Recursion Schemes and the WMSO+U Logic*

Paweł Parys

University of Warsaw
Warsaw, Poland
parys@mimuw.edu.pl

Abstract

We study the weak MSO logic extended by the unbounding quantifier (WMSO+U), expressing the fact that there exist arbitrarily large finite sets satisfying a given property. We prove that it is decidable whether the tree generated by a given higher-order recursion scheme satisfies a given sentence of WMSO+U.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases higher-order recursion schemes, intersection types, WMSO+U logic, boundedness

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.54

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 [16, 22, 29, 24]. This formalism is equivalent via direct translations to simply-typed λY-calculus [37]. Collapsible pushdown systems [20] and ordered tree-pushdown systems [13] are other equivalent formalisms. Schemes cover some other models such as indexed grammars [1] and ordered multi-pushdown automata [8].

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 [29, 20, 25, 35], and also some extensions like global model checking [11], logical reflection [9], effective selection [12], existence of λ-calculus model [36]. 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 [24, 23, 10, 33, 28] (most tools cover only a fragment of MSO, though).

Recently, an interest arisen in model-checking trees generated by schemes against properties not expressible in the MSO logic. These are properties expressing boundedness and unboundedness of some quantities. More precisely, it was shown that the *diagonal problem* for schemes is decidable [19, 14, 31]. 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 [39], and the problem of separability by piecewise testable languages [15].

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

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 54; pp. 54:1–54:25
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper we show a result of a more general style. Instead of considering a particular property, like in the diagonal problem, we consider a logic capable to express properties talking about boundedness. More precisely, we choose the WMSO+U logic. This logic extends WMSO (a fragment of MSO in which one can quantify only over finite sets) by the unbounding quantifier, U [3]. A formula using this quantifier, $\mathsf{U}X.\varphi$, says that φ holds for arbitrarily large finite sets X. The WMSO+U logic was widely considered in the context of infinite words [4] and infinite trees [18, 7, 5].

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

▶ **Theorem 1.** *It is decidable whether the tree generated by a given scheme satisfies a given* WMSO+U *sentence.*

In our solution, we depend on several earlier results. First, we translate WMSO+U formulae to an equivalent automata model using the notion of logical types (aka. composition method) following a long series of previous work (some selection: [17, 38, 27, 2, 18, 32]). Second, we use the logical-reflection property of schemes [9]. 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, from our previous work on the diagonal problem [30, 31], we deduce an analogous property for the diagonal problem, which we call *diagonal reflection* (Theorem 6): given a scheme $\mathcal{G}$ we can construct a scheme $\mathcal{G}_{diag}$ generating the same tree as $\mathcal{G}$, where every node is additionally annotated by the solution of the diagonal problem in the subtree starting in this node. We believe that Theorem 6 is a contribution of independent interest. 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 [20].

We remark that the model-checking problem for the full MSO logic (equipped with quantification over infinite sets) combined with the U quantifier is undecidable already over the infinite word without labels [6], so even more over all fancy trees that can be generated by higher-order recursion schemes. For this reason it is necessary to restrict the quantification to finite sets.

Our paper is structured as follows. In Section 2 we introduce all necessary definitions. In Section 3 we show how to translate WMSO+U sentences to automata. In Section 4 we give a theorem concerning diagonal reflection. Next, in Section 5, we finish the proof of the main theorem. We conclude in Section 6 by listing some possible extensions of our results.

2 Preliminaries

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

While defining λ-terms we assume an infinite set of letters Σ (we use unranked letters; this subsumes the setting of ranked letters), and a set of variables $\mathcal{V} = \{x^\alpha, y^\beta, z^\gamma\}$ containing infinitely many variables of every sort (sort of a variable is written in superscript). *Infinitary λ-terms* (or just *λ-terms*) are defined by coinduction, according to the following rules:

- node constructor—if $a \in \Sigma$, and $K_1^{\mathsf{o}}, \dots, K_r^{\mathsf{o}}$ are λ-terms, then $(a\langle K_1^{\mathsf{o}}, \dots, K_r^{\mathsf{o}}\rangle)^{\mathsf{o}}$ is a λ-term,
- variable—every variable $x^\alpha \in \mathcal{V}$ is a λ-term,
- application—if $K^{\alpha\to\beta}$ and L^α are λ-terms, then $(K^{\alpha\to\beta}\, L^\alpha)^\beta$ is a λ-term, and
- λ-binder—if K^β is a λ-term and x^α is a variable, then $(\lambda x^\alpha.K^\beta)^{\alpha\to\beta}$ is a λ-term.

We naturally identify λ-terms differing only in names of bound variables. We often omit the sort annotations of λ-terms, but we keep in mind that every λ-term (and every variable) has a fixed sort. Free variables and subterms of a λ-term, as well as β-reductions, are defined as usually. A λ-term K is *closed* if it has no free variables. We restrict ourselves to those λ-terms for which the set of sorts of all subterms is finite.

Trees; Böhm Trees. A *tree* is defined as a λ-term that is built using only node constructors, that is, not using variables, applications, nor λ-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 v is a node of T_i for some $i \in \{1, \dots, r\}$, labeled by b, then iv 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. By $T{\restriction}_v$ we denote the subtree of T starting in the node v, defined as one expects. For a (usually finite) subset Σ_0 of Σ, and for $r_{\max} \in \mathbb{N}$, a *$(\Sigma_0, r_{\max})$-tree* is a tree in which all node labels belong to Σ_0, and in which every node has at most $r_{\max}$ children.

We consider Böhm trees only for closed λ-terms of sort o. For such a λ-term K, its *Böhm tree* is constructed by coinduction, as follows: if there is a sequence of β-reductions from K to a λ-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 β-reductions from K, then $\omega\langle\rangle$ is a Böhm tree of K (where $\omega \in \Sigma$ is a fixed letter). It is folklore that every closed λ-term of sort o has exactly one Böhm tree (the order in which β-reductions are performed does not matter); this tree is denoted by $BT(K)$.

A closed λ-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 λ-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 λ-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 $\mathcal{N} \subseteq \mathcal{V}$ is a finite set of nonterminals, $\mathcal{R}$ is a function that maps every nonterminal $N \in \mathcal{N}$ to a finite λ-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 λ-term K whose free variables are contained in $\mathcal{N}$, we define the infinitary λ-term *generated by* $\mathcal{G}$ from K, 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 λ-term of the same sort as K. The infinitary λ-term *generated 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 subset of Σ 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 generating fully-convergent λ-terms, which is possible due to the following standard result.

▶ **Fact 2** ([34, page 14]). *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.** Consider the scheme $\mathcal{G}_1 = (\{\mathsf{M}^{\mathsf{o}}, \mathsf{N}^{\mathsf{o}\to\mathsf{o}}\}, \mathcal{R}, \mathsf{M})$, where

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

We obtain $\Lambda(\mathcal{G}_1) = K\,(c\langle\rangle)$, where K is the unique λ-term such that $K = \lambda x^{\mathsf{o}}.a\langle x, K\,(b\langle x\rangle)\rangle$. 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 $i \geq 1$.

WMSO+U. For technical convenience, we use a variant of WMSO+U in which there are no first-order variables. It is easy to translate a formula from any standard syntax of WMSO+U to ours (at least when the maximal arity of considered trees is fixed). In the syntax of WMSO+U we have the following constructions:

$$\varphi ::= a(X) \mid X \curlywedge_i Y \mid X \subseteq Y \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi' \mid \exists_{\mathsf{fin}} X.\varphi' \mid \mathsf{U}X.\varphi' \qquad \text{where } a \in \Sigma, i \in \mathbb{N}_+.$$

We evaluate formulae of WMSO+U in Σ-labeled trees. Set variables are interpreted as finite sets of nodes, and the semantics of formulae is defined as follows:

- $a(X)$ holds when every node in X is labeled by a,
- $X \curlywedge_i Y$ holds when both X and Y are singletons, and the unique node in Y is the i-th child of the unique node in X,
- $X \subseteq Y$, $\varphi_1 \wedge \varphi_2$, and $\neg\varphi'$ are defined as expected,
- $\exists_{\mathsf{fin}} X.\varphi'$ holds when φ' holds for some finite set of nodes X, and
- $\mathsf{U}X.\varphi'$ holds when for every $n \in \mathbb{N}$, φ' holds for some finite set of nodes X of cardinality at least n.

3 Nested U-Prefix Automata

In this section we give a definition of nested U-prefix automata, a formalism equivalent to the WMSO+U logic. A U-*prefix automaton* is a pair $\mathcal{A} = (Q, Q_{\mathsf{imp}}, \Delta)$, where 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 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 v such that $\rho(v) \in Q$, and
- for every node v of T, with label a and r children, it holds that either $\rho(v) = \top = \rho(v1) = \dots = \rho(vr)$ or $(\rho(v), a, \rho(v1), \dots, \rho(vr)) \in \Delta$.

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 v is labeled by a function $f_v \colon Q \to \{0, 1, 2\}$, which 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 $T{\restriction_v}$ that assigns q to the root of $T{\restriction_v}$, 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 $T{\restriction_v}$ that assigns q to the root of $T{\restriction_v}$;
- 0, if none of the above holds.

By the *output alphabet* of $\mathcal{A}$ we mean the set of functions $\Sigma^{\mathsf{out}}(\mathcal{A}) = \{0,1,2\}^Q$; we assume that $\{0,1,2\}^Q \subseteq \Sigma$.

A *nested* U*-prefix automaton* is a sequence $\mathcal{A} = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_k$ of U-prefix automata, where $k \geq 1$. We define $\mathcal{A}(T)$ to be $\mathcal{A}_k(\dots(\mathcal{A}_1(T))\dots)$. The output alphabet of $\mathcal{A}$, denoted $\Sigma^{\mathsf{out}}(\mathcal{A})$, equals $\Sigma^{\mathsf{out}}(\mathcal{A}_k)$. The key property is that these automata can check properties expressed in WMSO+U (actually, they are equivalent to WMSO+U, but we need only the one direction here).

