Weak MSO+U with Path Quantifiers over Infinite Trees

Mikołaj Bojańczyk[*]

University of Warsaw

Abstract. This paper shows that over infinite trees, satisfiability is decidable for weak monadic second-order logic extended by the unbounding quantifier U and quantification over infinite paths. The proof is by reduction to emptiness for a certain automaton model, while emptiness for the automaton model is decided using profinite trees.

This paper presents a logic over infinite trees with decidable satisfiability. The logic is *weak monadic second-order logic with* U *and path quantifiers* (WMSO+UP). A formula of the logic is evaluated in an infinite binary labelled tree. The logic can quantify over: nodes, finite sets of nodes, and paths (a path is a possibly infinite set of nodes totally ordered by the descendant relation and connected with respect to the child relation). The predicates are as usual in MSO for trees: a unary predicate for every letter of the input alphabet, binary left and right child predicates, and membership of a node in a set (which is either a path or a finite set). Finally, formulas can use the *unbounding quantifier*, denoted by

$$\mathsf{U}X\ \varphi(X),$$

which says that $\varphi(X)$ holds for arbitrarily large finite sets X. As usual with quantifiers, the formula $\varphi(X)$ might have other free variables except for X. The main contribution of the paper is the following theorem.

Theorem 1. *Satisfiability is decidable for* WMSO+UP *over infinite trees.*

Background. This paper is part of a program researching the logic MSO+U, i.e. monadic second-order logic extended with the U quantifier. The logic was introduced in [1], where it was shown that satisfiability is decidable over infinite trees as long as the U quantifier is used once and not under the scope of set quantification. A significantly more powerful fragment of the logic, albeit for infinite words, was shown decidable in [3] using automata with counters. These automata where further developed into the theory of cost functions initiated by Colcombet in [8]. Cost functions can be seen as a special case of MSO+U in the sense that decision problems regarding cost functions, such as limitedness or domination, can be easily encoded into satisfiability of MSO+U formulas. This encoding need not be helpful, since the unsolved problems for cost functions get encoded into unsolved problems from MSO+U.

[*] Supported by ERC Starting Grant "Sosna"

The logic MSO+U can be used to solve problems that do not have a simple solution in MSO alone. One example is the finite model problem for two-way μ-calculus, which can be solved by a reduction to satisfiability of MSO+U on infinite trees; the reduction and the decidability of the fragment used by the reduction are shown in [1]. A more famous problem is the star height problem, which can be solved by a reduction to the satisfiability of MSO+U on infinite words; the particular fragment of MSO+U used in this reduction is decidable by [3]. In Section 1 we give more examples of problems which can be reduced to satisfiability for MSO+U, examples which use the fragment that is solved in this paper. An example of an unsolved problem that reduces to MSO+U is the decidability of the nondeterministic parity index problem, see [9].

The first strong evidence that MSO+U can be too expressive was given in [11], where it was shown that MSO+U can define languages of infinite words that are arbitrarily high in the projective hierarchy. In [4], the result from [11] is used to show that there is no algorithm which decides satisfiability of MSO+U on infinite trees and has a correctness proof using the axioms of ZFC. A challenging open question is whether satisfiability of MSO+U is decidable on infinite words.

The principal reason for the undecidability result above is that MSO+U can define languages of high topological complexity. Such problems go away in the weak variant, where only quantification over finite sets is allowed, because weak quantification can only define Borel languages. Indeed, satisfiability is decidable for WMSO+U over infinite words [2] and infinite trees [6]. This paper continues the research on weak fragments from [2,6]. Note that WMSO+UP can, unlike WMSO+U, define non Borel-languages, e.g. "finitely many a's on every path", which is complete for level $\mathbf{\Pi}^1_1$ of the projective hierarchy. The automaton characterization of WMSO+UP in this paper implies that WMSO+UP definable languages are contained in level $\mathbf{\Delta}^1_2$.

What is the added value of path quantifiers? One answer is given in the following section, where we show how WMSO+UP can be used to solve games winning conditions definable in WMSO+U; here the use of path quantifiers is crucial. Another answer is that solving a logic with path quantifiers is a step in the direction of tackling one of the most notorious difficulties when dealing with the unbounding quantifier, namely the interaction between quantitative properties (e.g. some counters have small values) with qualitative limit properties (e.g. the parity condition). The difficulty of this interaction is one of the reasons why the boundedness problem for cost-parity automata on infinite trees remains open [9]. Such interaction is also a source of difficulty in the present paper, arguably more so than in the previous paper on WMSO+U for infinite trees [6]. One of the main contributions of the paper is a set of tools that can be used to tackle this interaction. The tools use profinite trees.

Acknowledgment. I would like to thank Szymon Toruńczyk and Martin Zimmermann for months of discussions about this paper; in particular Szymon Toruńczyk suggested the use of profinite trees. Also, I would like to thank the anonymous referees for their comments.

1 Notation and some applications

Let us begin by fixing notation for trees and parity automata. Notions of root, leaf, sibling, descendant, ancestor, parent are used in the usual sense. A tree in this paper is labelled, binary, possibly infinite and not necessarily complete. In other words, a tree is a partial function from $\{0,1\}^*$ to the input alphabet, whose domain is closed under parents and siblings. The logic WMSO+UP, as defined in the introduction, is used to define languages of such trees. To recognise properties of trees, we use the following variant of parity automata. A parity automaton is given by an input alphabet A, a set of states Q, an initial state, a total order on the states, a set of accepting states, and finite sets of transitions

$$\delta_0 \subseteq Q \times A \quad \text{and} \quad \delta_2 \subseteq Q \times A \times Q^2.$$

A run of the automaton is a labeling of the input tree by states such that for every node with $i \in \{0,2\}$ children, the set δ_i contains the tuple consisting of the node's state, label and the sequence of states in its children. A run is accepting if it has the initial state in the root, and on every infinite path, the maximal state appearing infinitely often is accepting. Parity automata defined this way have the same expressive power as MSO.

Before continuing, we underline the distinction between paths, which are connected sets of nodes totally ordered by the ancestor relation, and chains which can be possibly disconnected. Having chain quantification and the U quantifier would be sufficient to express all properties of the leftmost path definable in MSO+U, and therefore its decidability would imply decidability of MSO+U on infinite words, which is open.

The rest of this section is devoted to describing some consequences of Theorem 1, which says that satisfiability is decidable for WMSO+UP on infinite trees.

Stronger than MSO. When deciding satisfiability of WMSO+UP in Theorem 1, we ask for the existence of a tree labelled by the input alphabet. Since the labelling is quantified existentially in the satisfiability problem, the decidability result immediately extends to formulas of *existential* WMSO+UP, which are obtained from formulas of WMSO+UP by adding a prefix of existential quantifiers over arbitrary, possibly infinite, sets. A result equivalent to Theorem 1 is that the existential WMSO+UP theory of the unlabeled complete binary tree is decidable.

Existential WMSO+UP contains all of MSO, because it can express that a parity tree automaton has an accepting run. The existential prefix is used to guess the accepting run, while the path quantifiers are used to say that it is accepting. One can prove a stronger result. Define WMSO+UP *with* MSO *subformulas*, to be the extension of WMSO+UP where quantification over arbitrary sets is allowed under the following condition: if a subformula $\exists X\ \varphi(X)$ quantifies over an arbitrary set X, then $\varphi(X)$ does not use the unbounding quantifier.

Fact 1 WMSO+UP *with* MSO *subformulas is contained in existential* WMSO+UP.

The idea behind the fact is to use the existential prefix to label each node with the MSO-theory of its subtree.

Example 1. Consider the modal μ-calculus with backward modalities, as introduced in [16]. As shown in [1], for every formula φ of the modal μ-calculus with backward modalities, one can compute a formula $\psi(X)$ of MSO such that φ is true in some finite Kripke structure if and only if

$$\mathsf{U} X \varphi(X) \tag{1}$$

is true in some infinite tree. The paper [1] gives a direct algorithm for testing satisfiability of formulas of the form as in (1). Since this formula is in WMSO+UP with MSO subformulas, Theorem 1 can be used instead.

By inspecting the proofs of [6], one can show that also [6] would be enough for the above example. This is no longer the case for the following example.

Example 2. Consider a two-player game over an arena with a finite set of vertices V, where the winning condition is a subset of V^ω defined in WMSO+U over infinite words. For instance, the winning condition could say that a node $v \in V$ is visited infinitely often, but the time between visits is unbounded. A winning strategy for player 1 in such a game is a subset $\sigma \subseteq V^*$, which can be visualized as a tree of branching at most V. The properties required of a strategy can be formalised in WMSO+UP over infinite trees, using path quantifiers to range over strategies of the opposing player. Therefore, one can write a formula of WMSO+UP over infinite trees, which is true in some tree if and only if player 1 has a winning strategy in the game. Therefore Theorem 1 implies that one can decide the winner in games over finite arenas with WMSO+U winning conditions.

The games described in Example 2 generalize cost-parity games from [10] or energy consumption games from [7], so Theorem 1 implies the decidability results from those papers (but not the optimal complexities).

Example 3. Consider a game as in the previous example, but where the winning condition is defined by a formula φ of WMSO+U which can also use a binary predicate "x and y are close". For $n \in \mathbb{N}$, consider the winning condition φ_n to be the formula φ with "x and y are close" replaced by "the distance between x and y is at most n". Consider the following problem: is there some $n \in \mathbb{N}$, such that player 1 has a winning strategy according to the winning condition φ_n? This problem can also be reduced to satisfiability of WMSO+UP on infinite trees. The idea is to guess a strategy $\sigma \subseteq V^*$, and a set of nodes $X \subseteq \sigma$, such that 1) there is a common upper bound on the length of finite paths that do not contain nodes from X; 2) every infinite path consistent with σ satisfies the formula φ with "x and y are close" replaced by "between x and y there is at most one node from X". Using the same idea, one can solve the realizability problem for Prompt LTL [12].

2 Automata

In this section, we define an automaton model with the same expressive power as existential WMSO+UP, which is called a WMSO+UP *automaton*. The automaton uses a labellings of trees by counter operations called *counter trees*, so we begin by describing these.

Counter trees. Let C be a finite set of counters. A *counter tree* over a set of counters C is defined to be a tree where every node is labelled by a subset of

$$C \times \{\text{parent}, \text{self}\} \times \{\text{increment}, \text{transfer}\} \times C \times \{\text{parent}, \text{self}\}, \tag{2}$$

where every tuple contains "self" at least once. The counter tree induces a graph with edges labelled by "increment" or "transfer", called its associated *counter configuration graph.* The vertices of this graph, called *counter configurations*, are pairs (x, c) where x is a node of the counter tree and c is a counter. The counter configuration graph contains an edge from (x_0, c_0) to (x_1, c_1) labelled by o if and only if there exists a node x in the counter tree whose label contains a tuple

$$(c_0, \tau_0, o, c_1, \tau_1) \qquad \text{with} \qquad \tau_0, \tau_1 \in \{\text{parent}, \text{self}\}$$

such that x_i is x or its parent depending on whether τ_i is "self" or "parent".

A path in the counter configuration graph, using possibly both kinds of edges, is called a *counter path.* Its value is defined to be the number of "increment" edges. The value of a counter configuration is defined to be the supremum of values of counter paths that end in it. When t is a counter tree, then we write $[\![t]\!]$ for the tree with the same nodes but with alphabet $\bar{\mathbb{N}}^C$, where the label of a node x maps $c \in C$ to the value of (x, c) in the associated counter graph.

WMSO+UP *automata.* We now present the automaton model used to decide WMSO+UP. The syntax of a WMSO+UP *consists* of:

1. A parity automaton;
2. A set of counters C, partitioned into *bounded* and *unbounded* counters;
3. For every state q of the parity automaton:
   (a) a set $cut(q)$ of bounded counters, called the counters cut by q;
   (b) a set $check(q)$ of unbounded counters, called the counters checked by q;
   (c) a subset $counterops(q)$ of the set in (2).

The automaton inputs a tree over the input alphabet of the parity automaton in the first item. A *run* of the automaton is a labelling of the input tree by states, consistent with the transition relation of the parity automaton. Using the sets $counterops(q)$, we get a counter tree with counters C, call it $counterops(\rho)$. By abuse of notation, we write $[\![\rho]\!]$ for the tree $[\![counterops(\rho)]\!]$, which is a tree over $\bar{\mathbb{N}}^C$. Using the sets $cut(q)$ and $check(q)$, we can talk about the nodes in a run where a bounded counter gets cut, or an unbounded counter gets checked. A run is accepting if it has the initial state in the root, and it satisfies all three acceptance conditions defined below. In the conditions, we define the limsup of a function ranging over a countable set to be

$$\limsup_{x \in X} f(x) \quad \stackrel{\text{def}}{=} \quad \limsup_{n \in \mathbb{N}} f(x_n) \qquad \text{for some enumeration of } X = \{x_1, x_2, \ldots\},$$

which is well-defined because it does not depend on the enumeration.

– *Parity.* On every path the maximal state seen infinitely often is accepting.
– *Boundedness.* If a bounded counter c is never cut in a connected[1] set of nodes X, then

$$\limsup_{x \in X} [\![\rho]\!](x,c) < \infty$$

– *Unboundedness.* If an unbounded counter c is checked infinitely often on a path π, then

$$\limsup_{x} [\![\rho]\!](x,c) = \infty$$

with x ranging over those nodes in π where c is checked.

The automaton accepts an input tree if it admits an accepting run.

Equivalence to logic and emptiness. Below are the two main technical results about WMSO+UP automata. The two results immediately imply that satisfiability is decidable for WMSO+UP logic.

Theorem 2. *For every formula of existential* WMSO+UP *one can compute a* WMSO+UP *automaton that accepts the same trees, and vice versa.*

Theorem 3. *Emptiness is decidable for* WMSO+UP *automata.*

The proof of Theorem 2 is in the the appendix. The rest of this paper is devoted to describing the proof of Theorem 3. The proof itself is described in Section 4, while the next section is about profinite trees, which are used in the proof.

Remark 1. If in the definition of the unboundedness acceptance condition, we replace lim sup by lim inf, we get a more powerful model. The same proof as for Theorem 3 also shows that this more powerful model has decidable emptiness.

3 Profinite trees and automata on them

In the emptiness algorithm for WMSO+UP automata, we use profinite trees. The connection between boundedness problems and profiniteness was already explored in [14], in the case of words. Profinite trees are similar to profinite words, because the recognizers are MSO formulas, the difference is that the objects are (infinite) trees. Consider an input alphabet A. Fix an enumeration of all MSO formulas over the alphabet A. We define the distance between two trees to be $1/n$ where n is the smallest number such that the n-th formula is true in one of the trees but not the other. The distance itself depends on the enumeration, but the notion of an open set or Cauchy sequence does not. Cauchy sequences

[1] It suffices to restrict attention to maximal connected sets of nodes where c is not cut, such sets are called c-cut factors.

are considered equivalent if some (equivalently, every) shuffle of them is also a Cauchy sequence. A *profinite tree* is defined to be an equivalence class of Cauchy sequences. To avoid confusion with profinite trees, we use from now on the term *real tree* instead of tree. Therefore, a profinite tree is a limit of a sequence of real trees. Every real tree is also a profinite tree, as a limit of a constant sequence.

Evaluating MSO *formulas on profinite trees.* A Cauchy sequence is said to satisfy an MSO formula if almost all trees in the sequence satisfy it. A Cauchy sequence satisfies either an MSO formula, or its negation. Equivalent Cauchy sequences satisfy the same MSO formulas, and therefore satisfaction of MSO formulas is meaningful for profinite trees: a profinite tree is said to satisfy an MSO formula if this is true for some (equivalently, every) Cauchy sequence that tends to it. Formulas of MSO are the only ones that can be extended to profinite trees in this way; one can show that if L is a set of real trees that is not MSO-definable (for instance, L is defined by a formula of WMSO+UP that is not in MSO), then there is a Cauchy sequence which has infinitely many elements in L and infinitely many elements outside L. Summing up, it makes sense to ask if a profinite tree satisfies a formula of MSO, but it does not make sense to ask if it satisfies a formula of WMSO+UP.

Profinite subtrees. The *topological closure* of a binary relation on real trees is defined to be the pairs of profinite trees that are limits of pairs of real trees in the binary relation; with the metric in the product being the maximum of distances over coordinates. Define the *profinite subtree* relation to be the topological closure of the subtree relation. A real tree might have profinite subtrees that are not real. For example, consider a real tree t such that for every n, some subtree s_n of t has exactly one a, which occurs at depth n on the leftmost branch. By compactness, the sequence $s_1, s_2, \ldots$ has a convergent subsequence, whose limit is not a real tree, but is a profinite subtree of t.

Partially colored trees. Let A and Q be finite sets. A *partially Q-colored tree over A* is a tree, possibly profinite, over the alphabet $A \times (Q \cup \{\bot\})$. Suppose that ρ is a real partially Q-colored tree over A. If a node has second coordinate $q \in Q$, then we say that it is colored by q. When the second coordinate is $\bot$, then the node is called uncolored. A *color zone* of ρ is a connected set of nodes X in ρ such that:

– the unique minimal element of X is either the root of ρ or is colored;
– maximal elements of X are either leaves of ρ or are colored;
– all other elements of X are uncolored.

A real tree is called *real factor* of ρ if it is obtained from ρ by only keeping the nodes in some color zone. These notions are illustrated in Figure 1. The notions of defined color, color zone and real factor are only meaningful when ρ is a real tree. When ρ is not a real tree, then we can still use MSO-definable properties, such as "the root has undefined color" or "only the leaves and root have defined color". Define the *profinite factor* relation to be the topological closure of the real factor relation.

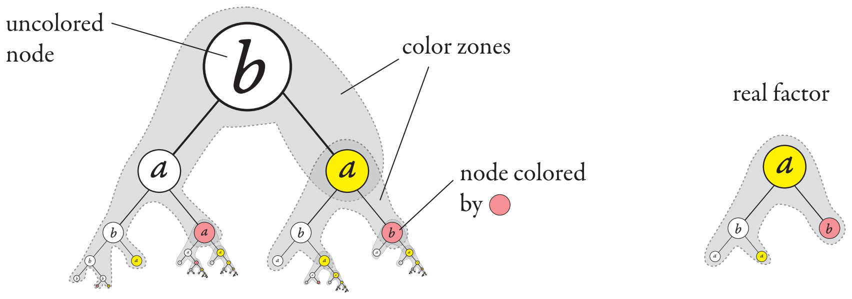


Fig. 1. A real {●, ●}-colored tree over $\{a, b\}$, together with a real factor. Uncolored nodes are white. Note how color zones overlap on colored nodes.

