Weak MSO+U over infinite trees[*]

Mikołaj Bojańczyk[1] and Szymon Toruńczyk[2]

1 University of Warsaw
2 INRIA and ENS Cachan

Abstract

We prove that, over infinite trees, satisfiability is decidable for Weak Monadic Second-Order Logic extended by the unbounding quantifier U. We develop an automaton model, prove that it is effectively equivalent to the logic, and that the automaton model has decidable emptiness.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic

Keywords and phrases Infinite trees, distance automata, MSO+U, profinite words

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

Introduction. The general topic of this paper is monadic second-order logic extended with the unbounding quantifier. The unbounding quantifier is a kind of set quantifier, which says that a formula $\varphi(X)$ holds for arbitrarily large finite sets X:

$$\mathsf{U}X\varphi(X) \qquad \stackrel{\text{def}}{=} \qquad \bigwedge_{n\in\mathbb{N}} \exists X\ n \leq |X| < \infty\ \wedge\ \varphi(X).$$

The unbounding quantifier was introduced in [1], along with some rudimentary decidability results. The quantifier is part of a research program, which investigates the notion of "regular language" for infinite words and trees. The general theme of the research program is that some features, such as the unbounding quantifier, can be added to monadic second-order logic over infinite objects, while preserving properties one would expect from a regular language. For instance, consider a language L of infinite words. Define a Myhill-Nerode-like equivalence relation $\sim_L$ on finite words:

$$w \sim_L w' \quad \text{if for every finite word } u \text{ and every infinite word } v, \quad uwv \in L \iff uw'v \in L.$$

One can show that if L is defined in monadic second-order logic with the unbounding quantifier (MSO+U), then $\sim_L$ has finitely many equivalence classes. Furthermore, each equivalence class is a regular language of finite words. The research program is discussed in [3].

The expressive power of the logic MSO+U is still not properly understood. It is an open problem whether satisfiability is decidable over infinite words.

So far, research has dealt with fragments of the logic. The paper [4] introduces two classes of automata on infinite words, called ωB- and ωS-automata, and proves that they correspond to fragments of MSO+U with restricted quantifier use. It is not clear if there can be an automaton model for the whole logic MSO+U, as opposed to an automaton model for fragments of the logic. These doubts are based on the paper [11], which proves that MSO+U can define non-Borel languages of infinite words. This implies that there can be no nondeterministic automaton model for MSO+U that has a Borel acceptance condition, which excludes all known nondeterministic automata models that use counters. One has to

* This research has been partially supported by the Polish MNiSW grant N N206 567840.

keep in mind that the non-Borel result still leaves room for automata; a distant analogy is that parity automata on infinite trees recognize non-Borel sets.

The topological problems described above disappear when one considers *weak* monadic second-order logic (WMSO), where set quantifiers are restricted to finite sets. In countable structures, such as infinite words or trees, formulas of WMSO, even extended with the unbounding quantifier, can only define Borel languages. Over infinite words, and without the unbounding quantifier, WMSO has the same expressive power as MSO, thanks to the McNaughton/Safra determinization theorem. This coincidence fails when the unbounding quantifier is introduced: WMSO+U is strictly less powerful than MSO+U. The crucial advantage of the weak logic is that it supports the classical automaton-logic connection: it admits an automaton model, the max-automaton from [2]. The automaton-logic connection also works for other extensions of WMSO on infinite words, see [5]. The topological complexity of WMSO+U has been studied in [6].

Content of this paper. The goal of this paper is the following theorem:

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

We prove the theorem in three steps.

1. In Section 1, we define a new automaton model for infinite trees, called a nested limsup automaton, which has the same expressive power as WMSO+U, and show effective translations from the logic to the automaton and back again.
2. In Section 2, we define a second new automaton model for infinite trees, called a puzzle, which is more expressive than a nested limsup automaton, and show an effective translation from nested limsup automata to puzzles.
3. In Section 3, we provide a decision procedure for nonemptiness of puzzles.

The proof, especially step 3, is maybe more interesting than the result itself. The general theme is to extend concepts of automata theory from finite sets to infinite sets equipped with compact metric topologies.

Related work. The automata models studied in this paper work on infinite objects. Two of the models, namely the ωB- and ωS-automata from [4], have natural counterparts working on finite words, called B- and S-automata. These finite word counterparts have recently seen a lot of interest. Instead of defining boolean-valued functions, which accept or rejects words, B- and S-automata define number-valued functions, which map words to numbers. These number-valued functions on finite words have been studied in depth by Colcombet in [8], under the name of regular cost functions. The theory of regular cost functions looks very promising, see [10] and [9] for some developments.

On a technical level, this paper uses profinite words [12] to model the limit behavior of finite words. This approach has been successfully applied in [13], as an alternative for cost functions in the study of the limitedness problem – B- or S-automata do define boolean-valued functions, which accept or reject profinite words.

Acknowledgement. We are grateful to the anonymous reviewer for his detailed remarks.

1 WMSO+U and Nested Limsup Automata

In Section 1 and 2, when talking about a tree over an alphabet A, we mean a full infinite binary tree with nodes labeled by A. (In Section 3, we switch to edge-labeled graphs.)

WMSO+U. A tree is interpreted as a logical structure, with unary predicates for labels, and two binary predicates for left and right successors. To express properties of this logical structure, we use weak monadic second-order logic, which means that formulas can quantify over nodes and *finite* sets of nodes. We use the convention where first-order variables are denoted x, y, z and set variables are denoted X, Y, Z. Also, we allow the unbounding quantifier U defined in the introduction.

▶ Running example. Consider an alphabet $A = \{a, b, c\}$. Define a *b-factor* in a tree t to be a connected set of nodes with label b. Being a b-factor is definable in WMSO:

$$\mathrm{bfactor}(X) \overset{\mathrm{def}}{=} \exists x\ x \in X \wedge \big(\forall z\ z \in X \implies \big(x \leq z\ \wedge\ \forall y\ x \leq y \leq z \implies (y \in X \wedge b(y))\big)\big).$$

Here, $\leq$ denotes the ancestor relation – the transitive reflexive closure of the parent relation – which is definable in WMSO. The running example in this paper is the tree language over A, call it L, which contains a tree if and only if the root has label a, and for every node x:

(a) If x has label a, then its subtree has b-factors of unbounded size.
(b) If x has label b or c, then in its subtree, the size of b-factors is bounded.

The language L is defined by the following formula of WMSO+U

$$(\exists x\ \forall y\ (x \leq y)\ \wedge\ a(x))\quad \wedge\quad \big(\forall x\ a(x)\ \iff\ \mathsf{U}X\big((\mathrm{bfactor}(X) \wedge \forall y\ (y \in X \Rightarrow y \geq x)\big)\big).$$

Let L be the set of trees satisfying the property above. What does a tree $t \in L$ look like? Observe first that every b-factor has to be finite, since an infinite b-factor contains finite b-factors of unbounded size, violating condition (b). Also, a node with label b or c cannot have a descendant with label a. This is because a tree with b-factors of bounded size cannot have a subtree with b-factors of unbounded size. It follows that all the nodes with label a form a connected set, call it X, which contains the root. There must be b-factors of unbounded size below every node from X, however every such b-factor must be finite, and have bounded size b-factors in its subtree. It follows that every node from X has at least one child in X, and the size of b-factors with parents in X is unbounded. An example is depicted in Figure 1. In the figure, we distinguish the maximal b-factors and call them $F_1, F_2, \ldots$, because they will get a lot of attention in the later analysis. The language L contains no regular tree, because in a regular tree either b-factors have bounded size, or some b-factor is infinite. In particular, L is not a regular language of infinite trees. Observe that in Figure 1, the only part of the tree that behaves in a non-regular way is the b-factors $F_1, F_2, \ldots$. ◀

1.1 Nested Limsup Automata

In this section, we define an automaton model which, over infinite trees, has the same expressive power as WMSO+U. The automaton is obtained by nesting two types of automata: prefix automata and limsup automata. We begin by defining prefix automata and limsup automata, then we show how they are nested.

Prefix automata. A prefix automaton is used to test regular properties of a finite prefix of a tree. Typical languages recognized by this kind of automaton are reachability properties "some node has label a", or "there is an antichain with five labels a". A *prefix* of a tree is an ancestor-closed set of nodes in the tree. A *prefix automaton* is given by the following ingredients:

- An *input alphabet* A.
- A finite set of *states* Q, together with an initial state $q_I \in Q$.