▶ **Lemma 3.** *Let $\Sigma_0 \subseteq \Sigma$ be a finite fragment of the alphabet, and let $r_{\max} \in \mathbb{N}$. Then for every* WMSO+U *sentence φ we can construct a nested* U*-prefix automaton $\mathcal{A}$, and a subset $\Sigma_{\mathsf{F}} \subseteq \Sigma^{\mathsf{out}}(\mathcal{A})$ such that for every $(\Sigma_0, r_{\max})$-tree T, it holds that T satisfies φ if and only if the root of $\mathcal{A}(T)$ is labeled by a letter in Σ_{F}.*

We remark that Bojańczyk and Toruńczyk [7] 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, but the components are of different form.

Recall that our aim is to evaluate φ in a tree T generated by a particular recursion scheme $\mathcal{G}$, so the restriction to $(\Sigma_0, r_{\max})$-trees is not harmful: as $(\Sigma_0, r_{\max})$ we are going to take $(\Sigma_{\mathcal{G}}, r_{\max}(\mathcal{G}))$.

We now come to the proof of Lemma 3. We notice that due to the nested structure, our automata are quite close to the logic. Nondeterminism on particular levels of the automaton may realize the choices done by particular quantifiers of the formula. Moreover, in effect of applying an automaton we check whether something is unbounded, which corresponds to the U quantifiers. As states of the automaton we will take *phenotypes* (aka. logical types), which are defined next.

Fix some finite set $\mathcal{F}$ of variables, such that all variables appearing in WMSO+U formulae under consideration come from this set. Let φ be a formula of WMSO+U, let T be a tree, and let ν be a valuation assigning finite sets of nodes of T to variables from $\mathcal{F}$. 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 $a(X)$ (for some symbol $a \in \Sigma$) or $X \subseteq Y$ 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 $X \curlywedgedownarrow_i Y$, then $[T]^\nu_\varphi$ equals:
  - tt if $T, \nu \models \varphi$,
  - empty if $\nu(X) = \nu(Y) = \emptyset$,
  - root if $\nu(X) = \emptyset$ and $\nu(Y) = \{\varepsilon\}$, and
  - ff otherwise,
- if $\varphi = (\psi_1 \wedge \psi_2)$, then $[T]^\nu_\varphi = ([T]^\nu_{\psi_1}, [T]^\nu_{\psi_2})$,
- if $\varphi = (\neg\psi)$, then $[T]^\nu_\varphi = [T]^\nu_\psi$, and
- if $\varphi = \exists_{\mathsf{fin}} X.\psi$ or $\varphi = \mathsf{U}X.\psi$, then

  $$[T]^\nu_\varphi = \left(\{\sigma \mid \exists X^T.\, [T]^{\nu[X\mapsto X^T]}_\psi = \sigma\}, \{\sigma \mid \forall n.\, \exists X^T.\, [T]^{\nu[X\mapsto X^T]}_\psi = \sigma \wedge |X^T| \geq n\}\right),$$

  where X^T ranges over finite sets of nodes of T and n ranges over $\mathbb{N}$.

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, 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 v, by $\nu\restriction_v$ we mean the valuation that restricts ν to the subtree starting at v, that is, maps every variable $X \in \mathcal{F}$ to $\{w \mid vw \in \nu(X)\}$.

▶ **Lemma 4** (cf. [18, 32])**.** *For every letter $a \in \Sigma$, every $r \in \mathbb{N}$, and every formula φ, one can compute a function $Comp_{a,r,\varphi} \colon \mathcal{P}(\mathcal{F}) \times (Pht_\varphi)^r \to Pht_\varphi$ such that for every tree T whose root has label a and r children, and for every valuation ν,*

$$[T]^\nu_\varphi = Comp_{a,r,\varphi}(\{X \in \mathcal{F} \mid \varepsilon \in \nu(X)\}, [T\restriction_1]^{\nu\restriction_1}_\varphi, \ldots, [T\restriction_r]^{\nu\restriction_r}_\varphi)\,.$$

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

When φ is of the form $b(X)$ or $X \subseteq Y$, 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 $\varphi = b(X)$ as $Comp_{a,r,\varphi}(R, \tau_1, \ldots, \tau_r)$ we take tt when $\tau_i = \mathsf{tt}$ for all $i \in \{1, \ldots, r\}$ and either $a = b$ or $X \notin R$. For $\varphi = (X \subseteq Y)$ the last part of the condition is replaced by "if $X \in R$ then $Y \in R$".

Next, suppose that $\varphi = (X \downarrow_k Y)$. Then as $Comp_{a,r,\varphi}(R, \tau_1, \ldots, \tau_r)$ we take

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

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

When $\varphi = (\neg\psi)$, we simply take $Comp_{a,r,\varphi} = Comp_{a,r,\psi}$, and when $\varphi = (\psi_1 \wedge \psi_2)$, as $Comp_{a,r,\varphi}(R, (\tau_1^1, \tau_1^2), \ldots, (\tau_r^1, \tau_r^2))$ we take the pair of $Comp_{a,r,\psi_i}(R, \tau_1^i, \ldots, \tau_r^i)$ for $i \in \{1, 2\}$.

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

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

The two possibilities, $R \cup \{X\}$ and $R \setminus \{X\}$, correspond to the fact that when quantifying over X, the root of T may be either taken to X or not. The second coordinate is computed correctly due to the pigeonhole principle: if for every n we have a set X_n^T 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 $j \in \{1, \ldots, r\}$ such that the sets contain unboundedly many descendants of j. ◀

We now concentrate on phenotypes under the valuation $\nu_\emptyset$ that maps every variable to the empty set.

▶ **Lemma 5.** *Let $\Sigma_0 \subseteq \Sigma$ be a finite fragment of the alphabet, and let $r_{\max} \in \mathbb{N}$. Then for every WMSO+U formula φ we can construct a nested U-prefix automaton $\mathcal{A}$, and a function $f \colon \Sigma^{\mathsf{out}}(\mathcal{A}) \to Pht_\varphi$ such that for every $(\Sigma_0, r_{\max})$-tree T the root of $\mathcal{A}(T)$ is labeled by a letter η such that $f(\eta) = [T]^{\nu_\emptyset}_\varphi$.*

Proof. Induction on the size of φ. Since all variables are mapped to the empty set, if φ is of the form $a(X)$ or $X \subseteq Y$, then the φ-phenotype of every tree is $\mathtt{tt}$. Thus every $\mathcal{A}$ works fine, only f has to map whole its output alphabet to $\mathtt{tt}$. Similarly, if $\varphi = (X \curlywedge_i Y)$, the φ-phenotype is always empty. For $\varphi = (\neg\psi)$ the situation is also trivial: we can directly use the induction assumption since $[T]^{\nu_\emptyset}_\varphi = [T]^{\nu_\emptyset}_\psi$.

Suppose that $\varphi = (\psi_1 \wedge \psi_2)$. Then from the induction assumption we have automata $\mathcal{B}$ and $\mathcal{C}$ (together with functions g and h) computing ψ_1-phenotypes and ψ_2-phenotypes. It is a routine to alter $\mathcal{B}$ so that from the label of every node v in the output tree $\mathcal{B}(T)$ one can read the original label of v from T (this amounts to adding Σ_0 to the state set of every layer, together with appropriate transitions). We also alter $\mathcal{C}$ so that it reads the output alphabet of $\mathcal{B}$ instead Σ_0; it bases its operation on the original labels from T that can be recovered from the letters, and it copies the information about ψ_1-phenotypes, so that it can be read at the end. After these modifications, we take $\mathcal{A} = \mathcal{B} \circ \mathcal{C}$. Then from the label of the root of $\mathcal{A}(T)$ one can read both $[T]^{\nu_\emptyset}_{\psi_1}$ (copied from the output of $\mathcal{B}$) and $[T]^{\nu_\emptyset}_{\psi_2}$ (calculated by $\mathcal{C}$), so $[T]^{\nu_\emptyset}_\varphi$ can be determined.

Finally, suppose that $\varphi = \exists_{\mathsf{fin}} X.\psi$ or $\varphi = \mathsf{U} X.\psi$. By the induction assumption we have an automaton $\mathcal{B}$ and a function g such that for every node v of T, the root of $\mathcal{B}(T{\restriction_v})$ is labeled by a letter η_v such that $g(\eta_v) = [T{\restriction_v}]^{\nu_\emptyset}_\psi$. As before, we can also assume that there is a function h such that additionally $h(\eta_v)$ is the original label of v in T. Recall that $\mathcal{B}(T)$ has the same shape as T, and actually $(\mathcal{B}(T)){\restriction_v} = \mathcal{B}(T{\restriction_v})$ for every node v. We construct a new layer $\mathcal{A}'$, which calculates φ-phenotypes basing on ψ-phenotypes, and we take $\mathcal{A} = \mathcal{B} \circ \mathcal{A}'$. As the state set of $\mathcal{A}'$ 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 4. More precisely, for every $r \leq r_{\max}$, every $\eta \in \Sigma^{\mathsf{out}}(\mathcal{B})$, and all $((i_1,\sigma_1),\dots,(i_r,\sigma_r)) \in Q^r$ we have transitions

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

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

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

We notice that there is a direct correspondence between runs of $\mathcal{A}'$ and choices of a set of nodes X^T to which the variable X is mapped. The first coordinate of the state is set to 1 in nodes chosen to the set X^T. The second coordinate contains the ψ-phenotype under the valuation mapping X to X^T and every other variable to the empty set. In some nodes below the chosen set X^T we use the 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^T corresponds to the fact that the run of $\mathcal{A}'$ can assign non-$\top$ states only to a finite prefix of the tree. Moreover, the cardinality of X^T 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^T of nodes of T such that $[T]^{\nu_\emptyset[X \mapsto X^T]}_\psi = \sigma$ if and only if for some $i \in \{0,1\}$ there is a run of $\mathcal{A}'$ on $\mathcal{B}(T)$ that assigns (i,σ) to the root, and
- for every $n \in \mathbb{N}$ there exists a finite set X^T_n of nodes of T such that $[T]^{\nu_\emptyset[X \mapsto X^T_n]}_\psi = \sigma$ and $|X^T_n| \geq n$ if and only if for some $i \in \{0,1\}$ and for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}'$ on $\mathcal{B}(T)$ 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 $\mathcal{A}(T)$ we can determine $[T]^{\nu_\emptyset}_\varphi$. ◀

Now the proof of Lemma 3 follows easily. Indeed, when φ is a sentence (has no free variables), $[T]^{\nu_\emptyset}_\varphi$ determines whether φ holds in T. Thus it is enough to take the automaton $\mathcal{A}$ constructed in Lemma 5, and replace the function f by the set $\Sigma_{\mathsf{F}} = \{\eta \in \Sigma^{\mathsf{out}}(\mathcal{A}) \mid tv_\varphi(f(\eta))\}$.