Generalized parity automata. A transition in a parity automaton can be visualized as a little tree, with one or three nodes, all of them colored by states. We introduce a generalized model, where transitions can be arbitrary trees, possibly infinite, and possibly profinite. A *generalized parity automaton* consists of: a totally ordered set of states Q, a subset of accepting states, an input alphabet, and a set of transitions, which is an arbitrary set of Q-colored profinite trees over the input alphabet. An input to the automaton is a profinite tree over the input alphabet. A run over such an input is a partially Q-colored profinite tree over the input alphabet, call it ρ, which projects to the input on the coordinate corresponding to the input alphabet. By projection we mean the topological closure of the projection relation on real trees. A run ρ is accepting if all of its profinite factors are transitions, and it satisfies the MSO properties "the root is uncolored" and "on every infinite path where colored nodes appear infinitely often, the maximal color seen infinitely often is accepting". (The transitions where the root is uncolored play the role of the initial state.) There might be some infinite paths which have colors finitely often, because some transitions might have infinite paths. Every profinite factor of a run will necessarily satisfy the MSO property "every node that is not the root or a leaf is uncolored", therefore it only makes sense to have transitions that satisfy this property. It is not difficult to show that if a run satisfies the property "the root is uncolored", which is the case for every accepting run, then the run has a unique profinite factor that satisfies this property.

A run is called *regular* if it has finitely many profinite subtrees rooted in colored nodes. For a generalized parity automaton $\mathcal{A}$, define $L(\mathcal{A})$ to be the set of profinite trees accepted by $\mathcal{A}$, and let $L_{\text{reg}}(\mathcal{A})$ be the subset of those profinite trees which are accepted via a regular run. The following theorem shows that two sets have the same topological closure (denoted by a bar on top), i.e. the smaller set is dense in the bigger one.

Theorem 4. $\overline{L_{\text{reg}}(\mathcal{A})} = \overline{L(\mathcal{A})}$ *holds for every generalized parity automaton* $\mathcal{A}$.

3.1 Automaton chains

Generalised parity automata are too general to be useful. For instance, every set of profinite trees is recognised by a generalised parity automaton, which has no states, and uses the recognised set as its transitions. Also, these automata do not allow a finite representation, and therefore cannot be used in algorithms. The emptiness algorithm for WMSO+UP automata uses a special case of generalised parity automata, called *automaton chains*, which can be represented in a finite way. Roughly speaking, an automaton chain is a generalised parity automaton where the set of transitions is the set of profinite trees defined by a simpler automaton chain, with the additional requirement that one cost function is bounded and another cost function is unbounded. The definitions of cost functions and automaton chains are given below.

Cost functions. A *cost function* on trees is a function α from real trees to $\bar{\mathbb{N}}$, such that the inverse image of every finite number $n \in \mathbb{N}$ is definable in MSO. As proposed by Toruńczyk in [14], a cost function α can be applied to a profinite tree t by defining $\alpha(t)$ to be a finite number $n \in \mathbb{N}$ if t satisfies the MSO property "has value n under α", and to be ∞ otherwise. Cost functions on finite words were introduced by Colcombet in [8] and then extended to finite trees, infinite words and infinite trees. The specific variant of cost functions that we use is the logic *cost* WMSO that was proposed by Vanden Boom in [15]. A sentence of this logic is built the same way as a sentence of WMSO over infinite trees, except that it can use an additional predicate "X is small", which takes a set X as a parameter, and can only be used under an even number of negations. The predicate can be used for different sets, like in the following example, call it α:

$$\exists X\ \exists Y\ X \text{ is small} \ \wedge\ Y \text{ is small} \ \wedge\ (\forall x\ a(x) \Rightarrow x \in X)\ \wedge\ (\forall y\ b(y) \Rightarrow y \in Y)$$

The cost function defined by a sentence of cost WMSO maps a tree to the smallest number n such that the sentence becomes true after "X is small" is replaced by $|X| < n$. If such a number does not exist, the result is ∞. In the case of the example α above, the function maps a tree to the number of a's or to the number of b's, whichever is bigger.

Automaton chains. We now define automaton chains, by induction on a parameter called *depth*. A automaton chain of depth 0 is any parity automaton. For $n > 0$, an automaton chain of depth n is a generalised parity automaton $\mathcal{A}$ whose set of transitions is

$$\{t : t \text{ is accepted by } \mathcal{B} \text{ and } \alpha(t) < \infty \text{ and } \beta(t) = \infty\}$$

for some automaton chain $\mathcal{B}$ of smaller depth and some cost functions α, β that are definable in cost WMSO. An automaton chain can be represented in a finite way and therefore used as an input for an algorithm, such as in the following lemmas.

Lemma 1. *Nonemptiness is decidable for automaton chains.*

Lemma 2. *Automaton chains are effectively closed under intersection with* MSO *formulas.*

4 Emptiness of WMSO+UP automata

In this section, we describe the proof of Theorem 3, which says that emptiness is decidable for WMSO+UP automata. We reduce emptiness for WMSO+UP automata to emptiness of automaton chains, which is decidable by Lemma 1.

A normal form. We begin by normalising the automaton. A counter c is called *separated* in a counter tree if the counter tree does not contain edges that involve c and and some other counter. A counter c is called *root-directed* if every counter edge involving c is directed toward the root. A WMSO+UP automaton is said to be in *normal form* if:

(a) for every run, in the counter graph generated by the automaton, every bounded counter is separated and root-directed.
(b) there is a total order on the states which is consistent with the order from the parity condition, and a mapping which maps every state q to sets of counters $larcut(q)$ and $larcheck(q)$ with the following property. For every run and every finite path in the run that starts and ends in state q and does not visit bigger states in the meantime,
  - the counters checked in the path are exactly $larcheck(q)$;
  - the counters cut in the path are exactly $larcut(q)$.

Lemma 3. *For every* WMSO+UP *automaton one can compute an equivalent one in normal form.*

In the proof, to achieve property (b), we use the latest appearance record data structure introduced by McNaughton in [13].

Partial runs. Let $\mathcal{A}$ be a WMSO+UP automaton that we want to test for emptiness. Thanks to Lemma 3, we assume without loss of generality that it is in normal form. In the emptiness algorithm, we describe properties of pieces of runs of $\mathcal{A}$, called partial runs, and defined below. Recall that in a parity automaton, there are two types of transitions δ_0 and δ_2, for leaves and non-leaves, respectively. A *partial run* of a parity automaton is a labelling of the input tree by states which respect δ_2 in nodes with two children, but need not respect δ_0 in leaves. A *partial run* of a WMSO+UP automaton is a partial run of the underlying parity automaton. A partial run is called *accepting* if it satisfies the parity, boundedness and unboundedness acceptance conditions. An accepting run of $\mathcal{A}$ is a partial accepting run where the root has the initial state and for every leaf, its (state, label) pair is in δ_0. Note that every finite partial run is an accepting partial run.

Chain automata recognising accepting runs. For a state q of $\mathcal{A}$, consider the following sets of real trees over the alphabet $A \times Q$, where A is the input alphabet of $\mathcal{A}$ and Q is its state space:

R_q accepting partial runs where states strictly bigger than q appear only in nodes with finitely many descendants;

R_{q*} the subset of R_q where state q is allowed only finitely often on every path.

Note that if q is a parity-rejecting state of the automaton $\mathcal{A}$, then $R_q = R_{q*}$. By induction on q in the order on states from the assumption on $\mathcal{A}$ being in normal form, we define automaton chains $\mathcal{R}_q$ and $\mathcal{R}_{q*}$ such that

$$\overline{R_{q*}} = \overline{L(\mathcal{R}_{q*})} \qquad \text{and} \qquad \overline{R_q} = \overline{L(\mathcal{R}_q)}. \tag{3}$$

The definition of $\mathcal{R}_q$ and $\mathcal{R}_{q*}$ is given below. The proof of (3) is in the appendix.

The automaton $\mathcal{R}_{q*}$. The automaton $\mathcal{R}_{q*}$ has a unique state, call it "state", which is rejecting, meaning that it must appear finitely often on every path. A transition of this automaton is any profinite partially {"state"}-colored tree σ over $A \times Q$ such that:

1. the projection of σ onto the $A \times Q$ coordinate belongs to $\overline{R_p}$, where p is the predecessor of q in the order on states; and
2. for every root-to-leaf path in σ which ends in a leaf with defined color "state", the maximal value of the Q coordinate is q.

Property 1 is recognised by an automaton chain by the induction assumption. Property 2 is MSO-definable, and therefore the conjunction of properties 1 and 2 is recognised by an automaton chain thanks to Lemma 2. It follows that $\mathcal{R}_{q*}$ is a degenerate form of an automaton chain where the cost functions α and β are not used. This degenerate form is a special case of an automaton chain, by taking α to be the constant 0 and β to be the constant ∞.

The automaton $\mathcal{R}_q$. If q is a parity-rejecting state of $\mathcal{A}$, then $\mathcal{R}_q$ is equal to $\mathcal{R}_{q*}$. Otherwise, it is defined as follows. The automaton $\mathcal{R}_q$ has a unique state, call it "state", which is accepting, meaning that it can appear infinitely often on a path. A transition of this automaton is any profinite partially {"state"}-colored tree σ over $A \times Q$ such that:

1. the projection of σ onto the $A \times Q$ coordinate belongs to $\overline{R_{q*}}$; and
2. for every root-to-leaf path in σ which ends in a leaf with defined color "state", the maximal value of the Q coordinate is q.
3. $\alpha(\sigma) < \infty$ holds for the cost function defined by

$$\alpha(\sigma) = \max_c \max_x \llbracket \sigma \rrbracket(x, c)$$

   with c ranging over bounded counters not in $larcut(q)$ and x ranging over nodes which do not have an ancestor where c is cut.

4. $\beta(\sigma) = \infty$ holds for the cost function defined by

$$\beta(\sigma) = \begin{cases} \min_c \min_x \max_y & [\![\sigma]\!](y,c) \quad \text{if the root of } \sigma \text{ has defined color "state"} \\ \infty & \text{otherwise} \end{cases}$$

with c ranging over unbounded counters in $larcheck(q)$, x ranging over leaves with defined color "state", and y ranging over ancestors of x where c is checked.

As for the automaton $\mathcal{R}_{q*}$, the conjunction of properties 1 and 2 is recognised by an automaton chain, and therefore $\mathcal{R}_q$ is an automaton chain.

Proof (of Theorem 3). If q is the maximal state of $\mathcal{A}$, then R_q is the set of all partial accepting runs. Therefore, the automaton $\mathcal{A}$ is nonempty if and only if $\mathcal{R}_q$ accepts some tree which is an accepting run of the underlying parity automaton in $\mathcal{A}$. This is decidable by Lemmas 1 and 2 □

5 Conclusions

This paper shows that satisfiability is decidable for WMSO+UP on infinite trees. We conjecture the logic remains decidable after adding the R quantifier from [5]. We also conjecture that the methods developed here, maybe the automaton mentioned in Remark 1, can be used to decide satisfiability of tree languages of the form "every path is in L", with L being ωB- or ωS-regular languages of infinite words, as defined in [3].

References

1. Mikołaj Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.
2. Mikołaj Bojańczyk. Weak MSO with the unbounding quantifier. *Theory Comput. Syst.*, 48(3):554–576, 2011.
3. Mikołaj Bojańczyk and Thomas Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296, 2006.
4. Mikołaj Bojańczyk, Tomasz Gogacz, Henryk Michalewski, and Michał Skrzypczak. On the decidability of MSO+U. *ICALP*, 2014.
5. Mikołaj Bojańczyk and Szymon Toruńczyk. Deterministic automata and extensions of weak MSO. In *FSTTCS*, pages 73–84, 2009.
6. Mikołaj Bojańczyk and Szymon Toruńczyk. WMSO+U over infinite trees. In *STACS*, pages 648–660, 2012.
7. Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In *CAV*, pages 23–38, 2012.
8. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, pages 139–150, 2009.
9. Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.
10. Nathanaël Fijalkow and Martin Zimmermann. Cost-parity and cost-Streett games. In *FSTTCS*, pages 124–135, 2012.

11. Szczepan Hummel and Michal Skrzypczak. The topological complexity of MSO+U and related automata models. *Fundam. Inform.*, 119(1):87–111, 2012.
12. Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.
13. Robert McNaughton. Finite state infinite games. *Project MAC Report, MIT*, 1965.
14. Szymon Toruńczyk. Languages of profinite words and the limitedness problem. In *ICALP (2)*, pages 377–389, 2012.
15. Michael Vanden Boom. Weak cost monadic logic over infinite trees. In *MFCS*, pages 580–591, 2011.
16. Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.

Appendix Part I, consisting of Sections A-D

Equivalence of Logic and Automata

In this part of the appendix, we prove Theorem 2, which says that WMSO+UP automata recognise exactly the tree languages definable in existential WMSO+UP logic. The whole energy of the proof goes into the logic-to-automata direction. The proof is spread across Section A – D. Here is an overview of the content of these sections.

A. In Section A, we define the notion of nesting closure of a class of languages $\mathcal{L}$. Basing on [6], we show sufficient conditions for the nesting closure to be closed under weak set quantification and unbounding quantification.
B. In [2], it is shown that, over infinite infinite words, WMSO+U has the same expressive power as a deterministic automaton model called max-automata. In the proof of Theorem 2, we use a slightly generalised version of this result, for *weighted words*, i.e. words where every position carries, apart from its label, a vector of natural numbers. The equivalence of WMSO+U and max-automata for weighted infinite words is shown in Section B.
C. In Section C, we introduce nested counter languages, which are the nesting closure (as defined in Section A) of a class of languages called atomic counter languages. We prove that nested counter languages are exactly the languages definable in WMSO+UP. For the more difficult right-to-left inclusion, closure of nested counter languages under weak set quantification and unbounding quantification follows from the results presented in Section A. The difficult part is closure under path quantification, which is the main content of Section C. When proving closure under path quantification, we use the equivalence of WMSO+U and max-automata over weighted words, as proved in Section B.
D. In Section D, we complete the proof of Theorem 2, by showing that every nested counter language is recognised by a WMSO+UP automaton. The nesting itself is not difficult, the main difficult is showing that the atomic counter languages are recognised by WMSO+UP automata.

A Nesting closure

In this section, we define an abstract notion of the *nesting closure* of a class of languages. The nesting closure is essentially the same as [6], but the definition used in this paper passes through transducers. The idea of nesting languages is not new, for instance it is implicit in weak alternating automata, and explicit in the Comp classes known from hierarchies in the μ-calculus.

Lookahead transducers. A *lookahead transducer* consists of

– An input alphabet A and an output alphabet B;
– Tree languages $L_1, \ldots, L_k$ over the input alphabet, called the *lookahead*;
– A set of states Q with a distinguished initial state;
– A transition function

$$\delta : Q \times A \times \{0,1\}^k \to B \times Q \times Q.$$

The transducer is run on a tree over the input alphabet. It begins in the root in the initial state. Suppose that the automaton is in a node x in state q, and that a is the label of x in the input tree. Let

$$\delta(q, a, a_1, \ldots, a_k) = (b, q_0, q_1) \qquad \text{where } a_i = \begin{cases} 1 & \text{if } t|_x \in L_i \\ 0 & \text{otherwise} \end{cases}$$

In the above $t|_x$ denotes the subtree of t rooted in x. The automaton assigns label b to the node x, and sends states q_0 and q_1 to the left and right children of x, respectively. This way, the transducer induces a function

$$f : \mathrm{trees}(A) \to \mathrm{trees}(B),$$

which is called the function recognised by the transducer.

Note that in principle, the transducer does not even need to have explicit access to the label of the current node, because this can be simulated by having for every label a of the input alphabet a lookahead language L_a which contains trees with root label a. Nevertheless, we include the label of the current node in the transition function so that simple transformations, e.g. the identity transformation, can be realised without using lookahead. Lookahead transducers without any lookahead correspond to deterministic top-down transducers.

Nesting closure. The *nesting closure* of a class of languages, called the *basis*, is defined to be the least class of languages and transducers such that:

1. the nesting closure contains every lookahead transducer where all the lookahead languages are in the nesting closure;
2. if the nesting closure contains

$$f : \mathrm{trees}(A) \to \mathrm{trees}(B),$$

and L is a tree language over B that is in the basis or definable in WMSO, then the nesting closure contains $f^{-1}(L)$.

Note that a language or transducer in the nesting closure has a natural finite representation, and therefore it makes sense to talk about computing such a language or transducer. In particular, the *nesting depth* of a language or transducer is defined in the natural way: the nesting depth of a transducer is one plus the maximal depth of its lookahead languages, while the nesting depth of $f^{-1}(L)$ is defined to be one plus the nesting depth of f.

The first two levels of nesting depth are as follows. Nesting depth one is lookahead transducers without lookahead, i.e. deterministic top-down transducers. Nesting depth two is inverse images of basis languages or WMSO definable languages under deterministic top-down transducers.

Lemma 4. *If the nesting closure of a class of languages contains*

$$f : \text{trees}(A) \to \text{trees}(B) \qquad g : \text{trees}(B) \to \text{trees}(C) \qquad L, K \subseteq \text{trees}(B)$$

then it also contains the composition $f \circ g$, the inverse image $f^{-1}(L)$, the complement of L, the union $L \cup K$ and the intersection $L \cap K$.

Proof. A lookahead transducer can label the root of the tree by the result of any Boolean combination of its lookahead languages. This shows the case of Boolean operations.

Note that the inverse image does not immediately follow from the definition, since the definition only allows the inverse image of a language in the basis, and not in the nesting closure. The proof of closure under composition and inverse image is by parallel induction on the nesting depth of g and L. □

The lemma above implies that nesting closure is indeed a closure operator, i.e. applying the nesting closure a second time does not add any new languages or transducers.