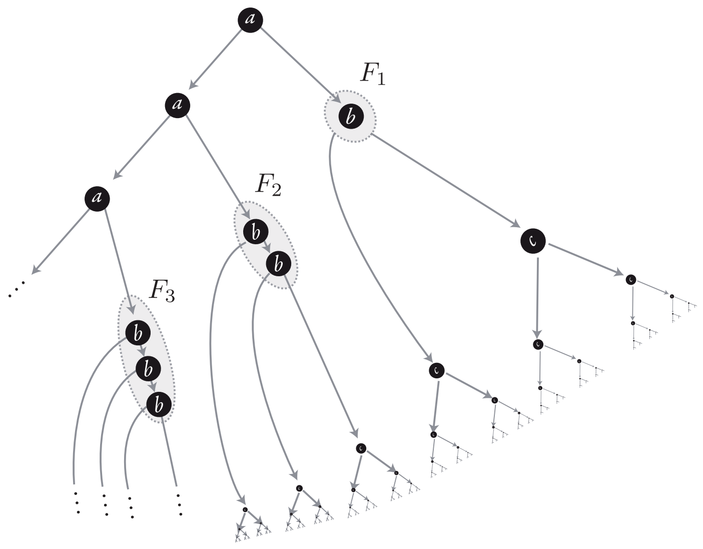


Figure 1 A tree $t \in L$. Every c node has only c descendants.

- A *(nondeterministic) transition relation*

$$\delta \subseteq Q \times A \times Q \times Q.$$

- A set of *accepting states* $F \subseteq Q$

The automaton accepts an infinite tree if there is a finite prefix $X \subseteq \{0,1\}^*$ and a run $\rho : X \to Q$, such that ρ respects the transition relation, has the initial state in the root, and all maximal nodes of X have labels in the accepting set F.

A prefix automaton has an existential nature: it tests if there exists a finite prefix with a certain (regular) property. In particular, languages recognized by prefix automata are open sets, under the usual topology over infinite trees.

Atomic limsup automata. We now define a second kind of automaton, called an atomic limsup automaton. A typical language recognized by this kind of automaton is "for every $n \in \mathbb{N}$, there is some path in the tree with at least n labels a". Observe that this typical language is not the same as "there is some path with infinitely many labels a".

The general idea is that the automaton has a counter, which stores natural numbers. The transition function chooses states in a top-down deterministic fashion. The transition function also induces a labeling of edges in the tree by sequences of counter operations. There are two counter operations: increment (written *inc*) and reset (written *reset*). Unlike the model for WMSO+U on infinite words defined in [2], there is no max operation here. The automaton accepts an input tree if the counter has unbounded values, ranging over nodes in the tree. We give a formal definition below.

An *atomic limsup automaton* is given by the following ingredients:

- An *input alphabet* A.
- A finite set of *states* Q, together with an initial state $q_I \in Q$.
- A *(top-down deterministic) transition function*

$$\delta : Q \times A \to (\{inc, reset\}^* \times Q)^2.$$

Let t be a tree over the input alphabet A. Using the deterministic transition function δ and the initial state in the root, one labels in a unique way the nodes of t by states and the edges of t by sequences in $\{inc, reset\}^*$. Suppose that the counter has value 0 in the root. For any finite path π in t, by reading the operations along the path, we get a counter value. The automaton accepts the tree t if the counter value is unbounded, when ranging over all finite paths in the tree. In other words, the automaton accepts if there are arbitrarily long sequences of increments that are not interrupted by reset.

Nested limsup automata. We now combine the two automata above into a single model, by using nesting. We define nested limsup automata by induction on the *nesting depth*. A nested limsup automaton of nesting depth 1 is either a prefix automaton, or an atomic limsup automaton.

An automaton of nesting depth $k+1$ is defined as follows. Suppose that $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are nested limsup automata of nesting depth k, over a common input alphabet A. Let $\mathcal{B}$ be either a prefix automaton, or an atomic limsup automaton, with input alphabet $\{0,1\}^n$. Then the expression $\mathcal{B}[\mathcal{A}_1, \ldots, \mathcal{A}_n]$ defines a nested limsup automaton. This new automaton has nesting depth $k+1$ and input alphabet A. When does it accept a tree t? Consider the tree $\hat{t}$ over alphabet $\{0,1\}^n$, where the label of a node x is a bit-vector, which has 1 on coordinate $i \in \{1, \ldots, n\}$ if and only if $\mathcal{A}_i$ accepts the subtree of t rooted in x. The automaton $\mathcal{B}[\mathcal{A}_1, \ldots, \mathcal{A}_n]$ accepts t if and only if the automaton $\mathcal{B}$ accepts the tree $\hat{t}$.

Observe that nested limsup automata are closed under complementation – the complement of $\mathcal{A}$ is recognized by an automaton $\mathcal{B}[\mathcal{A}]$, where $\mathcal{B}$ is a prefix automaton checking for 0 at the root.

Like all nested models of automata, nested limsup automata are something of a hybrid, sitting between logical formulas and automata.

▶ Running example. We now present a nested limsup automaton which recognizes the complement of the language L from the running example. Consider first an auxiliary automaton $\mathcal{B}$, a limsup automaton, which increments its counter whenever it sees a b, and resets it whenever it sees a or c. Since a large b-factor must contain a long path, the automaton $\mathcal{B}$ accepts a tree if and only if the tree has b-factors of unbounded size. A tree belongs to the complement of L if and only if the root is not labeled by an a, or if there is some node x, such that:

- The label of x is a, and $\mathcal{B}$ rejects the subree of x; or
- The label of x is either b or c, and $\mathcal{B}$ accepts the subtree of x.

Therefore, the complement of L is recognized by a limsup automaton nested inside a prefix automaton. ◀

1.2 Equivalence

The model of nested limsup automata is designed to be equivalent to WMSO+U, as stated in the following theorem.

▶ **Theorem 2.** *A language of infinite trees is definable in* WMSO+U *if and only if it is recognized by a nested limsup automaton. Translations both ways are effective.*

The proof of this theorem is in part I of the appendix. The proof ideas are based on [2].

Recall that our goal in this paper is to decide satisfiability of WMSO+U. The above theorem reduces the satisfiability problem of WMSO+U to the emptiness problem for nested limsup automata. However, due to the nesting operation, nested limsup automata are still too difficult to solve for emptiness. That is why, in the next section, we present a further reduction, which removes the nesting in a nested limsup automaton.

2 Puzzles

We now turn to the second automaton model in this paper, which is called a puzzle. The name is silly because we do not expect this model to be relevant outside this paper.

2.1 Puzzles, a denested version of nested limsup automata.

The ingredients of a *puzzle* are:

- a finite set Q of *states*
- a finite set C of *counters*
- an input alphabet A
- an *initial state* $q_I \in Q$
- a (nondeterministic) *transition relation*

$$\delta \subseteq Q \times A \times (\{\mathit{inc}, \mathit{reset}, \mathit{cut}\} \times C)^* \times Q)^2$$

- an *unbounding acceptance condition* $q \in Q \mapsto \mathsf{U}_q \subseteq C$, which maps each state q to the set of counters that are called *unbounded in* q.
- a *parity acceptance condition* $q \in Q \mapsto \Omega_q \in \mathbb{N}$, which maps each state to a natural number, called its *parity rank*.

Given an input tree t over the input alphabet, a *run* of the puzzle is an infinite binary tree where the nodes are labeled by states, the root has the initial state, and the edges are labeled by $(\{\mathit{inc}, \mathit{reset}, \mathit{cut}\} \times C)^*$, in a way consistent with the transition relation of the puzzle.

Observe that there is a new counter operation, called *cut*. The idea is that in the acceptance condition, the lim sup operation is only calculated along paths without cut. More formally, for a sequence of counter operations

$$\sigma \in (\{\mathit{inc}, \mathit{reset}, \mathit{cut}\} \times C)^*$$

we define the value of σ on counter c, denoted by $\mathrm{val}(\sigma, c)$, to be the maximal number n, such that some prefix of σ without a cut on counter c has n increments on counter c that are not interrupted by a reset on counter c. For example,

$$\mathrm{val}(\sigma, c) = 2 \qquad \text{for } \sigma = \mathit{inc}(c)\mathit{cut}(d)\mathit{inc}(c)\mathit{cut}(c)\mathit{inc}(d)\mathit{inc}(c)\mathit{inc}(c)\mathit{inc}(c)\mathit{reset}(c).$$

even though there are 3 consecutive increments on c after the cut on c. For a finite path π in a run ρ, we define

$$\mathrm{val}(\rho, \pi, c) \stackrel{\text{def}}{=} \mathrm{val}(\sigma, c) \qquad \text{where } \sigma \text{ is the sequence of edge labels on } \pi.$$

Finally, for a node x in a run ρ, we define

$$\mathrm{val}(\rho, x, c) \stackrel{\text{def}}{=} \sup\{\mathrm{val}(\rho, \pi, c) : \pi \text{ is a finite path originating in } x\ \} \in \mathbb{N} \cup \{\infty\}.$$

A run ρ is *accepting* if on every path, the parity acceptance condition is satisfied, and

$$\mathsf{U}_q = \{c \in C : \mathrm{val}(\rho, x, c) = \infty\} \qquad \text{for every node } x \text{ with state } q. \tag{1}$$

The key differences between a puzzle and a nested limsup automaton are:

- The set of bounded counters is tested in every subtree, as defined in (1);
- The model is not nested, but nondeterministic;
- There is the new *cut* counter operation.

