RECURSION SCHEMES, THE MSO LOGIC, AND THE U QUANTIFIER

PAWEŁ PARYS

Institute of Informatics, University of Warsaw
e-mail address: parys@mimuw.edu.pl

Abstract. We study the model-checking problem for recursion schemes: does the tree generated by a given higher-order recursion scheme satisfy a given logical sentence. The problem is known to be decidable for sentences of the MSO logic. We prove decidability for an extension of MSO in which we additionally have an unbounding quantifier U, saying that a subformula is true for arbitrarily large finite sets. This quantifier can be used only for subformulae in which all free variables represent finite sets (while an unrestricted use of the quantifier leads to undecidability).

We also show that the logic has the properties of reflection and effective selection for trees generated by recursion schemes.

1. Introduction

Higher-order recursion schemes (*schemes* in short) are used to faithfully represent the control flow of programs in languages with higher-order functions [Dam82, KNU02, Ong06, Kob13]. This formalism is equivalent via direct translations to simply-typed λY-calculus [SW16]. Collapsible pushdown systems [HMOS08] and ordered tree-pushdown systems [CPSW15] are other equivalent formalisms. Schemes cover some other models such as indexed grammars [Aho68] and ordered multi-pushdown automata [BCCC96].

In our setting, a scheme is a finite description of an infinite tree. A useful property of schemes is that the *MSO-model-checking problem* for schemes is decidable. This means that given a scheme $\mathcal{G}$ and an MSO sentence φ, it can be algorithmically decided whether the tree generated by $\mathcal{G}$ satisfies φ. This result has several different proofs [Ong06, HMOS08, KO09, SW14], and also some extensions like global model checking [BO09], logical reflection [BCOS10], effective selection [CS12], existence of lambda-calculus model [SW15a]. When the property of trees is given as an automaton, not as a formula, the model-checking problem can be solved efficiently, in the sense that there exist implementations working in a reasonable running time [Kob13, Kob11, BK13, RNO14, NO14] (most tools cover only a fragment of MSO, though).

Recently, an interest has arisen in model-checking trees generated by schemes against properties not expressible in the MSO logic. These are properties expressing boundedness

Key words and phrases: higher-order recursion schemes, intersection types, WMSO+U logic, boundedness.

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

and unboundedness of some quantities. More precisely, it was shown that the *simultaneous unboundedness problem* (aka. *diagonal problem*) for schemes is decidable [HKO16, CPSW16, Par17]. This problem asks, given a scheme $\mathcal{G}$ and a set of letters A, whether for every $n \in \mathbb{N}$ there exists a path in the tree generated by $\mathcal{G}$ such that every letter from A appears on this path 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 [Zet15], and the problem of separability by piecewise testable languages [CMvRZ15].

In this paper we show a result of a more general style. Instead of considering a particular property, like in the simultaneous unboundedness problem, we consider a logic capable to express properties talking about boundedness. More precisely, we extend the MSO logic by the unbounding quantifier, U [Boj04]. A formula using this quantifier, $\mathsf{U}\mathsf{X}.\varphi$, says that φ holds for arbitrarily large finite sets X. We impose a restriction that $\mathsf{U}\mathsf{X}.\varphi$ can be used only in a context where all free variables of φ represent finite sets. We call the resulting logic MSO+$\mathsf{U}^{\mathsf{fin}}$.

The goal of this paper is to prove the following theorem.

Theorem 1.1. *It is decidable whether the tree generated by a given scheme satisfies a given* MSO+$\mathsf{U}^{\mathsf{fin}}$ *sentence.*

We remark that the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic extends the WMSO+U logic, which was widely considered in the context of infinite words [Boj11] and infinite trees [GK10, BT12, Boj14]. The difference is that in WMSO+U only quantification over finite sets is allowed. In consequence, WMSO+U cannot express all properties of MSO [HM12]. On the other hand, in MSO+$\mathsf{U}^{\mathsf{fin}}$ we allow quantification over infinite sets like in standard MSO, and we only restrict the use of the U quantifier to subformulae in which all free variables represent finite sets.

Furthermore, we remark that some restriction for the U quantifier is necessary. Indeed, the model-checking problem for the full MSO+U logic (where the U quantifier can be used in an unrestricted way) is undecidable already over the infinite word without labels [BPT16], so even more over all fancy trees that can be generated by higher-order recursion schemes.

While proving Theorem 1.1, we depend on several earlier results. First, we translate MSO+$\mathsf{U}^{\mathsf{fin}}$ formulae to an equivalent automata model using the notion of logical types (aka. composition method) following a long series of previous work (some selection: [FV59, She75, Lä68, BCL08, GK10, PT16]). Second, we use the logical-reflection property of schemes [BCOS10]. It says that given a scheme $\mathcal{G}$ and an MSO sentence φ one can construct a scheme $\mathcal{G}_\varphi$ generating the same tree as $\mathcal{G}$, where in every node it is additionally written whether φ is satisfied in the subtree starting in this node. Third, we use an analogous property for the simultaneous unboundedness problem, called *SUP reflection* [Par18b]: given a scheme $\mathcal{G}$ we can construct a scheme $\mathcal{G}_{SUP}$ generating the same tree as $\mathcal{G}$, where every node is additionally annotated by the solution of the simultaneous unboundedness problem in the subtree starting in this node. Finally, we use the fact that schemes can be composed with finite tree transducers transforming the generated trees; this follows directly from the equivalence between schemes and collapsible pushdown systems [HMOS08].

Although our algorithm depends on a solution to the simultaneous unboundedness problem, it is not known whether the simultaneous unboundedness problem itself can be expressed in MSO+$\mathsf{U}^{\mathsf{fin}}$. The difficulty is that every U quantifier can entail unboundedness only of a single quantity, and it seems difficult to express simultaneous unboundedness of multiple quantities in MSO+$\mathsf{U}^{\mathsf{fin}}$.

This paper is an extended version of a conference paper [Par18a], where the result is shown for the WMSO+U logic. Besides the fact that we work here with a slightly stronger logic (namely, with MSO+$\mathsf{U}^{\mathsf{fin}}$ instead of WMSO+U), our proofs follow basically the same ideas as proofs contained in the conference paper [Par18a]. We remark that the conference paper [Par18a] contained additionally a justification of the SUP-reflection property for schemes. This justification was already expanded in another paper [Par18b], and for this reason we do not include it here.

Our paper is structured as follows. In Section 2 we introduce all necessary definitions. In Section 3 we show how to translate MSO+$\mathsf{U}^{\mathsf{fin}}$ sentences to automata. In Section 4 we prove the main theorem. Section 5 contains a few extensions of the main theorem.

2. Preliminaries

The powerset of a set X is denoted $\mathcal{P}(X)$, and the set of finite subsets of X is denoted $\mathcal{P}^{\mathsf{fin}}(X)$. For a relation r, we write r^* for the reflexive transitive closure of r. 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 $f(z)$.

Infinitary lambda-calculus. We consider infinitary, simply-typed lambda-calculus. In particular, each lambda-term has an associated sort (aka. simple type). The set of *sorts* is constructed from a unique ground sort o using a binary operation $\rightarrow$; namely o is a sort, and if α and β are sorts, so is $\alpha \rightarrow \beta$. By convention, $\rightarrow$ associates to the right, that is, $\alpha \rightarrow \beta \rightarrow \gamma$ is understood as $\alpha \rightarrow (\beta \rightarrow \gamma)$.

While defining lambda-terms we assume a set of variables $\mathit{Vars}^\lambda = \{x^\alpha, y^\beta, z^\gamma, \dots\}$ containing infinitely many variables of every sort (sort of a variable is written in superscript). *Infinitary lambda-terms* (or just *lambda-terms*) are defined by coinduction, according to the following rules:

- node constructor—if $K_1^{\mathsf{o}}, \dots, K_r^{\mathsf{o}}$ are lambda-terms, and a is an arbitrary object, called a *letter*, then $(a\langle K_1^{\mathsf{o}}, \dots, K_r^{\mathsf{o}}\rangle)^{\mathsf{o}}$ is a lambda-term,
- variable—every variable $x^\alpha \in \mathit{Vars}^\lambda$ is a lambda-term,
- application—if $K^{\alpha\rightarrow\beta}$ and L^α are lambda-terms, then $(K^{\alpha\rightarrow\beta}\, L^\alpha)^\beta$ is a lambda-term, and
- lambda-binder—if K^β is a lambda-term and x^α is a variable, then $(\lambda x^\alpha.K^\beta)^{\alpha\rightarrow\beta}$ is a lambda-term.

Sets of letters are called *alphabets*. We use unranked letters; this subsumes the setting of ranked letters. We naturally identify lambda-terms differing only in names of bound variables. We often omit the sort annotations of lambda-terms, but we keep in mind that every lambda-term (and every variable) has a fixed sort. Free variables and subterms of a lambda-term, as well as beta-reductions, are defined as usual. A lambda-term K is *closed* if it has no free variables. We restrict ourselves to those lambda-terms for which the set of sorts of all subterms is finite.

Trees; Böhm trees. A *tree* is defined as a lambda-term that is built using only node constructors, that is, not using variables, applications, nor lambda-binders. For a tree $T = a\langle T_1, \dots, T_r\rangle$, its set of nodes is defined as the smallest set such that

- ε is a node of T, labeled by a, and
- if u is a node of T_i for some $i \in \{1, \dots, r\}$, labeled by b, then iu is a node of T, also labeled by b.

A node v is the *i-th child* of u if $v = ui$. We say that two trees T, T' are of *the same shape* if they have the same nodes. A tree T is *over alphabet* Σ if all labels of its nodes belong to Σ, and it has *maximal arity* $r_{\max} \in \mathbb{N}$ if its every node has at most $r_{\max}$ children. When both these conditions are satisfied, we say that T is a $(\Sigma, r_{\max})$*-tree*. For a tree T and its node u, by $T{\restriction}_u$ we denote the *subtree* of T starting at u, defined in the expected way.

We consider Böhm trees only for closed lambda-terms of sort o. For such a lambda-term K, its *Böhm tree* is constructed by coinduction, as follows: if there is a sequence of beta-reductions from K to a lambda-term of the form $a\langle K_1, \dots, K_r\rangle$, and $T_1, \dots, T_r$ are Böhm trees of $K_1, \dots, K_r$, respectively, then $a\langle T_1, \dots, T_r\rangle$ is a Böhm tree of K; if there is no such sequence of beta-reductions from K, then $\omega\langle\rangle$ is a Böhm tree of K (where ω is a fixed letter). It is folklore that every closed lambda-term of sort o has exactly one Böhm tree (the order in which beta-reductions are performed does not matter); this tree is denoted by $BT(K)$.

A closed lambda-term K of sort o is called *fully convergent* if every node of $BT(K)$ is explicitly created by a node constructor from K (e.g., $\omega\langle\rangle$ is fully convergent, while $K = (\lambda x^{\mathsf{o}}.x)\, K$ is not). More formally: we consider the lambda-term $K_{-\omega}$ obtained from K by replacing ω with some other letter ω', and we say that K is fully convergent if in $BT(K_{-\omega})$ there are no ω-labeled nodes.

Higher-order recursion schemes. Our definition of schemes is less restrictive than usually, as we see them only as finite representations of infinite lambda-terms. Thus a *higher-order recursion scheme* (or just a *scheme*) is a triple $\mathcal{G} = (\mathcal{N}, \mathcal{R}, N_0^{\mathsf{o}})$, where

- $\mathcal{N} \subseteq \mathit{Vars}^\lambda$ is a finite set of nonterminals,
- $\mathcal{R}$ is a function that maps every nonterminal $N \in \mathcal{N}$ to a finite lambda-term whose all free variables are contained in $\mathcal{N}$ and whose sort equals the sort of N, and
- $N_0^{\mathsf{o}} \in \mathcal{N}$ is a starting nonterminal, being of sort o.

We assume that elements of $\mathcal{N}$ are not used as bound variables, and that $\mathcal{R}(N)$ is not a nonterminal for any $N \in \mathcal{N}$.