4 Diagonal Reflection

The goal of this section is to justify the property of diagonal reflection (Theorem 6).

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 $\mathsf{Diag}_A(\mathcal{L})$, 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 diagonal 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 $\mathsf{nd} \in \Sigma$, denoting a nondeterministic choice. We write $T \to_{\mathsf{nd}} U$ if U is obtained from T by choosing some nd-labeled node u not having any nd-labeled ancestors, 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 $\#_{\mathsf{nd}}(U) = 0$ and $T \to^*_{\mathsf{nd}} U$.

▶ **Theorem 6** (diagonal reflection). *For every scheme $\mathcal{G}$ generating a tree T one can construct a scheme $\mathcal{G}_{diag}$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair $(a, \mathcal{D})$, where a is the label of v in T, and $\mathcal{D} = \{A \subseteq \Sigma_\mathcal{G} \mid \mathsf{Diag}_A(\mathcal{L}(T{\restriction_v}))\}$.*

While proving this theorem, we depend on our previous work on the diagonal problem [31]. We have developed there a type system, in which for a closed λ-term K we derive type judgments of the form $\vdash_{m,A} K : \hat\tau \triangleright c$, where

- m is a natural number,
- $A \subseteq \Sigma$ is the set of types for which we want to solve the diagonal problem (originally it was not written in the type judgment, but anyway the type system depends on this set),
- $\hat\tau$ comes from a finite set $\mathcal{TT}^\alpha_{m,A}$, depending on m, on A, and on the sort α of K,
- c is a function from A to $\mathbb{N}$.

We refer to type judgments only for closed λ-term, but we remark that they were defined also for λ-terms with free variables (and then one writes a type environment to the left of $\vdash$). While working with some scheme $\mathcal{G}$, as K we only take λ-terms in which all variables have the same sort as some variables appearing in $\Lambda(\mathcal{G})$. Under this assumption, it is enough to consider as m only one fixed value, denoted $m_\mathcal{G}$ (equal to the so-called order of $\mathcal{G}$).

Having in mind some scheme $\mathcal{G}$, we define the *value* of a closed λ-term K, denoted $[\![K]\!]$, as the pair consisting of:

- the set of pairs $(A, \hat\tau)$ such that $A \subseteq \Sigma_\mathcal{G}$ and there exists $c\colon A \to \mathbb{N}$ for which $\vdash_{m_\mathcal{G},A} K : \hat\tau \triangleright c$ can be derived, and
- the set of pairs $(A, \hat\tau)$ such that $A \subseteq \Sigma_\mathcal{G}$ and for every $n \in \mathbb{N}$ there exists $c_n\colon A \to \mathbb{N}$ satisfying $c_n(a) \geq n$ for all $a \in A$, and for which $\vdash_{m_\mathcal{G},A} K : \hat\tau \triangleright c_n$ can be derived.

When K is of sort α, $[\![K]\!]$ belongs to the finite set $\mathcal{S}^\alpha = (\mathcal{P}(\bigcup_{A \subseteq \Sigma_\mathcal{G}} \{A\} \times \mathcal{TT}^\alpha_{m_\mathcal{G},A}))^2$.

The considered type system is compositional, in the sense that knowing what can be derived for closed λ-terms $K^{\alpha\to\beta}$ and L^α, we can determine what can be derived for $K\,L$.

In other words, we can define a composition operation "$\cdot$" on values, going from $\mathcal{S}^{\alpha\to\beta} \times \mathcal{S}^\alpha$ to $\mathcal{S}^\beta$ and such that $[\![K\,L]\!] = [\![K]\!] \cdot [\![L]\!]$ for every closed λ-term $K\,L$.

We now extend the definition of the value to λ-terms K that are not closed. To this end, we need a valuation ν mapping some variables x^α to elements of $\mathcal{S}^\alpha$, which is defined at least for all free variables of K. Then the *value* of K (with respect to ν), denoted $[\![K]\!]^\nu$, is defined as $[\![\lambda x_1.\cdots.\lambda x_k.K]\!] \cdot \nu(x_1) \cdot \ldots \cdot \nu(x_k)$, where $x_1, \ldots, x_k$ are free variables of K, listed according to some fixed order.

Using an algorithm from [31] we can compute $[\![\lambda x_1.\cdots.\lambda x_k.K]\!]$ for every subterm K of $\Lambda(\mathcal{G})$, where, as above, $x_1, \ldots, x_k$ are free variables of K (recall that $\Lambda(\mathcal{G})$ has finitely many subterms).

Having the above properties in hand, it is easy to deduce the following lemma.

▶ **Lemma 7.** *For every scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, N_0)$ generating a tree T one can construct a scheme $\mathcal{G}'$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair $(a, [\![K]\!])$, where K is some λ-term (closed, of sort o) such that $BT(K) = T{\restriction}_v$.*

Proof. This lemma is proven by literally repeating the construction of Salvati and Walukiewicz [34, Section 5]. We recall it here for completeness. Without loss of generality we assume that $\Lambda(\mathcal{G})$ is fully convergent (cf. Fact 2).

For every sort α, let $[\alpha] = \underbrace{\mathsf{o}\to\ldots\to\mathsf{o}\to}_{|\mathcal{S}^\alpha|}\mathsf{o}$. When $\tau_1, \ldots, \tau_{|\mathcal{S}^\alpha|}$ are all elements of $\mathcal{S}^\alpha$, listed in some fixed order, we let $(\tau_i)_\lambda = \lambda x_1^\mathsf{o}.\cdots.\lambda x_{|\mathcal{S}^\alpha|}^\mathsf{o}.x_i$ for $i \in \{1, \ldots, |\mathcal{S}^\alpha|\}$; these λ-terms are of sort $[\alpha]$. Given a λ-term K of sort $[\alpha]$, and $K_1, \ldots, K_{|\mathcal{S}^\alpha|}$ of sort $\beta_1\to\ldots\to\beta_s\to\mathsf{o}$, we write $\mathtt{case}\,K\,\{\tau_i \rightsquigarrow K_i\}_{\tau_i\in\mathcal{S}^\alpha}$ for

$$\lambda y_1^{\beta_1}.\cdots.\lambda y_s^{\beta_s}.K\,(K_1\,y_1\,\ldots\,y_s)\,\ldots\,(K_{|\mathcal{S}^\alpha|}\,y_1\,\ldots\,y_s)\,.$$

We notice that for $K = (\tau_j)_\lambda$ this λ-term β-reduces to $\lambda y_1^{\beta_1}.\cdots.\lambda y_s^{\beta_s}.K_j\,y_1\,\ldots\,y_s$, which in turn is η-equivalent to K_j.

We transform every finite λ-term K of sort α to a λ-term $(\!|K|\!)^\nu$ of sort $\alpha^\bullet$, where sorts $\alpha^\bullet$ are defined by induction: $(\alpha\to\beta)^\bullet = \alpha^\bullet\to[\alpha]\to\beta^\bullet$ and $\mathsf{o}^\bullet = \mathsf{o}$. The translation is defined as follows:

$$\begin{aligned}
(\!|a\langle K_1, \ldots, K_r\rangle|\!)^\nu &= (a, [\![\Lambda_\mathcal{G}(a\langle K_1, \ldots, K_r\rangle)]\!]^\nu)\langle(\!|K_1|\!)^\nu, \ldots, (\!|K_r|\!)^\nu\rangle\,,\\
(\!|x^\alpha|\!)^\nu &= x^{\alpha^\bullet}\,,\\
(\!|K\,L|\!)^\nu &= (\!|K|\!)^\nu\,(\!|L|\!)^\nu\,([\![\Lambda_\mathcal{G}(L)]\!]^\nu)_\lambda\,,\\
(\!|\lambda x^\alpha.K|\!)^\nu &= \lambda x^{\alpha^\bullet}.\lambda y^{[\alpha]}.\mathtt{case}\,y\,\{\tau \rightsquigarrow (\!|K|\!)^{\nu[x^\alpha\mapsto\tau]}\}_{\tau\in\mathcal{S}^\alpha}\,.
\end{aligned}$$

In the above translation nonterminals are treated as any other variables.

To the resulting scheme $\mathcal{G}'$ we take a nonterminal $N^{\alpha^\bullet}$ for every nonterminal N^α of $\mathcal{G}$, and we define $\mathcal{R}'(N^{\alpha^\bullet}) = (\!|\mathcal{R}(N^\alpha)|\!)^\emptyset$, where $\emptyset$ is the valuation with empty domain. It is not difficult to see that such a scheme $\mathcal{G}'$ has the expected properties. We remark that when in effect of performing β-reductions one obtains a λ-term $K = (a, \tau)\langle K_1, \ldots, K_r\rangle$, then $\tau = [\![L]\!]$ for some λ-term L β-equivalent to K, but not necessarily for $L = K$ (it is not clear from [31] whether for β-equivalent λ-terms K and L it holds that $[\![K]\!] = [\![L]\!]$). This is enough for us, as β-equivalent λ-terms have the same Böhm tree. ◀

It was shown [31, Theorem 3] that, for a closed λ-term K of sort o, the set $\mathcal{D} = \{A \subseteq \Sigma_\mathcal{G} \mid \mathsf{Diag}_A(\mathcal{L}(BT(K)))\}$ can be computed out of the value $[\![K]\!]$. We can thus easily convert the scheme $\mathcal{G}'$ from Lemma 7 to a scheme $\mathcal{G}_{diag}$ as needed in Theorem 6. Indeed, it is enough to replace, in every node constructor appearing in $\mathcal{G}'$, the pair (a, τ) by the pair $(a, \mathcal{D})$ for the set $\mathcal{D}$ computed out of the value τ.

5 Proof of the Main Theorem

In this section we prove our main theorem—Theorem 1. To this end, we have to recall two properties of recursion schemes: logical reflection, and closure under composition with finite tree transducers.