We will now show sufficient conditions for the nesting closure to be closed under weak set quantification and under unbounding quantification. To discuss closure under quantification of a class of languages, free variables need to be encoded into the trees. We write $A \times 2$ instead of $A \times \{0, 1\}$. To encode a set of nodes X in a tree t over alphabet A, we write $t \otimes X$ for the tree over alphabet $A \times 2$ obtained from t by extending the label of every node by a bit that indicates if the node belongs to X. Therefore, the quantifiers studied in this paper can be seen as a language operations, defined by

$$\begin{aligned}
\exists_{\text{fin}} X L &\overset{\text{def}}{=} \{t : t \otimes X \in L \text{ for some finite set of nodes } X\} \\
\mathsf{U} X L &\overset{\text{def}}{=} \{t : t \otimes X \in L \text{ for arbitrarily large finite sets } X\}. \\
\exists_{\text{path}} \pi L &\overset{\text{def}}{=} \{t : t \otimes \pi \in L \text{ for some path } \pi\}
\end{aligned}$$

In the following sections, we give sufficient conditions for the nesting closure to be closed under weak set quantification and unbounded quantification. This part of the paper uses the techniques from [6], and in the case of unbounded quantification we simply use a result from [6].

A.1 Closure under weak quantification.

A *tree congruence* over alphabet A is an equivalence relation on trees over A such that the equivalence class of a tree is uniquely determined by its root label and the equivalence classes of its left and right subtrees. An equivalence relation on trees is said to *saturate* a language if the language is a union of equivalence classes of the equivalence relation. A class of languages is called *derivative closed* if for every language L there exists a tree congruence which saturates L, and has finitely many equivalence classes, all of which are languages in $\mathcal{L}$. We also require the construction to be effective, i.e. based on a representation of L one can compute representations of the equivalence classes.

Lemma 5. *If a class $\mathcal{L}$ is derivative closed, then so is its nesting closure.*

Proof. Induction on the nesting depth. A language in the nesting closure is of the form $f^{-1}(L_0)$ where $L_0 \in \mathcal{L}$ and

$$f : \text{trees}(A) \to \text{trees}(B)$$

is a lookahead transducer with lookahead languages $L_1, \ldots, L_k$. By induction assumption, for every $i \in \{0, \ldots, k\}$ there is a tree congruence $\sim_i$ which saturates L_i and has finitely many equivalence classes in the nesting closure of $\mathcal{L}$. Define an equivalence relation $\sim$ on trees over A, which considers trees equivalent if they are $\sim_i$-equivalent for every $i \in \{1, \ldots, k\}$, and their images under f are $\sim_0$-equivalent. It is not difficult to see that this is a tree congruence and all of its equivalence classes are in the nesting closure of $\mathcal{L}$. □

For a tree congruence $\sim$ over trees over alphabet A, define its characteristic transducer to be the function

$$f : \text{trees}(A) \to \text{trees}((\text{trees}(A))/_\sim)$$

which labels every node in a tree t by the $\sim$-equivalence class of its subtree. The proof of the following lemma is similar to the transformation from WMSO to weak alternating automata, as shown by Muller, Saoudi and Schupp in reference [1] from the bibliography of the appendix. A similar construction, with an abstract framework, is in [5].

Lemma 6. *If a class $\mathcal{L}$ is derivative closed, then languages in its nesting closure is closed under weak quantification.*

Proof. Let L be a language over alphabet $A \times \{0, 1\}$ that is in the nesting closure of $\mathcal{L}$. By Lemma 5, there is a tree congruence $\sim$ which saturates L and whose characteristic transducer, call it f, is also in the nesting closure of $\mathcal{L}$. Without loss of generality, we assume that f also outputs the original label of each node. It is not difficult to define a formula φ of WMSO such that

$$f(t \otimes \emptyset) \models \varphi \qquad \text{iff} \qquad t \otimes X \in L \text{ holds for some finite } X.$$

The formula φ guesses the set X, and then evaluates the tree congruence $\sim$, using the values supplied by $f(t \otimes \emptyset)$ for nodes that have no elements of X in their subtree. □

A.2 Closure under unbounding quantification

Consider the set of trees over alphabet $\{\varepsilon, \mathrm{inc}, \mathrm{reset}\}$, such that for every n, there is some finite path with at least n occurrences of the letter inc, but no occurrence of the letter reset. This language is called the *basic counter language*. Note that this language is prefix-independent, and saturated by a tree congruence with two equivalence classes: the language and its complement.

It is not difficult to see that the nesting closure of the class that contains only the basic counter language is equal to the nested limsup automata defined in [6]. Therefore, by Proposition 7 from [6], we have the following result.

Lemma 7. *If $\mathcal{L}$ contains the basic counter language, then for every* WMSO*-definable language L, the nesting closure of $\mathcal{L}$ contains*

$$\{t : t \otimes X \in L \textit{ for arbitrarily large sets } X\}.$$

Lemma 8. *If a class $\mathcal{L}$ is derivative closed and contains the basic counter language, then its nesting closure is closed under unbounding quantification.*

Proof. We need to show that if a language over the alphabet $A \times \{0,1\}$ is in the nesting closure of $\mathcal{L}$, then so is:

$$\{t : t \otimes X \in L \text{ for arbitrarily large finite sets } X\} \tag{4}$$

By Lemma 5, there is a tree congruence $\sim$ that saturates L. Let f be the characteristic transducer of $\sim$, which is in the nesting closure of $\mathcal{L}$. It follows that the function $t \mapsto t \otimes f(t \otimes \emptyset)$ is also in the nesting closure of $\mathcal{L}$. By using the definition of a tree congruence, it is not difficult to show that for every $\sim$-equivalence class τ, there is a WMSO formula such that

$$t \otimes f(t \otimes \emptyset) \models \varphi(X) \qquad \text{iff} \qquad t \otimes X \in L.$$

By Lemma 7, and closure under inverse images of transducers, we get the desired result. □

B Weighted words

In this part of the appendix, we introduce weighted words. We prove that the correspondence of WMSO+U and max-automata, which is proved for words without weights in [2], extends to weighted words.

Weighted words. Define a *weighted alphabet* to be a set Σ partitioned into sets

$$\Sigma = \Sigma_{\mathrm{lab}} \cup \Sigma_{\mathrm{we}}$$

called the *label symbols* and *weight symbols*. We use Σ instead of A to distinguish between weighted alphabets and normal alphabets. A *weighted word or tree* is one where the alphabet is the (infinite) set

$$\Sigma_{\mathrm{lab}} \times \bar{\mathbb{N}}^{\Sigma_{\mathrm{we}}}.$$

The sets of weighted words and trees over Σ are denoted by

$$\text{weightedwords}(\Sigma) \qquad \text{and} \qquad \text{weightedtrees}(\Sigma).$$

For the words, we only use words of infinite length, for the trees we use possibly infinite trees where every node has zero or two children.

For a weight symbol b, define the b-weight of a position in a weighted word to be the number on coordinate b in the vector labelling that position. Define the b-weight of a set of positions to be the sum of all b-weights of positions in the set. For instance, if every position has b-weight 1, then the b-weight of a set is its size. If some position has b-weight ∞, then the b-weight of every set containing it will also be ∞.

Weighted WMSO+U. To express properties of weighted words, we use *weighted* WMSO+U. The syntax is the same as for WMSO+U on infinite words over the alphabet containing the label symbols, except that for weight symbol b of the weight alphabet there are:

– a predicate $b(x)$ which is true in positions with nonzero b-weight;
– a quantifier $\mathsf{U}_b X \varphi(X)$ which is true if the b-weights of sets satisfying X are unbounded (which may be achieved by a single set with b-weight ∞).

Weighted max-automata. For a position in a weighted word

$$w \in \text{weightedwords}(\Sigma)$$

define its profile to be the following information: the label of the position, the set of weight symbols for which the position has nonzero weight, and the set of weight symbols for which the position has infinite weight. A weighted max-automaton over consists of:

– An input weighted alphabet Σ.
– A set C of counters.
– A deterministic finite automaton whose input alphabet is profiles over Σ
– For every transition of the deterministic finite automaton, a sequence of counter operations from the set

$$c := c+1 \qquad c := c+b \qquad c := 0 \qquad c := \max(d,e)$$

where c, d, e range over counters in C and b ranges over weight symbols in Σ.
– An acceptance condition, which is a family $\mathcal{U}$ of sets of counters.

The automaton is executed as follows on a weighted word w. It begins in the initial state with a counter valuation

$$v_0 : C \to \mathbb{N}$$

that maps every counter to 0. Suppose that after reading the first $n-1$ letters of w, the state of the automaton is q_{n-1} and its counter valuation is

$$v_{n-1} : C \to \mathbb{N}.$$

The automaton performs the transition corresponding to the profile of the n-th position in the input word, and it performs the counter operations for that transition, with $c := c + b$ meaning that c is incremented by the b-weight of position n. The automaton accepts if the set $\mathcal{U}$ contains

$$\{c \in C : \limsup v_n(c) = \infty\},$$

which is the set of counter that are unbounded throughout the run. The set $\mathcal{U}$ is similar to Muller acceptance condition.

Without weights, weighted WMSO+U is the same as WMSO+U, and weighted max-automata are the same as the max-automata defined in [2]. Theorem 5 in [2] of says that, without weights, max-automata and WMSO+U have the same expressive power, and translations both ways are effective. Below we generalise this result to weighted words.

Theorem 5. *For every formula of weighted* WMSO+U *there is an equivalent weighted max-automaton, and vice versa.*

The rest of Appendix B is devoted to proving the above theorem. The proof is by a reduction to the case without weights. We only prove the more difficult logic-to-automata direction. The automata-to-logic part is shown the same way as Lemma 6 in [2], and is not used in this paper.

Fix a weighted alphabet Σ. Let $b_1, \ldots, b_k$ be the weight symbols of Σ. Define the block encoding of a word

$$w \in \text{weightedwords}(\Sigma)$$

to be the word $[w]$ obtained from w by by replacing position each position of the word w with the word

$$a v_1 \cdots v_k \qquad \text{with } v_i = \begin{cases} b_i^{n_i} & \text{when } n_i < \infty \\ \infty & \text{otherwise} \end{cases}$$

where a is the label of the position, and n_i is the b_i-weight of the position. The alphabet of $[w]$, which is a normal (unweighted word) is all symbols in Σ (both label and weighted) plus the letter ∞. By doing a syntactic transformation, it is not difficult to compute for every formula φ of weighted WMSO+U a formula $[\varphi]$ of WMSO+U such that

$$w \models \varphi \qquad \text{iff} \qquad [w] \models [\varphi].$$

By equivalence of WMSO+U and max-automata in the case without weights, for the formula $[\varphi]$ one can compute a max-automaton $\mathcal{A}_\varphi$ such that for every weighted word w,

$$[w] \models [\varphi] \qquad \text{iff} \qquad \mathcal{A}_\varphi \text{ accepts } [w].$$

If w is a weighted word, and n is a nonzero natural number, then define $n \cdot w$ by multiplying all weights by n, in particular zero weights remain zero weights. Since weighted WMSO+U is invariant under such multiplication, we have

$$w \models \varphi \qquad \text{iff} \qquad n \cdot w \models \varphi \qquad \text{for every } n > 0.$$

Let n be the number of states in the automaton $\mathcal{A}_\varphi$. By combining the above observations, we see that

$$w \models \varphi \qquad \text{iff} \qquad \mathcal{A}_\varphi \text{ accepts } [n! \cdot w].$$

The proof is concluded by the following lemma.

Lemma 9. *There is a weighted max-automaton $\mathcal{B}$ such that*

$$\mathcal{B} \textit{ accepts } w \quad \textit{iff} \quad \mathcal{A}_\varphi \textit{ accepts } [n! \cdot w] \qquad \textit{for every } w \in \text{weightedwords}(\Sigma).$$

Proof. Consider what happens when the automaton $\mathcal{A}_\varphi$ is processing the block encoding of the i-th letter of $n! \cdot w$. Let the encoding of this letter be

$$av_1 \cdots v_k \qquad \text{with } v_i = \begin{cases} b_i^{n_i} & \text{when } n_i < \infty \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

Each number n_i is divisible by $n!$. The key observation is that when a deterministic automaton has n states, then for every word w, the state transformation induced by $w^{n! \cdot m}$ does not depend on m, as long as m is nonzero. By this observation, for every fixed Σ-profile τ and every state q of the automaton $\mathcal{A}_\varphi$ there are sequences of counter operations

$$u_1, \ldots, u_k \qquad \text{and} \qquad w_1, \ldots, w_k$$

on the counters of the automaton $\mathcal{A}_\varphi$ such that when the automaton is in state q and reads a word as in (5) corresponding to a position with profile τ, then sequence of counter operations that it produces is exactly

$$u_1 w_1^{m_1} \cdots u_k w_k^{m_k} \qquad \text{with } m_i = \begin{cases} n_i & \text{if } n_i < \infty \\ 0 & \text{otherwise} \end{cases}$$

This behaviour can be simulated by a weighted max-automaton. □

C Nested counter languages

In this part of the appendix, we introduce nested counter languages and transducers, which are obtained by applying the nesting closure, as defined in Section A, to a particular class of languages called atomic counter languages. The main result is Theorem 6, which says that nested counter languages are exactly the same as languages definable in WMSO+UP.

Atomic counter languages. Let C be a set of counters and $\mathcal{U}$ a family of subsets of C. A counter $c \in C$ is called *tail unbounded* on a path π in a counter tree

$$t \in \text{countertrees}(C)$$

if the value of counter c is unbounded on path π in every counter tree that agrees with t on the labels of all but at most finitely many nodes. For a family $\mathcal{U}$ of subsets of C, we say that a path π is $\mathcal{U}$-accepting in t if $\mathcal{U}$ contains the set of counters which are tail unbounded on the path. An *atomic counter language* is a language of the form

$$\{t \in \text{countertrees}(C) : \text{every infinite path is } \mathcal{U}\text{-accepting}\}$$

for some set of counters C and family of subsets $\mathcal{U}$. Note also that the basic counter language from Lemma 8 is a special case of an atomic counter language (up to a different encoding of the input). The notion of tail unbounded is defined so that every atomic counter language is prefix independent in the following sense: if L is an atomic counter language, then a tree t belongs to L if and only if every subtree of t belongs to L.

In this section, we prove that the nesting closure of atomic counter languages is equivalent to the logic WMSO+UP. We use the name *nested counter language* for a language in the nesting closure of atomic counter languages. Likewise we define *nested counter transducers.*

Theorem 6. *Nested counter languages are exactly the languages definable in* WMSO+UP *and translations both ways are effective.*

The rest of Appendix C is devoted to proving the theorem. We begin with the easier inclusion, from nested counter languages to WMSO+UP.

Nested counter languages are WMSO+UP *definable.* The automata-to-logic direction is by a straightforward induction on the nesting depth. Call a tree transducer

$$f : \text{trees}(A) \to \text{trees}(B)$$

WMSO+UP definable if for every letter $b \in B$ of the output alphabet, there is a formula $\varphi_b(x)$ over the input alphabet A with one free node variable, such that for every tree t over the input alphabet A, a node of t has label b in $f(t)$ if and only if formula $\varphi_b(x)$ is true in t.

By induction on the nesting depth, we prove that every nested counter language and transducer is definable in WMSO+UP. The induction step is straightforward, because it is not difficult to see that languages definable in WMSO+UP are preserved under inverse images of WMSO+UP definable transducers, and every deterministic transducer with lookahead languages in WMSO+UP is definable in WMSO+UP. The only interesting case is the induction base, i.e. that atomic counter languages are definable in WMSO+UP.

To show this, it suffices to show that for every set of counters C and counter c, there is a formula of WMSO+UP $\varphi_c(\pi)$, with a free path variable π, which says that counter c is tail unbounded on path π in a counter tree t over counter C. Using Boolean combinations of this formula, one can say that a path is $\mathcal{U}$-accepting, and using path quantification, one can say that a tree is $\mathcal{U}$-accepting. To define the formula $\varphi_c(\pi)$, we observe that for every counters c, d one can write a formula $\varphi_{cd}(x, y)$ of WMSO, which selects a pair of nodes (x, y) if and only if there is a counter path from (x, c) to (x, y) which does at least one increment. Using this formula, one can write a formula $\psi_c(x, X)$ of WMSO, which selects a set of nodes if and only if there is a counter path that begins in x, and for every node $y \in X$ the path does an increment when leaving X. Therefore, a counter c is unbounded on a path π if

$$\mathsf{U}X\ \exists x\ x \in \pi\ \wedge\ \psi_c(x, X).$$

To say that the automaton is tail unbounded, one needs to additionally say that the above formula remains true after the counter tree is modified in any finite way, which can easily be done using quantification over finite sets.

WMSO+UP *definable languages are nested counter languages.* As usual, the more difficult implication in Theorem 6 is going from the logic to nested counter languages and transducers. We need to show that nested counter languages are closed under weak set quantification, unbounding quantification, and path quantification.

By being prefix closed, for every atomic counter languages L the equivalence relation with two equivalence classes, namely L and its complement, is a tree congruence. Therefore, atomic counter languages are closed under derivatives, and so Lemma 6, can be used to conclude that nested counter languages are closed under weak set quantification. As we have remarked above, the basic counter language from Lemma 8 is a special case of an atomic counter language, and so Lemma 8 can be used to conclude that nested counter languages are closed under unbounding quantification. The rest of this section is devoted to path quantification.

The rest of Section C is devoted to showing that nested counter languages are closed under path quantification. In the presence of first-order quantification, it suffices to quantify over *full paths*, i.e. paths that begin in the root. This is because a non-full path π can be described by the pair (σ, x) where x is the first node of π and $\sigma \supseteq \pi$ is the full path obtained from π by adding all ancestors of x. The proof strategy is as follows. Let L be a nested counter language over alphabet $A \times 2$. Our goal is to show that that

$$\{t \in \text{trees}(A) : t \otimes \pi \in L \text{ for every full path } \pi\} \tag{6}$$

is also a nested counter language. The main result is Lemma 11, which says that the language (6) is equivalent to first running a transducer f on the tree t, and then checking that every path in $f(t)$ satisfies a formula of weighted WMSO+U, with the weights corresponding to counter values encoded in the tree $f(t)$. In Lemma 10, we show that running a weighted WMSO+U formula on all paths can be implemented by a nested counter language, thus proving closure under universal path quantification. The precise definitions of how weighted trees are computed by transducers are given below, followed by Lemmas 10 and Lemma 11.