For a scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, N_0)$, and for a lambda-term K whose free variables are contained in $\mathcal{N}$, we define the infinitary lambda-term *represented by* K with respect to $\mathcal{G}$, denoted $\Lambda_{\mathcal{G}}(K)$, by coinduction: to obtain $\Lambda_{\mathcal{G}}(K)$ we replace in K every nonterminal $N \in \mathcal{N}$ with $\Lambda_{\mathcal{G}}(\mathcal{R}(N))$. Observe that $\Lambda_{\mathcal{G}}(K)$ is a closed lambda-term of the same sort as K. The infinitary lambda-term *represented by* $\mathcal{G}$, denoted $\Lambda(\mathcal{G})$, equals $\Lambda_{\mathcal{G}}(N_0)$.

By the *tree generated by* $\mathcal{G}$ we mean $BT(\Lambda(\mathcal{G}))$. We write $\Sigma_{\mathcal{G}}$ for the finite alphabet containing ω and letters used in node constructors appearing in $\mathcal{G}$, and $r_{\max}(\mathcal{G})$ for the maximal arity of node constructors appearing in $\mathcal{G}$. Clearly $BT(\Lambda(\mathcal{G}))$ is a $(\Sigma_{\mathcal{G}}, r_{\max}(\mathcal{G}))$-tree.

In our constructions it is convenient to consider only schemes representing fully-convergent lambda-terms, which is possible due to the following standard result.

Fact 2.1 ([Had12, SW15b]). *For every scheme $\mathcal{G}$ we can construct a scheme $\mathcal{G}'$ generating the same tree as $\mathcal{G}$, and such that $\Lambda(\mathcal{G}')$ is fully convergent.* □

Example 2.1. Consider the scheme $\mathcal{G}_1 = (\{M^{\mathsf{o}}, N^{\mathsf{o}\to\mathsf{o}}\}, \mathcal{R}, M)$, where

$$\mathcal{R}(N) = \lambda x^{\mathsf{o}}.a\langle x, N\,(b\langle x\rangle)\rangle\,, \qquad \text{and} \qquad \mathcal{R}(M) = N\,(c\langle\rangle)\,.$$

We obtain $\Lambda(\mathcal{G}_1) = K\,(c\langle\rangle)$, where K is the unique lambda-term for which it holds $K = \lambda x^{\mathsf{o}}.a\langle x, K\,(b\langle x\rangle)\rangle$. The tree generated by $\mathcal{G}_1$ equals $a\langle T_0, a\langle T_1, a\langle T_2, \dots\rangle\rangle\rangle$, where $T_0 = c\langle\rangle$ and $T_i = b\langle T_{i-1}\rangle$ for all $i \geq 1$.

Remark. An usual definition of schemes is more restrictive than ours: it is required that $\mathcal{R}(N)$ is a of the form $\lambda x_1.\cdots.\lambda x_s.K$, where K is of sort o and does not use any lambda-binders. We do not have this requirement, so possibly $\mathcal{R}(N)$ does not start with a full sequence of lambda-binders, and possibly some lambda-binders are nested deeper in the lambda-term. It is, though, not difficult to convert a scheme respecting only our definition to a scheme satisfying these additional requirements (at the cost of introducing more non-terminals). We can, for example, use a translation between schemes and λY-terms from Salvati and Walukiewicz [SW16]: their translation from schemes to λY-terms works well with our definition of schemes, while the translation from λY-terms to schemes produces schemes respecting the more restrictive definition.

Another difference is that in the definition of the Böhm tree we allow arbitrary beta-reductions, while it is sometimes assumed that only outermost beta-reductions are allowed. It is a folklore that these two definitions are equivalent.

There is one more difference: we expand a scheme to an infinite lambda-term, and then we operate on this lambda-term, while often finite lambda-terms containing nonterminals are considered, and appearances of nonterminals are expanded only when needed. This is a purely syntactical difference.

MSO+$\mathsf{U}^{\mathsf{fin}}$. 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 (at least when the maximal arity of considered trees is fixed). We assume two infinite sets of variables, $\mathcal{V}^{\mathsf{fin}}$ and $\mathcal{V}^{\mathsf{inf}}$, and we let $\mathcal{V} = \mathcal{V}^{\mathsf{fin}} \uplus \mathcal{V}^{\mathsf{inf}}$. Variables from $\mathcal{V}^{\mathsf{fin}}$ are used to quantify over finite sets, while variables from $\mathcal{V}^{\mathsf{inf}}$ over arbitrary (potentially infinite) sets. In the syntax of MSO+$\mathsf{U}^{\mathsf{fin}}$ we have the following constructions:

$$\varphi ::= a(\mathsf{X}) \mid \mathsf{X} \curlywedge_i \mathsf{Y} \mid \mathsf{X} \subseteq \mathsf{Y} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi' \mid \exists \mathsf{Z}.\varphi' \mid \exists_{\mathsf{fin}}\mathsf{F}.\varphi' \mid \mathsf{U}\mathsf{F}.\varphi'$$

where a is a letter, and $i \in \mathbb{N}_+$, and $\mathsf{X}, \mathsf{Y} \in \mathcal{V}$, and $\mathsf{Z} \in \mathcal{V}^{\mathsf{inf}}$, and $\mathsf{F} \in \mathcal{V}^{\mathsf{fin}}$. Free variables of a formula are defined as usual; in particular UF is a quantifier, hence it bounds the variable F. We impose the restriction that $\mathsf{UF}.\varphi'$ can be used only when all free variables of φ' are from $\mathcal{V}^{\mathsf{fin}}$.

The MSO logic is defined likewise, with the exception that the U quantifier is disallowed. (The fact that a set of tree nodes is finite is expressible in MSO without using the $\exists_{\mathsf{fin}}$ quantifier, thus presence of this quantifier does not change the expressive power of MSO).

We evaluate formulae of MSO+$\mathsf{U}^{\mathsf{fin}}$ in Σ-labeled trees. In order to evaluate a formula φ in a tree T, we also need a *valuation*, that is, a partial function ν from $\mathcal{V}$ to sets of nodes of T, such that $\nu(\mathsf{F})$ is finite whenever $\mathsf{F} \in \mathcal{V}^{\mathsf{fin}} \cap \mathrm{dom}(\nu)$. The function should be defined at least for all free variables of φ. The semantics is defined as follows:

- $a(\mathsf{X})$ holds when every node in $\nu(\mathsf{X})$ is labeled by a,

- $\mathsf{X} \curlywedge_i \mathsf{Y}$ holds when both $\nu(\mathsf{X})$ and $\nu(\mathsf{Y})$ are singletons, and the unique node in $\nu(\mathsf{Y})$ is the i-th 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,
- $\exists\mathsf{Z}.\varphi'$ holds when φ' holds for an extension of ν that maps Z to some set of nodes of T,
- $\exists_{\mathsf{fin}}\mathsf{F}.\varphi'$ holds when φ' holds for an extension of ν that maps F to some finite set of nodes of T, and
- $\mathsf{U}\mathsf{F}.\varphi'$ holds when for every $n \in \mathbb{N}$, φ' holds for an extension of ν that maps F some finite set of nodes of T of cardinality at least n.

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

In order to see that our definition of MSO+$\mathsf{U}^{\mathsf{fin}}$ is not too poor, let us write a few example formulae.

- The fact that X represents an empty set can be expressed as $\mathit{empty}(\mathsf{X}) \equiv a(\mathsf{X}) \wedge b(\mathsf{X})$ (where a, b are any two different letters).
- The fact that X represents a set of size at least 2 can be expressed as $\mathit{big}(\mathsf{X}) \equiv \exists\mathsf{Y}.(\mathsf{Y} \subseteq \mathsf{X} \wedge \neg(\mathsf{X} \subseteq \mathsf{Y}) \wedge \neg\mathit{empty}(\mathsf{Y}))$.
- The fact that X represents a singleton can be expressed as $\mathit{sing}(\mathsf{X}) \equiv \neg\mathit{empty}(\mathsf{X}) \wedge \neg\mathit{big}(\mathsf{X})$.
- When we only consider trees of a fixed maximal arity $r_{\max}$, the fact that X and Y represent singletons $\{x\}, \{y\}$, respectively, such that y is a child of x can be expressed as
$$(\mathsf{X} \curlywedge_1 \mathsf{Y}) \vee \dots \vee (\mathsf{X} \curlywedge_{r_{\max}} \mathsf{Y}),$$
where $\varphi_1 \vee \varphi_2$ stands for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$.
- Let $A = \{a_1, \dots, a_k\}$ be a finite set of letters. The fact every node in the set represented by X has label in A can be expressed as
$$\forall\mathsf{Y}.(\mathit{sing}(\mathsf{Y}) \wedge \mathsf{Y} \subseteq \mathsf{X}) \rightarrow (a_1(\mathsf{Y}) \vee \dots \vee a_k(\mathsf{Y})),$$
where $\forall\mathsf{Y}.\varphi$ stands for $\neg\exists\mathsf{Y}.\neg\varphi$, and $\varphi_1 \rightarrow \varphi_2$ stands for $\neg(\varphi_1 \wedge \neg\varphi_2)$.

Like in most logics, in MSO+$\mathsf{U}^{\mathsf{fin}}$ we can relativize formulae, as described by Fact 2.2.

Fact 2.2. *Let $r_{\max} \in \mathbb{N}$. For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *sentence φ we can construct an* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula $\widehat{\varphi}(\mathsf{X})$ with one free variable X such that for every tree T of maximal arity $r_{\max}$ and every valuation ν, it holds $T, \nu \models \widehat{\varphi}$ if and only if φ holds in $T{\restriction}_u$ for every $u \in \nu(\mathsf{X})$.*

Proof sketch. Suppose first that we want to construct a formula $\varphi'(\mathsf{X})$ that satisfies the fact only for valuations mapping X to singleton sets $\{u\}$. To this end, we need to relativize quantification in φ to the subtree starting in u. This means that we replace subformulae of the form $\exists\mathsf{Y}.\psi$ (and likewise $\exists_{\mathsf{fin}}\mathsf{Y}.\psi$ and $\mathsf{U}\mathsf{Y}.\psi$) by $\exists\mathsf{Y}.\eta(\mathsf{X},\mathsf{Y}) \wedge \psi$, where $\eta(\mathsf{X},\mathsf{Y})$ says that the set represented by Y contains only (not necessarily proper) descendants of the node represented by X.

We conclude by taking $\widehat{\varphi}(\mathsf{X}) \equiv \forall\mathsf{X}'.(\mathit{sing}(\mathsf{X}') \wedge \mathsf{X}' \subseteq \mathsf{X}) \rightarrow \varphi'(\mathsf{X}')$, saying that the formula $\varphi'(\mathsf{X}')$ holds whenever X' represents a singleton subset of the set represented by X. $\square$

3. Nested U-prefix MSO automata

In this section we give a definition of nested U-prefix MSO automata, a formalism equivalent to the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic. These are compositions of U-prefix automata and MSO automata, defined below.

A U-*prefix automaton* is a tuple $\mathcal{A} = (\Sigma, Q, Q_{\mathsf{imp}}, \Delta)$, where Σ is a finite alphabet, Q is a finite set of states, $Q_{\mathsf{imp}} \subseteq Q$ is a set of *important* states, and $\Delta \subseteq Q \times \Sigma \times (Q \cup \{\top\})^*$ is a finite transition relation (we assume $\top \notin Q$). A *run* of $\mathcal{A}$ on a Σ-labeled tree T is a mapping ρ from the set of nodes of T to $Q \cup \{\top\}$ such that

- there are only finitely many nodes u such that $\rho(u) \in Q$, and
- for every node u of T, with label a and r children, it holds that either $\rho(u) = \top = \rho(u1) = \cdots = \rho(ur)$ or $(\rho(u), a, \rho(u1), \ldots, \rho(ur)) \in \Delta$.

We use U-prefix automata as transducers, relabeling nodes of T: we define $\mathcal{A}(T)$ to be the tree of the same shape as T, and such that its every node u originally labeled by a_u becomes labeled by the pair (a_u, f_u), where $f_u \colon Q \to \{0, 1, 2\}$ is the function that assigns to every state $q \in Q$

- 2, if for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}$ on $T{\restriction}_u$ that assigns q to the root of $T{\restriction}_u$, and such that for at least n nodes w it holds that $\rho_n(w) \in Q_{\mathsf{imp}}$;
- 1, if the above does not hold, but there is a run of $\mathcal{A}$ on $T{\restriction}_u$ that assigns q to the root of $T{\restriction}_u$;
- 0, if none of the above holds.