By MSO we mean the logic defined similarly to WMSO+U, but where there are no U quantifiers, and where existential quantifiers range over infinite sets. The MSO logic over infinite trees is equivalent to μ-calculus and to nondeterministic parity automata.

▶ **Fact 8** ([9, Theorem 2(ii)]). *For every scheme $\mathcal{G}$ generating a tree T and every MSO sentence φ one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair (a, b), where a is the label of v in T, and b is* tt *if φ is satisfied in $T{\upharpoonright}_v$ and* ff *otherwise.*

A *(deterministic, top-down) finite tree transducer* is a tuple $\mathcal{T} = (Q, q_0, \Sigma_0, r_{\max}, \delta)$, where Q is a finite set of states, $q_0 \in Q$ is an initial state, $\Sigma_0 \subseteq \Sigma$ is a finite alphabet, $r_{\max}$ is the maximal arity of considered trees, and δ is a transition function mapping $Q \times \Sigma_0 \times \{0, \dots, r_{\max}\}$ to finite λ-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 λ-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 $(\Sigma_0, r_{\max})$-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 9.** *For every scheme $\mathcal{G}$ generating a tree T, and for every finite tree transducer $\mathcal{T}$ 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 [20], as it is straightforward to compose a collapsible pushdown system with $\mathcal{T}$ (where due to Fact 2 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).

Having Facts 8 and 9, we now come to our main technical lemma.

▶ **Lemma 10.** *For every scheme $\mathcal{G}$ generating a tree T and every* U*-prefix automaton $\mathcal{A}$ one can construct a scheme $\mathcal{G}_\mathcal{A}$ that generates the tree $\mathcal{A}(T)$.*

It is easy to deduce Theorem 1 out of Lemma 10. Indeed, consider a WMSO+U sentence φ and a scheme $\mathcal{G}_0$ generating a tree T_0. By Lemma 3, φ is equivalent to a nested U-prefix automaton $\mathcal{A} = \mathcal{A}_1 \circ \dots \circ \mathcal{A}_k$, together with an accepting set Σ_F. By consecutively applying Lemma 10 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, we can assume that $\Lambda(\mathcal{G}_k)$ is fully convergent).

We now come to the proof of Lemma 10. We are thus given a U-prefix automaton $\mathcal{A} = (Q, Q_\mathsf{imp}, \Delta)$, and a scheme $\mathcal{G}$ generating a tree T; 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 $(Q \cup \{q_0, \top\}, q_0, \Sigma_\mathcal{G}, r_{\max}(\mathcal{G}), \delta)$, where $q_0 \notin Q$ 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 $(q, a, q_{11}, \ldots, q_{1r}), \ldots, (q, a, q_{k1}, \ldots, q_{kr})$ 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 $r \leq r_{\max}(\mathcal{G})$ we take

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

We see that $\mathcal{T}(T)$ contains all nodes of the original tree T. Additionally, below every node v coming from T we have $|Q|$ new children, such that subtrees starting in these children describe runs of $\mathcal{A}$ on $T{\restriction_v}$, starting in particular states. More precisely, when v has r children in T, for every $i \in \{1, \ldots, |Q|\}$ there is a bijection between trees U in $\mathcal{L}(\mathcal{T}(T){\restriction_{v(r+i)}})$ and runs ρ of $\mathcal{A}$ on $T{\restriction_v}$ such that $\rho(\varepsilon) = p_i$. The label of every node u in such a tree U contains the state assigned by ρ to u, 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 $T{\restriction_v}$, which corresponds to the fact that $\mathcal{L}(\mathcal{T}(T){\restriction_{v(r+i)}})$ 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 $q \notin Q_{\mathsf{imp}}$. Then, for a node v of T having r children, and for $i \in \{1, \ldots, |Q|\}$, we have the following equivalence: $\mathsf{Diag}_{\{1\}}(\mathcal{T}'(T){\restriction_{v(r+i)}})$ holds if and only if for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}$ on $T{\restriction_v}$ that assigns p_i to the root of $T{\restriction_v}$, and such that for at least n nodes w it holds that $\rho_n(w) \in Q_{\mathsf{imp}}$.

We now apply Fact 9 to $\mathcal{G}$ and $\mathcal{T}'$; we obtain a scheme $\mathcal{G}_{\mathcal{T}'}$ that generates the tree $\mathcal{T}'(T)$. Then, we apply Theorem 6 (diagonal 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 w there is additionally written a set $\mathcal{D}$ containing these sets $A \subseteq \Sigma$ for which $\mathsf{Diag}_A(\mathcal{L}(T{\restriction_w}))$ holds. Next, using Fact 8 (logical reflection) $2|Q|$ times, we annotate every node v of T', having r' children, by logical values of the following properties, for $i = 1, \ldots, |Q|$:

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

Clearly both these properties can be expressed in MSO. For nodes v coming from T, the first property holds when there is a run of $\mathcal{A}$ on $T{\restriction_v}$ that assigns p_i to the root of $T{\restriction_v}$, and the second property holds when for every $n \in \mathbb{N}$ there is a run ρ_n of $\mathcal{A}$ on $T{\restriction_v}$ that assigns p_i to the root of $T{\restriction_v}$, 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 $(a, \mathcal{D}, \sigma_1, \tau_1, \ldots, \sigma_{|Q|}, \tau_{|Q|})\langle P_1, \ldots, P_{r+|Q|}\rangle$ with $f\langle P_1, \ldots, P_r\rangle$, 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 $f(p_i) = 2$ otherwise, for all $i \in \{1, \ldots, |Q|\}$ (we do not do anything with node constructors of arity smaller than $|Q|$). In effect only the nodes coming from T remain, and they are appropriately relabeled.

6 Extensions

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

▶ **Theorem 11.** *For every scheme $\mathcal{G}$ generating a tree T and every* WMSO+U *sentence φ one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair (a, b), where a is the label of v in T, and b is* tt *if φ is satisfied in $T{\restriction_v}$ and* ff *otherwise.*

Proof. In the proof of Theorem 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 v of $\mathcal{A}(T)$ it is written whether $T{\restriction}_v$ satisfies φ. Moreover, by appropriately altering $\mathcal{A}$, we can assume that 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 11, the formula φ talks only about the subtree starting in v. One can obtain a stronger version of logical reflection, where φ is allowed to talk about v in the context of the whole tree. This version can be obtained as a simple corollary of Theorem 11 by using the same methods as in Broadbent, Carayol, Ong, and Serre [9, Proof of Corollary 2].

▶ **Corollary 12.** *For every scheme $\mathcal{G}$ generating a tree T and every* WMSO+U *formula $\varphi(X)$ with one free variable X, one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair (a, b), where a is the label of v in T, and b is* tt *if φ is satisfied in T with X valuated to $\{v\}$, and* ff *otherwise.*

For MSO it is possible to prove another property, called effective selection [12]. This time we are given an MSO sentence φ of the form $\exists X.\psi$. Assuming that φ is satisfied in the tree T generated by a scheme $\mathcal{G}$, one wants to compute an example set X^T of nodes of T, such that ψ is true in T with the variable X valuated to this set X^T. In particular, it is possible to create a scheme $\mathcal{G}_\varphi$ which generates a tree of the same shape as T, in which nodes belonging to some such example set X^T are marked. In WMSO+U we can only quantify over finite sets, so the analogous property for $\varphi = \exists_{\mathsf{fin}} X.\psi$ can be trivially obtained (and hence it is not so interesting). Indeed, there are only countably many finite sets X^T, so we may try one after another, until we find some set for which ψ is satisfied; it is easy to hardcode a given set X^T in the formula (or in the scheme).

We notice that WMSO+U is incomparable to MSO, with respect to the expressive power. As model-checking of MSO sentences is also decidable on trees generated by schemes, we can consider a hybrid logic, covering both MSO and WMSO+U. To obtain such a logic, we introduce to WMSO+U quantifiers $\exists X$ ranging over infinite sets X, but with the requirement that if $\mathsf{U}Y.\psi$ is a subformula of $\exists X.\varphi$ then X is not a free variable of $\mathsf{U}Y.\psi$. In nested automata equivalent to sentences of this logic, beside of U-prefix automata (responsible for U quantifiers) we also have nondeterministic parity automata (responsible for subformulae using $\exists$ quantifiers). As we have the reflection property for both kinds of automata, our results generalize to this logic.

Our algorithm 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 diagonal problem, we only use the single-letter case. On the other hand, it seems that WMSO+U (and full MSO as well) is not capable to express the diagonal problem, only its single-letter case. Thus another direction for a future work is to extend WMSO+U to a logic that can actually express the diagonal problem. As a possible candidate we see the qcMSO logic introduced in Kaiser, Lang, Leßenich, and Löding [21], in which the diagonal problem is expressible.

Acknowledgment. We thank Szymon Toruńczyk for a discussion on topics covered by this paper.

References

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

2 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.

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

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

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

6 Mikołaj Bojańczyk, 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. doi:10.4230/LIPIcs.STACS.2016.21.

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

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

9 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. doi:10.1109/LICS.2010.40.

10 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. doi:10.4230/LIPIcs.CSL.2013.129.

11 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. doi:10.1007/978-3-642-00596-1_9.

12 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. doi:10.1109/LICS.2012.73.

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

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

15 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. doi:10.1007/978-3-319-22177-9_14.

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

17 Solomon Feferman and Robert Lawson Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47(1):57–103, 1959. URL: http://eudml.org/doc/213526.

18 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. doi:10.1007/978-3-642-15205-4_29.

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

20 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. doi:10.1109/LICS.2008.34.

21 Łukasz Kaiser, Martin Lang, Simon Leßenich, and Christof Löding. A unified approach to boundedness properties in MSO. In Kreutzer [26], pages 441–456. doi:10.4230/LIPIcs.CSL.2015.441.

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

23 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. doi:10.1007/978-3-642-19805-2_18.

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

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

26 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.

27 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. doi:10.1016/S0049-237X(08)70525-1.

28 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. doi:10.1145/2632362.2632381.

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

30 Paweł Parys. Intersection types and counting. In Naoki Kobayashi, editor, *Proceedings Eighth Workshop on Intersection Types and Related Systems, ITRS 2016, Porto, Portugal, 26th June 2016.*, volume 242 of *EPTCS*, pages 48–63, 2016. doi:10.4204/EPTCS.242.6.