Encoding a weighted tree. Consider a weighted alphabet Σ. To encode a weighted tree over this alphabet, we use a tree $s \otimes t$, where s is a tree over the label alphabet of Σ and t is a counter tree over the weight alphabet of Σ. The tree $s \otimes t$ is said to be the *counter tree encoding* of a weighted tree over Σ if for every node of the weighted tree, its label is given by s and its b-weight is given by the value of counter b. A *counter tree encoding* of a function

$$f : \text{trees}(A) \to \text{weightedtrees}(\Sigma)$$

is a function which, given an input t, produces a weighted tree encoding of the weighted tree $f(t)$. If a function f admits a counter tree encoding (which is not necessarily unique) that is a nested counter transducer, then we say that f is *recognised by a counter transducer.*

Define the *path word* of a path π in a tree t, denoted by $t \upharpoonright \pi$, to be the infinite word consisting of the labels occurring on π. We will be mainly interested in the path word for full paths, i.e. infinite paths that begin in the root.

Lemma 10. *Let φ be a weighted* WMSO+U *formula over a weighted alphabet Σ, and let*

$$f : \text{trees}(A) \to \text{weightedtrees}(\Sigma)$$

be recognised by a nested counter automaton. The set of trees t such that

$$f(t) \upharpoonright \pi \models \varphi \qquad \textit{for every full path } \pi$$

is a nested counter language over the alphabet A.

Proof. Let B be the label of alphabet of Σ, and let C be the counter alpahbet. The assumption of the lemma is that there is a nested counter transducer

$$g : \text{trees}(A) \to \text{trees}(B) \otimes \text{countertrees}(C)$$

such that for every input tree t, the output $g(t)$ is a counter encoding of $f(t)$. By Theorem 5, the formula φ is equivalent to a weighted max-automaton, call it $\mathcal{A}$. Let D be the counters of the automaton. By describing the transitions of a weighted max-automaton in terms of a counter tree, one can a nested counter transducer

$$h : \text{trees}(B) \otimes \text{countertrees}(C) \to \text{countertrees}(C \cup D)$$

such that for every input $s \otimes t$ and every node x of this tree and every counter $c \in C$, the value of counter $d \in D$ in node x of $h(s \otimes t)$ is equal to the value of counter d in the run of the automaton $\mathcal{A}$ after reading the path word that corresponds to the path in the weighted tree encoded by $s \otimes t$ which goes from the root to node x. We leave the definition of the transducer h to the reader, we only notice that it in order to correctly simulate the automaton $\mathcal{A}$, the transducer h needs lookahead automata which compute for each node x and counter $c \in C$, whether the value of counter c in node x is zero, nonzero but finite, or infinite.

The statement of the lemma is obtained by composing g with h, and then testing an atomic counter language on the image of this composition, which tests that the acceptance condition of the weighted max-automaton is satisfied. $\square$

Lemma 11. *For every nested counter language*

$$L \subseteq \text{trees}(A \times 2)$$

there exists a weighted alphabet Σ, a function

$$\text{help}_L : \text{trees}(A) \to \mathit{weightedtrees}(\Sigma)$$

recognised by a nested counter transducer, and a formula φ of weighted WMSO+U *over Σ such that for every tree t over A and every full path π,*

$$t \otimes \pi \in L \qquad \textit{iff} \qquad \text{help}_L(t) \restriction \pi \models \varphi.$$

As remarked previously, Lemmas 10 and 11 imply that nested counter languages are closed under universal path quantification, therefore completing the proof of Theorem 6.

The rest of Section C is devoted to proving Lemma 11. The proof is by induction on the nesting depth of the language. However, when doing the induction, we pass through nested counter transducers, so we need a variant of Lemma 11 for transducer, which is stated in Lemma 12. Before stating the lemma, we define transducers from weighted words to normal infinite words.

In the same spirit as when defining tree transducers definable in WMSO+U, we use weighted WMSO+U transducers which map an (infinite) weighted word to an infinite word over a finite alphabet. We say that a function

$$f : \text{weightedwords}(\Sigma) \to A^\omega$$

is definable in weighted WMSO+U, if for every letter a of the output alphabet A, there is a formula $\varphi_a(x)$ of weighted WMSO+U over the input alphabet Σ, such that for every input word w, the label of position x in $f(w)$ is b if and only if $\varphi_b(x)$ is satisfied in w.

Lemma 12. *For every nested counter transducer*

$$f : \text{trees}(A \times 2) \to \text{trees}(B)$$

there exists a nested counter transducer recognising a function

$$\text{help}_f : \text{trees}(A) \to \mathit{weightedtrees}(\Sigma)$$

and a weighted WMSO+U *transducer*

$$\varphi : weightedwords(\Sigma) \to B^\omega$$

such that for every tree t over A and every full path π,

$$f(t \otimes \pi) \restriction \pi = \varphi(\mathrm{help}_f(t) \restriction \pi)$$

Lemmas 11 and 12 are proved by mutually recursive induction on the nesting depth. The induction step for transducers, corresponding to Lemma 12, is simple and given below. The induction steps for languages, corresponding to Lemma 11, is more involved. In Section C.1 we present some results on counter trees, and the actual proof of Lemma 11 is given in Section C.2.

Proof (of Lemma 12). Consider a nested counter transducer

$$f : \mathrm{trees}(A \times 2) \to \mathrm{trees}(B).$$

Let $L_1, \dots, L_k$ be the lookahead languages of f. Let t and π be as in the assumption of the lemma. Define x_i to be the i-th node on π, and define

$$a_i \in A \times 2^k$$

to be the label of t in x_i and the bit vector indicating which lookahead languages contain the subtree of $t \otimes \pi$ rooted in a_i. Apply the induction assumption of Lemma 11 to the lookahead languages, yielding for every $i \in \{1, \dots, k\}$ a transducer help_{L_i} and a formula φ_i. By using these transducers and formlas, one can compute a weighted WMSO+U transducer which inputs the word

$$t \otimes \mathrm{help}_{L_1}(t) \otimes \cdots \otimes \mathrm{help}_{L_k}(t) \restriction \pi$$

and outputs $a_1 a_2 \cdots$. By simulating the control structure of the transducer f, one can see that there is a WMSO transducer

$$g : (A \times 2^k)^\omega \to B^\omega$$

which inputs $a_1 a_2 \cdots$ and outputs $f(t \otimes \pi) \restriction \pi$. Composing these two transducers we get the statement of the lemma. □

C.1 Side values

This section is devoted to proving Lemmas 14 and 15, which are used in the proof of Lemma 11. Roughly speaking, Lemma 15 says that in order to determine whether a counter c is unbounded on a path in a counter tree, one only needs to look at the labels of the counter tree on the path, as well as an appropriate summary of what is happening outside the path. Furthermore, as shown in Lemma 14, these summaries can be produced by a nested counter transducer.

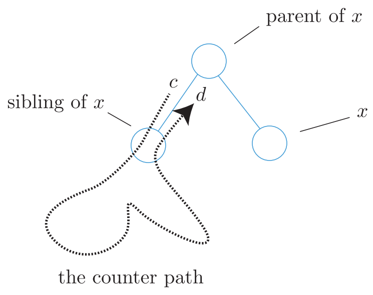
We begin by defining the summaries, which say for each node of a counter tree, what happens in the subtree of its sibling. For a counter tree t over counter C, a node x of t and counters $c, d \in C$ define

$$\text{sidevalue}(t, c, d, x) \in \bar{\mathbb{N}} \tag{7}$$

to be the least upper bound on the number of increments in a counter path in t such that:

1. the counter path begins in counter c in the parent of x;
2. the counter path ends in counter d in the parent of x;
3. the rest of the counter path is in the subtree of the sibling of x.

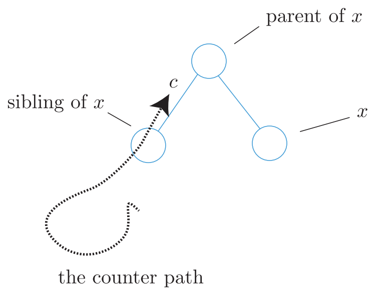
The counter path mentioned above looks like this:



If the parameter c is not supplied, then

$$\text{sidevalue}(t, d, x) \in \bar{\mathbb{N}} \tag{8}$$

is defined as above, except that instead of condition 1, we say that the counter path begins in the subtree of the sibling of x, in some counter. Such a counter path looks like this



Lemma 13. *The values* (7) *and* (8) *depend only on the subtree of* t *rooted in the sibling of* x. *In other words, there is a function*

$$\text{svalue} : \text{countertrees}(C) \times (C \cup C^2) \to \bar{\mathbb{N}}$$

such that every counter tree t, *node* x *and counter* c *and* d *satisfy*

$$\text{sidevalue}(t, c, d, x) = \text{svalue}(t|_x, c, d) \qquad \textit{and} \qquad \text{sidevalue}(t, d, x) = \text{svalue}(t|_x, d).$$

Proof. Follows straight from the definition, with the only subtle point being that the values do not depend on the label of the parent of x. This is because in a counter tree, the edges between a node and its child are encoded in the label of the child. □

Consider a nested counter transducer

$$f : \mathrm{trees}(A) \to \mathrm{countertrees}(C).$$

Let Q be the states of f. For a state $q \in Q$, define f_q to be the transducer obtained from f by changing the initital state to q. Let $\mathrm{svalue}_f(t)$ to be the weighted tree over weight alphabet $Q \times C^2 \cup Q \times C$ defined by

$$\mathrm{svalue}_f(t)(x, q, c, d) = \mathrm{svalue}(f_q(t|_x), c, d)$$
$$\mathrm{svalue}_f(t)(x, q, d) = \mathrm{svalue}(f_q(t|_x), d)$$

Lemma 14. *For every nested counter transducer*

$$f : \mathrm{trees}(A) \to \mathrm{countertrees}(C).$$

the function svalue_f *is recognised by a nested counter transducer, with the same nesting depth as* f*.*

Proof. The same method as in converting a two-way automaton on trees to a one-way automaton, e.g. when proving that tree-walking automata can be simulated by bottom-up branching automata. □

Lemma 15. *For every nested counter transducer*

$$f : \mathrm{trees}(A) \to \mathrm{countertrees}(C).$$

and counter $c \in C$ there exists a formula φ of weighted WMSO+U *such that for every tree t over A and path π, counter c is unbounded on path π if and only if*

$$\mathrm{svalue}_f(t) \otimes f(t) \restriction \pi \models \varphi.$$

Proof. The word $\mathrm{svalue}_f(t) \restriction \pi$ gives all information about the behaviour of counter paths outside the path π. □

C.2 Proof of Lemma 11

We are now ready to prove Lemma 11, which finishes the proof of Theorem 6. Consider a nested counter language

$$L \subseteq \mathrm{trees}(A \times 2)$$

as in the assumption of Lemma 11. We need to show the conclusion of Lemma 11, which says that there exists a weighted alphabet Σ, a function

$$\text{help}_L : \text{trees}(A) \to \text{weightedtrees}(\Sigma)$$

recogognised by a nested counter transducer, and a formula φ of weighted WMSO+U over Σ such that for every tree t over A and every full path π,

$$t \otimes \pi \in L \qquad \text{iff} \qquad \text{help}_L(t) \restriction \pi \models \varphi.$$

By the definition of a nested counter language,

$$L = f^{-1}(K) \qquad \text{for some } f : \text{trees}(A \times 2) \to \text{trees}(B),$$

such that f is a nested counter transducer and K is either definable in WMSO or an atomic counter language. Let Q be the states of the transducer f. Define

$$\rho : \text{trees}(A \times 2) \to \text{trees}(Q)$$

to be the transducer which relabels each node by the state of the transducer f that is used in that node. Note that if t is a tree over A and π is a path, then the label of a node x in $f(t \otimes \pi)$ is uniquely determined by its labels in $t \otimes \pi$ and $\rho(t \otimes \pi)$. The transducer ρ has the same nesting depth as f, and therefore by induction assumption we can apply Lemma 12 to it, yielding a transducer help_ρ.

We treat separately the cases when K is definable in WMSO, and when K is an atomic counter language.

***K* is definable in WMSO.** Let k be the quantifier depth of the WMSO formula defining K. Define the k-type of a tree to be the set of WMSO formulas of quantifier depth that are true in the tree. It is well known that there are finitely many k-types, and each one is definable in WMSO. Also, having the same k-type is a tree congruence.

For an infinite path π define the *π-child* of a node $x \in \pi$ to be the unique child of x that is on the path π. Likewise we define the *non-π-child*, which is the sibling of the π-child. Define the *k-profile* of an infinite path π in a tree t to be the infinite word, where the i-th letter contains the following information about the i-th node of the path: the label of the node, the k-type of the subtree of t rooted in its non-π-child, and a bit saying if the π-child of the node is a left or right child. Using the composition method, one shows that for every k-type τ, there is a formula φ_τ of WMSO over infinite words such that $t \otimes \pi$ has k-type τ if and only if φ_τ is true in the infinite word that is the k-profile of π in t.

Recall that we write f_q for the transducer obtained from f by changing the initial state to q. Define γ be the function which inputs a tree t over A and outputs a tree γ with the same nodes such that the label of every node x is the function

$$(q, i) \in Q \times 2 \qquad \mapsto \qquad k\text{-type of the subtree of } f_q(t \otimes \emptyset) \text{ rooted in the } i\text{-th child of } x.$$

It is not difficult to show that γ is a nested counter transducer, from the assumption that f is a nested counter transducer. Define

$$\mathrm{help}_L(t) \quad \overset{\mathrm{def}}{=} \quad t \otimes \mathrm{child}(t) \otimes \gamma(t) \otimes \mathrm{help}_\rho(t)$$

where $\mathrm{child}(t)$ labels each node by the information saying whether or not the node is the root, a left child, or a right child. We claim that help_L satisfies the properties in the statement of Lemma 11. It is not difficult to see that there is weighted WMSO+U transducer which inputs the word $\mathrm{help}_L(t) \upharpoonright \pi$ and outputs the k-profile of the path π in the tree $f(t \otimes \pi)$. Based on this k-profile, membership in K can be determined using only weak quantification.

***K* is an atomic counter language.** Consider now the case where L is the preimage $f^{-1}(K)$ where K is an atomic counter language, i.e. there is a set of counter C and a family $\mathcal{U}$ of subsets of C such that K is the set of counter trees over C where every path is $\mathcal{U}$-accepting. The condition $f(t \otimes \pi) \in K$ can be decomposed as a conjunction of two conditions:

1. path π is $\mathcal{U}$-accepting in $f(t \otimes \pi)$;
2. every path $\sigma \neq \pi$ is $\mathcal{U}$-accepting in $f(t \otimes \pi))$.

For each condition $i \in \{1, 2\}$ above, we define a transducer $\mathrm{help}_{L,i}$ and a formula φ_i of weighted WMSO+U such that t and π satisfy condition i if and only if

$$\mathrm{help}_{L,i}(t) \upharpoonright \pi \models \varphi_i.$$

The lemma will then follow by taking help_L to be the product of the transducers $\mathrm{help}_{L,i}$ and taking φ to be the conjunction of the formulas φ_i.

Condition 1. Define

$$\mathrm{help}_{L,1}(t) \quad \overset{\mathrm{def}}{=} \quad \mathrm{help}_f(t) \otimes \mathrm{svalue}_f(t \otimes \emptyset)$$

We need to show that there is formula of weighted WMSO+U, which is true in

$$\mathrm{help}_{L,1}(t) \upharpoonright \pi \tag{9}$$

if and only if path π in the counter tree $f(t \otimes \pi)$ is $\mathcal{U}$-accepting. For every counter $c \in C$, apply Lemma 15 to f, yielding a formula φ_c such that for every tree t over A and path π in t, counter c is unbounded on path π in $f(t \otimes \pi)$ if and only if formula φ_c is true in the word

$$\mathrm{svalue}_f(t \otimes \pi) \otimes f(t \otimes \pi) \upharpoonright \pi \tag{10}$$

Therefore, it suffices to show that there is a weighted WMSO+U transducer α such that for every tree t and path π, when given input (9), it produces output (10). Once such a transducer α has been defined, condition 1 is implemented by checking that $\mathcal{U}$ contains the set of counter c such that φ_c is true in the image of the word (9) under the transducer α.

By induction assumption, the word $f(t \otimes \pi) \upharpoonright \pi$ can be produced by a weighted WMSO+U transducer based on the word $\text{help}_f(t) \upharpoonright \pi$. To compute the word $\text{svalue}_f(t \otimes \pi) \upharpoonright \pi$, we observe that for every node x in a path π, the subtree of the sibling of x is the same in $t \otimes \pi$ and in $t \otimes \emptyset$. Therefore, from Lemma 13, it follows that

$$\text{svalue}_f(t \otimes \pi) \upharpoonright \pi \qquad = \qquad \text{svalue}_f(t \otimes \emptyset) \upharpoonright \pi. \tag{11}$$

By Lemma 14, the function $t \mapsto \text{svalue}_f(t \otimes \emptyset)$ is recognised by a nested counter transducer.

Condition 2. Recall that condition 2 says that every path $\sigma \neq \pi$ is $\mathcal{U}$-accepting in $f(t \otimes \pi)$. Because being $\mathcal{U}$-accepting is prefix-independent, this is equivalent to saying that for every node $x \notin \pi$, the language K contains the subtree of $f(t \otimes \pi)$ rooted in x. Furthermore, since K is closed under subtrees, it follows that only nodes x with parent in π need be checked. Summing up, condition 2 is equivalent to

$$f(t \otimes \pi)|_x \in K \qquad \text{for every } x \text{ whose sibling is in } \pi \tag{12}$$

Recall that f_q is obtained from f by changing the initial state to q. Define

$$L_q \quad \stackrel{\text{def}}{=} \quad \{t \in \text{trees}(A) : f_q(t \otimes \emptyset) \in K\},$$

which is a nested counter language. If q_x denotes be the state of the transducer f that is used in node x, then (12) is equivalent to

$$t|_x \in L_{q_x} \qquad \text{for every } x \text{ whose sibling is in } \pi \tag{13}$$

Let γ be the transducer which labels every node x of a tree t by the set of states q such that the subtree rooted in the sibling of x belongs to L_q. (The root, which has no siblings, gets labelled by the empty set.) This is a nested counter transducer, since every language L_q is a nested counter language. Define

$$\text{help}_{L,2}(t) \quad \stackrel{\text{def}}{=} \quad t \otimes \text{childnumber}(t) \otimes \gamma(t) \otimes \text{help}_\rho(t).$$

We claim that the word $\text{help}_{L,2}(t) \upharpoonright \pi$ provides sufficient information to check condition 2. By (13), condition 2 holds if and only if for every nonroot node $x \in \pi$, the label of x in $\gamma(t)$ contains that state of the transducer f in the sibling of x. This state can be deduced based on the child number of x, the label of x and the state of f in the parent of x, all of these are given by $\text{help}_{L,2}(t)$. This completes the proof of Lemma 11, and therefore the proof of Theorem 6.