Example 3.1. Consider the U-prefix automaton $\mathcal{A}_1 = (\{a\}, \{q_{\exists lf}, q_{\mathsf{fin}}\}, \{q_{\mathsf{fin}}\}, \Delta)$, where Δ contains transitions

$$\begin{aligned} &(q_{\mathsf{fin}}, a), (q_{\mathsf{fin}}, a, q_{\mathsf{fin}}), (q_{\mathsf{fin}}, a, q_{\mathsf{fin}}, q_{\mathsf{fin}}), (q_{\exists lf}, a), && \text{and} \\ &(q_{\exists lf}, a, q), (q_{\exists lf}, a, q, \top), (q_{\exists lf}, a, \top, q) && \text{for } q \in \{q_{\exists lf}, q_{\mathsf{fin}}\}\,. \end{aligned}$$

Suppose now that a $(\{a\}, 2)$-tree T comes. When a state q_{fin} is assigned to some node u of T, then it has to be assigned as well to all descendants of u. Thus, there is a run of $\mathcal{A}_1$ on T with state q_{fin} in the root exactly when the tree is finite. This is because the definition of a run allows to assign states (other than $\top$) only to finitely many nodes of the tree. Going further, there is a run of $\mathcal{A}$ on T with state $q_{\exists lf}$ in the root exactly when there is a leaf in the tree. The run can assign $q_{\exists lf}$ to all nodes on the branch leading to a selected leaf, and $\top$ to all other nodes. Alternatively, it can assign $q_{\exists lf}$ to nodes on a branch leading to some node u, and then q_{fin} to all descendants of u, assuming that the subtree starting in u is finite.

Let B_i be the full binary tree of height i, for $i \in \mathbb{N}$. Let T_1 be the tree consisting of an infinite branch, with tree B_i attached below the i-th child of the branch; that is, we take $T_i = a\langle T_{i+1}, B_i\rangle$ for $i \in \mathbb{N}_+$. By definition, $\mathcal{A}_1(T_1)$ has the same shape as T_1. Nodes inside all B_i become relabeled to $(a, [q_{\exists lf} \mapsto 1, q_{\mathsf{fin}} \mapsto 1])$. This is because every subtree of every B_i is finite and has a leaf. Moreover, the number of nodes of this subtree to which q_{fin} is assigned is bounded by the size of the subtree (and hence we do not use the value 2 in the new label). Nodes of the leftmost branch of T_1 are, in turn, relabeled to $(a, [q_{\exists lf} \mapsto 2, q_{\mathsf{fin}} \mapsto 0])$. The value 2 in the i-th node of the branch means that for every $n \in \mathbb{N}$ there is a run ρ_n on T_i that assigns $q_{\exists lf}$ to the root of T_i, and assigns q_{fin} to at least n nodes. Such a run ρ_n assigns $q_{\exists lf}$ on a branch entering some B_j with at least n nodes, and assigns q_{fin} to all nodes of this B_j.

An MSO *automaton* is a triple $\mathcal{A} = (\Sigma, Q, (\varphi_q)_{q\in Q})$, where Σ is a finite alphabet, Q is a finite set of states, and $(\varphi_q)_{q\in Q}$ is a bundle of MSO sentences indexed by elements of Q. An effect of running such an automaton $\mathcal{A}$ on a Σ-labeled tree T is the tree $\mathcal{A}(T)$ that is of the same shape as T, and such that its every node u originally labeled by a_u becomes labeled by the pair (a_u, f_u), where $f_u \colon Q \to \{0, 1, 2\}$ is the function that assigns to every index $q \in Q$

- 1 if φ_q is true in $T{\restriction_u}$;
- 0 otherwise.[1]

Example 3.2. Let $\mathcal{A}_2 = (\{a\} \times \{0,1,2\}^{\{q_{\exists lf}, q_{\mathsf{fin}}\}}, \{q_1, q_2\}, (\varphi_{q_1}, \varphi_{q_2}))$, where φ_{q_1} says that the second child of the root exists and is labeled by $(a, [q_{\exists lf} \mapsto 1, q_{\mathsf{fin}} \mapsto 1])$, and φ_{q_2} says that there exists an infinite branch with all nodes labeled by $(a, [q_{\exists lf} \mapsto 2, q_{\mathsf{fin}} \mapsto 0])$. Let us analyze $\mathcal{A}_2(\mathcal{A}_1(T_1))$, for the tree T_1 from Example 3.1. It has the same shape as $\mathcal{A}_1(T_1)$, and as T_1. All leaves become labeled by $((a, [q_{\exists lf} \mapsto 1, q_{\mathsf{fin}} \mapsto 1]), [q_1 \mapsto 0, q_2 \mapsto 0])$, other nodes of B_i become labeled by $((a, [q_{\exists lf} \mapsto 1, q_{\mathsf{fin}} \mapsto 1]), [q_1 \mapsto 1, q_2 \mapsto 0])$, and nodes on the leftmost branch of the tree become labeled by $((a, [q_{\exists lf} \mapsto 2, q_{\mathsf{fin}} \mapsto 0]), [q_1 \mapsto 1, q_2 \mapsto 1])$.

By the *input alphabet* of $\mathcal{A}$, where $\mathcal{A}$ is either a U-prefix automaton $(\Sigma, Q, Q_{\mathsf{imp}}, \Delta)$ or an MSO automaton $(\Sigma, Q, (\varphi_q)_{q\in Q})$, we mean the set $\Sigma^{\mathsf{in}}(\mathcal{A}) = \Sigma$. The *output alphabet* of $\mathcal{A}$ is $\Sigma^{\mathsf{out}}(\mathcal{A}) = \Sigma^{\mathsf{in}}(\mathcal{A}) \times \{0,1,2\}^Q$.

A *nested* U*-prefix* MSO *automaton* is a sequence $\mathcal{A} = \mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k$ (with $k \geq 1$), where every $\mathcal{A}_i$ is either a U-prefix automaton or an MSO automaton, and where $\Sigma^{\mathsf{in}}(\mathcal{A}_{i+1}) = \Sigma^{\mathsf{out}}(\mathcal{A}_i)$ for $i \in \{1, \dots, k-1\}$. We define $\mathcal{A}(T)$ to be $\mathcal{A}_k(\dots(\mathcal{A}_1(T))\dots)$. The input and output alphabets of $\mathcal{A}$, denoted $\Sigma^{\mathsf{in}}(\mathcal{A})$ and $\Sigma^{\mathsf{out}}(\mathcal{A})$, equal $\Sigma^{\mathsf{in}}(\mathcal{A}_1)$ and $\Sigma^{\mathsf{out}}(\mathcal{A}_k)$, respectively. The key property is that these automata can check properties expressed in MSO+$\mathsf{U}^{\mathsf{fin}}$, as we state in Lemma 3.1, and we prove in the remainder of this section.

Lemma 3.1. *Let Σ be a finite alphabet, and let $r_{\max} \in \mathbb{N}$. For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *sentence φ we can construct a nested* U*-prefix* MSO *automaton $\mathcal{A}_\varphi$ with $\Sigma^{\mathsf{in}}(\mathcal{A}_\varphi) = \Sigma$, and a subset $\Sigma_{\mathsf{F}} \subseteq \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi)$ such that for every $(\Sigma, r_{\max})$-tree T, the root of $\mathcal{A}_\varphi(T)$ is labeled by a letter in Σ_{F} if and only if φ holds in T.*

Recall that our aim is to evaluate φ in a tree T generated by a recursion scheme $\mathcal{G}$, so the restriction to $(\Sigma, r_{\max})$-trees is not harmful: as $(\Sigma, r_{\max})$ we are going to take $(\Sigma_{\mathcal{G}}, r_{\max}(\mathcal{G}))$.

It is not difficult to see that in MSO+$\mathsf{U}^{\mathsf{fin}}$ we can express properties checked by nested U-prefix MSO automata. This means that the two formalisms are actually equivalent. Although we do not need this second direction in order to prove Theorem 1.1, we state it in Lemma 3.2 for cognitive purposes.

Lemma 3.2. *Let $r_{\max} \in \mathbb{N}$. For every nested* U*-prefix* MSO *automaton $\mathcal{A}$, and every letter $\eta \in \Sigma^{\mathsf{out}}(\mathcal{A})$ we can construct an* MSO+$\mathsf{U}^{\mathsf{fin}}$ *sentence $\varphi_{\mathcal{A},\eta}$ such that for every $(\Sigma^{\mathsf{in}}(\mathcal{A}), r_{\max})$-tree T, the root of $\mathcal{A}(T)$ is labeled by η if and only if $\varphi_{\mathcal{A},\eta}$ holds in T.*

Proof sketch. When $\mathcal{A}$ is a single MSO automaton, $\mathcal{A} = (\Sigma, Q, (\psi_q)_{q\in Q})$, it is straightforward to construct $\varphi_{\mathcal{A},\eta}$ in question. Namely, when $\eta = (a, f)$, as $\varphi_{\mathcal{A},\eta}$ we take

$$\xi_a \wedge \bigwedge_{q : f(q)=1} \psi_q \wedge \bigwedge_{q : f(q)=0} \neg\psi_q\,,$$

[1]MSO automata never assign the value 2; nevertheless, for uniformity between U-prefix automata and MSO automata, we assume that the set of values is $\{0, 1, 2\}$.

where ξ_a says that the root is labeled by a.

It is also not difficult to deal with a single U-prefix automaton. Indeed, it is standard to express in MSO that a run of an automaton exists. The fact that there exist runs with arbitrarily many important states is expressed using the U quantifier.

It remains to simulate composition of automata. Suppose that $\mathcal{A} = \mathcal{A}_1 \rhd \mathcal{A}_2$ (where $\mathcal{A}_1, \mathcal{A}_2$ may be nested again), and that we already have sentences $\psi_{\mathcal{A}_1,a}$ corresponding to $\mathcal{A}_1$ for $a \in \Sigma^{\mathsf{out}}(\mathcal{A}_1) = \Sigma^{\mathsf{in}}(\mathcal{A}_2)$, and $\psi_{\mathcal{A}_2,\eta}$ corresponding to $\mathcal{A}_2$. Out of every sentence $\psi_{\mathcal{A}_1,a}$ we construct a formula $\widehat{\psi}_{\mathcal{A}_1,a}(\mathsf{Z})$ saying that $\psi_{\mathcal{A}_1,a}$ holds in all subtrees starting in elements of the set represented by Z (cf. Fact 2.2). The formula $\psi_{\mathcal{A}_2,\eta}$ is evaluated in $\mathcal{A}_1(T)$, while the formula $\varphi_{\mathcal{A},\eta}$ that we are going to construct is evaluated in T. Thus, whenever $\psi_{\mathcal{A}_2,\eta}$ uses $a(\mathsf{Z})$ for some letter $a \in \Sigma^{\mathsf{in}}(\mathcal{A}_2)$ and some variable Z, in $\varphi_{\mathcal{A},\eta}$ we replace it by $\widehat{\psi}_{\mathcal{A}_1,a}(\mathsf{Z})$. □

We now come to the proof of Lemma 3.1. We notice that due to the nested structure, our automata are quite close to the logic. It is clear that MSO automata can simulate all of MSO. On the other hand, U-prefix automata check whether something is unbounded, which corresponds to U quantifiers. As states of the U-prefix automata we take *phenotypes* (aka. logical types), which are defined next.

Let φ be a formula of MSO+$\mathsf{U}^{\mathsf{fin}}$, let T be a tree, and let ν be a valuation (defined at least for all free variables of φ). We define the *φ-phenotype* 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) 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 $\mathsf{X} \curlywedge_i \mathsf{Y}$, 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 $\nu(\mathsf{Y}) = \{\varepsilon\}$, and
  - ff otherwise,
- if $\varphi \equiv (\psi_1 \wedge \psi_2)$, then $[T]^\nu_\varphi = ([T]^\nu_{\psi_1}, [T]^\nu_{\psi_2})$,
- if $\varphi \equiv (\neg\psi)$, then $[T]^\nu_\varphi = [T]^\nu_\psi$,
- if $\varphi \equiv \exists \mathsf{X}.\psi$, then
$$[T]^\nu_\varphi = \{\sigma \mid \exists X.[T]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma\},$$
- if $\varphi \equiv \exists_{\mathsf{fin}}\mathsf{X}.\psi$, then
$$[T]^\nu_\varphi = \{\sigma \mid \exists X.[T]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma \wedge |X| < \infty\},$$ and
- if $\varphi \equiv \mathsf{U}\mathsf{X}.\psi$, then
$$[T]^\nu_\varphi = (\{\sigma \mid \exists X.[T]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma \wedge |X| < \infty\},$$
$$\{\sigma \mid \forall n \in \mathbb{N}.\exists X.[T]^{\nu[\mathsf{X}\mapsto X]}_\psi = \sigma \wedge n \leq |X| < \infty\}),$$

where X ranges over sets of nodes of T.

