm{dom}(T_1{\restriction}\mathsf{R}^{kN!})$, and likewise for T_2. By Lemma 5.3, for every ψ-type of $T_1{\restriction}\mathsf{R}^{kN!}$ there exists a set X_2 giving the same ψ-type for $T_2{\restriction}\mathsf{R}^{\ell N!}$, and conversely by Lemma 5.4, so the two φ-types are equal (recall that N was chosen such that $|\mathit{Typ}_{\exists_{\mathsf{fin}}\mathsf{X}.\psi}| \leq N$ whenever ψ is a subformula of φ_{SUP}, hence the two lemmata can indeed be applied).

Finally, suppose that $\varphi = \mathsf{U}\mathsf{X}.\psi$. Then the φ-type consists of two coordinates. On the first coordinate we simply have the $\exists_{\mathsf{fin}}\mathsf{X}.\psi$-type – these types are equal for $T_1{\restriction}\mathsf{R}^{kN!}$ and $T_2{\restriction}\mathsf{R}^{\ell N!}$ by the previous case. On the second coordinate we have the set of ψ-types τ such that $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_1]} = \tau$ for arbitrarily large finite sets X_1, and likewise for X_2. But $[\![T_1{\restriction}\mathsf{R}^{kN!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_1]} = \tau$ for arbitrarily large finite sets X_1 if and only if $[\![T_2{\restriction}\mathsf{R}^{\ell N!}]\!]_\psi^{\varnothing[\mathsf{X}\mapsto X_2]} = \tau$ for arbitrarily large finite sets X_2, by Lemmata 5.3 and 5.4. This gives us equality of the two φ-types.

Recall that by assumption φ_{SUP} is a formula of WMSO+U, without quantification over tuples, so the above exhausts all possible cases. ◀

Lemma 5.5 implies in particular that $[\![T_1]\!]^{\varnothing}_{\varphi_{SUP}} = [\![T_2]\!]^{\varnothing}_{\varphi_{SUP}}$, which by Fact 3.2 means that φ_{SUP} is satisfied in T_1 if and only if it is satisfied in T_2. This way we reach a contradiction with the fact that φ_{SUP} should be true in T_1, but not in T_2. Thus, the simultaneous unboundedness property for two letters cannot be expressed by a formula φ_{SUP} not involving the U quantifiers for tuples of variables; we obtain Theorem 1.1.

▶ Remark 5.6. We have shown that SUP with respect to a two-element set $\{\mathsf{a}, \mathsf{b}\}$ cannot be expressed without quantification over pairs of variables. It is easy to believe that using a very similar proof one can show that SUP with respect to a k-element set cannot be expressed without quantification over k-tuples of variables, for every $k \geq 2$.

References

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

2 David Barozzini, Lorenzo Clemente, Thomas Colcombet, and Paweł Parys. Cost automata, safe schemes, and downward closures. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 109:1–109:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.109.

3 Mikołaj Bojańczyk. A bounding quantifier. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004. doi:10.1007/978-3-540-30124-0_7.

4 Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011. doi:10.1007/s00224-010-9279-2.

5 Mikołaj Bojańczyk. Weak MSO+U with path quantifiers over infinite trees. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2014. doi:10.1007/978-3-662-43951-7_4.

6 Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 285–296. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.17.

7 Mikołaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th – March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 648–660. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.STACS.2012.648.

8 Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification – 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2012. doi:10.1007/978-3-642-31424-7_8.

9 Luca Breveglieri, Alessandra Cherubini, Claudio Citrini, and Stefano Crespi-Reghizzi. Multi-push-down languages and grammars. *Int. J. Found. Comput. Sci.*, 7(3):253–292, 1996. doi:10.1142/S0129054196000191.

10 Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Higher-order recursion schemes and collapsible pushdown automata: Logical properties. *ACM Trans. Comput. Log.*, 22(2):12:1–12:37, 2021. doi:10.1145/3452917.

11 Julius Richard Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

12 Claudia Carapelle, Alexander Kartzow, and Markus Lohrey. Satisfiability of CTL* with constraints. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 – Concurrency Theory – 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2013. doi:10.1007/978-3-642-40184-8_32.

13 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. Ordered tree-pushdown systems. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 163–177. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.163.

14 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 96–105. ACM, 2016. doi:10.1145/2933575.2934527.

15 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

16 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A characterization for decidable separability by piecewise testable languages. *Discret. Math. Theor. Comput. Sci.*, 19(4), 2017. doi:10.23638/DMTCS-19-4-1.

17 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982. doi:10.1016/0304-3975(82)90009-3.

18 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98(1):21–51, 1961. doi:10.2307/1993511.

19 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185392.

20 Nathanaël Fijalkow and Martin Zimmermann. Cost-parity and cost-Streett games. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 124–135. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.124.

21 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 – 22, 2016*, pages 151–163. ACM, 2016. doi:10.1145/2837614.2837627.

22 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. *ACM Trans. Comput. Log.*, 18(3):25:1–25:42, 2017. doi:10.1145/3091122.

23 Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. doi:10.1007/3-540-45931-6_15.

24 Naoki Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3):20:1–20:62, 2013. doi:10.1145/2487241.2487246.

25 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 179–188. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.29.

26 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods Syst. Des.*, 34(2):83–103, 2009. doi:10.1007/s10703-009-0067-z.

27 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.

28 Paweł Parys. Recursion schemes and the WMSO+U logic. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.53.

29 Paweł Parys. Recursion schemes, the MSO logic, and the U quantifier. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:20)2020.

30 Paweł Parys. A type system describing unboundedness. *Discret. Math. Theor. Comput. Sci.*, 22(4), 2020. doi:10.23638/DMTCS-22-4-2.

31 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969. doi:10.2307/1995086.

32 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Inf. Comput.*, 239:340–355, 2014. doi:10.1016/j.ic.2014.07.012.

33 Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 229–243. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.229.

34 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Math. Struct. Comput. Sci.*, 26(7):1304–1350, 2016. doi:10.1017/S0960129514000590.

35 Boris Trakhtenbrot. Finite automata and the logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961.

36 Georg Zetzsche. An approach to computing downward closures. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015. doi:10.1007/978-3-662-47666-6_35.