▶ Running example. We define a puzzle which recognizes the language L from the running example (for simplicity, we ignore the condition on the root label). The states Q are q_a, q_b and q_c. There is one counter, call it d. State q_b increments the counter, which corresponds to counting the size of a path in a b-block, while the other states q_a and q_c reset the counter. This behavior is captured by the following set of transitions:

$$\{(q_\sigma, \sigma, (reset(d), q_0), (reset(d), q_1)) : \sigma \in \{a, c\}, q_0, q_1 \in Q\} \cup \\ \{(q_b, b, (inc(d), q_0), (inc(d), q_1)) : q_0, q_1 \in Q\}.$$

In this particular puzzle, the parity acceptance condition plays no role, and all states have accepting parity rank 0. Also, this puzzle does not use the *cut* operation. The key role is played by the unbounding acceptance condition, which is defined by

$$\mathsf{U}_{q_a} = \{d\} \qquad \mathsf{U}_{q_b} = \emptyset \qquad \mathsf{U}_{q_c} = \emptyset.$$

In other words, any node with state q_a in an accepting run must have unbounded values of the counter in its subtree, and every other node must have bounded values of the counter in its subtree. ◀

▶ **Theorem 3.** *For every nested limsup automaton one can compute a puzzle that recognizes the same language.*

The proof of this theorem is in part II of the appendix. The theorem can be interpreted as trading nesting for nondeterminism. From the point of view of deciding emptiness, this is a good trade: nesting is cumbersome for an emptiness algorithm, while nondeterminism is irrelevant.

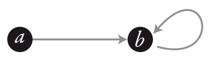
The converse of Theorem 3 fails: thanks to the parity condition, puzzles recognize non-Borel tree languages, while WMSO+U defines only Borel tree languages. Another reason is shown in the appendix: languages recognized by puzzles are not closed under complements.

3 Emptiness for puzzles

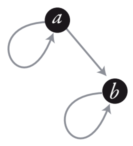
This section is about the emptiness procedure for puzzles.

▶ **Theorem 4.** *Emptiness is decidable for puzzles.*

The general idea is that even though an accepting run of a puzzle is an infinite object, there should be some way of drawing it in a finite way. This idea works for Büchi automata, because every nonempty Büchi automaton accepts the unfolding of a lasso graph such as:



This idea also works for parity tree automata, because every nonempty parity tree automaton accepts the unfolding of some finite graphs, such as:



Runs as graphs. In the proof of Theorem 4, we will treat a run ρ of a puzzle as an edge-labeled graph G_ρ. The graph G_ρ has the same nodes as ρ. It has no labels on the nodes. An edge in the graph is labeled by the word

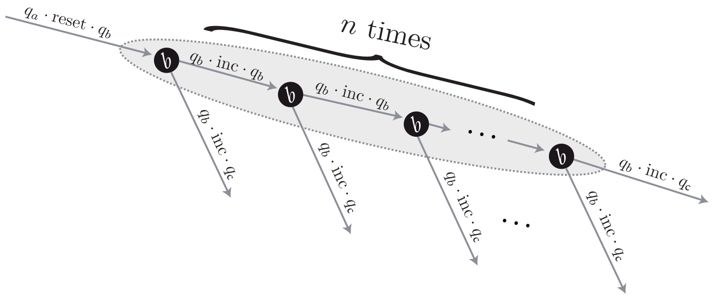
$$\sigma \in Q \cdot (\{inc, reset, cut\} \times C)^* \cdot Q,$$

which begins with the state in the source node of the edge, followed by the sequence of counter operations on the edge, and ending with the state in target node of the edge. From now on, when writing ρ, we will refer to the graph G_ρ. The labels on the edges of G_ρ are words over the alphabet

$$\Lambda \overset{\text{def}}{=} Q \cup \{inc, reset, cut\} \times C.$$

We fix this alphabet for the rest of this section.

▶ Running example. Recall the tree t from Figure 1. The puzzle in the running example has only one run over any tree, and over t the run is accepting. The part of this run that concerns that b-factor F_n is illustrated in Figure 2. In the rest of this section, we will try to define a limit F_ω. ◀


$q_a \cdot \text{reset} \cdot q_b$
n times
$q_b \cdot \text{inc} \cdot q_b$
$q_b \cdot \text{inc} \cdot q_b$
$q_b \cdot \text{inc} \cdot q_c$
$q_b \cdot \text{inc} \cdot q_c$
$q_b \cdot \text{inc} \cdot q_c$
$q_b \cdot \text{inc} \cdot q_c$
$q_b \cdot \text{inc} \cdot q_c$
b
...

■ **Figure 2** The run of the puzzle inside the b-factor F_n of the tree t from Figure 1.

Factor. A *factor* in a tree is a connected set of nodes. Every factor has a root node, which is the least node in the factor. A *port* in a factor is a node outside the factor that has its parent in the factor. A *root-to-port path* in a factor is a path from the root to some port, seen as a sequence of edges.

Signature. The signature of a (finite or infinite) path in a run is the concatenation of all the labels on that path, which is a word over the alphabet Λ. We use the letters σ or τ to denote signatures of paths.

▶ Running example. In Figure 2, the signature of the rightmost root-to-port path in F_n is

$$\sigma_n \overset{\text{def}}{=} (q_b \cdot inc \cdot q_b)^{n-1} \cdot (q_b \cdot inc \cdot q_c).$$ ◀

For signatures of factors, we use multisets, which are sets where the number of occurrences an element can be in $\mathbb{N} \cup \{\infty\}$. Consider a *finite* factor F. The *signature* of the factor is the multiset of path signatures, ranging over root-to-port paths in the factor. All path signatures in this multiset have the same source state, namely the state in the root of the factor. This state is called the root state of a factor signature. It is important that factor signatures describe finite factors. In an infinite factor, it may be the case that a path is not included in root-to-port paths. We use the letter Σ to denote factor signatures.

▶ Running example. The signature of the factor in Figure 2 is the multiset, call it Σ_n, which contains all path signatures $\sigma_1, \ldots, \sigma_{n-1}$ once, and the path signature σ_n twice. ◀

3.1 Limits of signatures

The key technique in this paper is to use limits. We are mainly interested in the limits of signatures, both signatures of paths, and signatures of factors. In this section, we establish the notion of limit that we use. Our approach to limits of path signatures is to treat path signatures as a special case of profinite words. Our approach to limits of factor signatures is to use a variant of Hausdorff distance on multisets of profinite words. The definitions follow.

Profinite words. Consider the following distance on finite words over the alphabet Λ.

$$\text{distance}(\sigma, \tau) = \max\{\frac{1}{2^n} : \text{ some DFA of } n \text{ states accepts } \sigma \text{ but not } \tau\}.$$

It is not difficult to see that this is indeed a distance, even an ultrametric:

$$\text{distance}(\sigma_1, \sigma_2) \leq \max(\text{distance}(\sigma_1, \tau), \text{distance}(\tau, \sigma_2)) \qquad \text{for every } \sigma_1, \sigma_2, \tau \in \Lambda^*.$$

A sequence of words $(\tau_n)_n$ is called *Cauchy* if for every $\varepsilon > 0$ there is some n such that all the words $\tau_n, \tau_{n+1}, \ldots$ lie in a ball of diameter ε.

▶ Running example. Recall the sequence $(\sigma_n)_n$ of signatures of rightmost paths in the factors F_n. This sequence is not Cauchy, because even-numbered words have an even number of increments, and odd-numbered words do not, and evenness can be tested by a DFA of 2 states. However, the sequence $(\sigma_n)_n$ has several Cauchy subsequences, including the sequences $(\sigma_{n!})_n$ and $(\sigma_{n!+1})_n$. ◀

Consider two Cauchy sequences $(\sigma_n)_n$ and $(\tau_n)_n$ to be *equivalent* if

$$\sigma_1, \tau_1, \sigma_2, \tau_2, \ldots$$

is also a Cauchy sequence. This is an equivalence relation, call it $\sim$. An equivalence class of this relation is called a *profinite word* (see [12] for more on profinite words). The set of profinite words is denoted by $\widehat{\Lambda^*}$. We model signatures of paths and their limits by profinite words. Here are the key properties of profinite words that we use:

1. It makes sense to say that a profinite word belongs or does not belong to a regular language $L \subseteq \Lambda^*$. Indeed, if (σ_n) is a Cauchy sequence, then either all but finitely many elements belong to L, or all but finitely many elements do not belong to L. Therefore, it makes sense to say that a Cauchy sequence belongs or does not belong to a regular language. Also this property is preserved when going to an equivalent Cauchy sequence. In particular, it makes sense to say that a profinite word does at least one increment on some counter c, or does at least 4 increments, or begins with state q, because all of these are regular properties.
2. There is a distance on profinite words, namely:

$$\text{distance}((\sigma_n)_n, (\tau_n)_n) = \lim_{n\to\infty} \text{distance}(\sigma_n, \tau_n).$$

   By the triangle inequality, the above limit exists for Cauchy sequences and does not depend on the choice of a sequence in a class of $\sim$. Equipped with this distance, the set of profinite words is a compact metric space. This means that every sequence has a converging subsequence. Also, for every regular language $L \subseteq \Lambda^*$, there is some distance ε such that any two words at distance at most ε either both belong to L, or both do not belong to L.