31 Paweł Parys. Complexity of the diagonal problem for recursion schemes. Submitted, 2017. URL: https://www.mimuw.edu.pl/~parys/publications/2017-types-diagonal.pdf.

32 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. doi:10.4230/LIPIcs.CSL.2016.11.

33 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. doi:10.1145/2535838.2535873.

34 Sylvain Salvati and Igor Walukiewicz. Using models to model-check recursive schemes. In Masahito Hasegawa, editor, *Typed Lambda Calculi and Applications, 11th International Conference, TLCA 2013, Eindhoven, The Netherlands, June 26-28, 2013. Proceedings*, volume 7941 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2013. doi:10.1007/978-3-642-38946-7_15.

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

36 Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. In Kreutzer [26], pages 229–243. doi:10.4230/LIPIcs.CSL.2015.229.

37 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Mathematical Structures in Computer Science*, 26(7):1304–1350, 2016. doi:10.1017/S0960129514000590.

38 Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, 102(3):379–419, 1975. doi:10.2307/1971037.

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

A On the Definition of Schemes

We remark that 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 λ-binders. We do not have this requirement, so possibly $\mathcal{R}(N)$ does not start with a full sequence of λ-binders, and possibly some λ-binders are nested deeper in the λ-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 nonterminals). We can, for example, use a translation between schemes and λY-terms from Salvati and Walukiewicz [37]: 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 β-reductions, while it is sometimes assumed that only outermost β-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 λ-term, and then we operate on this λ-term, while often finite λ-terms containing nonterminals are considered, and appearances of nonterminals are expanded only when needed. This is a purely syntactical difference.

B On the Definition of WMSO+U

In order to see that our definition of WMSO+U is not too poor, let us write a few example formulae.

- The fact that $X = \emptyset$ can be expressed as $\mathit{empty}(X) \equiv a(X) \wedge b(X)$ (where a, b are any two different letters).
- The fact that $|X| \geq 2$ can be expressed as $\mathit{big}(X) \equiv \exists Y. (Y \subseteq X \wedge \neg(X \subseteq Y) \wedge \neg \mathit{empty}(Y))$.
- The fact that $|X| = 1$ can be expressed as $\neg \mathit{empty}(X) \wedge \neg \mathit{big}(X)$.
- When every node of the considered tree has at most $r_{\max}$ children, the fact that $|X| = |Y| = 1$ and the element of Y is a child of the element of X can be expressed as $(X \curlywedge_1 Y) \vee \cdots \vee (X \curlywedge_{r_{\max}} Y)$.

C Proof of Lemma 7, Continued

In this section we formally prove that the construction from Lemma 7 is correct. The construction comes from Salvati and Walukiewicz [34, Section 5] but, first, they provide no proof, and second, our assumptions are slightly weaker (they assume that the values do not change during substitutions and β-reductions). Recall that in Lemma 7 we are given a scheme $\mathcal{G} = (\mathcal{N}, \mathcal{R}, N_0)$ generating a tree T; basing on $\mathcal{G}$ we have defined a scheme $\mathcal{G}'$. We say that a tree T_2 is a *reflection* of a tree T_1 if T_2 is of the same shape as T_1, and its every node v is labeled by a pair $(a, [\![K]\!])$, where K is some λ-term (closed, of sort o) such that $BT(K) = T_1{\restriction_v}$. Our goal is to prove that the tree generated by $\mathcal{G}'$ is a reflection of the tree T.

Let us first introduce some notation:

- a function (in particular a valuation) with an empty domain is denoted $\emptyset$;
- $FV(K)$ denotes the set of free variables of a λ-term K;
- $K[L_1/x_1, \dots, L_k/x_k]$ denotes the (capture-avoiding) substitution, where we substitute L_i for every appearance of x_i in K, simultaneously for all $i \in \{1, \dots, k\}$;
- $\to_\beta$ denotes the relation of β-reduction;

- $\xrightarrow{h}_\beta$ denotes the relation of *head β-reduction*, defined by

$$(\lambda x.K)\, L_0\, L_1\, \dots\, L_s \xrightarrow{h}_\beta K[L_0/x]\, L_1\, \dots\, L_s\,.$$

Let $\prec$ by the fixed order on variables used in the definition of $[\![K]\!]^\nu$; we have that $[\![K]\!]^\nu = [\![\lambda x_1.\cdots.\lambda x_k.K]\!] \cdot \nu(x_1) \cdot \ldots \cdot \nu(x_k)$ where $FV(K) = \{x_1, \dots, x_k\}$ and $x_1 \prec \cdots \prec x_k$.

We now introduce three kinds of sets, used to relate λ-terms before the transformation and λ-terms after the transformation.

▶ **Definition C.1.** For a closed λ-term F, let $V_\beta(F)$ be the set containing values $[\![F']\!]$ of all closed λ-terms F' such that $F' \to_\beta^* F$.

▶ **Definition C.2.** For an arbitrary λ-term F, and for a valuation ν such that $FV(F) \subseteq \mathrm{dom}(\nu)$, we define $V(F,\nu)$ to be the set of all values τ that can be obtained in the following process:

- decompose $F = R[S_1/y_1, \dots, S_l/y_l]$, where $S_1, \dots, S_l$ are closed, and $FV(R) \setminus FV(F) = \{y_1, \dots, y_l\}$, and $y_1 \prec \cdots \prec y_l$;
- let $\nu'(x) = \nu(x)$ for all $x \in FV(F)$, and for $i \in \{1, \dots, l\}$ let $\nu'(y_i)$ be any element of $V_\beta(S_i)$;
- then as τ take $[\![R]\!]^{\nu'}$.

▶ **Definition C.3.** For every λ-term F, and every valuation ν such that $FV(F) \subseteq \mathrm{dom}(\nu)$, we define a set $\Theta(F,\nu)$ by coinduction on the structure of F:

- $\Theta(a\langle K_1, \dots, K_r\rangle, \nu)$ contains λ-terms $(a,\tau)\langle R_1, \dots, R_r\rangle$ such that $R_i \in \Theta(K_i,\nu)$ for $i \in \{1, \dots, r\}$ and $\tau \in V(a\langle K_1, \dots, K_r\rangle, \nu)$;
- $\Theta(x^\alpha, \nu) = \{x^{\alpha^\bullet}\}$;
- $\Theta(K\, L, \nu)$ contains λ-terms $R\, S\, (\tau)_\lambda$ such that $R \in \Theta(K,\nu)$ and $S \in \Theta(L,\nu)$;
- $\Theta(\lambda x^\alpha.K, \nu)$ contains λ-terms $(\lambda x^{\alpha^\bullet}.\lambda y^{[\alpha]}.\mathtt{case}\, y\, \{\tau \rightsquigarrow R_\tau\}_{\tau \in \mathcal{S}^\alpha})$ such that $R_\tau \in \Theta(K, \nu[x^\alpha \mapsto \tau])$ for all $\tau \in \mathcal{S}^\alpha$.

In the next part, we prove multiple properties of the introduced sets.

▶ **Lemma C.4.** *Let F be a λ-term, and ν, μ valuations such that $FV(F) \subseteq \mathrm{dom}(\nu) \cap \mathrm{dom}(\mu)$. If $\nu(x) = \mu(x)$ for all $x \in FV(F)$, then $V(F,\nu) = V(F,\mu)$ and $\Theta(F,\nu) = \Theta(F,\mu)$.*

Proof. The part about sets $V(F,\cdot)$ follows directly from Definition C.2: we refer to the valuation only for free variables of F. The second part follows by a trivial coinduction on the structure of F, where in the cases of a node constructor and of an application we use the first part. ◀

▶ **Lemma C.5.** *For every λ-term F, and for every valuation ν such that $FV(F) \subseteq \mathrm{dom}(\nu)$, it holds that $[\![F]\!]^\nu \in V(F,\nu)$.*

Proof. In Definition C.2 we consider the trivial decomposition of F with $R = F$ and $l = 0$, and as ν' we take the restriction of ν to $FV(F)$. This way we obtain $[\![F]\!]^{\nu'} \in V(F,\nu)$. We recall, however, from the definition of the value of a λ-term that the valuation is referred only for free variables of the λ-term, so $[\![F]\!]^{\nu'} = [\![F]\!]^\nu$. ◀

▶ **Lemma C.6.** *Let F be a finite λ-term, and ν a valuation such that $FV(F) \setminus \mathcal{N} \subseteq \mathrm{dom}(\nu)$. Then $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(F), \nu)$.*

Proof. The proof is by coinduction on the structure of $\Lambda_{\mathcal{G}}(F)$. Suppose first that $F = N^\alpha$ is a nonterminal. Then we have $\Lambda_{\mathcal{G}}(F) = \Lambda_{\mathcal{G}}(\mathcal{R}(N^\alpha))$. Moreover, we can easily see that $\Lambda_{\mathcal{G}}(F)$ is closed, so from Lemma C.4 we obtain that $\Theta(\Lambda_{\mathcal{G}}(F), \nu) = \Theta(\Lambda_{\mathcal{G}}(F), \emptyset) = \Theta(\Lambda_{\mathcal{G}}(\mathcal{R}(N^\alpha)), \emptyset)$. On the other hand, $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) = \Lambda_{\mathcal{G}'}(N^{\alpha^\bullet}) = \Lambda_{\mathcal{G}'}(\mathcal{R}'(N^{\alpha^\bullet})) = \Lambda_{\mathcal{G}'}((\!|\mathcal{R}(N^\alpha)|\!)^\emptyset)$. Since the only free variables of $\mathcal{R}(N^\alpha)$ are nonterminals from $\mathcal{N}$, we have $FV(\mathcal{R}(N^\alpha)) \setminus \mathcal{N} \subseteq \emptyset = \mathrm{dom}(\emptyset)$. We can thus equally well consider $\mathcal{R}(N^\alpha)$ and $\emptyset$ instead of F and ν. Recalling that $\mathcal{R}(N^\alpha)$ cannot be equal to a nonterminal, we have reduced this case to the case when F is not a nonterminal.