For each φ, let Pht_φ denote the set of all potential φ-phenotypes. Namely, $Pht_\varphi = \{\mathsf{tt}, \mathsf{ff}\}$ in the first case, $Pht_\varphi = \{\mathsf{tt}, \mathsf{empty}, \mathsf{root}, \mathsf{ff}\}$ in the second case, $Pht_\varphi = Pht_{\psi_1} \times Pht_{\psi_2}$ in the third case, $Pht_\varphi = Pht_\psi$ in the fourth case, $Pht_\varphi = \mathcal{P}(Pht_\psi)$ in the fifth and sixth case, and $Pht_\varphi = (\mathcal{P}(Pht_\psi))^2$ in the last case.

We immediately see two facts. First, Pht_φ is finite for every φ. Second, the fact whether φ holds in T, ν is determined by $[T]^\nu_\varphi$. This means that there is a function $tv_\varphi \colon Pht_\varphi \to \{\mathsf{tt}, \mathsf{ff}\}$ such that $tv_\varphi([T]^\nu_\varphi) = \mathsf{tt}$ if and only if $T, \nu \models \varphi$.

Next, we observe that phenotypes behave in a compositional way, as formalized below. Here for a valuation ν and a node u, by $\nu{\restriction}_u$ we mean the valuation that restricts ν to the subtree starting at u, that is, maps every variable $\mathsf{X} \in \mathrm{dom}(\nu)$ to $\{w \mid uw \in \nu(\mathsf{X})\}$.

Lemma 3.3 (cf. [GK10, PT16]). *For every letter a, every $r \in \mathbb{N}$, and every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula φ, one can compute a function $Comp_{a,r,\varphi} \colon \mathcal{P}^{\mathsf{fin}}(\mathcal{V}) \times (Pht_\varphi)^r \to Pht_\varphi$ such that for every tree T whose root has label a and r children, and for every valuation ν,*

$$[T]^\nu_\varphi = Comp_{a,r,\varphi}(\{\mathsf{X} \in \mathrm{dom}(\nu) \mid \varepsilon \in \nu(\mathsf{X})\}, [T{\restriction}_1]^{\nu{\restriction}_1}_\varphi, \dots, [T{\restriction}_r]^{\nu{\restriction}_r}_\varphi) .$$

Proof. 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 every subtree $T{\restriction}_i, \nu{\restriction}_i$ and in the root of T. Thus, for $\varphi \equiv b(\mathsf{X})$ as $Comp_{a,r,\varphi}(R, \tau_1, \dots, \tau_r)$ we take tt when $\tau_i = \mathsf{tt}$ for all $i \in \{1, \dots, r\}$ and either $a = b$ or $\mathsf{X} \notin R$. For $\varphi \equiv (\mathsf{X} \subseteq \mathsf{Y})$ the last part of the condition is replaced by "if $\mathsf{X} \in R$ then $\mathsf{Y} \in R$".

Next, suppose that $\varphi \equiv (\mathsf{X} \swarrow_k \mathsf{Y})$. Then as $Comp_{a,r,\varphi}(R, \tau_1, \dots, \tau_r)$ we take

- tt if $\tau_j = \mathsf{tt}$ for some $j \in \{1, \dots, r\}$, and $\tau_i = \mathsf{empty}$ for all $i \in \{1, \dots, r\} \setminus \{j\}$, and $\mathsf{X} \notin R$, and $\mathsf{Y} \notin R$,
- tt also if $\tau_k = \mathsf{root}$, and $\tau_i = \mathsf{empty}$ for all $i \in \{1, \dots, r\} \setminus \{k\}$, and $\mathsf{X} \in R$, and $\mathsf{Y} \notin R$,
- empty if $\tau_i = \mathsf{empty}$ for all $i \in \{1, \dots, r\}$, and $\mathsf{X} \notin R$, and $\mathsf{Y} \notin R$,
- root if $\tau_i = \mathsf{empty}$ for all $i \in \{1, \dots, r\}$, and $\mathsf{X} \notin R$, and $\mathsf{Y} \in R$, and
- ff otherwise.

By comparing this definition with the definition of the phenotype we immediately see that the thesis is satisfied.

When $\varphi \equiv (\neg\psi)$, we simply take $Comp_{a,r,\varphi} = Comp_{a,r,\psi}$, and when $\varphi \equiv (\psi_1 \wedge \psi_2)$, as $Comp_{a,r,\varphi}(R, (\tau^1_1, \tau^2_1), \dots, (\tau^1_r, \tau^2_r))$ we take the pair of $Comp_{a,r,\psi_i}(R, \tau^i_1, \dots, \tau^i_r)$ for $i \in \{1, 2\}$.

Suppose now that $\varphi \equiv \exists \mathsf{X}.\psi$ or $\varphi \equiv \exists_{\mathsf{fin}} \mathsf{X}.\psi$. As $Comp_{a,r,\varphi}(R, \tau_1, \dots, \tau_r)$ we take

$$\{Comp_{a,r,\psi}(R \cup \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r), Comp_{a,r,\psi}(R \setminus \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r) \\ \mid (\sigma_1, \dots, \sigma_r) \in \tau_1 \times \dots \times \tau_r\} .$$

The two possibilities, $R \cup \{\mathsf{X}\}$ and $R \setminus \{\mathsf{X}\}$, correspond to the fact that when quantifying over X, the root of T may be either taken to the set represented by X or not. Notice that the cases of $\exists \mathsf{X}$ and $\exists_{\mathsf{fin}} \mathsf{X}$ are handled in the same way: for a local behavior near the root it does not matter whether we quantify over all sets or only over finite sets.

Finally, suppose that $\varphi \equiv \mathsf{U}\mathsf{X}.\psi$. The arguments of $Comp_{a,r,\varphi}$ are pairs $(\tau_1, \rho_1), \dots, (\tau_r, \rho_r)$. Let A be the set of tuples $(\sigma_1, \dots, \sigma_r) \in \tau_1 \times \dots \times \tau_r$, and let B be the set of tuples $(\sigma_1, \dots, \sigma_r)$ such that $\sigma_j \in \rho_j$ for some $j \in \{1, \dots, r\}$ and $\sigma_i \in \tau_i$ for all $i \in \{1, \dots, r\} \setminus \{j\}$. As $Comp_{a,r,\varphi}(R, (\tau_1, \rho_1), \dots, (\tau_r, \rho_r))$ we take

$$(\{Comp_{a,r,\psi}(R \cup \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r), Comp_{a,r,\psi}(R \setminus \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r) \mid (\sigma_1, \dots, \sigma_r) \in A\}, \\ \{Comp_{a,r,\psi}(R \cup \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r), Comp_{a,r,\psi}(R \setminus \{\mathsf{X}\}, \sigma_1, \dots, \sigma_r) \mid (\sigma_1, \dots, \sigma_r) \in B\}) .$$

The first coordinate is defined as for the existential quantifiers. The second coordinate is computed correctly due to the pigeonhole principle: if for every n we have a set X_n of cardinality at least n (satisfying some property), then we can choose an infinite subsequence of these sets such that either the root belongs to all of them or to none of them, and one can choose some $j \in \{1, \dots, r\}$ such that the sets contain unboundedly many descendants of j. $\square$

In order to prove Lemma 3.1 by induction on the structure of the sentence φ, we need to generalize it a bit; this is done in Lemma 3.4. In particular, we need to use phenotypes, instead of the truth value of the sentence (because phenotypes are compositional, unlike truth values). We also need to allow formulae with free variables, not just sentences, as well as arbitrary valuations. A special role is played by the valuation $\nu_\emptyset$ that maps every variable to the empty set; for this valuation we have a stronger version of the lemma.

Lemma 3.4. *Let Σ be a finite alphabet, and let $r_{\max} \in \mathbb{N}$. Then for every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula φ we can construct*

(1) a nested U-prefix MSO *automaton $\mathcal{A}_\varphi$ with $\Sigma^{\mathsf{in}}(\mathcal{A}_\varphi) = \Sigma$, and* MSO *formulae $\xi_{\varphi,\tau}$ for all $\tau \in \mathit{Pht}_\varphi$, such that for every $(\Sigma, r_{\max})$-tree T, every valuation ν in T, and every $\tau \in \mathit{Pht}_\varphi$ it holds $\mathcal{A}_\varphi(T), \nu \models \xi_{\varphi,\tau}$ if and only if $[T]^\nu_\varphi = \tau$, and*

(2) a nested U-prefix MSO *automaton $\mathcal{B}_\varphi$ with $\Sigma^{\mathsf{in}}(\mathcal{B}_\varphi) = \Sigma$, and a function f_φ: $\Sigma^{\mathsf{out}}(\mathcal{B}_\varphi) \to \mathit{Pht}_\varphi$, such that for every $(\Sigma, r_{\max})$-tree T the root of $\mathcal{B}_\varphi(T)$ is labeled by a letter η such that $f_\varphi(\eta) = [T]^{\nu_\emptyset}_\varphi$.*

Proof. Induction on the size of φ. We start by observing how Item (2) follows from Item (1). Item (1) gives us an automaton $\mathcal{A}_\varphi$ and MSO formulae $\xi_{\varphi,\tau}$ for all $\tau \in \mathit{Pht}_\varphi$. We change these formulae into sequences $\xi'_{\varphi,\tau}$, assuming that all their free variables are valuated to the empty set. More precisely, for every free variable X we change subformulae of $\xi_{\varphi,\tau}$ of the form $a(\mathsf{X})$ and $\mathsf{X} \subseteq \mathsf{Y}$ into tt, subformulae of the form $\mathsf{X} \curlywedge_i \mathsf{Y}$ and $\mathsf{Y} \curlywedge_i \mathsf{X}$ into ff, and subformulae of the form $\mathsf{Y} \subseteq \mathsf{X}$, where Y is a bound variable, into formulae checking that the set represented by Y is empty. Then, we take $\mathcal{B}_\varphi = \mathcal{A}_\varphi \rhd \mathcal{C}$ for $\mathcal{C} = (\Sigma^{\mathsf{out}}(\mathcal{A}_\varphi), \mathit{Pht}_\varphi, (\xi'_{\varphi,\tau})_{\tau \in \mathit{Pht}_\varphi})$. If $\eta = (a, h)$ for a function h mapping exactly one phenotype τ to 1, we define $f_\varphi(\eta)$ to be this phenotype τ, and for $\eta = (a, h)$ with $|h^{-1}(1)| \neq 1$ we define $f_\varphi(\eta)$ arbitrarily.

Consider now a $(\Sigma, r_{\max})$-tree T. By Item (1), for $\tau = [T]^{\nu_\emptyset}_\varphi$ we have $\mathcal{A}_\varphi(T), \nu_\emptyset \models \xi_{\varphi,\tau}$ (equivalently, $\mathcal{A}_\varphi(T) \models \xi'_{\varphi,\tau}$) and for $\tau \in \mathit{Pht}_\varphi \setminus \{[T]^{\nu_\emptyset}_\varphi\}$ we have $\mathcal{A}_\varphi(T), \nu_\emptyset \not\models \xi_{\varphi,\tau}$ (equivalently, $\mathcal{A}_\varphi(T) \not\models \xi'_{\varphi,\tau}$). It follows that the root of $\mathcal{B}_\varphi(T)$ is labeled by $\eta = (a, h)$ where a is the label of the root in $\mathcal{A}_\varphi(T)$, and $h([T]^{\nu_\emptyset}_\varphi) = 1$, and $h(\tau) = 0$ for $\tau \in \mathit{Pht}_\varphi \setminus \{[T]^{\nu_\emptyset}_\varphi\}$. Then $f_\varphi(\eta) = [T]^{\nu_\emptyset}_\varphi$, as required.