3. It makes sense to concatenate profinite words. This is because the relation $\sim$ is a congruence with respect to concatenation of sequences:

$$(\sigma_n)_n \sim (\sigma'_n)_n \quad \text{and} \quad (\tau_n)_n \sim (\tau'_n)_n, \qquad \text{implies} \qquad (\sigma_n \cdot \tau_n)_n \sim (\sigma'_n \cdot \tau'_n)_n.$$

▶ Running example. Recall the Cauchy sequence $(\sigma_{n!})_n$. We write σ_ω for the profinite word represented by this sequence. This profinite word begins with letter q_b and ends with letter q_c. Also, for every n, there are more than n increments in σ_ω, which is something that can only happen in a profinite word. ◀

Hausdorff distance on sets. So far, we have defined a compact metric space to model path signatures, namely the set of profinite words over Λ. We now want to do the same thing for multisets of path signatures. Our approach is to use a multiset variant of the Hausdorff distance on sets. We begin by recalling the distance on sets (not multisets), because this definition is easier to digest. A metric on a set A (we are interested in the case when A is the set of profinite words over Λ) can be lifted to a metric on closed subsets of A, using the Hausdorff distance. For two closed subsets $X, Y \subseteq A$, their Hausdorff distance is defined by

$$\text{distance}(X,Y) \stackrel{\text{def}}{=} \qquad \max\big(\sup_{x\in X}\inf_{y\in Y}\text{distance}(x,y), \sup_{y\in Y}\inf_{x\in X}\text{distance}(x,y)\big).$$

This is a metric on closed subsets. This definition can be extended to so-called *closed multisets*. The precise definition of closed multisets and their metric is given in the appendix. As an example, consider multisets of real numbers. Like any finite multiset, the multiset

$$X_n = \{\frac{1}{n}, \frac{1}{n^2}, \ldots, \frac{1}{n^n}\}$$

is closed. The sequence $(X_n)_n$ tends to the (closed) multiset where 0 appears infinitely often.

▶ Running example. Consider the signature Σ_n of the factor F_n. One can prove that the sequence $(\Sigma_{n!})_n$ is Cauchy. Its limit is the multiset, call it Σ_ω, where every σ_n appears once, and every limit of a subsequence of (σ_n) appears infinitely often. Among others, for every $k \in \mathbb{N}$, $\sigma_{\omega+k}$ appears infinitely often. ◀

3.2 Signature graphs

We are now ready to define the key concept of this paper, which is a signature graph. A signature graph is used to represent limits of accepting runs. A signature graph is going to have labeled parallel edges, so it is really a multigraph. When talking about an *edge labeled multigraph*, we mean a directed graph with edges labeled by some alphabet A. The edges form a multiset, so for any label σ and pair of vertices x, y, the number of edges from x to y with label σ may potentially be $0, 1, \ldots,$ or countably infinite.

Definition of a signature graph. A *path signature* is a profinite word over Λ that begins with a state (called the *source* state) and ends with a state (called the *target* state). A *factor signature* is a closed multiset of path signatures, which agree on the source state. A *signature graph* is a multigraph with edges labeled by path signatures, subject to the following consistency condition. For every node x, there is some state $q \in Q$, such that all edges entering x have target state q, and all edges leaving x have source state q. This state q is called the *node label* of x, although technically speaking a signature graph supplies only edge labels, and the node labels are derived information.

In a signature graph, the labeling assigns signature paths to individual edges. However, using the monoid structure of path signatures, we can assign a path signature to every finite edge path, by concatenating the labels of the edges in the path.

Fans. Suppose that x is a node in an edge labeled graph. We define the *fan* of x to be the multiset of labels of edges leaving x. If $\mathscr{P}$ is a family of multisets over A, then we say that a graph has *fans in* $\mathscr{P}$ if the fan of each node is in $\mathscr{P}$. In the proof, when dealing with signature graphs, we are interested in two sets $\mathscr{P}$ of factor signatures. The first set, call it

$$\mathscr{P}_{\text{fin}}$$

is the set of factor signatures of finite factors that appear in some run of the puzzle, not necessarily accepting. This set depends on the transitions and counter in the puzzle. It does not depend, however, on the acceptance conditions (boundedness and parity) in the puzzle, because these are only used to distinguish accepting runs. The second set is the closure of the first set, under the Hausdorff distance, in the space of closed multisets of profinite words:

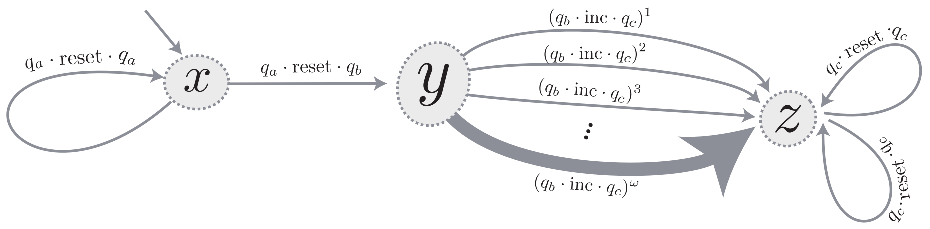
$$\overline{\mathscr{P}_{\text{fin}}}.$$

Limit accepting signature graph. Recall that thanks to the properties of the profinite monoid, it makes sense to say that a path signature does an increment/cut/reset on some counter. We say that a path signature has *value ω on counter c* if for every $n \in \mathbb{N}$, the path signature has value at least n on counter c (recall that the value refers to the maximal number of increments, not interrupted by a reset, before the cut). Also, one can ask about the maximum rank, in the parity acceptance condition, of states visited by the limit path signature. For a node x in a signature graph G, define $\mathsf{U}(G,x)$ to be the set of counters U_q, where q is the state in the label of x.

We now present the key definition in the emptiness procedure for puzzles, the definition of a limit accepting signature graph. The idea is that a limit accepting signature graph is the limit of a converging sequence of accepting runs.

A signature graph G with a distinguished *root node* is called *limit accepting* if

1. The root node is labeled by the initial state of the automaton,
2. Every node in G is reachable from the root node,
3. The parity condition is satisfied on every infinite path,
4. For every node x and counter c, counter c belongs to $\mathsf{U}(G,x)$ if and only if
   a. There is an infinite path from x, such that every prefix of the path can be extended to a finite path that does not cut c, and reaches a node whose fan contains an edge with ω on counter c; or
   b. There is an infinite path from x, which does not cut c, resets it finitely often, and increments it infinitely often.



$q_a \cdot \text{reset} \cdot q_a$
x
$q_a \cdot \text{reset} \cdot q_b$
y
$(q_b \cdot \text{inc} \cdot q_c)^1$
$(q_b \cdot \text{inc} \cdot q_c)^2$
$(q_b \cdot \text{inc} \cdot q_c)^3$
$\vdots$
$(q_b \cdot \text{inc} \cdot q_c)^\omega$
z
$q_c \cdot \text{reset} \cdot q_c$
$q_c \cdot \text{reset} \cdot q_c$

Figure 3 This signature graph represents the accepting run in Figure 2, in the following sense. Node x represents all nodes with label a. The self-loop in x stands for the lefmost path in the graph from Figure 2. Node y represents a limit of the factors F_n. The fan of y is Σ_ω – the limit of the fans $(\Sigma_{n!})_n$. The thick edge stands for infinitely many edges, including infinitely may with label $(q_b \cdot \mathit{inc} \cdot q_c)^\omega$. Node z together with its two self-loops stands for a subtree with infinitely many c's.

Main technical theorem. To prove Theorem 4, we present a stronger result, Theorem 5, which is the main technical contribution of this paper.