Thus suppose that F is not a nonterminal. We have four cases depending on the shape of F. Suppose first that $F = a\langle K_1, \dots, K_r\rangle$. Then $(\!|F|\!)^\nu = (a, [\![\Lambda_{\mathcal{G}}(F)]\!]^\nu)\langle (\!|K_1|\!)^\nu, \dots, (\!|K_r|\!)^\nu\rangle$, and thus $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) = (a, [\![\Lambda_{\mathcal{G}}(F)]\!]^\nu)\langle \Lambda_{\mathcal{G}'}((\!|K_1|\!)^\nu), \dots, \Lambda_{\mathcal{G}'}((\!|K_r|\!)^\nu)\rangle$. On the other hand, $\Lambda_{\mathcal{G}}(F) = a\langle \Lambda_{\mathcal{G}}(K_1), \dots, \Lambda_{\mathcal{G}}(K_r)\rangle$. By the assumption of coinduction we obtain that $\Lambda_{\mathcal{G}'}((\!|K_i|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(K_i), \nu)$ for all $i \in \{1, \dots, r\}$, and due to Lemma C.5 we have that $[\![\Lambda_{\mathcal{G}}(F)]\!]^\nu \in V(\Lambda_{\mathcal{G}}(F), \nu)$. Thus from Definition C.3 we can deduce that $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(F), \nu)$.

Next, suppose that $F = x^\alpha$. Recall that F is not a nonterminal, thus we have $\Lambda_{\mathcal{G}}(F) = x^\alpha$ and $\Lambda_{\mathcal{G}'}((\!|x^\alpha|\!)^\nu) = \Lambda_{\mathcal{G}'}(x^{\alpha^\bullet}) = x^{\alpha^\bullet}$. It follows from Definition C.3 that $x^{\alpha^\bullet} \in \Theta(x^\alpha, \nu)$.

Suppose now that $F = K\,L$. Then $(\!|F|\!)^\nu = (\!|K|\!)^\nu\,(\!|L|\!)^\nu\,([\![\Lambda_{\mathcal{G}}(L)]\!]^\nu)_\lambda$, and thus $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) = \Lambda_{\mathcal{G}'}((\!|K|\!)^\nu)\,\Lambda_{\mathcal{G}'}((\!|L|\!)^\nu)\,([\![\Lambda_{\mathcal{G}}(L)]\!]^\nu)_\lambda$. On the other hand, $\Lambda_{\mathcal{G}}(F) = \Lambda_{\mathcal{G}}(K)\,\Lambda_{\mathcal{G}}(L)$. By the assumption of coinduction we obtain that $\Lambda_{\mathcal{G}'}((\!|K|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(K), \nu)$ and $\Lambda_{\mathcal{G}'}((\!|L|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(L), \nu)$, and due to Lemma C.5 we have that $[\![\Lambda_{\mathcal{G}}(L)]\!]^\nu \in V(\Lambda_{\mathcal{G}}(L), \nu)$. Thus from Definition C.3 we can deduce that $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(F), \nu)$.

Finally, suppose that $F = \lambda x^\alpha.K$; then $(\!|F|\!)^\nu = \lambda x^{\alpha^\bullet}.\lambda y^{[\alpha]}.\texttt{case}\,y\,\{\tau \rightsquigarrow (\!|K|\!)^{\nu[x^\alpha \mapsto \tau]}\}_{\tau \in \mathcal{S}^\alpha}$, and thus $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) = \lambda x^{\alpha^\bullet}.\lambda y^{[\alpha]}.\texttt{case}\,y\,\{\tau \rightsquigarrow \Lambda_{\mathcal{G}'}((\!|K|\!)^{\nu[x^\alpha \mapsto \tau]})\}_{\tau \in \mathcal{S}^\alpha}$. On the other hand, $\Lambda_{\mathcal{G}}(F) = \lambda x^\alpha.\Lambda_{\mathcal{G}}(K)$. By the assumption of coinduction we obtain that $\Lambda_{\mathcal{G}'}((\!|K|\!)^{\nu[x^\alpha \mapsto \tau]}) \in \Theta(\Lambda_{\mathcal{G}}(K), \nu[x^\alpha \mapsto \tau])$ for every $\tau \in \mathcal{S}^\alpha$. Thus from Definition C.3 we can deduce that $\Lambda_{\mathcal{G}'}((\!|F|\!)^\nu) \in \Theta(\Lambda_{\mathcal{G}}(F), \nu)$. ◀

▶ **Lemma C.7.** *Let F and G^α be λ-terms, where G is closed, let z^α be a variable, let ν be a valuation such that $FV(F) \setminus \{z\} \subseteq \mathrm{dom}(\nu)$, and let $\sigma \in V_\beta(G)$. Then $V(F, \nu[z \mapsto \sigma]) \subseteq V(F[G/z], \nu)$.*

Proof. If z is not free in F, we have $F[G/z] = F$, so the thesis follows from Lemma C.4. In the sequel we assume that z is free in F. Take some $\tau \in V(F, \nu[z \mapsto \sigma])$, and consider some $R, S_1, \dots, S_l$ and ν' for which this value τ was obtained in Definition C.2. Since $F = R[S_1/y_1, \dots, S_l/y_l]$ and $S_1, \dots, S_l$ are closed, we also have $F[G/z] = R[G/z, S_1/y_1, \dots, S_l/y_l]$. Recall that G is closed as well, and observe that $FV(R) \setminus FV(F[G/z]) = \{z, y_1, \dots, y_l\}$, and that z cannot be equal to any y_i. We can thus consider this decomposition of $F[G/z]$. Let us also see that ν' can be taken as the valuation corresponding to this decomposition. Indeed, we have $\nu'(x) = \nu(x)$ for all $x \in FV(F[G/z]) = FV(F) \setminus \{z\}$, and $\nu'(z) = \nu[z \mapsto \sigma](z) = \sigma \in V_\beta(G)$, and $\nu'(y_i) \in V_\beta(S_i)$ for all $i \in \{1, \dots, l\}$. In consequence we obtain that $\tau \in V(F[G/z], \nu)$. ◀

▶ **Lemma C.8.** *Let F and G^α be λ-terms, where G is closed, let z^α be a variable, let ν be a valuation such that $FV(F) \setminus \{z^\alpha\} \subseteq \mathrm{dom}(\nu)$, and let $\sigma \in V_\beta(G)$. Then for all $P \in \Theta(F, \nu[z^\alpha \mapsto \sigma])$ and $Q \in \Theta(G, \emptyset)$ it holds that $P[Q/z^{\alpha^\bullet}] \in \Theta(F[G/z^\alpha], \nu)$.*

Proof. The proof is by coinduction on the structure of F. Observe first that if z^α is not free in F, then $z^{\alpha^\bullet}$ is not free in P, and in consequence $F[G/z^\alpha] = F$ and $P[Q/z^{\alpha^\bullet}] = P$, so the thesis follows from Lemma C.4. In the sequel we assume that z^α is free in F. We have four cases depending on the shape of F.

Suppose first that $F = a\langle K_1, \dots, K_r\rangle$. Then $P \in \Theta(F, \nu[z^\alpha \mapsto \sigma])$ implies by Definition C.3 that P is of the form $(a, \tau)\langle R_1, \dots, R_r\rangle$, where $R_i \in \Theta(K_i, \nu[z^\alpha \mapsto \sigma])$ for $i \in \{1, \dots, r\}$, and $\tau \in V(F, \nu[z^\alpha \mapsto \sigma])$. We have that $R_i[Q/z^{\alpha^\bullet}] \in \Theta(K_i[G/z^\alpha], \nu)$ for all $i \in \{1, \dots, r\}$ by the assumption of coinduction, and $\tau \in V(F[G/z^\alpha], \nu)$ by Lemma C.7. Thus $P[Q/z^{\alpha^\bullet}] = (a, \tau)\langle R_1[Q/z^{\alpha^\bullet}], \dots, R_r[Q/z^{\alpha^\bullet}]\rangle \in \Theta(a\langle K_1[G/z^\alpha], \dots, K_r[G/z^\alpha]\rangle, \nu) = \Theta(F[G/z^\alpha], \nu)$ by Definition C.3.

Next, suppose that $F = z^\alpha$ (recall that z^α is free in F, so F cannot be a variable other than z^α). Then $P \in \Theta(F, \nu[z^\alpha \mapsto \sigma])$ implies by Definition C.3 that $P = z^{\alpha^\bullet}$. Thus $P[Q/z^{\alpha^\bullet}] = Q \in \Theta(G, \emptyset) = \Theta(G, \nu) = \Theta(F[G/z^\alpha], \nu)$ by Lemma C.4.

Suppose now that $F = K\,L$. Then $P \in \Theta(F, \nu[z^\alpha \mapsto \sigma])$ implies by Definition C.3 that P is of the form $R\,S\,(\tau)_\lambda$, where $R \in \Theta(K, \nu[z^\alpha \mapsto \sigma])$, and $S \in \Theta(L, \nu[z^\alpha \mapsto \sigma])$, and $\tau \in V(L, \nu[z^\alpha \mapsto \sigma])$. We have that $R[Q/z^{\alpha^\bullet}] \in \Theta(K[G/z^\alpha], \nu)$ and $S[Q/z^{\alpha^\bullet}] \in \Theta(L[G/z^\alpha], \nu)$ by the assumption of coinduction, and $\tau \in V(L[G/z^\alpha], \nu)$ by Lemma C.7. Thus $P[Q/z^{\alpha^\bullet}] = R[Q/z^{\alpha^\bullet}]\,S[Q/z^{\alpha^\bullet}]\,(\tau)_\lambda \in \Theta(K[G/z^\alpha]\,L[G/z^\alpha], \nu) = \Theta(F[G/z^\alpha], \nu)$ by Definition C.3.