We now come to the proof of Item (2), where we proceed by case distinction. When φ is an atomic formula, that is, equals $a(\mathsf{X})$, $\mathsf{X} \subseteq \mathsf{Y}$, or $\mathsf{X} \curlywedge_i \mathsf{Y}$, then the automaton $\mathcal{A}_\varphi$ is not needed: as $\mathcal{A}_\varphi$ we can take the MSO automaton with empty set of states (and input alphabet Σ). For such an automaton we have that $\mathcal{A}_\varphi(T) = T$ for every $(\Sigma, r_{\max})$-tree T. As $\xi_{\varphi,\mathsf{tt}}$ we take φ. When φ equals $a(\mathsf{X})$ or $\mathsf{X} \subseteq \mathsf{Y}$, the only phenotypes are tt and ff, and thus we take $\xi_{\varphi,\mathsf{ff}} \equiv \neg\varphi$. In the case of $\varphi \equiv \mathsf{X} \curlywedge_i \mathsf{Y}$, the situation when the formula is false is divided into three phenotypes: empty, root, and ff. Nevertheless, it is easy to express in MSO that we have a particular phenotype, following the definition of $[T]^\nu_\varphi$.

Suppose now that φ is of the form $\psi_1 \wedge \psi_2$. From the induction assumption, Item (1), we have two automata, $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$, as well as formulae ξ_{ψ_1,τ_1} for all $\tau_1 \in \mathit{Pht}_{\psi_1}$ and ξ_{ψ_2,τ_2} for all $\tau_2 \in \mathit{Pht}_{\psi_2}$. We combine the two automata into a single automaton $\mathcal{A}_\varphi$ with $\Sigma^{\mathsf{in}}(\mathcal{A}_\varphi) = \Sigma$. More precisely, we take $\mathcal{A}_\varphi = \mathcal{A}_{\psi_1} \rhd \mathcal{A}'_{\psi_2}$, where $\mathcal{A}'_{\psi_2}$ works exactly like $\mathcal{A}_{\psi_2}$, but instead of reading a tree T over alphabet Σ, it reads the tree $\mathcal{A}_{\psi_1}(T)$ and ignores the part of its labels added by $\mathcal{A}_{\psi_1}$. We also amend ξ_{ψ_1,τ_1} and ξ_{ψ_2,τ_2} so that they can read the output of $\mathcal{A}_\varphi$: formulae ξ'_{ψ_1,τ_1} work like ξ_{ψ_1,τ_1} but ignore the parts of labels added by $\mathcal{A}'_{\psi_2}$, and formulae ξ'_{ψ_2,τ_2} work like ξ_{ψ_2,τ_2} but ignore the parts of labels added by $\mathcal{A}_{\psi_1}$. Having the above, for every $\tau = (\tau_1, \tau_2) \in \mathit{Pht}_\varphi$ we take $\xi_{\varphi,\tau} \equiv \xi'_{\psi_1,\tau_1} \wedge \xi'_{\psi_2,\tau_2}$. Because a tree has φ-phenotype τ when it has ψ_1-phenotype τ_1 and simultaneously ψ_2-phenotype τ_2, it should be clear that the thesis of Item (1) becomes satisfied.

When φ is of the form $\neg\psi$, or $\exists \mathsf{X}.\psi$, or $\exists_{\mathsf{fin}}\mathsf{X}.\psi$, as $\mathcal{A}_\varphi$ we take $\mathcal{A}_\psi$ existing by the induction assumption, Item (1). The induction assumption gives us also formulae $\xi_{\psi,\tau}$ for all $\tau \in \mathit{Pht}_\psi$. If $\varphi \equiv \neg\psi$, we take $\xi_{\varphi,\tau} \equiv \xi_{\psi,\tau}$. If $\varphi \equiv \exists\mathsf{X}.\psi$, we take

$$\xi_{\varphi,\tau} \equiv \bigwedge_{\sigma\in\tau} (\exists\mathsf{X}.\xi_{\psi,\sigma}) \wedge \bigwedge_{\sigma\in \mathit{Pht}_\psi\setminus\tau} (\neg\exists\mathsf{X}.\xi_{\psi,\sigma})\,.$$

If $\varphi \equiv \exists_{\mathsf{fin}}\mathsf{X}.\psi$, we take the same formula, but with $\exists_{\mathsf{fin}}$ quantifiers instead of $\exists$.

Finally, suppose that $\varphi \equiv \mathsf{U}\mathsf{X}.\psi$. We cannot proceed like in the previous cases, because the U quantifier cannot be expressed in MSO; we rather need to append a new U-prefix automaton at the end of the constructed automaton. In this case we first prove Item (2), and then we deduce Item (1) out of Item (2). By Item (2) of the induction assumption we have an automaton $\mathcal{B}_\psi$ and a function $f_\psi\colon \Sigma^{\mathsf{out}}(\mathcal{B}_\psi) \to \mathit{Pht}_\psi$ such that for every node u of T, the root of $\mathcal{B}_\psi(T{\restriction_u})$ is labeled by a letter η_u such that $f_\psi(\eta_u) = [T{\restriction_u}]_\psi^{\nu_\emptyset}$. Moreover, there is a function $g\colon \Sigma^{\mathsf{out}}(\mathcal{B}_\psi) \to \Sigma$ such that $g(\eta_u)$ is the original label of u in T (such a function exists, because the labels from T remain as a part of the labels in $\mathcal{B}_\psi(T)$). Recall that $\mathcal{B}_\psi(T)$ has the same shape as T, and actually $(\mathcal{B}_\psi(T)){\restriction_u} = \mathcal{B}_\psi(T{\restriction_u})$ for every node u. We construct a new layer $\mathcal{C}$, which calculates φ-phenotypes basing on ψ-phenotypes, and we take $\mathcal{B}_\varphi = \mathcal{B}_\psi \rhd \mathcal{C}$. As the state set of $\mathcal{C}$ we take $Q = \{0,1\} \times \mathit{Pht}_\psi$; states from $\{1\} \times \mathit{Pht}_\psi$ are considered as important. Transitions are determined by the *Comp* predicate from Lemma 3.3. More precisely, for every $r \le r_{\max}$, every $\eta \in \Sigma^{\mathsf{out}}(\mathcal{B}_\psi)$, and all $((i_1,\sigma_1),\dots,(i_r,\sigma_r)) \in Q^r$ we have transitions

$$((0, \mathit{Comp}_{g(\eta),r,\psi}(\emptyset,\sigma_1,\dots,\sigma_r)), \eta, (i_1,\sigma_1),\dots,(i_r,\sigma_r))\,, \qquad \text{and}$$
$$((1, \mathit{Comp}_{g(\eta),r,\psi}(\{\mathsf{X}\},\sigma_1,\dots,\sigma_r)), \eta, (i_1,\sigma_1),\dots,(i_r,\sigma_r))\,.$$

Moreover, we have transitions that read the ψ-phenotype from the label:

$$((0, f_\psi(\eta)), \eta, \underbrace{\top,\dots,\top}_{r}) \qquad \text{for } r \le r_{\max}.$$

We notice that there is a direct correspondence between runs of $\mathcal{C}$ and choices of a set of nodes X to which the variable X is mapped. The first coordinate of the state is set to 1 in nodes chosen to belong to the set X. The second coordinate contains the ψ-phenotype under the valuation mapping X to X and every other variable to the empty set. In some nodes below the chosen set X we use transitions of the second kind, reading the ψ-phenotype from the label; it does not matter in which nodes this is done, as everywhere a correct ψ-phenotype is written. The fact that we quantify only over finite sets X corresponds to the fact that the run of $\mathcal{C}$ can assign non-$\top$ states only to a finite prefix of the tree. Moreover, the cardinality of X is reflected by the number of important states assigned by a run. It follows that for every $\sigma \in \mathit{Pht}_\psi$,

- there exists a finite set X of nodes of T such that $[T]_\psi^{\nu_\emptyset[\mathsf{X}\mapsto X]} = \sigma$ if and only if for some $i \in \{0,1\}$ there is a run of $\mathcal{C}$ on $\mathcal{B}_\psi(T)$ that assigns (i,σ) to the root, and

- for every $n \in \mathbb{N}$ there exists a finite set X_n of nodes of T such that $[T]_\psi^{\nu_\emptyset[\mathsf{X}\mapsto X_n]} = \sigma$ and $|X_n| \geq n$ if and only if for some $i \in \{0,1\}$ and for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{C}$ on $\mathcal{B}_\psi(T)$ that assigns (i,σ) to the root, and such that ρ_n assigns an important state to at least n nodes.

Thus, looking at the root's label in $\mathcal{B}_\varphi(T)$ we can determine $[T]_\varphi^{\nu_\emptyset}$. This finishes the proof of Item (2) in the case of the U quantifier.

Next, still supposing that $\varphi \equiv \mathsf{U}\mathsf{X}.\psi$, we prove Item (1) using Item (2), which is already proved. It is essential that, by the definition of the MSO+$\mathsf{U}^{\mathsf{fin}}$ logic, all free variables of φ come from $\mathcal{V}^{\mathsf{fin}}$, that is, represent finite sets. This means that only nodes from a finite prefix of a considered tree can belong to $\nu(\mathsf{Y})$ for Y free in φ (since clearly the number of free variables is finite). Outside of this finite prefix we can read the φ-phenotype from the output of $\mathcal{B}_\varphi$ (because the valuation is empty there), and in the finite prefix we can compute them using the *Comp* function.

More precisely, as $\mathcal{A}_\varphi$ we take $\mathcal{B}_\varphi$, coming from Item (2). Item (2) gives us a function $f_\varphi \colon \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi) \to \mathit{Pht}_\varphi$ reading the φ-phenotype of a $(\Sigma, r_{\max})$-tree T out of the root label of $\mathcal{A}_\varphi(T)$; we also have a function $g \colon \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi) \to \Sigma$ that extracts original labels out of labels in $\mathcal{A}_\varphi(T)$. For every $\tau \in \mathit{Pht}_\varphi$ we define the formula $\xi_{\psi,\tau}$ as follows. It starts with a sequence of $|\mathit{Pht}_\varphi|$ existential quantifiers, quantifying over variables X_ρ for all $\rho \in \mathit{Pht}_\varphi$. The intention is that, in a tree T, every X_ρ represents the set of nodes u such that $[T{\restriction}_u]_\varphi^{\nu\restriction_u} = \rho$. Inside the quantification we say that

- the sets represented by these variables are disjoint, and every node belongs to some of them,
- the root belongs to X_τ,
- if a node with label $\eta \in \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi)$ belongs to X_ρ, and its children belong to $\mathsf{X}_{\rho_1}, \dots, \mathsf{X}_{\rho_r}$, respectively (where $r \leq r_{\max}$), and R is the set of free variables Y of φ for which the node belongs to $\nu(\mathsf{Y})$, then $\rho = \mathit{Comp}_{g(\eta),r,\varphi}(R, \rho_1, \dots, \rho_r)$ (there are only finitely many possibilities for $\rho, \rho_1, \dots, \rho_r \in \mathit{Pht}_\varphi$, for $r \in \{0, \dots, r_{\max}\}$, for $\eta \in \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi)$, and finitely many free variables of φ, thus the constructed formula can be just a big alternative listing all possible cases), and
- if a node with label $\eta \in \Sigma^{\mathsf{out}}(\mathcal{A}_\varphi)$ belongs to X_ρ and none of $\nu(\mathsf{Y})$ for Y free in φ contains this node or some its descendant, then $\rho = f_\varphi(\eta)$.

Consider now a $(\Sigma, r_{\max})$-tree T, and a valuation ν in this tree. If $[T]_\varphi^\nu = \tau$, then we can show that $\xi_{\varphi,\tau}$ is true by taking for X_ρ the set of nodes u for which $[T{\restriction}_u]_\varphi^{\nu\restriction_u} = \rho$ (for every $\rho \in \mathit{Pht}_\varphi$). Conversely, suppose that $\xi_{\varphi,\tau}$ is true. Then we can prove that a node u can belong to the set represented by X_ρ (for $\rho \in \mathit{Pht}_\varphi$) only when $[T{\restriction}_u]_\varphi^{\nu\restriction_u} = \rho$. The proof is by a straightforward induction on the number of descendants of u that belong to $\nu(\mathsf{Y})$ for some Y free in φ; we use Lemma 3.3 for the induction step. $\square$

Now the proof of Lemma 3.1 follows easily. Indeed, when φ is a sentence (has no free variables), $[T]_\varphi^{\nu_\emptyset}$ determines whether φ holds in T. Thus, it is enough to take the automaton $\mathcal{B}_\varphi$ constructed in Lemma 3.4, and replace the function f_φ by the set $\Sigma_{\mathsf{F}} = \{\eta \in \Sigma^{\mathsf{out}}(\mathcal{A}) \mid \mathit{tv}_\varphi(f_\varphi(\eta))\}$ (where tv_φ, defined on page 10, given a φ-phenotype says whether φ holds in trees having this φ-phenotype).