▶ **Theorem 5.** *The following conditions are equivalent.*

1. *There is a limit accepting signature graph with fans in* $\overline{\mathscr{P}_{\text{fin}}}$*,*
2. *There is a limit accepting signature graph with fans in* $\overline{\mathscr{P}_{\text{fin}}}$ *with finitely many nodes,*
3. *The puzzle has an accepting run.*

Furthermore, given a puzzle one can decide if the conditions hold.

▶ Running example. By Theorem 5, the puzzle from the running example should have a limit accepting signature graph with fans in $\overline{\mathscr{P}_{\text{fin}}}$, with finitely many nodes. Such a graph is illustrated in Figure 3, and has 3 nodes, but infinitely many edges. ◀

The proof of Theorem 5 is in part III of the appendix. A rough sketch is as follows.

- **Implication from 1 to 2**. The key point is that we can design an automaton model, closely resembling alternating automata on graphs, which recognizes limit accepting signature graphs. This automaton model shares the following property with alternating automata on graphs: a nonempty automaton accepts a graph with finitely many nodes.
- **Implication from 2 to 3.** The key point is to get rid of the limits, and replace them by actual finite pieces of runs. The idea is of course to use finite pieces of runs from a sequence approximating the limit, but the implementation of this idea requires some technical effort. We use a notion of bisimulation that is adapted to converging sequences.
- **Implication from 3 to 1.** The key point is to extract limits from an arbitrary accepting run of a puzzle. For this, we use a version of Ramsey's theorem adapted to metric spaces.
- **Decidability.** The key point is to compute a finite representation of the set $\overline{\mathscr{P}_{\text{fin}}}$. In the end, we reduce this to the domination problem for B-automata over finite trees [10].

We believe that the technique of limits of graphs is quite general, and can be applied to other automaton models for trees.

References

1 M. Bojańczyk. A bounding quantifier. In *CSL*, pages 41–55, 2004.

2 M. Bojańczyk. Weak MSO with the unbounding quantifier. In *STACS*, pages 159–170, 2009.

3 M. Bojańczyk. Beyond ω-regular languages. In *STACS*, pages 11–16, 2010.

4 M. Bojańczyk and T. Colcombet. Bounds in ω-regularity. In *LICS*, pages 285–296, 2006.

5 M. Bojańczyk and S. Toruńczyk. Deterministic automata and extensions of weak mso. In *FSTTCS*, pages 73–84, 2009.

6 J. Cabessa, J. Duparc, A. Facchini, and F. Murlak. The wadge hierarchy of max-regular languages. In *FSTTCS*, pages 121–132, 2009.

7 T. Colcombet. Factorisation forests for infinite words. In *FCT'07*, 2007.

8 T. Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP (2)*, pages 139–150, 2009.

9 T. Colcombet, D. Kuperberg, and S. Lombardy. Regular temporal cost functions. In *ICALP (2)*, pages 563–574, 2010.

10 T. Colcombet and C. Löding. Regular cost functions over finite trees. In *LICS*, pages 70–79, 2010.

11 S. Hummel, M. Skrzypczak, and S. Toruńczyk. On the topological complexity of MSO+U and related automata models. In *MFCS*, pages 429–440, 2010.

12 J-E. Pin. Profinite methods in automata theory. In *STACS*, pages 31–50, 2009.

13 S. Toruńczyk. *Languages of profinite words and the limitedness problem.* PhD thesis, Warsaw University, 2011.

Appendix Part I

Equivalence of Logic and Nested Limsup Automata

A Equivalence of logic and automata

In this part of the appendix, we prove Theorem 2, which says that a language of infinite trees is definable in Weak MSO with the unbounding quantifier if and only if it is recognized by a nested limsup automaton. Also, translations in both ways are effective.

As usual, the easier direction is to convert an automaton into a formula, as stated in the following lemma.

▶ **Lemma 6.** *Every language recognized by a nested limsup automaton is definable in* WMSO+U.

Proof. Since formulas of WMSO+U are closed under nesting, all we need to show is that both a prefix automaton and an atomic limsup automaton can be converted into a formula of WMSO+U. For prefix automata, this is immediate, since a prefix automaton asks for the existence of a *finite* prefix which is recognized by an automaton over *finite* trees and this is expressible in WMSO.

Now, let $\mathcal{A}$ be an atomic limsup automaton over the alphabet A. Let

$$o_1, \ldots, o_n \in \{inc, reset\}^*$$

be the sequences of counter operations that appear in the transition function of the automaton. Because the automaton is deterministic, for every $i \in \{1, \ldots, n\}$, one can write a formula $\varphi_i(x)$ of WMSO which holds in a node x if the edge connecting x to its parent is labeled by the sequence o_i in the accepting run. The automaton accepts a tree if there is no bound on sets of nodes X which are:

- linearly ordered, as expressed by

$$\forall x, y \in X \; x \leq y \vee y \leq x,$$

- contain only incrementing nodes, as expressed by

$$\forall x \in X \bigvee_{o_i \in inc^+} \varphi_i(x),$$

- and are not separated by resets, as expressed by

$$\forall x, y, z (x \in X \wedge z \in X \wedge x \leq y \leq z \Rightarrow \bigvee_{o_i \in inc^*} \varphi_i(y)).$$

◀

The rest of Part I of the appendix is devoted to converting a formula of WMSO+U into a nested limsup automaton. We need to show that nested limsup automata are closed under the operations which constitute WMSO+U. The proof is split into two steps.

The first step is to show that adding one quantifier U atop of a formula of WMSO (without the quantifier U) results in a language recognizable by a nested limsup automaton.

▶ **Proposition 7.** *Nested limsup automata recognize all languages of the form*

$$\mathsf{U}X \; \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \text{WMSO}.$$

This step relies on the specific properties of the U quantifier and uses the factorization theorem of Simon (or, more precisely, its enhanced version of Colcombet).

The second step is to prove an abstract property, for an abstract notion of quantifiers Q, which says that if a class of automata recognizes formulas of the form

$$\mathsf{Q}X\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \mathrm{WMSO},$$

then the class of automata recognizes all languages defined in WMSO $+\mathsf{Q}$, not only those where the quantifier Q is used once, and in the outermost position of the formula. This property holds for arbitrary nested automata and a wide range of quantifiers, but for simplicity, our proof will assume that $\mathsf{Q} = \mathsf{U}$.

▶ **Proposition 8.** *Let $\mathscr{C}$ be a class of automata which is closed under Boolean combinations, nesting and which recognizes all languages*

$$\mathsf{U}X\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \mathrm{WMSO}.$$

Then automata from $\mathscr{C}$ recognize all languages definable in WMSO+U.

The two propositions above give the translation of logic into automata, thus completing the proof of Theorem 2. Indeed, all of the assumptions of Proposition 8, are easily seen to hold for nested limsup automata, with the exception of the assumption provided by Proposition 7.

The proof of Proposition 8 is in Appendix B. The proof of Proposition 7 is in Appendix C.

B Proof of Proposition 8

Let $\mathscr{C}$ be a class of automata which satisfies the assumptions of Proposition 8, namely it is closed under Boolean combinations, nesting and recognizes all languages

$$\mathsf{U}X\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \mathrm{WMSO}. \tag{1}$$

We prove that the class $\mathscr{C}$ captures WMSO+U. Note that the assumption implies in particular that $\mathscr{C}$-automata cover WMSO, since any formula φ of WMSO can be written as $\mathsf{U}X.\varphi$, where φ does not even depend on X.

Sketch of proof The technique is standard – a given formula φ of WMSO+U partitions the set of all trees into a finite family of *φ-types* – a standard notion, which we recall later. Whether a tree t satisfies φ depends only on its φ-type. We prove by induction on the size of φ that there is an automaton in the class $\mathscr{C}$ recognizing the φ-type of a given tree.

The interesting case is when the formula φ is of the form

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

where ψ is any formula of WMSO+U. The idea is roughly as follows. Let t be an input tree for the formula φ. We use the inductive assumption and obtain a $\mathscr{C}$-transducer (defined below) $\mathcal{T}$ which labels the input tree t by the ψ-types of the subtrees of t at each node. Let t_ψ be the resulting tree. Note that ψ has one free variable more than the formula φ so in order to determine the ψ-type of a subtree of t, we need to specify also the valuation of the free variable X – we set $X = \emptyset$. Now, if X is any finite set of positions of t, then there is a formula γ of WMSO (without U) over trees labeled by ψ-types, such that

$$t, X \models \psi \qquad \textit{iff} \qquad t_\psi, X \models \gamma.$$

In particular,

$$t \models \mathsf{U}X\ \psi(X) \qquad \textit{iff} \qquad t_\psi \models \mathsf{U}X\ \gamma(X).$$

Finally, we use the assumption on the class $\mathscr{C}$ and conclude that the set of trees t_ψ which satisfy the right-hand side of the above equivalence is recognized by a nested limsup automaton $\mathcal{A}$. Then, nesting $\mathcal{A}$ on top of $\mathcal{T}$ yields a nested limsup automaton which accepts precisely the trees t which satisfy the left-hand side of the above equivalence.