D WMSO+UP automata

In this part of the appendix, we prove Theorem 2, which says that for every formula of existential WMSO+UP one can compute a WMSO+UP automaton that accepts the same trees, and vice versa. The automata-to-logic is straightforward, shown the same way as the automata-to-logic direction in Theorem 6. For the logic-to-automata part, we only need to translate nested counter languages into WMSO+UP automata, since Theorem 6 says that the logic WMSO+UP defines exactly nested counter languages.

We prove a slightly stronger result, because we will show that it suffices to use WMSO+UP automata which satisfy condition (a) of the definition of normal form. Recall that condition (a) says that for every run, in the counter graph generated by the automaton, every bounded counter is separated and root-directed.

Theorem 7. *Every nested counter language is recognised by a* WMSO+UP *automaton, which satisfies condition (a) of the definition of normal form.*

For the rest of this section, all produced WMSO+UP automata satisfy condition (a), so we do not mention it explicitly. In other words, from now on by "WMSO+UP automaton" we mean "WMSO+UP automaton which satisfies condition (a)". In Section D.2, we show that every atomic counter language is recognised by a WMSO+UP automaton. In Section D.3, we generalise this result to nested counter languages.

D.1 WMSO+UP automata contain languages definable in WMSO+U

In this section we show that WMSO+UP automata capture all languages definable in WMSO+U, i.e. definable without the path quantifier. To show this, we prove that WMSO+UP automata generalise the automaton model from [6], which is at least as expressive as WMSO+U.

We begin by proving some simple closure properties of languages recognised by WMSO+UP automata. A *relabelling* is an arbitrary function from an input alphabet to an output alphabet. Such a function is lifted to trees in the natural way.

Lemma 16. *Languages recognised by* WMSO+UP *automata are closed under union, intersection, inverse images under relabellings, images under relabellings and contain all* MSO*-definable languages.*

Proof. Inverse images are immediate: an automaton can ignore part of its input. Union and images are by nondeterminism. For intersection, the cartesian product works. The underlying parity automaton can be used to recognise any MSO-definable language, without using any counters. □

Puzzles. Define a *cut-increment-reset tree* to be a tree where every node is labelled by a subset of "cut", "increment", "reset", and "∞". An increment node is one whose label contains "increment", likewise we define cut and reset nodes. Define the value of a path in such a tree to be the maximal number of increment nodes in a subpath that does not contain reset nodes. Define the value of a node to be the supremum of values of paths that begin in the node and do not contain cut nodes. Define the *puzzle language* to be the set of cut-increment-reset trees where a node has ∞ in its label if and only if its value is ∞.

Note that unlike counter paths in counter trees, which can go both ways, the paths in the definition of the puzzle language are normal paths in trees, i.e. totally ordered connected sets of nodes. In this sense, the puzzle language uses one-way counter paths.

Lemma 17. *The puzzle language is recognised by a* WMSO+UP *automaton.*

Proof. The puzzle language is the intersection of two languages: the *bounded puzzle language*, which says that every node without ∞ in its label has value strictly smaller than ∞; and the *unbouned puzzle language*, which says that every node with ∞ in its label has value ∞. By closure of WMSO+UP automata under intersection, it suffices to show that each of these languages is recognised by a WMSO+UP automaton.

The bounded puzzle language is immediate, by using the cuts in the definition of WMSO+UP automata. This lemma is actually the only place in the proof where we produce a new bounded counter, in all other places bounded counters are only inherited from already produced automata. Note that in the bounded puzzle language, there is only one bounded counter, so it is necessarily separated, and it is root-directed. Therefore, the WMSO+UP automaton for the bounded puzzle language satisfies condition (a) in the definition of normal form.

The rest of the proof is devoted to the unbounded counter language.

A cut-increment-reset tree t can be encoded as a counter tree $\underline{t}$ over counters c, d as follows. The nodes of t and $\underline{t}$ are the same. If the label of a node x in t does not contain "cut" then the counter tree $\underline{t}$ contains a transfer edge from d in x to d in its parent, and a transfer edge from c in x to d in its parent. If the label of x contains neither "reset" nor "cut", then $\underline{t}$ contains an edge from counter c in x to counter c in its parent; this edge is an increment edge if x is an increment node, otherwise the edge is a transfer edge. It is easy to see that the value of a node x in t is equal to the value of counter d in the same node of $\underline{t}$.

The WMSO+UP automaton for the unbounded puzzle language works as follows. Given an input cut-increment-reset tree, it computes the tree $\underline{t}$. The counters of the automaton are c and d, as in $\underline{t}$, and both are unbounded counters. The automaton also guesses a subset of positions where counter d is checked (in an accepting run, counter d is unbounded on any infinite chain of positions where it is checked). The parity automaton underlying the WMSO+UP automaton verifies that for every node that has ∞ in its label, there is some infinite path that begins in the position, contains infinitely many nodes where d is checked, and does pass through a node whose label includes "cut". □

Lemma 18. *Every language definable in* WMSO+U *is recognised by a* WMSO+UP *automaton.*

Proof. In [6] an automaton model called a *puzzle* is introduced, and is proved to be at least as expressive as WMSO+U. Every language recognised by a puzzle is of the form

$$\{t : t \otimes t_1 \otimes \cdots \otimes t_n \in L \text{ holds for some } t_1, \ldots, t_n \text{ in the puzzle langauge}\}$$

where L is MSO-definable. By Lemma 17, and the closure properties from Lemma 16, every language recognised by a puzzle is also recognised by a WMSO+UP automaton. □

D.2 Atomic counter languages

This section is devoted to proving the following lemma.

Lemma 19. *Atomic counter languages are recognised by* WMSO+UP *automata.*

If X is a set of nodes in a counter tree t over counters C, then define the X-restricted value of counter c in node x, denoted by

$$[\![t]\!]_X(x, c),$$

to be the supremum of values of counter paths that end in (c, x), and which do not visit any ancestors of x in the set X. A set of nodes X in a counter tree t over counters C is called a witness for counter $c \in C$ if

1. for every connected set Y disjoint with X,
$$\limsup_{x \in Y} [\![t]\!]_X(x, c) < \infty$$
2. for every path π that passes infinitely often through X,
$$\limsup_{x \in \pi} [\![t]\!]_X(x, c) = \infty$$

In Lemma 20 we show that if X is a witness for c, then counter c is tail unbounded exactly on those paths that visit X infinitely often. Lemma 21 shows that a witness always exists, and Lemma 22 shows that being a witness can be recognised by a WMSO+UP automaton. These lemmas imply Lemma 19.

Lemma 20. *If X is a witness for c in a counter tree, then counter c is tail unbounded exactly on those paths that visit X infinitely often.*

Proof. For a node x, define t_x to be the counter tree obtained from t by removing all counter edges that involve ancestors of x. Clearly t and t_x differ on finitely many nodes, and for every descendant y of x, the value of counter c in y is lowest in t_x, as compared to other counter trees that agree with t on descendants of

x. It follows that counter c is tail unbounded on path π in counter tree t if and only if

$$\limsup_{y\in\pi} [\![t_x]\!](y,c) = \infty \qquad \text{for all } x \in \pi \tag{14}$$

Suppose that π passes through finitely many nodes of X. Let x be the last of node X that is visited by π, if no such node is visited then let x be the root. The definition of X being a witness for c implies that

$$\limsup_{y\in\pi} [\![t_x]\!](y,c) < \infty$$

and therefore counter c is tail bounded on path π in t. For the converse implication, if π passes infinitely often through X, then (14) holds and therefore counter c is tail unbounded on patph π in t. □

Lemma 21. *In every counter tree, every counter admits a witness.*

Proof. Construct a family of sets $X_0, X_1, \ldots$ by induction as follows. The set X_0 contains only the root. Suppose that $X_0, \ldots, X_{i-1}$ have already been defined, and let $X_{<i}$ be their union. Define X_i to be the set of nodes where the $X_{<i}$-restricted value of counter c is at least i, and which are minimal (closest to the root) for this property. Define X to be the union of all sets X_i.

Let x be a node, and let i be the largest number such that x has an ancestor in X_i. It is not difficult to see that the X-restricted value of counter c in x is at most i when $x \notin X$ and at least i when $x \in X$. This implies that X is a witness for counter c. □

Lemma 22. *There exists a* WMSO+UP *automaton recognising the language*

$$\{t \otimes X : t \in \text{countertrees}(C) \textit{ and } X \textit{ witness for } c \textit{ in } t\}$$

Proof. The first condition in the definition of a witness for c is definable in WMSO+U, and therefore by Lemma 18 it is recognised by a WMSO+UP automaton. We focus on the second condition, which says that

$$\limsup_{x\in\pi} [\![t]\!]_X(x,c) = \infty \qquad \text{for every path } \pi \text{ that visits } X \text{ infinitely often.} \tag{15}$$

For counter trees s, t we write $s \subseteq t$ if the counter trees have the same nodes, and every counter edge in s is a counter edge in t. We claim that (15) is equivalent to saying that some $s \subseteq t$ satisfies:

1. For every path π that passes infinitely often through X,

$$\limsup_{x\in\pi\cap X} [\![s]\!](x,c) = \infty$$

2. If $x < y$ are in X, then no counter path in s passes through both x and y.

Condition 1 is easily seen to be recognised by a WMSO+UP automaton based on s; while condition 2 is MSO definable and therefore also recognised by a WMSO+UP automaton. Therefore, a WMSO+UP automaton can check if there is some $s \subseteq t$ which satisfies 1 and 2.

It remains to show that (15) is equivalent to the existence of $s \subseteq t$ satisfying conditions 1 and 2 above. The right-to-left implication is immediate. For the left-to-right implication, let t be a counter tree satisfying (15). Using induction, it is not difficult to define counter trees

$$s_0 \subseteq s_1 \subseteq \cdots \qquad \subseteq t$$

which satisfy condition 2, have finitely many counter edges, and such that for every n, if a path π passes infinitely often through X, then

$$\sup_{x \in X} [\![s_n]\!](x, c) \geq n.$$

The tree s is then taken to be the limit, i.e. union, of the trees s_n. $\square$

Proof (of Lemma 19). Let L be an atomic counter language, i.e. there exists a set of counters C and a family of subsets $\mathcal{U}$ such that L contains a counter tree over C if and only if for every infinite path, $\mathcal{U}$ contains the counters which are tail unbounded on the path. Let the counters be $c_1, \ldots, c_n$. Let K be the set of trees

$$t \otimes X_1 \otimes \cdots \otimes X_n \qquad t \in \text{countertrees}(C)$$

such that a) for every for every $i \in \{1, \ldots, n\}$, the set of nodes X_i is a witness for c_i; and b) for every infinite path π, the set $\mathcal{U}$ contains the set of counters c_i such that the path passes infinitely often through X_i. By Lemma 22, condition a) is recognised by a WMSO+UP automaton. Since WMSO+UP automata generalise parity automata, condition b) is also recognised by WMSO+UP automata. Therefore, K is recognised by a WMSO+UP automaton. By Lemma 21, for every counter tree t there exist sets $X_1, \ldots, X_n$ such that condition a) holds. By Lemma 20, condition b) holds if and only if t belongs to L. Therefore, L is the image of K under the relabeling which ignores the sets $X_1, \ldots, X_n$; and the lemma follows because the sets can be guessed using nondeterminism. $\square$

D.3 Nested counter languages

For a language L, define its *characteristic language* to be:

$$\{t \otimes X : t|_x \notin L \text{ if and only if } x \in X\},$$

In the following lemma, we use a transducer without lookahead, which is the same thing as a deterministic top-down tree transducer.

Lemma 23. *For every transducer f without lookahead, and every atomic counter language L, the characteristic language of $f^{-1}(L)$ is recognised by a* WMSO+UP *automaton.*

Before proving the above lemma, we show how it completes the proof of Theorem 7.

Proof (of Theorem 7). To complete the proof, we need to show that every nested counter language is recognised by a WMSO+UP automaton. We prove a stronger result: for ever nested counter language, its characteristic language is recognised by a WMSO+UP automaton. This is proved by induction on the nesting depth.

Consider a nested counter language, which is of the form $f^{-1}(L)$, where L is an atomic counter language or WMSO-definable, and f is in a nested counter transducer. Let the lookahead languages of the transducer f be $L_1, \ldots, L_k$. By definition of a lookahead transducer, there is a deterministic transducer g without lookahead such that for every tree t,

$$f(t) = g(t \otimes X_1 \otimes \cdots \otimes X_k)$$

where X_i is the set of nodes in t whose subtree belongs to L_i, or equivalently, X_i is the unique set such that $\{t \otimes X_i\}$ is in the characteristic language of L_i. The automaton recognising the characteristic language of $f^{-1}(L)$ works as follows. Given on input a tree $t \otimes X$, it guesses sets $X_1, \ldots, X_k$ and checks that a) the trees

$$t \otimes X_1, \ldots, t \otimes X_k$$

belong respectively to the characteristic languages of $L_1, \ldots, L_k$, which can be done by the induction assumption, and b) that the tree

$$t \otimes X_1 \otimes \cdots \otimes X_k \otimes X$$

belongs to the characteristic language of $g^{-1}(L)$. When L is an atomic counter language, then condition b) can be done by a WMSO+UP automaton thanks to Lemma 23. When L is definable in WMSO, then condition b) is also definable in WMSO, and therefore it can be done by a WMSO+UP automaton. □

We are therefore left with proving Lemma 23. Define the positive and negative characteristic languages of L to be, respectively:

$$\{t \otimes X : t|_x \in L \text{ for every } x \in X\} \qquad \{t \otimes X : t|_x \notin L \text{ for every } x \notin X\}.$$

Unlike for the characteristic language, where X is uniquely determined by t, for a given t there are many sets X such that $t \otimes X$ is in the positive characteristic language, although there is a unique greatest set X with $t \otimes X$ in the positive characteristic language. The dual property holds for the negative characteristic language. The characteristic language is the intersection of the positive and negative characteristic languages. Therefore, to prove Lemma 23, it suffices to show that both the positive and negative characteristic languages of $f^{-1}(L)$ are recognised by WMSO+UP automata. We begin with the positive one.

Lemma 24. *Let L and f be as in the assumption of Lemma 23. There exists a function*

$$g : \mathrm{trees}(A) \to \mathrm{weightedtrees}(\Sigma)$$

recognised by a transducer without lookahead and a formula of ψ of weighted WMSO+U *over weighted alphabet Σ such that a tree $t \otimes X$ belongs to the positive characteristic language of $f^{-1}(L)$ if and only if*

$$g(t \otimes X) \upharpoonright \pi \models \psi \qquad \text{for every full path } \pi.$$

Proof. Recall the function svalue_f defined in Section C.1. By Lemma 14, this function is recognised by a transducer without lookahead. By Lemma 15, for every counter $c \in C$ there is a formula φ_c of weighted WMSO+U such that for every full path π,

$$f(t) \otimes \mathrm{svalue}_f(t) \upharpoonright \pi \models \varphi_c$$

holds if and only if counter c is tail unbounded on π in $f(t)$. Let $\mathcal{U}$ be the family of accepting sets in the definition of the language L. By using a Boolean combination, one gets a formula φ of weighted WMSO+U such that for every full path π,

$$f(t) \otimes \mathrm{svalue}_f(t) \upharpoonright \pi \models \varphi$$

is true if and only if π is $\mathcal{U}$-accepting in $f(t)$. Since $f(t)$ has no lookahead, the labels of the output $f(t)$ on the path π can be computed, using weak quantification, based only on the labels on the input t on the same path. Therefore, there is a formula φ' of weighted WMSO+U such that for every full path π,

$$t \otimes \mathrm{svalue}_f(t) \upharpoonright \pi \models \varphi'$$

is true if and only if π is $\mathcal{U}$-accepting in $f(t)$. Finally, let ψ be obtained from φ' by extending the input alphabet with an additional bit, and requiring φ' to occur in every suffix of the input word which begins in a position marked by the additional bit. Because

$$\mathrm{svalue}_f(t)|_x = \mathrm{svalue}_f(t|_x) \qquad \text{for every node } x,$$

the formula ψ satisfies the condition in the statement of the lemma, with g defined by

$$t \otimes X \qquad \mapsto \qquad t \otimes \mathrm{svalue}_f(t) \otimes X$$

□

Lemma 25. *Let L and f be as in the assumption of Lemma 23. The positive characteristic language of $f^{-1}(L)$ is recognised by a* WMSO+UP *automaton.*

Proof. Apply Lemma 24, yielding a transducer g and a formula ψ such that the $t \otimes X$ belongs to the positive characteristic language of $f^{-1}(L)$ if and only if

$$g(t \otimes X) \restriction \pi \models \psi \qquad \text{for every full path } \pi. \tag{16}$$

By Lemma 10, there exists a nested counter transducer

$$h : \text{trees}(A \times 2) \to \text{countertrees}(D)$$

and an atomic counter language K over counters D such that (16) is equivalent to

$$h(t \otimes X) \in K.$$

By the proof of Lemma 10, the transducer h has no lookahead because g has no lookahead. The language K, as an atomic counter language, is recognised by a WMSO+UP automaton thanks to Lemma 19. The result follows by closure of WMSO+UP automata under inverse images of deterministic transducers without lookahead. □

It remains to show that the negative characteristic language

$$\{t \otimes X : f(t|_x) \notin L \text{ for every } x \notin X\}$$

is recognised by a WMSO+UP automaton. Using nondeterminism, we reduce this language to a positive characteristic language (for some different f and L). Let C be the counters and let $\mathcal{U}$ be the family of counters in the atomic counter language L. If $f(t|_x)$ is not in L, this means that there must be some path, call it π_x, which begins in x and is $\mathcal{U}$-rejecting in the tree $f(t|_x)$.