We remark that the WMSO+U logic (which is weaker than MSO+$\mathsf{U}^{\mathsf{fin}}$) corresponds to nested U-prefix automata, composed of U-prefix automata only (i.e., not using MSO automata). Indeed, MSO automata are needed only to deal with infinite sets; when all quantified sets are finite, we can simulate all the constructs using U-prefix automata [Par18a].

We also remark that Bojańczyk and Toruńczyk [BT12] introduce another model of automata equivalent to WMSO+U: nested limsup automata. A common property of these two models is that both of them are nested; the components of nested limsup automata are of a different form, though.

4. Proof of the main theorem

In this section we prove our main theorem—Theorem 1.1. To this end, we have to recall three properties of recursion schemes: logical reflection (Fact 4.1), SUP reflection (Fact 4.2), and closure under composition with finite tree transducers (Fact 4.3).

The property of logical reflection for schemes comes from Broadbent, Carayol, Ong, and Serre [BCOS10]. They state it for sentences of μ-calculus, but μ-calculus and MSO are equivalent over infinite trees [EJ91].

Fact 4.1 (logical reflection [BCOS10, Theorem 2(ii)]). *For every* MSO *sentence φ and every scheme $\mathcal{G}$ generating a tree T one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node u is labeled by a 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.* □

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

Originally, in the simultaneous unboundedness problem we consider nondeterministic higher-order recursion schemes, which instead of generating a single infinite tree, recognize a set of finite trees. 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 a special letter nd, denoting a nondeterministic choice. We write $T \to_{\mathsf{nd}} U$ if U 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 $\mathsf{nd}\langle T_1, \dots, T_r\rangle \to_{\mathsf{nd}} T_j$ for $j \in \{1, \dots, r\}$, and if $T_j \to_{\mathsf{nd}} T'_j$ for some $j \in \{1, \dots, r\}$, and $T_i = T'_i$ for all $i \in \{1, \dots, r\} \setminus \{j\}$, then $a\langle T_1, \dots, T_r\rangle \to_{\mathsf{nd}} a\langle T'_1, \dots, T'_r\rangle$. For a tree T, $\mathcal{L}(T)$ is the set of all finite trees U such that $\#_{\mathsf{nd}}(U) = \#_\omega(U) = 0$ and $T \to^*_{\mathsf{nd}} U$.

Fact 4.2 (SUP reflection [Par18b, Theorem 10.1]). *For every scheme $\mathcal{G}$ generating a tree T one can construct a scheme $\mathcal{G}_{SUP}$ that generates a tree of the same shape as T, and such that its every node u, having in T label a_u, is labeled by*

- *a pair $(a_u, \{A \subseteq \Sigma_{\mathcal{G}} \mid \mathsf{SUP}_A(\mathcal{L}(T{\restriction}_u))\})$, if $a_u \neq \mathsf{nd}$, and*
- *the letter nd, if $a_u = \mathsf{nd}$.* □

The third recalled fact (Fact 4.3) talks about finite tree transducers. A *(deterministic, top-down) finite tree transducer* is a tuple $\mathcal{T} = (\Sigma, r_{\max}, Q, q_0, \delta)$, where Σ is a finite alphabet, $r_{\max}$ is the maximal arity of considered trees, Q is a finite set of states, $q_0 \in Q$ is an initial state, and δ is a transition function mapping $Q \times \Sigma \times \{0, \dots, r_{\max}\}$ to finite lambda-terms.

A triple (q, a, r) should be mapped by δ to a term that uses only node constructors and variables of the form $\mathsf{x}_{i,p}$, where $i \in \{1, \dots, r\}$ and $p \in Q$ (applications and lambda-binders are not allowed); at least one node constructor has to be used (the whole $\delta(q, a, r)$ cannot be equal to a variable).

For a $(\Sigma, r_{\max})$-tree T and a state $q \in Q$, we define $\mathcal{T}_q(T)$ by coinduction, as follows: if $T = a\langle T_1, \dots, T_r\rangle$, then $\mathcal{T}_q(T)$ is the tree obtained from $\delta(q, a, r)$ by substituting $\mathcal{T}_p(T_i)$ for the variable $\mathsf{x}_{i,p}$, for all $i \in \{1, \dots, r\}$ and $p \in Q$. In the root we start from the initial state, that is, we define $\mathcal{T}(T) = \mathcal{T}_{q_0}(T)$. We have the following fact.

Fact 4.3. *For every finite tree transducer $\mathcal{T} = (\Sigma, r_{\max}, Q, q_0, \delta)$ and every scheme $\mathcal{G}$ generating a $(\Sigma, r_{\max})$-tree T, one can construct a scheme $\mathcal{G}_{\mathcal{T}}$ that generates the tree $\mathcal{T}(T)$.* □

This fact follows from the equivalence between schemes and collapsible pushdown systems [HMOS08], as it is straightforward to compose a collapsible pushdown system with $\mathcal{T}$ (where due to Fact 2.1 we can assume that $\Lambda(\mathcal{G})$ is fully convergent, i.e., that every node of T is explicitly generated by the collapsible pushdown system). Since we are not aware of any proof of this fact in the literature, we give more details in Appendix A.

Using Fact 4.1 we can compose schemes with MSO automata, as stated below.

Lemma 4.4. *For every* MSO *automaton $\mathcal{A}$ and every scheme $\mathcal{G}$ generating a tree T, where $\Sigma_{\mathcal{G}} \subseteq \Sigma^{\mathsf{in}}(\mathcal{A})$, one can construct a scheme $\mathcal{G}_{\mathcal{A}}$ that generates the tree $\mathcal{A}(T)$.*

Proof. Let $\mathcal{A} = (\Sigma, Q, (\varphi_q)_{q\in Q})$. Assume that $Q = \{1, \dots, n\}$, and take $\mathcal{G}_0 = \mathcal{G}$ and $T_0 = T$. Consecutively for $q = 1, \dots, n$ we want to apply Fact 4.1 to φ_q and $\mathcal{G}_{q-1}$, and obtain a scheme $\mathcal{G}_q$ that generates a tree T_q of the same shape as T, and such that its every node u is labeled by a tuple $(a, b_1, \dots, b_q)$, where a is the label of u in T, and b_i says whether φ_i is satisfied in $T{\restriction}_u$ for $i \in \{1, \dots, q\}$. Strictly speaking, we cannot apply Fact 4.1 to the original sentences φ_q (these sentences can be evaluated in T, but not in T_{q-1}). We need to slightly modify the sentences: out of φ_q we obtain φ'_q by changing every subformula of the form $a(\mathsf{X})$ to a formula saying that every node in the set represented by X is labeled by a letter from $\{a\} \times \{\mathsf{tt}, \mathsf{ff}\}^{q-1}$. Then φ_q is satisfied in $T{\restriction}_u$ if and only if φ'_q is satisfied in $T_{q-1}{\restriction}_u$; in consequence, we can apply Fact 4.1 to φ'_q and $\mathcal{G}_{q-1}$.

The last tree, T_n, contains truth values of all sentences φ_q. In order to obtain $\mathcal{G}_A$ as required, it is thus enough to rename letters appearing in $\mathcal{G}_n$: we change every letter $(a, b_1, \dots, b_n)$ to (a, f) for $f\colon Q \to \{0, 1, 2\}$ mapping every $q \in Q$ to 1 if $b_q = \mathsf{tt}$, and to 0 if $b_q = \mathsf{ff}$. □

As one can expect, we can also compose schemes with U-prefix automata, and for that we need Facts 4.2 and 4.3.

Lemma 4.5. *For every U-prefix automaton $\mathcal{A}$ and every scheme $\mathcal{G}$ generating a tree T, where $\Sigma_{\mathcal{G}} \subseteq \Sigma^{\mathsf{in}}(\mathcal{A})$, one can construct a scheme $\mathcal{G}_{\mathcal{A}}$ that generates the tree $\mathcal{A}(T)$.*

It is easy to deduce Theorem 1.1 out of Lemmata 4.4 and 4.5. Indeed, consider an MSO+$\mathsf{U}^{\mathsf{fin}}$ sentence φ and a scheme $\mathcal{G}_0$ generating a tree T_0. By Lemma 3.1, φ is equivalent to a nested U-prefix MSO automaton $\mathcal{A} = \mathcal{A}_1 \rhd \cdots \rhd \mathcal{A}_k$, together with an accepting set Σ_{F}. By consecutively applying Lemmata 4.4 and 4.5 for $i = 1, \dots, k$, we combine $\mathcal{G}_{i-1}$ with $\mathcal{A}_i$, obtaining a scheme $\mathcal{G}_i$ that generates the tree $T_i = \mathcal{A}_i(T_{i-1})$. The root of $T_k = \mathcal{A}(T_0)$ has label in Σ_{F} if and only if φ is satisfied in T_0. Surely this label can be read: having $\mathcal{G}_k$, we simply start generating the tree T_k, until its root is generated (by Fact 2.1, we can assume that $\Lambda(\mathcal{G}_k)$ is fully convergent).

We now come to the proof of Lemma 4.5. We are thus given a U-prefix automaton $\mathcal{A} = (\Sigma, Q, Q_{\mathsf{imp}}, \Delta)$, and a scheme $\mathcal{G}$ generating a tree T, where $\Sigma_{\mathcal{G}} \subseteq \Sigma$; our goal is to create a scheme $\mathcal{G}_{\mathcal{A}}$ that generates the tree $\mathcal{A}(T)$. As a first step, we create a finite tree transducer $\mathcal{T}$ that converts T into a tree containing all runs of $\mathcal{A}$ on all subtrees of T. Let us write $Q = \{p_1, \dots, p_{|Q|}\}$. As $\mathcal{T}$ we take $(\Sigma_{\mathcal{G}}, r_{\max}(\mathcal{G}), Q \cup \{q_0, \top\}, q_0, \delta)$, where $q_0 \not\in Q$ is a fresh state, and δ is defined as follows. For $q \in Q$, $a \in \Sigma_{\mathcal{G}}$, and $r \leq r_{\max}(\mathcal{G})$ we take

$$\delta(q, a, r) = \mathsf{nd}\langle q\langle \mathsf{x}_{1,q_{11}}, \dots, \mathsf{x}_{r,q_{1r}}\rangle, \dots, q\langle \mathsf{x}_{1,q_{k1}}, \dots, \mathsf{x}_{r,q_{kr}}\rangle\rangle\,,$$

where $(q, a, q_{11}, \dots, q_{1r}), \dots, (q, a, q_{k1}, \dots, q_{kr})$ are all elements of Δ being of length $r+2$ and having q and a on the first two coordinates. Moreover, for $a \in \Sigma_{\mathcal{G}}$ and $r \leq r_{\max}(\mathcal{G})$ (and for a special letter "?") we take

$$\delta(q_0, a, r) = a\langle \mathsf{x}_{1,q_0}, \dots, \mathsf{x}_{r,q_0}, ?\langle \delta(p_1, a, r)\rangle, \dots, ?\langle \delta(p_{|Q|}, a, r)\rangle\rangle\,, \qquad \text{and}$$
$$\delta(\top, a, r) = \top\langle\rangle\,.$$

We see that $\mathcal{T}(T)$ contains all nodes of the original tree T. Additionally, below every node u coming from T we have $|Q|$ new children labeled by ?, such that subtrees starting below these children describe runs of $\mathcal{A}$ on $T{\restriction}_u$, starting in particular states. More precisely, when u has r children in T, for every $i \in \{1, \dots, |Q|\}$ there is a bijection between trees U in $\mathcal{L}(\mathcal{T}(T){\restriction}_{u(r+i)1})$ and runs ρ of $\mathcal{A}$ on $T{\restriction}_u$ such that $\rho(\varepsilon) = p_i$. The label of every node v in such a tree U contains the state assigned by ρ to v, where U contains exactly all nodes to which ρ assigns a state from Q, and all minimal nodes to which ρ assigns $\top$ (i.e., such that ρ does not assign $\top$ to their parents). Recall that by definition ρ can assign a state from Q only to a finite prefix of the tree $T{\restriction}_u$, which corresponds to the fact that $\mathcal{L}(\mathcal{T}(T){\restriction}_{u(r+i)1})$ contains only finite trees.