We present a more detailed proof below.

$\mathscr{C}$-transducers By *$\mathscr{C}$-automaton* we mean an automaton from the class $\mathscr{C}$. A *$\mathscr{C}$-transducer* $\mathcal{T}$ is described by the following components.

- An *input* alphabet A
- An *output* alphabet B
- Underlying $\mathscr{C}$-automata, $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$
- A labeling function, $\lambda\colon \{0,1\}^k \to B$.

Let t be an input tree over the alphabet A. The *output* $\mathcal{T}(t)$ has label

$$\lambda(b_1, b_2, \ldots, b_k)$$

at position x, where $b_i = 1$ if and only if $\mathcal{A}_i$ accepts the subtree of t rooted at x. The definition is such that if $\mathcal{T}$ is a $\mathscr{C}$-transducer with input alphabet A and output alphabet B, and $\mathcal{A}$ is an $\mathscr{C}$-automaton with input alphabet B, then one can nest $\mathcal{A}$ on top of $\mathcal{T}$ and obtain a $\mathscr{C}$-automaton with input alphabet A.

Contexts and types Let A be a fixed (finite) alphabet. We define the standard notion of a *type*, or *φ-type*, of a tree with respect to a formula φ of WMSO+U. The formula φ might have several free variables – if it has n free variables, we treat it as a closed formula $\hat{\varphi}$ over an extended alphabet $A \times \{0,1\}^n$, where the additional coordinates serve for encoding the valuation of the free variables. This is a standard technique, which we describe now to fix the notation.

If $X_1, X_2, \ldots, X_n$ are sets of positions of the input tree t, then by

$$t \otimes X_1 \otimes \cdots \otimes X_n$$

we denote the tree t over the extended alphabet $A \times \{0,1\}^n$, whose label at position x is the tuple $(a, b_1, b_2, \ldots, b_n)$, where a is the label of x in t, and for $i = 1, \ldots, n$, the bit b_i is set to 1 if and only if $x \in X_i$. Such a tree will be called an *n-decorated tree.*

We then convert a formula φ with n free variables $X_1, \ldots, X_n$ to a o formula $\hat{\varphi}$ with no free variables, by replacing each predicate $x \in X_i$ by the unary predicate testing whether the label at position x has a 1 at the i-th position. Clearly,

$$t, X_1, \ldots, X_n \models \varphi \quad \textit{iff} \quad t \otimes X_1 \otimes \cdots \otimes X_n \models \hat{\varphi}.$$

An *input tree* for the formula $\hat{\varphi}$ is any n-decorated tree. Using the above encoding, we will often make the assumption that a given formula φ of WMSO+U is closed, and identify φ with $\hat{\varphi}$.

Let us fix a formula φ of WMSO+U. The *φ-type* of a tree t captures its behavior with respect to the formula φ, when t is plugged into different contexts. For our needs, a context is an input tree with one distinguished node, called the *port.* A context p can be applied to a tree t, by inserting t in place of port of p. The resulting tree is denoted $p \cdot t$.

If φ is a (closed) formula of WMSO+U, and t, t' are two input trees, then we write

$$t \equiv_\varphi t'$$

if for any context p,

$$p \cdot t \models \varphi \quad \iff \quad p \cdot t' \models \varphi.$$

It is easy to see that $\equiv_\varphi$ is an equivalence relation. Let type$_\varphi(t)$ denote the equivalence class of the input tree t under $\equiv_\varphi$ and let Types$_\varphi$ denote the set of equivalence classes of n-decorated trees. We denote the elements of Types$_\varphi$ by symbols τ, τ', and call them φ-types. For a φ-type τ, we write $\tau \models \varphi$ if $t \models \varphi$ for some (equivalently, all) $t \in \tau$.

Automata can compute types To prove Proposition 8, we will show the following.

▶ **Lemma 9.** *Let φ be a formula of* WMSO+U*.*

1. *Then there are finitely many φ-types.*
2. *Every φ-type τ, as a set of input trees, is recognized by a $\mathscr{C}$-automaton $\mathcal{A}_\tau$.*

The lemma implies Proposition 8, because the set of trees which satisfy the formula φ is precisely the finite union of all the φ-types that satisfy φ, and $\mathscr{C}$-automata are closed under finite unions. We prove both items of the lemma simultaneously, by induction on the structure of the formula φ. The base case is trivial, and so is the inductive step in the case of Boolean combinations, as the class $\mathscr{C}$ is closed under Boolean combinations.

The interesting case is when the formula φ is obtained from ψ by applying either of the quantifiers $\exists_{\mathsf{fin}}, \mathsf{U}$. In fact, both quantifiers can be treated in a unified way. In what follows, we will only deal with the quantifier U, but the argumentation for the quantifier $\exists_{\mathsf{fin}}$ is completely the same.

Let us assume that

$$\varphi = \mathsf{U}X.\ \psi(X). \tag{$\mathsf{U}\psi$}$$

We will show that if ψ has finitely many types, then φ also has finitely many types.

We show that, given an an input tree t for φ, only a finite information about t is sufficient to determine whether $p \cdot t \models \varphi$, if p is a given context. Therefore, this finite information determines the φ-type of t. More precisely, we have the following lemma.

▶ **Lemma 10.** *The validity of the formula*

$$\mathsf{U}X.((p \cdot t) \otimes X \models \psi), \tag{2}$$

depends only on the following information about t:

- *The ψ-types which can be obtained from decorating t by some set X_t:*

$$T_\exists \overset{\text{def}}{=} \{\tau \in \psi\text{-types}:\ \exists_{\mathsf{fin}} X_t.\, (\psi\text{-type}(t \otimes X_t) = \tau)\}$$

- *The ψ-types which can be obtained from decorating t by arbitrarily large sets X_t:*

$$T_\mathsf{U} \overset{\text{def}}{=} \{\tau \in \psi\text{-types}:\ \mathsf{U}X_t.\, (\psi\text{-type}(t \otimes X_t) = \tau)\}$$

Proof. The formula (2) says that the tree $p \cdot t$ can be decorated with arbitrarily large sets X for which $(p \cdot t) \otimes X$ satisfies ψ. Thanks to the assumption that there are finitely many ψ-types, this is equivalent to the disjunction of the two conditions:

(A) There exists a single ψ-type $\tau \in T_{\mathsf{U}}$ such that there exists a decoration $p \otimes X_p$ of the context p, for which

$$\big((p \otimes X_p) \cdot \tau\big) \models \psi,$$

(B) The context p can be decorated with arbitrarily large sets X_p, such that there exists a ψ-type $\tau \in T_{\exists}$ for which

$$\big((p \otimes X_p) \cdot \tau\big) \models \psi.$$

(Formally, in the formula $(p \otimes X_p) \cdot \tau$, the equivalence class τ should be replaced by any of its elements. The choice of the representative does not matter.)

It is now clear that both the above conditions depend only on the sets T_{U} and $T_{\exists}$. This ends the proof of the lemma. ◀

Therefore, if the set of ψ-types is finite, then also the set of φ-types is finite, as each φ-type is determined by the finite sets $T_{\exists}$ and T_{U}.

We now show that the sets $T_{\exists}$ and T_{U} can be computed by an automaton from the class $\mathscr{C}$. Let t_ψ be a labeling which associates with a node v of t the ψ-type of the subtree of $t \otimes \emptyset$ rooted at v. Thus, t_ψ can be viewed as a tree over the alphabet of ψ-types. By the inductive hypothesis, the labeling t_ψ can be carried out by a $\mathscr{C}$-transducer $\mathcal{T}$ working over the original input tree t.

Furthermore, for any finite set of positions X, the ψ-type of an input tree $t \otimes X$ can be computed by a bottom-up tree automaton working on any finite prefix of $t_\psi \otimes X$ which contains X. Hence, for any ψ-type τ, there exists a formula γ_τ of Weak MSO with one free variable X, working over the tree t_ψ, which determines whether $t \otimes X$ has ψ-type τ. Then, $\tau \in T_{\mathsf{U}}$ if and only if

$$t_\psi \models \mathsf{U}X.\gamma_\tau(X),$$

and $\tau \in T_{\exists_{\mathsf{fin}}}$ if and only if

$$t_\psi \models \exists_{\mathsf{fin}}X.\gamma_\tau(X).$$

We use the assumption on the class $\mathscr{C}$ stated in Proposition 8. Since γ_τ is a formula of Weak MSO, these formulas can be determined by $\mathscr{C}$-automata which work over the tree t_ψ. Nesting these automata with the $\mathscr{C}$-transducer $\mathcal{T}$ yields automata which work over the original input tree t. Therefore, the sets $T_{\exists}$ and T_{U} can be computed by a $\mathscr{C}$-automaton $\mathcal{A}$ working over the tree t. Hence, by Lemma 10, the φ-type of t can be computed by a $\mathscr{C}$-automaton.