This motivates the following definition. An (f, L)-*reject path* in a tree t over alphabet A is an infinite path π, whose source node x is not necessarily the root, such that π is $\mathcal{U}$-rejecting in $f(t|_x)$. A tree $t \otimes X$ belongs to the negative characteristic language of $f^{-1}(L)$ if and only if for every $x \notin X$ there is a (f, L)-reject path that begins in x.

We say that an infinite path π *merges* with a node x that is an ancestor of the source node of π if in the two runs of the transducer f obtained by starting the transducer in node x and in the source node of π, the same states are reached in some node of π.

Lemma 26. *Let L and f be as in the assumption of Lemma 23. For every node x in a tree t over alphabet A, $f(t|_x) \notin L$ if and only if some (f, L)-reject path π merges with x.*

Proof. For the left-to-right implication in the lemma, we can simply take a path that begins in x, and is $\mathcal{U}$-rejecting in $f(t|_x)$. For the right-to-left implication, we use prefix independence of $\mathcal{U}$-rejecting paths. □

If X, Y are sets of nodes, define an (X, Y)-path to be a path contained in X such that Y contains the source node of the path and no other nodes of the path. The path can be finite or infinite.

Lemma 27. *Let L and f be as in the assumption of Lemma 23. For every input tree t and state q of the transducer f there exist sets X_q, Y_q such that:*

1. *every infinite (X_q, Y_q)-path is an (f, L)-reject path for every q;*
2. *for every x with $f(t|_x) \notin L$, some infinite (X_q, Y_q)-path merges with x for some q;*

Proof. Let $x_1, x_2, \ldots$ be the enumeration of the nodes of t in breadth-first search order. By induction on $n \in \{1, 2, \ldots\}$ we define sets X_{qn}, Y_{qn} for states q of f such that the conditions in the statement of the lemma hold, with the second condition restricted only to nodes x in $\{x_1, \ldots, x_n\}$. The construction is left to the reader; the sets in the statement of the lemma are obtained by taking the limit. □

Lemma 28. *Let L and f be as in the assumption of Lemma 23. The following language is recognised by a* WMSO+UP *automaton:*

$$\{t \otimes X \otimes Y : \textit{every infinite } (X, Y)\textit{-path is an } (f, L)\textit{-reject path}\}$$

Proof. Let c be a fresh counter not in C. Define a transducer

$$f' : \text{trees}(A \times 2 \times 2) \to \text{countertrees}(C \cup \{c\})$$

such that for every tree t over alphabet A and sets of nodes X, Y, the output $f_1(t \otimes X \otimes Y)$ is defined as follows. The counter edges concerning C are defined as in $f(t)$. If the path from the root to a node x is an (X, Y)-path, then counter c is transferred with an increment from x to its parent. The operations on counter c are defined so that counter c is tail unbounded on a path π in

$$f'((t \otimes X \otimes Y)|_x) \tag{17}$$

if and only if π is an infinite (X, Y)-path. It follows that $t \otimes X$ belongs to the language in the statement of the lemma if and only if for every node x in t, the subtree (17) belongs to the atomic counter language over counters $C \cup \{c\}$ with the acceptance condition $\mathcal{U}'$ defined by

$$D \in \mathcal{U}' \qquad \text{iff} \qquad c \in D \Rightarrow D \cap C \notin \mathcal{U}.$$

This property can be recognised by a WMSO+UP automaton thanks to Lemma 25. □

The lemmas above imply that the negative characteristic language of $f^{-1}(L)$ is recognised by a WMSO+UP automaton. Given an input $t \otimes X$ the automaton works as follows. The automaton guesses sets X_q, Y_q as in Lemma 27. It checks that for every node $x \notin X$, some infinite (X_q, Y_q)-path merges with x, which can be done by a WMSO+UP automaton because this is an MSO-definable property. Finally, it checks condition 1 in Lemma 27, which can be done by a WMSO+UP automaton thanks to Lemma 28.

This completes the proof of Lemma 23, and therefore also the proof Theorem 7.

Appendix Part II, consisting of Sections E-H

Emptiness of WMSO+UP Automata

In this part of the appendix, we complete the proof of Theorem 3, which says that emptiness is deciable for WMSO+UP automata. The plan of this part is given below.

E. In Section E we prove results about generalised parity automata, in particular we show Theorem 4 which says that the regular accepting runs are dense in all accepting runs.

F. In Section F we show that the automaton chains have decidable emptiness. The proof is by reduction a theorem of Vanden Boom in [15], which establishes decidability for the domination problem for cost functions on infinite trees that are definable in cost WMSO.

G In Section G we prove Lemma 3 which says that every automaton can be transformed into normal form. Property (a) in the definition of normal form holds for any automaton produced as a result of Theorem 7. Property (b) is achieved using the LAR construction of McNaughton.

H. In Section H, we prove that the reduction from emptiness of WMSO+UP automata to emptiness for automaton chains, as presented in Section 4, is correct.

E Profinite trees

In this section of the appendix, we prove results about generalised parity automata. Section E.1 shows that automata with closed transitions recognise closed languages. Section E.2, shows that regular accepting runs are dense in all accepting runs. Section E.3 shows how a profinite tree can be converted into a similar tree, with the similarity growing on each path.

E.1 Automata with closed transitions

In this section we prove Lemma 29, which says that if the transitions of a generalised parity automaton are a closed set, then so is the recognised language.

Lemma 29. *If the transitions of a generalised parity automaton are a closed set, then so is the recognised language.*

Proof. Let (t_n) be a sequence of profinite trees accepted by the automaton, whose limit is a tree t. We need to show that the limit is also accepted. For each tree t_n, let ρ_n be an accepting run, which is again a profinite tree. By compactness and extracting a subsequence, we can assume without loss of generality that the sequence $(\rho_n)_n$ tends to a run, call it ρ. Therefore, the lemma boils down to showing that the limit of a sequence of accepting runs is also accepting. The parity condition is MSO-definable, it is closed, and therefore preserved in the limit. It remains to show that every profinite factor of the limit ρ is a transition of the automaton.

Let then σ be a profinite factor of ρ. Let $\varepsilon > 0$. Let X_ε be the set of trees which have a profinite factor at distance at most ε from σ. For every $\varepsilon > 0$, the set of trees at distance at most ε from ρ is MSO-definable, and therefore X_ε is also MSO-definable. Since the set of profinite factors of a given tree is closed, it follows that

$$\bigcap_{\varepsilon>0} X_\varepsilon$$

is the set of trees that have profinite factor σ. Since X_ε is MSO-definable, it is open, and since it contains ρ, it must also contain almost all runs ρ_n, in particular at least one run ρ_n. It follows that for every $\varepsilon > 0$, some run ρ_n has a profinite factor that is at distance at most ε from σ. Since all profinite factors of ρ_n are transitions, it follows that σ can be approximated arbtrarily closely by transitions. By the assumption that the set of transitions is closed, it follows that σ itself is a transition. □

E.2 Proof of Theorem 4

In this section, we prove Theorem 4, which says that

$$\overline{L(\mathcal{A})} = \overline{L_{\text{reg}}(\mathcal{A})}$$

holds for every generalised parity automaton $\mathcal{A}$. The nontrivial inclusion is

$$\overline{L(\mathcal{A})} \subseteq \overline{L_{\mathrm{reg}}(\mathcal{A})}.$$

By properties of closure, it is sufficient to show

$$L(\mathcal{A}) \subseteq \overline{L_{\mathrm{reg}}(\mathcal{A})}. \tag{18}$$

To prove the above inclusion, we will use the following lemma.

Lemma 30. *Let X, Y be sets of profinite trees. Then $X \subseteq \overline{Y}$ if and only if every* MSO *formula true in some tree from X is also true in some tree from Y.*

Proof. By unraveling the definitions, the inclusion $X \subseteq \overline{Y}$ means that for every profinite tree $t \in X$ and every $\varepsilon > 0$, there is a tree in Y which is ε-close to t.

For the right-to-left implication, we observe that the ε-ball around an arbitrary profinite tree t is MSO-definable. Therefore, if t is in X, then the ε-ball around it intersects X, and therefore it intersects Y. Therefore, every open set containing t intersects Y, which means that t belongs to $\overline{Y}$.

For the left-to-right implication, suppose that an MSO formula is true in some profinite tree $t \in X$. For sufficiently small ε, the ε-ball around t contains only trees that satisfy the MSO formula. This ε-ball must contain an element of Y. □

By the above lemma, to prove (18), it suffices to show that for every MSO formula true in some tree accepted by $\mathcal{A}$, the same MSO formula is true in some tree accepted by $\mathcal{A}$ via a regular run. The reason for this is that, as we will show in Lemma 32, if a generalised parity automaton has some accepting run, then it has a regular accepting run. Before proving Lemma 32, we introduce some terminology.

Let ρ be a regular partial Q-coloring of a tree t. We say that ρ has *degree of regularity at most k* if ρ has at most k distinct profinite subtrees that have a colored root. A partially colored tree is regular if and only if it has finite degree of regularity.

Lemma 31. *For every $k \in \mathbb{N}$, the set of regular partially Q-colored trees with degree of regularity $\leq k$ is closed.*

Proof. A partially Q-colored tree does not have degree of regularity at most k if and only if one can find a set Γ of $k+1$ MSO formulas which are mutually contradictory, and each one of them is true in some subtree of ρ that is rooted in a colored node. For every fixed choice of Γ, this property is MSO-definable and therefore clopen, and therefore the intersection ranging over all choices of Γ is closed. □

Let $\mathcal{A}$ be a generalised parity automaton. As for standard parity automata, nonemptiness for generalised parity automata can be described in terms of a parity game, called the *acceptance game for $\mathcal{A}$*, which is played by players Automaton and Pathfinder. Positions of player Automaton are states of the automaton plus a special initial position. Positions of player Pathfinder are transitions of the automaton. In the initial position, player Automaton chooses a transition where the root is uncolored; in a position corresponding to a state, player Automaton chooses a transition where the root colored by that state. In a position corresponding to a transition, player Pathfinder chooses a state that appears in some leaf that is colored by a state. The parity condition is on the sequence of states that appears in the play, with the order on states and notion of accepting state inherited from the automaton.

Lemma 32. *For a generalised parity automaton, the following are equivalent:*

1. *the automaton has an accepting run;*
2. *player Automaton wins the acceptance game;*
3. *the automaton has a regular accepting run.*

Proof. By using classical proofs, one can easily see that the lemma is true for real runs; i.e. when the runs of generalised parity automata are restricted to real trees. The point of the proof is to show that the equivalence extends to profinite trees.

Let Q be the states of the automaton. For a transition in the automaton, define its *profile* to be the pair (q, P) where: q is the color of the root or $\bot$ if the root is uncolored, and P is the set of states that appear as colors of leaves. Define the *profile of a run* ρ of the automaton to be the set of profiles the transitions used by the run. For a set X of transition profiles, consider the following parity game, call it *the profile game of* X, which is also played by Automaton and Pathfinder. The initial position is $\bot$. In a position $q \in Q \cup \{\bot\}$, player Automaton chooses a set $P \subseteq Q$ with $(q, P) \in X$, and the game continues from position P. In a position $P \subseteq Q$, player Pathfinder chooses an element $p \in P$, and the game continues from position p. The winning condition for player Automaton is that the maximal state seen infinitely often is accepting.

We now prove the equivalence of the conditions in the lemma. Clearly 3 implies 1, so we only to show the remaining implications.

1 implies 2. It is not difficult to see that player Automaton wins the acceptance game of the automaton if and only if player Automaton wins the profile game of X, where X is the set of all profiles of transitions used in the automaton. Therefore, to prove that 1 implies 2, it suffices to show that if ρ is an accepting run, then player Automaton wins the profile game of the profile of ρ. An accepting run with profile X satisfies the following two MSO-definable properties: a) it has profile X; and b) it satisfies the parity condition, i.e. on every infinite path with infinitely many colored nodes, the maximal color appearing infinitely often is accepting. The conjunction of these properties is MSO-definable. If an MSO formula is true in some profinite tree, then it is true in some real tree. Therefore some real tree satisfies a) and b). For real trees, a) and b) imply that Eve wins the game associated to X.

2 implies 3. Let the transitions of the automaton be Δ. A strategy of player Automaton in the acceptance game is a function

$$\sigma : \bot \cdot (\Delta \cdot Q)^* \to \Delta.$$

If the range of the strategy contains only real trees, then the strategy induces a real run, call it the *unfolding of* σ, which is accepting if and only if the strategy is winning. When the strategy is furthermore memoryless, then its unfolding is regular, and has degree of regularity bounded by the number of states. Since the acceptance game is a parity game, if player Automaton wins it, then player Automaton wins it via a memoryless strategy, which is a function

$$\sigma : Q \cup \{\bot\} \to \Delta.$$

Because every profinite tree can be approximated arbitrarily closely by real trees, we can choose for every $\varepsilon > 0$ a function

$$\sigma_\varepsilon : Q \cup \{\bot\} \to \text{real trees} \tag{19}$$

such that for every argument q, $\sigma_\varepsilon(q)$ is at distance at most ε from $\sigma(q)$ and has the same profile. Note that σ_ε is not necessarily a strategy in the acceptance game, since its values might not be transitions. Since the unfolding of σ satisfies the parity condition, and the function σ_ε uses the same profiles, it follows that the unfolding of σ_ε, call it ρ_ε, also satisfies the parity condition. The unfolding ρ_ε has degree of regularity bounded by the number of states. By compactness, we can assume without loss of generality that there is a limit of ρ_ε as ε tends to zero. By Lemma 31, the limit has degree of regularity bounded by the number of states and is therefore regular. The parity condition is satisfied in the limit, because it is MSO-definable and therefore preserved in the limit. Finally, every factor of the limit is of the form $\sigma(q)$, and therefore the limit is an accepting run of $\mathcal{A}$. For the last statement, we use the fact that every ρ_ε has degree of regularity bounded by the number of states. □

Proof (of Theorem 4). We only need to show the inclusion

$$L(\mathcal{A}) \subseteq \overline{L_{\text{reg}}(\mathcal{A})}.$$

By Lemma 30, it suffices to show that for every MSO formula true in some tree accepted by $\mathcal{A}$, the same MSO formula is true in some tree accepted by $\mathcal{A}$ via a regular run. In other words, we need to show that if $\mathcal{B}$ is a (non-generalised) parity automaton, then if it $\mathcal{A}$ and $\mathcal{B}$ accept some common tree, then they accept a tree where the accepting run of $\mathcal{A}$ is regular. This is proved by applying Lemma 32 to a product automaton.

□

E.3 Approximating a profinite factorised tree

Define a *factorised tree over alphabet* A to be a tree over alphabet A together with a partial coloring that uses one color. In other words, this is a (possibly profinite) tree over the alphabet $A \times \{c, \bot\}$, where c is the name of the unique color. We denote factorised trees by λ. A *factorisation* of a profinite tree t over A is a factorised tree that projects to t. In this section, we prove Lemma 36, which says that a profinite factorised tree λ can be approximated by a real factorised tree λ_r so that λ and λ_r are close, and λ_r-factors resemble λ-factors with the resemblance growing closer as the distance from the root increases.

Before proving Lemma 36, we need some auxiliary results. The following lemma shows that if f is a relabelling, and $f(t)$ is a limit of trees from a set R, then t is a limit of trees with images in R.

Lemma 33. *Let A, B be alphabets and let $f : A \to B$. Then*

$$f^{-1}(\overline{R}) \subseteq \overline{f^{-1}(R)} \qquad \text{for every set } R \text{ of profinite trees over } B.$$

Proof. Let s be an element of the left side in the inclusion of the lemma. For $n \in \mathbb{N}$, let S_n be the profinite trees over alphabet A that are at distance $1/n$ from s. This set is MSO-definable and therefore clopen. The image $f(S_n)$ is also MSO-definable because MSO-definable sets are closed under images of relabelings (by using existential quantification). Since $f(s)$ belongs to both $f(S_n)$ and $\overline{R}$, it follows that $f(S_n)$ and $\bar{R}$ have nonempty intersection. Since $f(S_n)$ is open, it follows that $f(S_n)$ and R have nonempty intersection, let s_n be an element of this intersection. Since the elements s_n have smaller and smaller distances to s, it follows that the limit of s_n must be s. Also every s_n is in $f^{-1}(R)$, and s belongs to the right side of the inclusion in the statement of the lemma. □

Define a *pre-parity automaton* like a parity (or generalised parity) automaton, except that it does not have the transitions. If $\mathcal{A}$ is a pre-parity automaton and Δ is a set of transitions, then let $\mathcal{A}[\Delta]$ denote the generalised parity automaton which is like $\mathcal{A}$ and has transitions Δ. Sometimes, we will denote generalised parity automata by $\mathcal{A}[\Delta]$, if we want to make explicit the transitions.

In a real paritally colored tree ρ, define the *color depth* of a node to be the number of colored ancestors of that node. A factor of ρ at color depth n is one obtained from a color zone whose root has color depth n. A subtree at color depth n is a subtree rooted in a node at color depth n. We emphasize that these notions are defined for real trees.