Actually, we need to consider a transducer $\mathcal{T}'$ obtained from $\mathcal{T}$ by a slight modification: we replace the letter q appearing in $\delta(q, a, r)$ by 1 if $q \in Q_{\mathsf{imp}}$, and by 0 if $q \not\in Q_{\mathsf{imp}}$. Then, for a node u of T having r children, and for $i \in \{1, \dots, |Q|\}$, we have the following equivalence: $\mathsf{SUP}_{\{1\}}(\mathcal{T}'(T){\restriction}_{u(r+i)})$ holds if and only if for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}$ on $T{\restriction}_u$ that assigns p_i to the root of $T{\restriction}_u$, and such that for at least n nodes v it holds that $\rho_n(v) \in Q_{\mathsf{imp}}$.

We now apply Fact 4.3 to $\mathcal{G}$ and $\mathcal{T}'$; we obtain a scheme $\mathcal{G}_{\mathcal{T}'}$ that generates the tree $\mathcal{T}'(T)$. Then, we apply Fact 4.2 (SUP reflection) to $\mathcal{G}_{\mathcal{T}'}$, which gives us a scheme $\mathcal{G}'$. The tree T' generated by $\mathcal{G}'$ has the same shape as $\mathcal{T}'(T)$, but in the label of every node v (originally having label other than nd) there is additionally written a set $\mathcal{U}$ containing these sets $A \subseteq \Sigma_{\mathcal{G}_{\mathcal{T}'}}$ for which $\mathsf{SUP}_A(\mathcal{L}(T{\restriction}_v))$ holds. Next, using Fact 4.1 (logical reflection) $2|Q|$ times, we annotate every node u of T', having r' children, by logical values of the following properties, for $i = 1, \dots, |Q|$:

- whether $r' \geq |Q|$ and $\mathcal{L}(T'{\restriction}_{u(r'-|Q|+i)1})$ is nonempty, and
- whether $r' \geq |Q|$ and the label $(a, \mathcal{U})$ of node $u(r' - |Q| + i)$ in T' satisfies $\{1\} \in \mathcal{U}$.

Clearly both these properties can be expressed in MSO. For nodes u coming from T, the first property holds when there is a run of $\mathcal{A}$ on $T{\restriction}_u$ that assigns p_i to the root of $T{\restriction}_u$, and the second property holds when for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}$ on $T{\restriction}_u$ that assigns p_i to the root of $T{\restriction}_u$, and such that for at least n nodes w it holds that $\rho_n(w) \in Q_{\mathsf{imp}}$. Let $\mathcal{G}''$ be the scheme generating the tree T'' containing these annotations.

Finally, we create $\mathcal{G}_{\mathcal{A}}$ by slightly modifying $\mathcal{G}''$: we replace every node constructor $(a, \mathcal{U}, \sigma_1, \tau_1, \dots, \sigma_{|Q|}, \tau_{|Q|})\langle P_1, \dots, P_{r+|Q|}\rangle$ with $(a, f)\langle P_1, \dots, P_r\rangle$, where $f \colon Q \to \{0, 1, 2\}$ is such that $f(p_i) = 2$ if $\tau_i = \mathsf{tt}$, and $f(p_i) = 1$ if $\sigma_i = \mathsf{tt}$ but $\tau_i = \mathsf{ff}$, and $f(p_i) = 0$ otherwise, for all $i \in \{1, \dots, |Q|\}$ (we do not do anything with node constructors of arity smaller than $|Q|$). As a result, only the nodes coming from T remain, and they are appropriately relabeled.

5. Extensions

In this section we give a few possible extensions of our main theorem, saying that we can evaluate MSO+$\mathsf{U}^{\mathsf{fin}}$ sentences on trees generated by recursion schemes. First, we notice that our solution actually proves a stronger result: logical reflection for MSO+$\mathsf{U}^{\mathsf{fin}}$.

Theorem 5.1. *For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *sentence φ and every scheme $\mathcal{G}$ generating a tree T one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node u is labeled by a 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.*

Proof. In the proof of Theorem 1.1 we have constructed a nested U-prefix automaton $\mathcal{A}$ equivalent to φ, and then a scheme $\mathcal{G}_\mathcal{A}$ that generates the tree $\mathcal{A}(T)$. In every node u of $\mathcal{A}(T)$ it is written whether $T\restriction_u$ satisfies φ. Moreover, labels of $\mathcal{A}(T)$ contain also original labels coming from T. Thus in order to obtain $\mathcal{G}_\varphi$ it is enough to appropriately relabel node constructors appearing in $\mathcal{G}_\mathcal{A}$. □

In Theorem 5.1, the sentence φ talks only about the subtree starting in u. One can obtain a stronger version of logical reflection (Theorem 5.2), where φ is a formula allowed to talk about u in the context of the whole tree. This version can be obtained as a simple corollary of Theorem 5.1 by using the same methods as in Broadbent et al. [BCOS10, Proof of Corollary 2]. As shown on page 18, it is also an immediate consequence of our next theorem (Theorem 5.3).

Theorem 5.2. *For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula $\varphi(\mathsf{X})$ with one free variable X and every scheme $\mathcal{G}$ generating a tree T, one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node u is labeled by a 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 with X valuated to $\{u\}$, and* ff *otherwise.*

Carayol and Serre [CS12] show one more property of MSO and schemes, called effective selection. This time we are given an MSO sentence ψ of the form $\exists\mathsf{X}.\varphi$. Assuming that ψ is satisfied in the tree T generated by a scheme $\mathcal{G}$, one wants to compute an example set X of nodes of T, such that φ is true in T with the variable X valuated to this set X. The theorem says that it is possible to create a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, in which nodes belonging to some such example set X are marked. We can show the same for MSO+$\mathsf{U}^{\mathsf{fin}}$.

Theorem 5.3. *For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula $\varphi(\mathsf{X})$ with one free variable $\mathsf{X} \in \mathcal{V}^{\mathsf{inf}}$ and every scheme $\mathcal{G}$ generating a tree T, if $\exists\mathsf{X}.\varphi(\mathsf{X})$ holds in T, then one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree T' of the same shape as T, and such that its every node u is labeled by a pair (a_u, b_u), where a_u is the label of u in T, and b_u belongs to $\{\mathsf{tt}, \mathsf{ff}\}$; when X is the set of nodes of T' having* tt *on the second coordinate of the label, φ is holds in T with X valuated to X.*

The proof of this theorem bases on the following lemma, which is also interesting in itself.

Lemma 5.4. *For every* MSO+$\mathsf{U}^{\mathsf{fin}}$ *formula φ and every scheme $\mathcal{G}$ generating a tree T one can construct a scheme $\mathcal{G}_+$ that generates a tree T' of the same shape as T, and an* MSO *formula φ_{MSO} (whose all free variables are also free in φ) such that for every valuation ν in T (defined at least for all free variables of φ) it holds that $T', \nu \models \varphi_{MSO}$ if and only if $T, \nu \models \varphi$. Moreover, the label of every node of T' contains as its part the label of that node in T.*

Proof. Recall that Lemma 3.4 gives us a nested U-prefix MSO automaton $\mathcal{A}_\varphi$ and MSO formulae $\xi_{\varphi,\tau}$ for all $\tau \in Pht_\varphi$ such that for every valuation ν in T (where T is now the fixed $(\Sigma_\mathcal{G}, r_{\max}(\mathcal{G}))$-tree generated by $\mathcal{G}$) it holds that $\mathcal{A}_\varphi(T), \nu \models \xi_{\varphi,\tau}$ if and only if $[T]^\nu_\varphi = \tau$.

Applying Lemmata 4.4 and 4.5 to components of the automaton $\mathcal{A}_\varphi$, out of the scheme $\mathcal{G}$ we can construct a scheme $\mathcal{G}_+$ that generates the tree $\mathcal{A}_\varphi(T)$.

Recall that $tv_\varphi(\tau)$ says whether φ is true in a tree having φ-phenotype τ, and consider the MSO formula

$$\varphi_{MSO} \equiv \bigvee_{\substack{\tau \in Pht_\varphi \\ tv_\varphi(\tau)}} \xi_{\varphi,\tau}\,.$$

By the above, for every valuation ν, it holds that $\mathcal{A}_\varphi(T), \nu \models \varphi_{MSO}$ if and only if $T, \nu \models \varphi$, as required in the thesis. □

Using the above lemma, we can easily deduce effective selection for MSO+$\mathsf{U}^{\mathsf{fin}}$ out of effective selection for MSO.

Proof of Theorem 5.3. Using effective selection for MSO (which is a theorem with the same statement as Theorem 5.3, but for the MSO logic [CS12]) for the formula φ_{MSO} and for the scheme $\mathcal{G}_+$ (created by Lemma 5.4) we obtain a scheme $\mathcal{G}'_\varphi$. It is almost as required: it generates a tree T' of the same shape as T (but with some additional parts of labels, added by $\mathcal{G}_+$), where additionally nodes of some set X are marked, so that φ_{MSO} holds in T' with X valuated to X. Lemma 5.4 implies that then also φ holds in T with X valuated to X. Thus, it is enough to modify node constructors of $\mathcal{G}'_\varphi$: out of every letter we leave only the original letter coming from $\mathcal{G}$, and the last component marking the set X, while we remove all the components added by $\mathcal{G}_+$. □

One may want to obtain an analogous theorem for $\mathsf{X} \in \mathcal{V}^{\mathsf{fin}}$, that is, for a sentence of the form $\exists_{\mathsf{fin}}\mathsf{X}.\varphi(X)$. It is, however, a special case of Theorem 5.3, which can be used with the sentence $\exists\mathsf{X}'.\exists_{\mathsf{fin}}\mathsf{X}.\mathsf{X} \subseteq \mathsf{X}' \land \mathsf{X}' \subseteq \mathsf{X} \land \varphi(\mathsf{X})$. We remark, though, that the version of Theorem 5.3 for $\mathsf{X} \in \mathcal{V}^{\mathsf{fin}}$ is actually also a corollary of Theorem 1.1, because there are only countably many finite sets X, so we may try one after another, until we find some set for which φ is satisfied; it is easy to hardcode a given set X in the formula (or in the scheme).

We now show how Theorem 5.2 follows from Theorem 5.3.

Proof of Theorem 5.2. We use Theorem 5.3 for

$$\varphi'(\mathsf{X}') \equiv \forall\mathsf{X}.sing(\mathsf{X}) \to (\mathsf{X} \subseteq \mathsf{X}' \to \varphi(\mathsf{X})) \land (\neg(\mathsf{X} \subseteq \mathsf{X}') \to \neg\varphi(\mathsf{X}))\,.$$

The only set X' for which φ' is true in a tree T is the set of all nodes u for which φ is true in T with X valuated to $\{u\}$. Thus the scheme $\mathcal{G}_{\varphi'}$ obtained from Theorem 5.3 satisfies the thesis of Theorem 5.2. □

Our algorithm for Theorem 1.1 has nonelementary complexity. This is unavoidable, as already model-checking of WMSO sentences on the infinite word over an unary alphabet is nonelementary. It would be interesting to find some other formalism for expressing unboundedness properties, maybe using some model of automata, for which the model-checking problem has better complexity. We leave this issue for future work.

Finally, we remark that in our solution we do not use the full power of the simultaneous unboundedness problem, we only use the single-letter case. On the other hand, it seems that MSO+$\mathsf{U}^{\mathsf{fin}}$ is not capable to express simultaneous unboundedness, only its single-letter case. Thus, another direction for a future work is to extend MSO+$\mathsf{U}^{\mathsf{fin}}$ to a logic that can actually express simultaneous unboundedness. As a possible candidate we see the qcMSO logic introduced in Kaiser, Lang, Leßenich, and Löding [KLLL15], in which simultaneous unboundedness is expressible.

References

[Aho68] Alfred V. Aho. Indexed grammars - an extension of context-free grammars. *J. ACM*, 15(4):647–671, 1968.

[BCCC96] 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.

[BCL08] Achim Blumensath, Thomas Colcombet, and Christof Löding. Logical theories and compatible operations. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 73–106. Amsterdam University Press, 2008.

[BCOS10] Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Recursion schemes and logical reflection. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 120–129. IEEE Computer Society, 2010.