This finishes the proof of Lemma 9, and thus of Proposition 8.

C Proof of Proposition 7

In this section, we prove Proposition 7, which says that nested limsup automata recognize all languages of the form

$$\mathsf{U}X.\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \text{WMSO}.$$

Fix a formula $\varphi(X)$ of WMSO. The rest of Section C is devoted to finding a nested limsup automaton recognizing the language $\mathsf{U}X\ \varphi(X)$. For a tree t, we use the name *witness for* t for any finite set X such that $t, X \models \varphi$. Our aim is to prove that the size of a maximal witness can be estimated by a nested limsup automaton. We make a preliminary reduction, which shows that we may (after suitably modifying the formula φ) restrict to witnesses which are chains. Then we will apply a reasoning which is very similar to the one used in [2], where infinite words where considered.

Reduction to chains Let us define the restriction of U to chains (i.e. sets of nodes X such that any two nodes are comparable with respect to the ancestor ordering), denoted $\hat{\mathsf{U}}$ via:

$$\hat{\mathsf{U}}X.\varphi(X) \qquad \iff \qquad \mathsf{U}X.\ (\varphi(X)\ \wedge\ \mathit{chain}(X)), \tag{3}$$

where $\mathit{chain}(X)$ is a first-order formula testing whether X is a chain.

▶ **Lemma 11.** *For any formula φ of* WMSO *there exists a formula ψ of* WMSO *such that*

$$\mathsf{U}X.\varphi(X) \quad \iff \quad \hat{\mathsf{U}}X.\psi(X).$$

This lemma implies that in the proof of Proposition 7 we may restrict to formulas using $\hat{\mathsf{U}}$, instead of formulas using U, and thus consider only chain witnesses.

Proof. Let X be a set of nodes. We say that a finite chain $Y = \{y_1 < \ldots < y_n\}$ is a *skeleton* of X if for every $y_i < y_{i+1}$ in Y, there is a node $x \in X$ with $y_i \le x$ and $y_{i+1} \not\le x$. The definition of a skeleton can be readily translated into a formula $\mathit{skeleton}(Y, X)$ of first-order logic. It is easy to see that for a set X of n nodes:

1. The size of any skeleton of X is bounded by n
2. X has a skeleton with at least $\log(n) - 1$ elements.

From the above, we conclude the following equivalence.

$$\mathsf{U}X.\varphi(X) \qquad \iff \qquad \hat{\mathsf{U}}Y.\ \exists_{\mathsf{fin}}X.\ \big(\mathit{skeleton}(Y,X) \wedge \varphi(X)\big).$$

◀

It follows that in the proof of Proposition 7 we only need to construct a nested limsup automaton which recognizes the language of the form

$$\hat{\mathsf{U}}X.\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \mathrm{WMSO}, \tag{4}$$

i.e. where the witnesses X are subsets of finite paths.

In the following Section C.1 we prove an auxiliary result which (roughly) states that over finite words, the approximate size of a largest witness can be computed by a deterministic automaton equipped with counters. In Section C.2, we apply this result pathwisely to trees and finish the proof of Proposition 7.

C.1 Estimating the maximal size of a φ-witness

In this section, we adapt the method used in [2] to obtain a slightly enhanced result, which we will later use to prove that nested limsup automata recognize languages of the form (4). The result which we prove now, Theorem 13, is in the context of *finite words*, and it roughly says the following. If we are given a formula φ of MSO with one free variable X, then there exists a deterministic counter automaton with lookahead which computes an approximation of the size of the largest φ-witness for the input word. We will state this result more formally in the context of monoids.

Fix a finite monoid M acting on a finite set U. An *associative sequence over* (M, U) is a sequence

$$w = (m_1, u_1), (m_2, u_2), \ldots, (m_n, u_n) \tag{w}$$

of elements of $M \times U$ such that for $i = 1, \ldots, n-1$,

$$u_i = m_{i+1} \cdot u_{i+1}. \tag{$*$}$$

The element $m_1 \cdot u_1 = m_1 \cdot m_2 \cdots m_n \cdot u_n$ is called *the type of* w and is denoted type(w). Note that w is uniquely determined by $m_1, \ldots, m_n \in M$ and $u_n \in U$.

We define the powerset monoid $P(M)$ in the usual way, by taking as elements subsets of M and defining multiplication by $X \cdot Y = \{x \cdot y : x \in X, y \in Y\}$. The identity is the singleton $\{1_M\}$. Similarly we define $X \cdot V$ for $X \subseteq M$ and $V \subseteq U$. This way $P(M)$ acts on the set $P(U)$. Let

$$W = (M_1, U_1), (M_2, U_2), \ldots, (M_n, U_n) \tag{W}$$

be an associative sequence over $(P(M), P(U))$ and let w be an associative sequence over (M, U) of the form (w). We write that $w \in W$ if $m_i \in M_i$ and $u_i \in U_i$ for $i = 1, \ldots, n$.

Recall that a *prime ideal* in a monoid M is a set $I \subseteq M$ such that for all $s, t \in M$, $s \cdot t \in I$ if and only if $s \in I$ or $t \in I$. An archetypical example of a prime ideal is as follows. Let $f \colon M \to \{0, 1\}$ be a homomorphism of a monoid to the monoid $\{0, 1\}$ equipped with the max operation. Then the elements which are mapped to 1 form a prime ideal in M.

Let I be a prime ideal in M and let V be a subset of U. Let us denote the *support* of w

$$\text{supp}_I(w) = \{1 \le i \le n : \ m_i \in I\}.$$

We define:

$$\text{max-count}_{I,V}(W) \quad = \quad \max\{|\,\text{supp}_I(w)| : \quad w \in W \text{ such that } \text{type}(w) \in V\}.$$

The aim is to construct a deterministic counter automaton, which, for a given associative sequence of the form (W), computes an approximation of the size max-count$_{I,V}(W)$. The *domination relation*, defined in [8], captures the precise notion of approximation that we need.

▶ **Definition 12.** Let X be any set. For two functions $f, g \colon X \to \overline{\mathbb{N}}$ we write that $f \preceq g$ if for any $K \subseteq X$, if $f|_K$ is unbounded then $g|_K$ is unbounded. In this case, we also say that *g dominates f*. We write that $f \simeq g$ if $f \preceq g$ and $g \preceq f$.

Let $\mathcal{A}$ be a deterministic automaton with counters Π, which can be incremented or reset. For a finite word w, we denote by $[\![\mathcal{A}]\!]_{max}(w)$ the maximal value attained by a counter of $\mathcal{A}$, while processing the word w.

▶ **Theorem 13.** *Let (M, U) be a finite transformation monoid, let I be a prime ideal in M and V a subset of U. There exists a deterministic counter automaton $\mathcal{A}_{I,V}$ such that the following equivalence holds*

$$[\![\mathcal{A}_{I,V}]\!]_{max} \simeq \text{max-count}_{I,V} \tag{$\simeq$}$$

over the set of associative sequences over $(P(M), P(U))$.

Let C be a finite, linearly ordered set of colors. Let w be a finite word of length n. We call the set $\{1, \ldots, n\}$ the *positions* of w. We also consider the additional dummy *initial position* and *end position*, corresponding to the numbers 0 and $n + 1$. An *edge* x of w is a pair of consecutive (possibly initial or end) positions denoted $b(x)$ and $e(x)$, respectively. The edges of w are ordered in a natural way.

A C-coloring of w is a coloring of the set of edges of w, by colors from C. Fix such a coloring. We say that two edges of w, x and y are *neighboring*, it they have the same color and all edges in between have a smaller color. If x and y are neighboring, then we call the sequence of positions of w between x and y *factor between x and y* of w, or *the y-factor* in w. Note that not for every edge y there is a well-defined y-factor, but if it is defined, it is

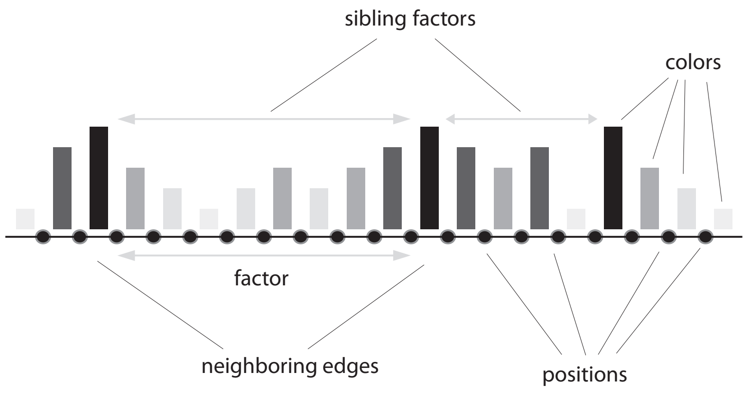


Figure 4 A colored word

unique. If $x_0 < x_1 < \ldots < x_n$ are such that for $i = 1, \ldots, n$, the edges x_{i-1} and x_i are neighboring, and if f_i is the factor between x_{i-1} and x_i, then we say that $f_1, \ldots, f_n$ is a sequence of *sibling* factors.