Finally, suppose that $F = \lambda x^\beta.K$. Since z^α is free in F, we have $x^\beta \neq z^\alpha$. Then $P \in \Theta(F, \nu[z^\alpha \mapsto \sigma])$ implies by Definition C.3 that $P = \lambda x^{\beta^\bullet}.\lambda y^{[\beta]}.\mathtt{case}\,y\,\{\tau \rightsquigarrow R_\tau\}_{\tau \in \mathcal{S}^\beta}$, where $R_\tau \in \Theta(K, \nu[z^\alpha \mapsto \sigma][x^\beta \mapsto \tau])$ for all $\tau \in \mathcal{S}^\beta$. For all $\tau \in \mathcal{S}^\beta$ we have that $R_\tau[Q/z^{\alpha^\bullet}] \in \Theta(K[G/z^\alpha], \nu[x^\beta \mapsto \tau])$ by the assumption of coinduction, thus $P[Q/z^{\alpha^\bullet}] = \lambda x^{\beta^\bullet}.\lambda y^{[\beta]}.\mathtt{case}\,y\,\{\tau \rightsquigarrow R_\tau[Q/z^{\alpha^\bullet}]\}_{\tau \in \mathcal{S}^\beta} \in \Theta(\lambda x^\beta.K[G/z^\alpha], \nu) = \Theta(F[G/z^\alpha], \nu)$ by Definition C.3. ◀

▶ **Lemma C.9.** *If a λ-term F is closed, then $V(F, \emptyset) \subseteq V_\beta(F)$.*

Proof. Take some $\tau \in V(F, \emptyset)$, and consider some $R, S_1, \dots, S_l$ and ν' for which this value τ was obtained Definition C.2. For all $i \in \{1, \dots, l\}$ we have $\nu'(y_i) \in V_\beta(S_i)$, so there is some closed λ-term S'_i such that $S'_i \to_\beta^* S_i$ and $\nu'(y_i) = [\![S'_i]\!]$. Denote $F' = (\lambda y_1.\cdots.\lambda y_l.R)\,S'_1\,\dots\,S'_l$. Since F is closed, we have that $FV(R) = \{y_1, \dots, y_l\}$, so

$$\begin{aligned}\tau = [\![R]\!]^{\nu'} &= [\![\lambda y_1.\cdots.\lambda y_l.R]\!] \cdot \nu'(y_1) \cdot \ldots \cdot \nu'(y_l)\\ &= [\![\lambda y_1.\cdots.\lambda y_l.R]\!] \cdot [\![S'_1]\!] \cdot \ldots \cdot [\![S'_l]\!] = [\![F']\!]\,.\end{aligned}$$

Clearly $F' \to_\beta^* F$ and F' is closed, so $\tau \in V_\beta(F)$. ◀

▶ **Lemma C.10.** *Let F and G be closed λ-terms of sort o such that $F \xrightarrow{h}_\beta G$. If $P \in \Theta(F, \emptyset)$, then there exists $Q \in \Theta(G, \emptyset)$ such that $P \to_\beta^* Q$.*

Proof. The condition $F \xrightarrow{h}_\beta G$ implies that F is of the form $(\lambda x^\alpha.\,K)\,L_0\,L_1\,\dots\,L_s$, and then $G = K[L_0/x^\alpha]\,L_1\,\dots\,L_s$. From Definition C.3 we deduce that P is of the form $(\lambda x^{\alpha^\bullet}.\lambda y^{[\alpha]}.\mathtt{case}\,y\,\{\tau \rightsquigarrow R_\tau\}_{\tau \in \mathcal{S}^\alpha})\,S_0\,(\tau_0)_\lambda\,S_1\,(\tau_1)_\lambda\,\dots\,S_s\,(\tau_s)_\lambda$, where $S_i \in \Theta(L_i, \emptyset)$ and $\tau_i \in V(L_i, \emptyset)$ for all $i \in \{0, \dots, s\}$, and $R_\tau \in \Theta(K, \emptyset[x^\alpha \mapsto \tau])$ for all $\tau \in \mathcal{S}^\alpha$. As Q we take $R_{\tau_0}[S_0/x^{\alpha^\bullet}]\,S_1\,(\tau_1)_\lambda\,\dots\,S_s\,(\tau_s)_\lambda$. Since L_0 is closed, from Lemma C.9 we obtain that $\tau_0 \in V_\beta(L_0)$. Since moreover $R_{\tau_0} \in \Theta(K, \emptyset[x^\alpha \mapsto \tau_0])$, by Lemma C.8 we have that $R_{\tau_0}[S_0/x^{\alpha^\bullet}] \in \Theta(K[L_0/x^{\alpha^\bullet}], \emptyset)$. Using again Definition C.3 we obtain that $Q \in \Theta(G, \emptyset)$. We now observe that $P \to_\beta^2 (\mathtt{case}\,(\tau_0)_\lambda\,\{\tau \rightsquigarrow R_\tau[S_0/x^{\alpha^\bullet}]\}_{\tau \in \mathcal{S}^\alpha})\,S_1\,(\tau_1)_\lambda\,\dots\,S_s\,(\tau_s)_\lambda$. Recalling the definition of the $\mathtt{case}$ construct, we observe that $\mathtt{case}\,(\tau_0)_\lambda\,\{\tau \rightsquigarrow R_\tau[S_0/x^{\alpha^\bullet}]\}_{\tau \in \mathcal{S}^\alpha} \to_\beta^* \lambda y_1.\cdots.\lambda y_{2s}.R_{\tau_0}[S_0/x^{\alpha^\bullet}]\,y_1\,\dots\,y_{2s}$. By further reducing variables $y_1, \dots, y_{2s}$ with arguments $S_1, (\tau_1)_\lambda, \dots, S_s, (\tau_s)_\lambda$ we obtain that $P \to_\beta^* Q$. ◀

▶ **Lemma C.11.** *Let F be a fully-convergent, closed λ-term of sort o. If $P \in \Theta(F, \emptyset)$ then $BT(P)$ is a reflection of $BT(F)$.*

Proof. Coinduction on the structure of $BT(F)$. Suppose that the root of $BT(F)$ has label a and r children. In such a situation $F \to_\beta^* a\langle K'_1, \dots, K'_r\rangle$, since F is fully convergent. It is a folklore that if $F \to_\beta^* a\langle K'_1, \dots, K'_r\rangle$, then some (not necessarily the same) λ-term of the form $a\langle K_1, \dots, K_r\rangle$ can be reached from F by using only head β-reductions: $F \xrightarrow{h}_\beta^* a\langle K_1, \dots, K_r\rangle$. By uniqueness of the Böhm tree we have that $BT(F) = a\langle BT(K_1), \dots, BT(K_r)\rangle$; in particular all $K_1, \dots, K_r$ are fully convergent. When we have some $P \in \Theta(F, \emptyset)$, using Lemma C.10 consecutively for every head β-reduction in a sequence witnessing $F \xrightarrow{h}_\beta^* a\langle K_1, \dots, K_r\rangle$, we obtain a λ-term $Q \in \Theta(a\langle K_1, \dots, K_r\rangle, \emptyset)$ such that $P \to_\beta^* Q$. From Definition C.3 it follows that Q is of the form $(a, \tau)\langle R_1, \dots, R_r\rangle$, where $R_i \in \Theta(K_i, \emptyset)$ for $i \in \{1, \dots, r\}$, and $\tau \in V(a\langle K_1, \dots, K_r\rangle, \emptyset)$. By the assumption of coinduction we have that $BT(R_i)$ is a reflection of $BT(K_i)$. Moreover $\tau \in V_\beta(a\langle K_1, \dots, K_r\rangle)$ by Lemma C.9, which means that there is a closed λ-term F' such that $F' \to_\beta^* a\langle K_1, \dots, K_r\rangle$ (and thus $BT(F') = BT(F)$) and $\tau = [\![F']\!]$. It follows that $BT(P)$ is a reflection of $BT(F)$. ◀

Having all this, we can easily conclude. Indeed, $\Lambda(\mathcal{G}') = \Lambda_{\mathcal{G}'}(N_0) = \Lambda_{\mathcal{G}'}((\!|N_0|\!)^\emptyset) \in \Theta(\Lambda_{\mathcal{G}}(N_0), \emptyset) = \Theta(\Lambda(\mathcal{G}), \emptyset)$ by Lemma C.6; at the very beginning, using Fact 2, we have ensured that $\Lambda(\mathcal{G})$ is fully convergent, and thus $BT(\Lambda(\mathcal{G}'))$ is a reflection of $BT(\Lambda(\mathcal{G}))$ by Lemma C.11.

D Proof of Fact 9

As already said, Fact 9 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 these 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 stack $\perp_n \in \mathcal{PD}_{n,\Gamma}$,
- 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 $\delta\colon Q \times \Gamma \to (Q \times Op_{n,\Gamma}) \cup (\Sigma \times Q^*)$ is a transition function. 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 D.1.** *[20] 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.*[1]

In Fact 9 we are given a finite tree transducer $\mathcal{T} = (P, p_0, \Sigma_0, r_{\max}, \delta_{\mathcal{T}})$, and a scheme $\mathcal{G}$ generating a $(\Sigma_0, r_{\max})$-tree T, and we want to construct a scheme $\mathcal{G}_{\mathcal{T}}$ that generates the tree $\mathcal{T}(T)$. By Fact 2 we can assume that $\mathcal{G}$ is fully convergent. We translate it to a fully convergent CPS $\mathcal{C} = (Q, q_0, n, \Gamma, \delta_{\mathcal{C}})$ generating T.

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 $q \in Q$ and U is a subterm of $\delta(p, a, r)$ for some $(p, a, r) \in P \times \Sigma_0 \times \{0, \dots, r_{\max}\}$. 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 $\delta((q,p), \chi) = ((q, \delta(p,a,r), \mathsf{id})$,
- 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 $\delta(p,a,r)$ 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 D.1.

E Proof of Corollary 12

Recall that we are given a scheme $\mathcal{G}$ generating a tree T, and a WMSO+U formula $\varphi(X)$ with one free variable X. We first use Lemma 5, and we obtain a nested U-prefix automaton $\mathcal{A}$, and a function $f \colon \Sigma^{\mathsf{out}}(\mathcal{A}) \to Pht_\varphi$ such that for every node v of T, the label of the root of $\mathcal{A}(T{\restriction_v})$ (and thus the label of v in $\mathcal{A}(T)$) is η_v such that $f(\eta_v) = [T]^{\nu_\emptyset}_\varphi$. As previously, we can also assume that there is a function g such that additionally $g(\eta_v)$ is the original label of

[1] 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.

v in T. Then, using Lemma 10 for every layer of $\mathcal{A}$, we create a scheme $\mathcal{G}_{\mathcal{A}}$ that generates the tree $\mathcal{A}(T)$.