[BK13] Christopher H. Broadbent and Naoki Kobayashi. Saturation-based model checking of higher-order recursion schemes. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 129–148. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[BO09] Christopher H. Broadbent and C.-H. Luke Ong. On global model checking trees generated by higher-order recursion schemes. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009.

[Boj04] 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.

[Boj11] Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.

[Boj14] 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.

[BPT16] Mikołaj Bojańczyk, Paweł Parys, and Szymon Toruńczyk. The MSO+U theory of (N, <) is undecidable. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 21:1–21:8. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[BT12] 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 fuer Informatik, 2012.

[CMvRZ15] Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, and Marc Zeitoun. A note on decidable separability by piecewise testable languages. In Adrian Kosowski and Igor Walukiewicz, editors, *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, volume 9210 of *Lecture Notes in Computer Science*, pages 173–185. Springer, 2015.

[CPSW15] 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 fuer Informatik, 2015.

[CPSW16] 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.

[CS12] Arnaud Carayol and Olivier Serre. Collapsible pushdown automata and labeled recursion schemes: Equivalence, safety and effective selection. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 165–174. IEEE Computer Society, 2012.

[Dam82] Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.

[EJ91] 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.

[FV59] Solomon Feferman and Robert Lawson Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47(1):57–103, 1959.

[GK10] Tobias Ganzow and Łukasz Kaiser. New algorithm for weak monadic second-order logic on inductive structures. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2010.

[Had12] Axel Haddad. IO vs OI in higher-order recursion schemes. In Dale Miller and Zoltán Ésik, editors, *Proceedings 8th Workshop on Fixed Points in Computer Science, FICS 2012, Tallinn, Estonia, 24th March 2012.*, volume 77 of *EPTCS*, pages 23–30, 2012.

[HKO16] 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.

[HM12] Szczepan Hummel and Michał Skrzypczak. The topological complexity of MSO+U and related automata models. *Fundam. Inform.*, 119(1):87–111, 2012.

[HMOS08] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 452–461. IEEE Computer Society, 2008.

[KLLL15] Łukasz Kaiser, Martin Lang, Simon Leßenich, and Christof Löding. A unified approach to boundedness properties in MSO. In Kreutzer [Kre15], pages 441–456.

[KNU02] 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.

[KO09] 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.

[Kob11] Naoki Kobayashi. A practical linear time algorithm for trivial automata model checking of higher-order recursion schemes. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2011.

[Kob13] Naoki Kobayashi. Model checking higher-order programs. *J. ACM*, 60(3):20:1–20:62, 2013.

[Kre15] Stephan Kreutzer, editor. *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[Lä68] Hans Läuchli. A decision procedure for the weak second order theory of linear order. *Studies in Logic and the Foundations of Mathematics*, 50:189–197, 1968.

[NO14] Robin P. Neatherway and C.-H. Luke Ong. TravMC2: Higher-order model checking for alternating parity tree automata. In Neha Rungta and Oksana Tkachuk, editors, *2014 International Symposium on Model Checking of Software, SPIN 2014, Proceedings, San Jose, CA, USA, July 21-23, 2014*, pages 129–132. ACM, 2014.

[Ong06] 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.

[Par17] Paweł Parys. The complexity of the diagonal problem for recursion schemes. In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[Par18a] 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 fuer Informatik, 2018.

[Par18b] Paweł Parys. A type system describing unboundedness. Submitted, 2018.

[PT16] Paweł Parys and Szymon Toruńczyk. Models of lambda-calculus and the weak MSO logic. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 11:1–11:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[RNO14] Steven J. Ramsay, Robin P. Neatherway, and C.-H. Luke Ong. A type-directed abstraction refinement approach to higher-order model checking. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 61–72. ACM, 2014.

[She75] Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, 102(3):379–419, 1975.

[SW14] Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Inf. Comput.*, 239:340–355, 2014.

[SW15a] Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. In Kreutzer [Kre15], pages 229–243.

[SW15b] Sylvain Salvati and Igor Walukiewicz. Using models to model-check recursive schemes. *Logical Methods in Computer Science*, 11(2), 2015.

[SW16] Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Mathematical Structures in Computer Science*, 26(7):1304–1350, 2016.

[Zet15] 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.

Appendix A. Proof of Fact 4.3

As already said, Fact 4.3 follows easily from the equivalence between schemes and collapsible pushdown systems. We do not even need to know a full definition of these systems. Let us recall those fragments that are relevant for us.

For every $n \in \mathbb{N}$, and every finite set Γ containing a distinguished initial symbol $\perp \in \Gamma$, there are defined:

- a set $\mathcal{PD}_{n,\Gamma}$ of collapsible pushdowns of order n over stack alphabet Γ,
- an initial pushdown $\perp_n \in \mathcal{PD}_{n,\Gamma}$,
- a finite set $Op_{n,\Gamma}$ of operations on these pushdowns, where every $op \in Op_{n,\Gamma}$ is a partial function from $\mathcal{PD}_{n,\Gamma}$ to $\mathcal{PD}_{n,\Gamma}$, and
- a function $top\colon \mathcal{PD}_{n,\Gamma} \to \Gamma$ (returning the topmost symbol of a pushdown).

We assume that $Op_{n,\Gamma}$ contains the identity operation id, mapping every element of $\mathcal{PD}_{n,\Gamma}$ to itself.

Having the above, we define a *collapsible pushdown system* (a *CPS* for short) as a tuple $\mathcal{C} = (Q, q_0, n, \Gamma, \delta)$, where Q is a set of states, $q_0 \in Q$ is an initial state, $n \in \mathbb{N}$ is an order, Γ is a finite stack alphabet, and $\delta\colon Q \times \Gamma \to (Q \times Op_{n,\Gamma}) \uplus (\Sigma \times Q^*)$ is a transition function (where Σ is some alphabet). A *configuration* of $\mathcal{C}$ is a pair $(q, s) \in Q \times \mathcal{PD}_{n,\Gamma}$. A configuration (p, t) is a *successor* of (q, s), written $(q, s) \to_{\mathcal{C}} (p, t)$, if $\delta(q, top(s)) = (p, op)$ and $op(s) = t$. We define when a tree is generated by $\mathcal{C}$ from (q, s), by coinduction:

- if $(q, s) \to^*_{\mathcal{C}} (p, t)$, and $\delta(p, top(t)) = (a, q_1, \dots, q_r) \in \Sigma \times Q^*$, and trees $T_1, \dots, T_r$ are generated by $\mathcal{C}$ from $(q_1, t), \dots, (q_r, t)$, respectively, then $a\langle T_1, \dots, T_r\rangle$ is generated by $\mathcal{C}$ from (q, s),
- if there is no (p, t) such that $(q, s) \to^*_{\mathcal{C}} (p, t)$ and $\delta(p, top(t)) \in \Sigma \times Q^*$, then $\omega\langle\rangle$ is generated by $\mathcal{C}$ from (q, s).

Notice that for every configuration (q, s) there is at most one configuration (p, t) such that $(q, s) \to^*_{\mathcal{C}} (p, t)$ and $\delta(p, top(t)) \in \Sigma \times Q^*$; in consequence exactly one tree is generated by $\mathcal{C}$ from every configuration. While talking about the tree generated by $\mathcal{C}$, without referring to a configuration, we mean generating from the initial configuration $(q_0, \perp_n)$.

We say that a CPS is *fully convergent* (from a configuration (q, s)) if it generates (from (q, s)) a tree without using the second item of the definition. More formally: we consider the CPS $\mathcal{C}_{-\omega}$ obtained from $\mathcal{C}$ by replacing ω with some other letter ω' (in all transitions), and we say that $\mathcal{C}$ is fully convergent (from (q, s)) if $\mathcal{C}_{-\omega}$ generates (from (q, s)) a tree without ω-labeled nodes. We have the following fact.

Fact A.1 ([HMOS08]). *For every scheme $\mathcal{G}$ one can construct a CPS $\mathcal{C}$ that generates the tree generated by $\mathcal{G}$ and, conversely, for every CPS $\mathcal{C}$ one can construct a scheme $\mathcal{G}$ that generates the tree generated by $\mathcal{C}$. Both translations preserve the property of being fully convergent.*[2] □

In Fact 4.3 we are given a finite tree transducer $\mathcal{T} = (\Sigma, r_{\max}, P, p_0, \delta_{\mathcal{T}})$, and a scheme $\mathcal{G}$ generating a $(\Sigma, r_{\max})$-tree T, and we want to construct a scheme $\mathcal{G}_{\mathcal{T}}$ that generates the tree $\mathcal{T}(T)$. By Fact 2.1 we can assume that $\mathcal{G}$ is fully convergent. As a first step, we translate $\mathcal{G}$ to a fully convergent CPS $\mathcal{C} = (Q, q_0, n, \Gamma, \delta_{\mathcal{C}})$ generating T, using Fact A.1.

Then, we create a CPS $\mathcal{C}_{\mathcal{T}} = (R, (q_0, p_0), n, \Gamma, \delta)$ by combining $\mathcal{C}$ with $\mathcal{T}$. Its set of states R contains states of two kinds: pairs $(q, p) \in Q \times P$, and pairs (q, U) where $q \in Q$

[2] Clearly only a fully-convergent CPS/scheme can generate a tree without ω-labeled nodes. Thus it is easy to preserve the property of being fully convergent: we can replace all appearances of ω by some fresh letter ω', switch to the other formalism, and then replace ω' back by ω.

and U is a subterm of $\delta_{\mathcal{T}}(p,a,r)$ for some $(p,a,r) \in P \times \Sigma \times \{0,\dots,r_{\max}\}$. We define the transitions as follows:

- if $\delta_{\mathcal{C}}(q,\chi) = (q', op) \in Q \times Op_{n,\Gamma}$, then $\delta((q,p),\chi) = ((q',p), op)$,
- if $\delta_{\mathcal{C}}(q,\chi) = (a,q_1,\dots,q_r) \in \Sigma \times Q^*$, then $\delta((q,p),\chi) = ((q,\delta_{\mathcal{T}}(p,a,r)), \mathsf{id})$,
- if $\delta_{\mathcal{C}}(q,\chi) \in \Sigma \times Q^*$, then $\delta((q, b\langle U_1,\dots,U_k\rangle),\chi) = (b,(q,U_1),\dots,(q,U_k))$,
- if $\delta_{\mathcal{C}}(q,\chi) = (a,q_1,\dots,q_r) \in \Sigma \times Q^*$ and $i \in \{1,\dots,r\}$, then $\delta((q,\mathsf{x}_{i,p}),\chi) = ((q_i,p),\mathsf{id})$, and
- all other transitions are irrelevant, and can be defined arbitrarily.

It is easy to prove by coinduction that if $\mathcal{C}$ is fully convergent from some configuration (q,s), then, for every state $p \in P$, $\mathcal{C}_{\mathcal{T}}$ generates $\mathcal{T}_p(T_{q,s})$ from $((q,p),s)$, where $T_{q,s}$ is the tree generated by $\mathcal{C}$ from (q,s). Indeed, because $\mathcal{C}$ is fully convergent from (q,s), for some (q',t) we have $(q,s) \to_{\mathcal{C}}^* (q',t)$ and $\delta(q', top(t)) = (a,q_1,\dots,q_r) \in \Sigma \times Q^*$. In such a situation $((q,p),s) \to_{\mathcal{C}_{\mathcal{T}}}^* ((q',p),t)$ (where we use transitions of the first kind). From $((q',p),t)$ the CPS $\mathcal{C}_{\mathcal{T}}$ uses a transition of the second kind, and then starts generating the tree $\delta_{\mathcal{T}}(p,a,r)$ until a variable is reached (using transitions of the third kind). When a variable $\mathsf{x}_{i,p'}$ is reached, $\mathcal{C}_{\mathcal{T}}$ enters the configuration $((q_i,p'),t)$ (a transition of the fourth kind), which, by the assumption of coinduction, means that it continues by generating the tree $\mathcal{T}_{p'}(T_{q_i,t})$, where $T_{q_i,t}$ is the tree generated by $\mathcal{C}$ from (q_i,t).

In particular we have that $\mathcal{C}_{\mathcal{T}}$ generates $\mathcal{T}(T)$. At the end we translate $\mathcal{C}_{\mathcal{T}}$ to a scheme $\mathcal{G}_{\mathcal{T}}$ generating the same tree, using again Fact A.1.