Lemma 34. *Let $\mathcal{A}$ be a pre-parity automaton, Δ a set of real trees, and $\Gamma \subseteq \bar{\Delta}$. Let $\varepsilon > 0$ be a real number. If $\mathcal{A}[\Gamma]$ has an accepting run, then $\mathcal{A}[\Delta]$ has a real accepting run ρ such that for every $n \in \mathbb{N}$:*

1. *there are finitely many subtrees of ρ with colored roots at color depth n;*
2. *every factor of ρ at color depth n is at distance $\frac{\varepsilon}{n}$ from some transition in Γ.*

Proof. Let the input alphabet of the automaton be A and the states be Q. By Lemma 32, if $\mathcal{A}[\Gamma]$ is nonempty, then player Automaton wins the acceptance game. As in the proof of Lemma 32, a winning strategy is a function

$$\sigma : Q \cup \{\bot\} \to \Gamma.$$

By the assumption that $\Gamma \subseteq \overline{\Delta}$, for every $n \in \mathbb{N}$ there is a function

$$\sigma_n : Q \cup \{\bot\} \to \Delta$$

such that for every argument q, $\sigma_n(q)$ is at distance at most $\frac{\varepsilon}{n}$ from $\sigma(q)$ and has the same profile. In the acceptance game for $\mathcal{A}[\Delta]$ consider a strategy for player Automaton where function σ_n is used in the n-th round. This is a winning strategy by the assumption that σ was a winning strategy and that the profiles match for σ and σ_n. Therefore the run of $\mathcal{A}[\Delta]$ constructed from unfolding this strategy is an accepting run. Item 1 from the statement of the lemma holds because the strategy depends only on the current state and the number of rounds played so far. Item 2 holds because $\sigma_n(q)$ is at distance $\frac{\varepsilon}{n}$ from $\sigma(q)$. □

Consider a generalised parity automaton $\mathcal{A}[\Delta]$ which inputs factorised trees over some alphabet A. This automaton is called *factor consistent* if for every run ρ over an input λ, every ρ-factor projects to a λ-factor when the states of $\mathcal{A}$ are forgotten. In other words, the automaton uses states in the places where λ has a defined colour.

Lemma 35. *Every* MSO*-definable set of profinite factorised trees is recognised by a generalised parity automaton which is factor consistent, and such that the transitions are* MSO*-definable*

Proof. Let $\mathcal{A}$ be a parity automaton that recognises the MSO-definable property in the assumption of the lemma. We construct an automaton which, on a given input λ, labels λ by a run of $\mathcal{A}$, but then removes the states from nodes that are not colored by λ. □

We are now ready to state the main result of Section E.3.

Lemma 36. *Let λ be a profinite factorised tree such that every λ-factor is in $\bar{R}$ for some set of real treess R, and let $\varepsilon > 0$ be a real number. There exists a real factorised tree λ_r such that*

1. *all λ_r-factors are in R;*
2. *every λ_r-factor at color depth n is at distance at most $\frac{\varepsilon}{n}$ from some λ-factor;*
3. *for every $n \in \mathbb{N}$, there are finitely many λ_r-factors at color depth n.*
4. *λ and λ_r are at distance at most ε_0;*

Proof. Consider the ε-ball around λ, which is MSO-definable. Apply Lemma 35 to this ball, yielding a factor consistent generalised parity automaton, call it $\mathcal{A}[\Sigma]$. Let Σ_r be the real trees in Σ. Like for every MSO-definable set, $\Sigma = \overline{\Sigma_\mathrm{r}}$. Define Δ to be the real trees in Σ and which project to R once the transitions of the automaton $\mathcal{A}$ are ignored. By assumption that all λ-factors are in $\bar{R}$, it follows that $\mathcal{A}[\bar{\Delta}]$ accepts λ, and is therefore nonempty. Let $\Gamma \subseteq \bar{\Delta}$ be those transitions which project to a λ-factor. The automaton $\mathcal{A}[\Gamma]$ still accepts λ.

Apply Lemma 34 to $\mathcal{A}$, Γ and Δ, yielding a real run ρ of $\mathcal{A}[\Delta]$ over some input λ_r. The automaton $\mathcal{A}[\Delta]$ is factor consistent, since it is obtained from the factor consistent automaton $\mathcal{A}[\Sigma]$ by removing transitions. Therefore, the λ_{r}-factors are exactly the projections of the ρ-factors. Since every element of Δ projects to R, we get item 1 in the statement of the lemma. Items 2 and 3 are the two properties in the statement of Lemma 34. The automaton $\mathcal{A}[\Delta]$ recognises a subset of the ε-ball around λ, which yields item 4. □

F Nonemptiness for automaton chains

In this section we prove Lemma 1, which says that nonemptiness is decidable for automaton chains. Lemma 1 follows immediately from Lemmas 37 and 38 given below.

Lemma 37. *Given cost functions α, β of cost* WMSO *and a formula φ of* MSO*, one can decide if there is a profinite tree t that satisfies*

$$t \models \varphi \qquad \text{and} \qquad \alpha(t) < \infty \qquad \text{and} \qquad \beta(t) = \infty.$$

Proof. We say that a cost function α is dominated by a cost function β over a set L of trees if for every set of trees $K \subseteq L$, if β is bounded over K then so is α. As shown by Vanden Boom in [15], the domination problem is decidable, assuming that α and β are defined in cost WMSO, and L is MSO definable. It is not difficult to see that the property in the statment of the lemma is equivalent to β not being dominated by α over φ. □

Lemma 38. *For every automaton chain $\mathcal{A}$ with input alphabet A, one can compute a relabeling $f : B \to A$, cost formulas α, β over cost* WMSO *over alpahbet B, and a formula φ of* MSO *over alphabet B, such that the language recognised by $\mathcal{A}$ is the image under f of the profinite trees t satisfy*

$$t \models \varphi \qquad \text{and} \qquad \alpha(t) < \infty \qquad \text{and} \qquad \beta(t) = \infty$$

Proof (sketch). The proof is by induction on the depth of the automaton chain. The language before the projection describes runs of the automaton chain. The key observation is that for every cost function α definable in cost WMSO, one can compute cost functions α_{sup} and α_{inf} of cost WMSO such that for every real tree t with a real partial coloring ρ,

$$\alpha_{\mathrm{sup}}(t \otimes \rho) = \sup_{\sigma} \alpha(\sigma) \qquad \alpha_{\mathrm{inf}}(t \otimes \rho) = \inf_{\sigma} \alpha(\sigma) \tag{20}$$

with σ ranging over ρ-factors of t. The formula defining α_{sup} says that for every ρ-colored node x (or the x being the root), the value of the factor with root x is small. The formula for α_{inf} is obtained using duality of cost functions definable in cost WMSO. It is not difficult to see that the equalities (20) extend to profinite trees with profinite partial colorings. □

G Normal form of WMSO+UP automata

In this part of the appendix, we prove Lemma 3, which says that every automaton can be converted into the following normal form:

(a) for every run, in the counter graph generated by the automaton, every bounded counter is separated and root-directed.
(b) for every state q there are sets of counters $larcut(q)$ and $larcheck(q)$ with the following property. For every run and every finite path in the run that starts and ends in state q and does not visit bigger states in the meantime,
   - the counters checked in the path are exactly $larcheck(q)$;
   - the counters cut in the path are exactly $larcut(q)$;

Recall that in Theorem 7, a nested counter language is converted into a WMSO+UP automaton that satisfies condition (a). Therefore, for the purposes of deciding satisfiability of WMSO+UP logic, the part of Lemma 3 which assures condition (a) is not actually needed, because in the satisfiability algorithm for WMSO+UP logic we only use automata produced via Theorem 7. Nevertheless, the part of Lemma 3 concerning (a) is still true: one can take an arbitrary WMSO+UP automaton, convert it to existential WMSO+UP logic, and then use Theorems 6 and 7 to convert it back into a WMSO+UP automaton that satisfies condition (a).

The main contribution of Lemma 3 concerns condition (b). This is achieved by using the following lemma, with A being the states of the WMSO+UP automaton.

Lemma 39. *Let A be an alphabet. There is a deterministic word automaton $\mathcal{A}$ with a totally ordered state space Q and a function*

$$lar : Q \to P(A)$$

such that for every word $w \in A^*$ and positions $i < j$, if state q appears in positions $i < j$, and no state bigger than q appears in positions $\{i, \dots, j\}$, then the set of letters that appears in positions $\{i, \dots, j\}$ is exactly $lar(q)$.

Proof. This automaton is the latest appearance record of McNaughton. A state of the automaton is a pair (w, v) such that $wv \in A^*$ does not contain any letter twice. In particular, the number of states is the finite number

$$\sum_{n \le |A|} (n+1) \cdot n!$$

One invariant that will be satisfied by the automaton is that:

> (*) If the state of the automaton after reading u is (w, v), then wv is obtained from u by only keeping the last appearance of each letter.

The division of wv into (w, v) is such that the comma indicates the place where the last letter of wv was in the previous state. More specifically, the initial state is $(\varepsilon, \varepsilon)$ and the transition function is defined by

$$\delta((w,v),a) = \begin{cases} (x, ya) & \text{if } wv = xay \text{ for some } x, y \\ (\varepsilon, wva) & \text{otherwise} \end{cases}$$

Consider any total order $\leq$ on the states of this automaton, such that if v is shorter than v', then $(w, v) < (w', v')$. Finally define $lar(w, v)$ to be the set of letters in v. To prove the statement of the lemma, consider a run

$$(\varepsilon, \varepsilon) = (w_0, v_0) \xrightarrow{a_1} (w_1, v_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (w_n, v_n)$$

such that for some $i < j$,

$$(w_i, v_i) = (w_j, v_j) \qquad \text{and} \qquad (w_k, v_k) \leq (w_i, v_i) \text{ for } k \in \{i, \ldots, j\}.$$

We need to show that the set words $a_{i+1} \cdots a_j$ and v_i have the same letters. The transitions of the automaton are defined in such a way that the combined length (w, v) grows or stays the same during a run, and therefore

$$|w_i v_i| = \cdots = |w_j v_j|.$$

Since no bigger states appear between positions i and j, it follows that all the words $v_i, \ldots, v_j$ are at most as long as v_i, and therefore all the words $w_i, \ldots, w_j$ are at least as long as w_i. It follows that no letters from w_i can appear in $a_{i+1} \cdots a_j$. It remains to show that all letters from v_i appear in $a_{i+1} \cdots a_j$. In the state (w_{j-1}, v_{j-1}), the letter $a_j = a_i$ is before all other letters that appear in v_i; therefore these other letters must have been seen after the last appearance of a_i, which was at position i or later. □

H A profinite characterisation of partial accepting runs

This part of the appendix is devoted to showing (3) from Section 4. Recall that in Section 4, based on a WMSO+UP automaton $\mathcal{A}$ in normal form, we define sets R_{q*}, R_q of real trees and generalised parity automata $\mathcal{R}_{q*}$ and $\mathcal{R}_q$. The main result in Section 4 is then

$$\overline{R_{q*}} = \overline{L(\mathcal{R}_{q*})} \qquad \text{and} \qquad \overline{R_q} = \overline{L(\mathcal{R}_q)}. \tag{3}$$

As shown at the end of Section 4, using (3) one can reduce emptiness of $\mathcal{A}$ to emptiness of the automata $\mathcal{R}_q$, which are automaton chains, and therefore have decidable emptiness by Lemma 1. Therefore, to complete the emptiness algorithm for WMSO+UP automata, it suffices to show (3), which we do in this part of the appendix.

For the reader's convenience, we recall the definitions of $R_{q*}, R_q, \mathcal{R}_{q*}$ and $\mathcal{R}_q$. In all cases, the input alphabet is the one used to describe runs of $\mathcal{A}$, i.e. it is the product $A \times Q$ where A is the input alphabet of $\mathcal{A}$ and Q is the state space of $\mathcal{A}$.

The sets R_q and R_{q*} are sets of real trees. The set R_q consists of the accepting partial runs where states strictly bigger than q appear only in nodes with finitely many descendants. The set R_{q*} is the subset of R_q where state q is allowed only finitely often on every path.

The automata $\mathcal{R}_{q*}$ and $\mathcal{R}_q$ are generalised parity automata which recognise properties of profinite trees. Both automata have one state only, call it "state", this state is rejecting in $\mathcal{R}_{q*}$ and accepting in $\mathcal{R}_q$. A transition of $\mathcal{R}_{q*}$ is any profinite partially {"state"}-colored tree σ over $A \times Q$ such that:

1. the projection of σ onto the $A \times Q$ coordinate belongs to $\overline{R_p}$, where p is the predecessor of q in the order on states; and
2. for every root-to-leaf path in σ which ends in a leaf with defined color "state", the maximal value of the Q coordinate is q.

A transition of this automaton is any profinite partially {"state"}-colored tree σ over $A \times Q$ such that:

1. the projection of σ onto the $A \times Q$ coordinate belongs to $\overline{R_{q*}}$; and
2. for every root-to-leaf path in σ which ends in a leaf with defined color "state", the maximal value of the Q coordinate is q.

3,4. $\alpha(\sigma) < \infty$ and $\beta(\sigma) = \infty$ holds for cost functions α, β defined in Section 4;

The left equality of (3) is shown in Section H.1, the right equality is shown in Section H.2.

H.1 $\overline{R_{q*}} = \overline{L(\mathcal{R}_{q*})}$

Section H.1 is devoted to proving that

$$\overline{R_{q*}} = \overline{L(\mathcal{R}_{q*})} \tag{21}$$

holds for every state q.

Left-to-right inclusion. We begin with the left-to-right inclusion in (21). We use the name *zone* for a connected set of nodes in a real tree, which is also closed under siblings. By connectedness, a zone has a unique minimal node (with respect to the descendant relation), this node is called the *root* of the zone. If X is a zone in a real tree t, then $t|X$ denotes the tree obtained from t by keeping only the nodes from X. We say a zone Y is a *well-founded extension* of a zone $X \supseteq Y$ if X and Y have the same root and there are no infinite paths in $Y - X$.

Lemma 40. *Every zone X in a partial accepting run ρ of $\mathcal{A}$ admits a well-founded extension to a zone Y such that $\rho|Y$ is a partial accepting run.*

Proof. Let c be an unbounded counter. We say that a run is c-accepting if the unboundedness condition holds only for c, i.e. on every infinite path that checks c infinitely often, the limsup is infinite for the value of counter c in nodes where c is checked. Note that if Y is a well-founded extension of X, and $\rho|X$ is c-accepting, then also $\rho|Y$ is c-accepting, because a) every infinite path that is contained in Y is already contained in X; and b) the values of counters can only grow when adding nodes to a counter tree.

Claim. For every unbounded counter c, every zone X admits a well-founded extension to a zone Y such that $\rho|Y$ is c-accepting.

Proof. Define[2] $[\![\rho]\!]^X(x,c)$ to be the value of counter c in node x, but only counting those counter paths that are entirely contained in X. This is the same as the value of counter c in the node corresponding to x in the counter tree corresponding to $\rho|X$. It is not difficult to see that every X admits a well-founded extension Y such that

$$[\![\rho]\!]^Y(x,c)\begin{cases}\text{is equal to } [\![\rho]\!](x,c) & \text{when } [\![\rho]\!](x,c)<\infty\\ \text{is at least } |x| & \text{when } [\![\rho]\!](x,c)=\infty\end{cases}\qquad \text{for every } x\in X.$$

This is the zone required by the claim. □

Using the claim, we prove the lemma. Let $c_1,\dots,c_n$ be the unbounded counters. Applying the claim n times, we find zones

$$X = X_0, X_1, \dots, X_n$$

such that every $X_i \in \{X_1,\dots,X_n\}$ is a well-founded extension of X_{i-1} and $\rho|X_i$ is c_i-accepting. The well-founded extension relation is transitive, so X_n is a well-founded extension of X. As we have observed, being c-accepting is closed under well-founded extensions, and therefore X_n is c_i-accepting for every unbounded counter. Finally, since ρ is a partial accepting run, and the boundedness and parity acceptance conditions are preserved under removing nodes, the run $\rho|X_n$ satisfies the boundedness and parity acceptance conditions. □

Define $R_{<q}$ to be the union of R_p ranging over states $p<q$.

Lemma 41. *Let $\rho \in R_q$ and let x be a node in ρ. There is a zone X with root x such that $\rho|X \in R_{<q}$ and for every maximal node $y \in X$, either y is a leaf of ρ, or q is the maximal state that appears on the path from x to y.*

Proof. Let Z be the set of (not necessarily proper) descendants of x that have a state bigger or equal to q, and are closest to x for that property. Apply Lemma 40 to the zone

$$\{z : x \le y \le z \text{ for some } z \in Z\} \tag{22}$$

[2] Similar notation, but with X being a lower index, was used in Section D.2 for a different concept. In Section D.2, the counter paths were prohibited to pass through an ancestor of x in the set X.

yielding a well-founded extension Y such that $\rho|Y$ is a partial accepting run. In the special case when x has a state bigger or equal to q, the set in (22) is equal to Y and is also equal to $\{x\}$. For every maximal node $y \in X$, the path from x to y contains a node from (22), and therefore a state bigger than equal to q. Define X by adding to Y all nodes that have an ancestor in Y with a state strictly bigger than q. The set X is a zone and a well-founded extension of Y, since the assumption $\rho \in R_q$ implies that every state strictly bigger than q has finitely many descendants. In the proof of Lemma 40, we showed that well-founded extension preserves partial accepting runs, and therefore $\rho|X$ is a partial accepting run. Since X was obtained from (22) by two consecutive well-founded extensions, and in (22) nodes with state q have zero descendants, it follows that nodes with state q have finitely many descendants in X, and therefore $\rho|X$ belongs to $R_{<q}$. Finally, every maximal node $y \in X$ which is not a leaf has an ancestor in Z, and therefore on the path from x to y there is a state bigger or equal to q. Furthermore, if the state is strictly bigger than q, then y is a leaf of ρ, which finishes the proof of the lemma. □

Lemma 42. *Every run in R_{q*} is accepted by $\mathcal{R}_{q*}$.*

Proof. Let $\rho \in R_{q*}$. For nodes x, y in ρ, define $x \preceq y$ to hold if x is a descendant of y (note that $\preceq$-bigger nodes are closer to the root) and there is at least one occurrence of state q on the path from y to x. The assumption that ρ belongs to R_{q*} implies that q appears finitely often on every path, and therefore $\preceq$ is a well-founded order. By induction on this order, we show that for every node x in ρ, there is a run of $\mathcal{R}_{q*}$ on the subtree $\rho|_x$.