We now write an MSO formula $\psi(X)$ with one free variable X. This formula will be evaluated in $\mathcal{A}(T)$ with X valuated as a singleton $\{v\}$ for some node v. For every node v of T, it should hold that[2]

$$\mathcal{A}(T) \models \psi(\{v\}) \iff T \models \varphi(\{v\})\,. \tag{1}$$

Denote $anc(v) = \{u \mid \exists w.\, uw = v\}$, which is the set of ancestors of v, including v itself. For every node u, let η_u be the label of u in $\mathcal{A}(T)$. We start ψ with $|Pht_\varphi|$ existential quantifiers, using variables X_τ for every $\tau \in Pht_\varphi$. We then express that every node of $anc(v)$ belongs to exactly one of the sets X_τ. For the node v, we say that $v \in X_\tau$ when

$$\tau = Comp_{g(\eta_v),r,\varphi}(\{X\}, f(\eta_{v1}), \dots, f(\eta_{vr}))\,,$$

where r is the number of children of v. For every other node $u \in anc(v)$ we say that $u \in X_\tau$ when

$$\tau = Comp_{g(\eta_u),r,\varphi}(\emptyset, f(\eta_{v1}), \dots, f(\eta_{v(i-1)}), \sigma, f(\eta_{v(i+1)}), \dots, f(\eta_{vr})\,,$$

where r is the number of children of u, ui is the child of u that belongs to $anc(v)$, and σ is such that $ui \in X_\sigma$. In effect, for every $u \in anc(v)$ we have the property that $u \in X_\tau$ exactly when $\tau = [T{\restriction}_u]_\varphi^{\nu_\emptyset[X\mapsto\{v\}]{\restriction}_u}$. In particular, by looking to which set the root belongs, we know $[T]_\varphi^{\nu_\emptyset[X\mapsto\{v\}]}$. At the end, we say that this phenotype is such that $\varphi(\{v\})$ holds in T (recall that the φ-phenotype allows us to determine whether φ holds). We thus obtain a formula ψ as declared, that is, such that (1) holds.

Finally, we use the following stronger version of logical reflection.

▶ **Fact E.1** ([9, Corollary 2(ii)]). *For every scheme $\mathcal{G}$ generating a tree T and every MSO formula $\varphi(X)$ with one free variable X, one can construct a scheme $\mathcal{G}_\varphi$ that generates a tree of the same shape as T, and such that its every node v is labeled by a pair (a, b), where a is the label of v in T, and b is* tt *if $\varphi(\{v\})$ is satisfied in T and* ff *otherwise.*

We apply this fact to $\mathcal{G}_{\mathcal{A}}$, which effects in annotating every node v of $\mathcal{A}(T)$ with an additional bit saying whether $\psi(\{v\})$ holds in $\mathcal{A}(T)$, that is, whether $\varphi(\{v\})$ holds in T. Since from labels of $\mathcal{A}(T)$ we can recover original labels from T, after relabelling node constructors we obtain a scheme $\mathcal{G}_\varphi$ as required.

F Mixture of WMSO+U and MSO

In this final section we justify that our results extend to the mixture of MSO and WMSO+U, introduced in Section 6. In the lack of a better name, we call this logic a Hybrid Logic (HL). Its syntax consists of the following constructions:

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

where we impose the restriction that if $\mathsf{U}Y.\psi$ is a subformula of $\exists X.\varphi$ then X is not a free variable of $\mathsf{U}Y.\psi$.

[2] We use this notation to say that ψ holds in $\mathcal{A}(T)$ when X is valuated to $\{v\}$; similarly for φ and T.

▶ **Theorem F.1.** *Theorems 1 and 11 hold as well when the considered sentence comes from the Hybrid Logic.*

Proof. We first need to extend the definition of φ-phenotypes to formulae φ of HL. Let us repeat the whole definition: the *φ-phenotype* of a tree T under a valuation ν, denoted $[T]^\nu_\varphi$, is defined by induction on the size of φ as follows:

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

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

  where X^T ranges over finite sets of nodes of T,
- if $\varphi = \exists X.\psi$, then

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

  where X^T ranges over all (not necessarily finite) sets of nodes of T, and
- if $\varphi = \mathsf{U}X.\psi$, then

  $$[T]^\nu_\varphi = (\{\sigma \mid \exists X^T.\, [T]^{\nu[X \mapsto X^T]}_\psi = \sigma\}, \{\sigma \mid \forall n.\, \exists X^T.\, [T]^{\nu[X \mapsto X^T]}_\psi = \sigma \wedge |X^T| \geq n\})\,,$$

  where X^T ranges over finite sets of nodes of T and n ranges over $\mathbb{N}$.

We have thus added the case of $\varphi = \exists X.\varphi$, but we have also modified the case of $\varphi = \exists_{\mathsf{fin}} X.\varphi$ (we have removed the second coordinate, which was meaningful only for $\varphi = \mathsf{U}X.\psi$). The definition of Pht_φ should be adjusted accordingly.

Clearly $[T]^\nu_\varphi$ determines whether φ holds in T, ν. Moreover Lemma 4 still holds. Indeed, the case of $\varphi = \exists_{\mathsf{fin}} X.\psi$ is now even easier than it was previously, and the case of $\varphi = \exists X.\varphi$ can be handled in the same way (for a local behaviour near the root it does not matter whether we quantify over all sets or only over finite sets).

We now define nested automata corresponding to HL. Beside of U-prefix automata, in the sequence of automata we can now have *MSO automata*. An MSO automaton, is just a bundle of MSO sentences, $\mathcal{A} = (\varphi_q)_{q \in Q}$, indexed by some finite set Q. An effect of running such an automaton $\mathcal{A}$ on a tree T is the tree $\mathcal{A}(T)$ that is of the same shape as T, and such that its every node v is labeled by a function $f_v \colon Q \to \{0, 1\}$, which assigns to every index $q \in Q$:

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

We define the output alphabet of such an automaton as $\Sigma^{\mathsf{out}}(\mathcal{A}) = \{0, 1\}^Q$, and we assume that $\{0, 1\}^Q \subseteq \Sigma$.

We now prove that every formula of HL can be translated into a nested automaton, that is, we prove an analogue of Lemma 5 (in consequence obtaining an analogue of Lemma 3). The statements of these lemmata are as previously, we only have a stronger logic, and a stronger model of automata. When φ does not start with a full existential quantifier, the induction step is as previously. We only need to handle the case of $\varphi = \exists X.\psi$, where of course we need to create a new layer, consisting of an MSO automaton.

Consider thus a tree T. By the induction hypothesis, we can assume that as the input to the MSO automaton we get a tree T' of the same shape as T, but where the label of every node v contains additionally the θ-phenotype $[T{\restriction_v}]^{\nu_\emptyset}_\theta$ (with respect to the valuation $\nu_\emptyset$ mapping every variable to the empty set) for every subformula θ of ψ. Our goal is to write for every $\sigma \in \mathit{Pht}_\varphi$ an MSO sentence $\Phi_{\varphi,\sigma}$ which is true in T' when $[T]^{\nu_\emptyset}_\varphi = \sigma$. We do this more generally, and for every subformula θ of φ, and for every $\sigma \in \mathit{Pht}_\varphi$ we write an MSO formula $\Phi_{\theta,\sigma}$ which is true in T' under valuation ν when $[T]^\nu_\theta = \sigma$ (under the assumption that only variables bound in φ by the $\exists$ quantifier can be mapped by ν to infinite sets). This is done by induction on the structure of θ. In most cases it is easy to write $\Phi_{\theta,\sigma}$, following the definition of $[T]^\nu_\theta$, and using formulae from the induction assumption. The only problematic case is when $\theta = \mathsf{U}X.\theta'$ (then we cannot express the definition of $[T]^\nu_\theta$ in MSO, since in MSO the U quantifier is not available).

In this case ($\theta = \mathsf{U}X.\theta'$) we proceed differently. First, we recall that by definition of HL, no free variable of θ is bound by the $\exists$ quantifier in φ. This means that all free variables of θ are mapped by ν to finite sets (and without loss of generality we can assume that ν is not defined for any variable not being free in θ); in other words $\nu{\restriction_v} \neq \nu_\emptyset$ only for finitely many nodes v. In such a situation our formula $\Phi_{\theta,\sigma}$ can work as follows. In every topmost node v such that $\nu{\restriction_v} = \nu_\emptyset$, the formula $\Phi_{\theta,\sigma}$ reads the value $[T{\restriction_v}]^{\nu_\emptyset}_\theta$ (i.e., $[T{\restriction_v}]^{\nu\restriction_v}_\theta$) from the label of v in T'. Then in every one among finitely many nodes v such that $\nu{\restriction_v} \neq \nu_\emptyset$, the formula $\Phi_{\theta,\sigma}$ computes $[T{\restriction_v}]^{\nu\restriction_v}_\theta$ using the $\mathit{Comp}_{a,r,\theta}$ function from Lemma 4 (basing on the label a of v in T, and on θ-phenotypes $[T{\restriction_{vi}}]^{\nu\restriction_{vi}}_\theta$ in every child vi of v). For $v = \varepsilon$ this gives $[T]^\nu_\theta$.

We remark that the MSO automaton for $\varphi = \exists X.\psi$ does not use directly the output of an automaton for ψ. It rather uses the output of automata for all outermost subformulae of the form $\theta = \mathsf{U}X.\theta'$, which is possible because all free variables of these subformulae are mapped to finite sets. In other words, the whole "MSO prefix" of the formula φ is converted to a single MSO automaton.

When an analogue of Lemma 3 is established, the rest of the proof goes smoothly. Recall that previously we were composing the given scheme $\mathcal{G}$ generating a tree T with the nested U-prefix automaton $\mathcal{A} = \mathcal{A}_1 \circ \cdots \circ \mathcal{A}_k$ corresponding to the formula, so that a scheme $\mathcal{G}_\mathcal{A}$ generating the tree $\mathcal{A}(T)$ was constructed. This was done consecutively for every $\mathcal{A}_i$. Now the difference is that among the automata $\mathcal{A}_i$ we also have MSO automata. However, we can process U-prefix automata as previously, while for MSO automata it is enough to use Fact 8 (logical reflection). ◀