Let ρ be a run in R_{q*} and let x be a node in ρ. Apply Lemma 41 to x and ρ, yielding a zone X with root x such that $\rho|X \in R_{<q}$. For every maximal node $y \in X$ that is not a leaf of ρ, state q appears on the path from x to y, and therefore $y \prec x$. By the induction assumption, the subtree $\rho|_x$ is accepted by $\mathcal{R}_{q*}$. Combining the run $\rho|X$ with the accepting of $\mathcal{R}_{q*}$ on those subtrees, we get an accepting run of $\mathcal{R}_{q*}$ on ρ. □

The above lemma says that

$$R_{q*} \subseteq L(\mathcal{R}_{q*}).$$

Since closure is a monotone operator, the left-to-right inclusion in (21) follows.

Right-to-left inclusion. By Lemma 4, the set of trees which admit a regular accepting run of $\mathcal{R}_{q*}$ is dense in all trees accepted by $\mathcal{R}_{q*}$. Therefore, to prove the right-to-left inclusion

$$\overline{R_{q*}} \supseteq \overline{L(\mathcal{R}_{q*})}$$

in (21), it suffices to show

$$\overline{R_{q*}} \supseteq L_{\text{reg}}(\mathcal{R}_{q*}))$$

which is stated in the following lemma.

Lemma 43. *$\overline{R_{q*}}$ contains every input accepted by $\mathcal{R}_{q*}$ via a regular run.*

Proof. Let ρ be a profinite tree which admits a regular run, call it λ, of the generalised parity automaton $\mathcal{R}_{q*}$. By definition of the automaton $\mathcal{R}_{q*}$, we know that every λ-factor is in $\overline{R_{<q}}$.

Let $k \in \mathbb{N}$ be such that λ has degree of regularity at most k. Let $\lambda_1, \lambda_2, \ldots$ be a sequence of real factorised trees that tends to λ. By Lemma 31, we can assume without loss of generality that every λ_n has degree of regularity at most k. Having a well-founded factorisation is MSO-definable, and therefore we can assume that every coloring λ_n is well-founded. Every real tree λ_n satisfies the property "on every path, there are at most k nodes colored by λ_n". Since this property is MSO-definable, it is preserved in the limit, and therefore in λ there are at most k nodes on every path which are colored by λ.

Choose some real number $\varepsilon > 0$. Apply Lemma 36 to λ, yielding a real factorised tree λ_r. Note that λ_r is a factorisation of some real partial run of $\mathcal{A}$, call this real partial run ρ_r. If ε is small enough with respect to k, item 4 of Lemma 36 which says that λ and λ_r are at distance at most ε, implies that the maximal λ_r-depth is also k. Item 3 says that at every λ_r-depth there are finitely many λ_r-factors, and since there are finitely many λ_r-depths, it follows that altogether there are finitely many different λ_r-factors. Item 1 implies that every λ_r-factor is an accepting partial run. This implies that every λ_r-factor satisfies the boundedness acceptance condition. A real partial run of $\mathcal{A}$ with finitely many factors, each of which satisfies the boundedness acceptance condition, must itself satisfy the boundedness acceptance condition. Therefore, ρ_r satisfies the boundedness acceptance condition. The parity and unboundedness acceptance conditions are also satisfied in ρ_r, because for every infinite path in ρ_r, all but finitely many nodes of the path are in some λ_r-factor, where the parity and unboundedness acceptance condition are satisfied. □

H.2 $\overline{R_q} = \overline{L(\mathcal{R}_q)}$

The rest of Section H is devoted to showing

$$\overline{R_q} = \overline{L(\mathcal{R}_q)}.$$

When q is a parity-rejecting state of $\mathcal{A}$, then the equality above is the same one as in the previous section. Therefore, we consider the case when q is parity-accepting.

Left-to-right inclusion. We begin by proving the left-to-right inclusion. As in Section H.1, it suffices to show that every run in R_q is accepted by $\mathcal{R}_q$.

For a bounded counter c, define a c-cut zone in a real run of $\mathcal{A}$ to be a maximal connected set of nodes where counter c is not cut. Define the c-cut zone of a node x to be the c-cut zone that contains x. The boundedness acceptance condition says that for every c-cut zone, the value of c has finite limsup.

Recall that a real factorisation of a real run ρ is a partition of its nodes into colored and uncolored nodes.

Lemma 44. *Every $\rho \in R_q$ admits a real factorisation γ such that*

1. *every γ-factor is in R_{q*};*
2. *for every finite path with source and target colored by γ:*
   (a) *the maximal visited state is q;*
   (b) *on nodes of the path, all bounded counters outside $larcut(q)$ are bounded by a finite number n_x that only depends on the source node x of the path.*

Proof. For every node x in ρ, apply Lemma 41, yielding a zone Y_x, and let X_x be the maximal nodes of Y_x that are not leaves of ρ. Lemma 41 says that $\rho|X_x$ belongs to $R_{<q}$ and that on every path from x to a node in X_x, the maximal visited state is q. Let X be the smallest set that contains X_x for x being the root, and such that if $x \in X$, then $X_x \subseteq X$. Note that X does not contain the root. Define γ so that it colors a node x if and only if $x \in X$ and on some path that begins in x, there are infinitely many nodes from X. By construction, γ satisfies items 1 and 2a in the lemma. Below we prove item 2b.

Let π be a path as in item 2b. For every bounded counter c, let n_c be the maximal value of counter c in the c-cut zone of the source node of π. This number is finite by assumption that ρ is accepting. Let n be the maximal value of of n_c, ranging over bounded counters $c \notin larcut(q)$. To prove item 2b, it suffices to show that no bounded counter $c \notin larcut(q)$ is cut on the path π. This follows from the following claim.

Claim. Every path π as in item 2b is contained in a finite path that begins and ends in state q, and visits no bigger nodes. In particular, no bounded counter $c \notin larcut(q)$ is cut on π.

Proof. Let the source and target nodes of π be x and y. By assumption that x is colored by γ, it belongs to $X_{x'}$ for some x', possibly x' being the root. Therefore, there is some path from x' to x such that the maximal visited state is q. By definition of the nodes colored by γ, the set X_y is nonempty and therefore there is some path that begins in y and such that the maximal visited state is q. □

□

In Lemma 44, we defined a real factorisation γ of a real run. In the next lemma, we produce a profinite factorisation of a real run. The lemma refers to the cost functions α and β that were defined in the definition of the automaton $\mathcal{R}_q$, which we recall here:

– the cost function α is defined by

$$\alpha(\sigma) = \qquad \max_c \max_x \quad [\![\sigma]\!](x, c)$$

with c ranging over bounded counters not in $larcut(q)$ and x ranging over nodes which do not have an ancestor where c is cut.

– the cost function β is defined by

$$\beta(\sigma) = \begin{cases} \min_c \min_x \max_y \quad [\![\sigma]\!](y,c) & \text{if the root of } \sigma \text{ has defined color “state”} \\ \infty & \text{otherwise} \end{cases}$$

with c ranging over unbounded counters in $larcheck(q)$, x ranging over leaves with defined color “state”, and y ranging over ancestors of x where c is checked.

Lemma 45. *Let ρ and γ be as in Lemma 44. For every γ-colored node x, there is a factorisation λ of the subtree $\rho|_x$ such that:*

1. *every λ-factor of ρ is in $\overline{R_{q*}}$;*
2. *the value of α is smaller than ∞ on every λ-factor of $\rho|_x$.*
3. *the value of β is ∞ on every λ-factor of $\rho|_x$.*

Proof. Let ρ, γ and x be as in the statement of the lemma. Let $n_x \in \mathbb{N}$ be the bound from item 2b in Lemma 44.

Claim. For every $n \in \mathbb{N}$, there is a factorisation λ_n of $\rho|_x$ such that:

1. every λ_n-factor of $\rho|_x$ is accepted by $\mathcal{R}_{q*}$;
2. the value of α is at most n_x on every λ_n-factor of $\rho|_x$.

$3n$. the value of β is at least n on every λ_n-factor of $\rho|_x$.

Proof. We define sets of nodes

$$Y_0, Y_1, \ldots \subseteq \{y : y \text{ is a } \gamma\text{-colored descendant of } x\}.$$

The set Y_0 contains only x. Suppose that Y_{n-1} has already been defined. Define Z_n to be those γ-colored nodes z, such that for some ancestor $y \in Y_{n-1}$, all unbounded counters $c \in larcheck(q)$ are checked between y and z and have value at least n. Note that every path which begins in x and visits γ-colored nodes infinitely often must eventually see a node from Z_n. This is because for such a path, the maximal state seen infinitely often is q, and therefore all counters in $larcheck(q)$ are checked infinitely often, and therefore they must have unbounded values on the path. Define Y_n to be the minimal nodes of Z_n. Define λ_n to be the Boolean coloring which is defined in the nodes

$$Y_n \cup Y_{n+1} \cup \cdots$$

From our observation on Z_n, it follows that if a path is entirely contained in an λ_n-factor, then it sees finitely many γ-colored nodes. This implies that every λ_n-factor belongs to R_{q*}, and is therefore accepted by $\mathcal{R}_{q*}$. Item 2 of the claim follows from item 3 of Lemma 44. Item 3 follows by definition of λ_n. □

A run of $\mathcal{R}_q$ is a factorisation that satisfies properties 1 and 2 in the claim, and satisfies property $3n$ for every n. Let λ_n be the run of $\mathcal{R}_q$ in the statement of the claim. Note that if $m > n$, then λ_m satisfies property $3n$ as well. By compactness, some subsequence of $\lambda_1, \lambda_2, \ldots$ has a limit, call it λ. Property 1 is MSO-definable, and therefore closed, and therefore satisfied in the limit λ. Property 2 is closed by Lemma 4, and therefore satisfied in the limit λ. For every n, property $3n$ is MSO-definable, and therefore closed. Since property $3n$ is satisfied by almost all elements of the sequence $\lambda_1, \lambda_2, \ldots$ it must be satisifed in the limit λ. Therefore λ is an accepting run of $\mathcal{R}_q$. □

Lemma 46. *Let $\rho \in R_q$ and let Y be a set of nodes in ρ such that:*

- *$\mathcal{R}_q$ accepts every subtree of ρ rooted in a node from Y;*
- *$\mathcal{R}_{q*}$ accepts the tree obtained from ρ by removing nodes from Y together with their subtrees.*

Then $\mathcal{R}_q$ accepts ρ.

Proof. By combining the accepting runs. □

We now show every $\rho \in R_q$ is accepted by $\mathcal{R}_q$, which completes the proof of

$$\overline{R_q} \subseteq \overline{L(\mathcal{R}_q)}$$

Apply Lemma 44 yielding a factorisation γ. The set Y of minimal γ-colored nodes satisfies items 1 and 2 of Lemma 46, with item 2 following from Lemma 45.

Right-to-left inclusion. We are left with showing

$$\overline{R_q} \supseteq \overline{L(\mathcal{R}_q)}.$$

As in Section H.1 it suffices to show the following lemma.

Lemma 47. *$\overline{R_q}$ contains every input accepted by $\mathcal{R}_q$ via a regular run.*

The rest of Section H.2 is devoted to showing this lemma. Let then ρ be a tree accepted by the automaton $\mathcal{R}_q$, via a regular accepting run λ. We need to show that ρ can be approximated by trees from R_q with arbitrary precision $\varepsilon > 0$. Choose some ε that is small enough (smaller than the ε from Lemma 49). Apply Lemma 36 to λ and ε with the set R being R_{q*}, yielding a real factorisation λ_r of a real partial run ρ_r of $\mathcal{A}$. By item 4 the lemma, the distance between λ and λ_r is at most ε. Since ρ is a projection of λ and ρ_r is a projection of λ_r, and projections are non-expansive, it follows that the distance between ρ and ρ_r is at most ε. It remains to prove that ρ_r is an accepting run. We first deal with the unboundedness condition, and then the boundedness condition.

Lemma 48. *The run ρ_r satisfies the unboundedness and parity conditions.*

Proof. We only do the more interesting case of the unboundedness condition.

Consider an infinite path π in ρ_r. If the path passes through finitely many nodes colored by λ_r, then all but a finite part of the path is included in some λ_r-factor. By item 1 of Lemma 36 this factor is in R_{q*}, and therefore the unboundedness and parity acceptance conditions are satisfied by the path. The more interesting case is when the path visits λ_{r}-colored nodes infinitely often. Toward a contradiction, suppose that some unbounded counter c is checked infinitely often by the path, but its values in checked nodes are bounded by some $k \in \mathbb{N}$.

Decompose π into finite paths where the source and target nodes are colored by λ_{r}, call them

$$\pi_1, \pi_2, \dots$$

For every n, let $\sigma_{\mathrm{r}n}$ be the λ_{r}-factor that contains the path π_n. By item 2 of Lemma 36, there is some λ-factor, call it σ_n, which is at distance at most $\frac{\varepsilon}{n}$ from $\sigma_{\mathrm{r}n}$. For every unbounded counter $c \in \mathit{larcheck}(q)$, and every $m \in \mathbb{N}$, in every λ-factor with root colored by λ, every path from the root to a λ-colored node satisfies the following MSO-definable properties:

1. the maximal state that appears on the path is q;
2. the maximal checked value of c on the path is at least m.

It follows that for every c and m, these properties are satisified by almost all the λ_{r}-paths π_n. This implies the unboundedness. □

Boundedness condition. We are now left with the boundedness condition, which is where we use the assumption that λ is regular.

Let c be a bounded counter. A node x in a real run is said to be a

- c-increment if counter c in node x is transferred with an increment to counter c in the parent of x;
- c-cut if the state in x cuts counter c;
- c-reset if counter c in node x is not transferred to

Define the c-value of a path π to be the least upper bound on the number of c-increment nodes that can be found between nodes $x < y \in \pi$ such that there are no c-resets between x and y, and there are no c-cuts between the source of π and y. Because the automaton is in normal form, the value of counter c in a node is the maximal c-value of paths that leave the node.

For a finite path π in a factorisation, define its *factor decomposition* to be the unique decomposition

$$\pi = \pi_0 \cdots \pi_n$$

such that the path π_0 contains no colored nodes (and is therefore possibly empty), and in the remaining paths the first node is colored and no other nodes are colored.

Lemma 49. *There is some $\varepsilon > 0$ such that if a real partial run ρ_{r} admits a factorisation that is at distance at most ε from λ, then*

1. *Every bounded counter in $\mathit{larcut}(q)$ is cut in every path in ρ_{r} that contains at least three colored nodes.*

2. *There is some k such that for every bounded counter $c \notin larcut(q)$, the c-value is at most k for paths in ρ_{r} that begin in a colored node and contain no other colored nodes.*
3. *There is some k such that for every path π in ρ_{r} with color decomposition $\pi = \pi_0 \cdots \pi_n$, there are at most k paths among $\pi_0, \ldots, \pi_n$ which have nonzero c-value for some bounded counter $c \notin larcut(q)$.*

Proof. Note that each of the properties is open in the topological sense. The first property is open because it is MSO definable, while the remaining two properties are open because they are unions of MSO definable properties, ranging over possible values of k.

Below we show that each of the properties from the statement of the lemma is true in ρ and λ. Therefore, by openness, if ρ_{r} and λ_{r} are sufficiently close to ρ and λ, then they will also satisfy the three properties.

1. By definition of the automaton $\mathcal{R}_q$, every path in ρ that goes from a colored node to a colored node must visit state q at least once, and no bigger states. Therefore, a path that visits three colored nodes must contain a path which goes from state q to state q and does not visit bigger states. Such a path must cut all counters in $larcut(q)$.
2. By definition of the automaton $\mathcal{R}_q$, the cost function α has finite value on every λ-factor of ρ. It follows that in every λ-factor of ρ, call it σ, there is some finite number k_σ such that all paths that begin in the root of σ have c-value bounded by k_σ on every bounded counter $c \notin larcut(q)$. Since λ is regular, there are finitely many λ-factors, and therefore finitely many possible values of k_λ, so we can take k to be the maximal value of k_σ. The run ρ has finitely many λ-factors. Therefore, property 2 holds for ρ and λ, giving some bound k. For a fixed value of k, property 2 is MSO definable. Therefore, if ε is small enough with respect to k, then property 2 also holds for ρ_{r} and λ_{r}.
3. Let k be such that λ is k-regular. Using a pumping argument, one shows that every real run with a k-regular real coloring satisfies the following property:

   (*) if item 3 fails for $k+1$, then some path in the run increments counter c infinitely often without ever resetting or cutting it.

   Since the implication (*) is MSO definable, it must also be true in ρ and λ. Since the conclusion of the implication (*) is false in ρ, it follows that item 3 must hold in ρ and λ.

□

Lemma 50. *Let ε, ρ_{r} and λ_{r} be as in Lemma 49, and suppose also that:*

1. *every factor of ρ_{r} satisfies the boundedness acceptance condition.*
2. *for every $n \in \mathbb{N}$, there are finitely many subtrees of ρ_{r} rooted in colored nodes at color depth n.*

Then ρ_{r} satisfies the boundedness acceptance condition.

Proof. We need to show that for every node x in ρ_{r} and every bounded counter c, the c-value is bounded for paths that begin in x. We can restrict attention to paths that do not cut counter c. Let π be a path that begins in x and does not cut counter c. Let the color decomposition of π be $\pi_0 \cdots \pi_n$. The path π_0 is entirely contained in a single factor, and therefore its c-value is bounded by a function of x by the assumption that every factor satisfies the boundedness acceptance condition. For the paths $\pi_1, \ldots, \pi_n$, consider two cases, depending on whether c belongs to $larcut(q)$.

– Let $c \in larcut(q)$. By item 1 of Lemma 49, the path π can pass through at most three colored nodes and therefore there are at most three paths $\pi_1, \ldots, \pi_n$. By assumption 2, there are finitely many subtrees of ρ_{r} that are rooted in colored nodes visited by such paths. Therefore, by assumption 1, each of the paths $\pi_1, \ldots, \pi_n$ has bounded c-value.
– Let $c \notin larcut(q)$. By item 3 of Lemma 49, at most k paths among $\pi_0, \ldots, \pi_n$ have nonzero c-value. By item 2 of Lemma 49, each of the paths in $\pi_0, \ldots, \pi_n$ has bounded c-value.

□

The real run ρ_{r} with factorisation λ_{r}, as defined at the beginning of the proof of Lemmma 47, satisfy the assumptions of Lemma 50. Therefore, the run ρ_{r} satisfies the boundedness acceptance condition.

References

1. David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata. the weak monadic theory of the tree, and its complexity. In *ICALP*, pages 275–283, 1986.