We will use the following simple lemma as an analytic tool for proving the equivalence ($\simeq$).

▶ **Lemma 14.** *Let W be a finite word and η a C-coloring of the edges of W and let X be a finite set of positions of W. Then the maximal length k of a sequence of sibling factors $f_1, \ldots, f_k$ of w, each of which contains an element of X, satisfies*

$$|X| \geq k \geq \log_{|C|+1} |X|.$$

Proof. Let us consider the tree whose nodes are factors containing an element of X, ordered by the subfactor relation, and augmented with a dummy root. This tree has height $|C| + 1$ and $|X|$ leaves. Then k is the same as the largest degree of all of its nodes, so it satisfies $|X| \leq (|C| + 1)^k$ and $k \leq |X|$, which proves the desired inequalities. ◀

If M is a monoid, $w = m_1, m_2, \ldots, m_n$ is a word over M (or the sequence of first coordinates of an associative sequence), then for two edges $x \leq y$ of w we denote

$$w(x..y) \overset{\text{def}}{=} m_i \cdot m_{i+1} \cdots m_{j-1} \cdot m_j \quad \in \quad M$$

where $i = e(x)$ and $j = b(y)$. Let us recall a theorem of Colcombet, which is an enhanced version of the factorization forest of Simon.

▶ **Theorem 15** (Colcombet, [7]). *Let S be a finite monoid. Then there exists a finite, linearly ordered set of colors C and a deterministic transducer $\mathcal{T}_S$ with input alphabet S and output alphabet C such that, given a word $w \in S^*$, $\mathcal{T}_S$ produces a C-coloring $\mathcal{T}_S(w) \in C^*$ of w which satisfies*

$$w(x..y) = w(x..z) \qquad \textit{for neighboring edges } x < y < z. \tag{1}$$

We will apply the above theorem to the monoid $P(M)$. Let W be a fixed associative word over $(P(M), P(U))$ of the form (W). Let x be an edge in W. We define the following notions:

1. The color of x, denoted $c(x)$ – the color c output by the transducer $\mathcal{T}_{P(M)}$ at the edge x, when processing W
2. The type of the x-factor, denoted $M_f(x)$ – the product of the labels along the positions of the x-factor; if the x-factor is undefined, its type is $\emptyset$
3. The type of the prefix before the x-factor, denoted $M_p(x)$ – the element $W(x_0..x_1)$, where x_0 is the initial dummy edge of W and x_1 is the first edge before the x-factor. If the x-factor is undefined, the type of the prefix before is $\emptyset$
4. The suffix type of x, denoted $U_s(x)$ – the element U_i, where $i = b(x)$.

Basing on the transducer $\mathcal{T}_{P(M)}$, it is straightforward to construct a deterministic transducer $\mathcal{T}$, which, for each edge x of the input word, computes the tuple

$$\big(c(x), M_p(x), M_f(x), U_s(x)\big).$$

The output alphabet of $\mathcal{T}$ is

$$\Gamma = C \times P(M) \times P(M) \times P(U).$$

From this transducer $\mathcal{T}$ we construct a deterministic automaton $\mathcal{A}_{I,V} = \mathcal{A}$ which has a counter c_λ corresponding to each tuple $\lambda \in \Gamma$. The counter c_λ is incremented whenever $\mathcal{T}$ outputs λ and reset whenever $\mathcal{T}$ outputs a tuple whose C-coordinate is larger than the one of λ.

There is a set of *important counters* $\Pi \subseteq \Gamma$, such that only their values are considered in the result of $[\![\mathcal{A}]\!]_{max}$. (Formally, the set of counters of $\mathcal{A}$ is $\Pi \subseteq \Gamma$, not Γ, and the operations performed on counters outside of Π are ignored.) The distinguished set Π of important counters of $\mathcal{A}$ is the set of all tuples $(c, M_p, M_f, U_s) \in \Gamma$ such that:

$$\begin{array}{c} \textit{there exist} \quad m_p \in M_p, \quad m_f \in M_f \cap I, \quad u_s \in U_s \\ \textit{such that} \quad m_p \cdot m_f \cdot u_s \in V \quad \textit{and} \quad m_f \textit{ is idempotent.} \end{array} \tag{Π}$$

We will now prove the equivalence

$$[\![\mathcal{A}_{I,V}]\!]_{max} \simeq \text{max-count}_{I,V} \tag{$\simeq$}$$

Proof. We will prove $(\simeq)$ by first showing that $\text{max-count}_{I,V} \preceq [\![\mathcal{A}]\!]_{max}$ and then showing that $\text{max-count}_{I,V} \succeq [\![\mathcal{A}]\!]_{max}$.

$(\preceq)$ Assume that K is a set of associative sequences over $(P(M), P(U))$, over which the function $\text{max-count}_{I,V}$ is unbounded. We will show that then, $[\![\mathcal{A}]\!]_{max}$ is also unbounded over the set K. To reach a contradiction, assume that there is a bound l such that $[\![\mathcal{A}]\!]_{max}(W) < l$ for every $W \in K$. We may assume that the bound l is large enough – we assume that $l > \log(3|M|)$.

Let $W \in K$ be an associative word of the form (W) such that $\text{max-count}_{I,V}(W)$ is very large. To be precise, we choose W so that

$$\log_{|C|+1}\Big(\text{max-count}_{I,V}(W)\Big) \geq l \cdot |\Gamma|. \tag{2}$$

We will show that (2) implies that $[\![\mathcal{A}]\!]_{max}(W) \geq l$, reaching a contradiction. First we show that when processing W, some counter λ is incremented at least l times (without any resets in between). Then we show that λ is an important counter.

A counter λ is incremented l times From the assumption (2), there exists an associative sequence w, such that for $X = \mathrm{supp}_I(w)$,

$$\begin{aligned} w &\in W, \\ \mathrm{type}(w) &\in V, \\ \log_{|C|+1}|X| &\geq l \cdot |\Gamma|, \end{aligned} \tag{3}$$

Apply Lemma 14 to W and X to conclude that there exist k sibling factors in W, each containing an element of X and such that

$$k \geq l \cdot |\Gamma|.$$

By the pigeonhole principle, there is a tuple $\lambda \in \Gamma$ and l sibling factors $f_1, \ldots, f_l$ such that after processing f_i, the transducer $\mathcal{T}$ outputs λ. Note that c_λ is incremented at the end of each factor f_i and is not reset in between them, so it reaches the value l. To prove that $[\![\mathcal{A}]\!]_{max}(W) \geq l$, it remains to show that $\lambda \in \Pi$.

The counter λ is important Assume that $\lambda = (c, M_p, M_f, U_s)$. For $i = 1, \ldots, l$, let x_i and y_i denote the first and last edge of the factor f_i, correspondingly. Let x_{l+1} denote y_l and for $i = 1, \ldots, l$, let

$$m_i = w(x_i..x_{i+1}).$$

Note that $m_i \in I$ since I is an ideal and f_i contains a position in the set X. Also

$$m_i \in W(x_i..x_{i+1}) = W(x_i..y_i) = M_f.$$

The first equality above follows from equation (1), while the second one comes from the assumption on λ. Note that from $M_f \cdot M_f = M_f$ it follows that M_f is a subsemigroup of M.

By a standard Ramsey argument, if l is sufficiently large ($l > \log(3|M|)$ suffices), there exist indices $i < j$ such that

$$m_f \overset{\mathrm{def}}{=} m_i \cdot m_{i+1} \cdots m_{j-1} \cdot m_j$$

is idempotent. Since M_f is a semigroup, it follows that $m_f \in M_f$ and moreover $m_f \in M_f \cap I$ since already $m_i \in I$ and I is an ideal.

Let m_p denote $w(x_0..x_i)$, and t_s denote u_j. Then, $m_p \cdot m_f \cdot u_s = \mathrm{type}(w)$, and by assumption, $\mathrm{type}(w) \in V$, implying that

$$m_p \cdot m_f \cdot u_s \in V.$$

Moreover, $m_p \in W(x_0..x_i) = M_p$ by the assumption on λ. Similarly, $u_s \in U_s$. Thus we have proved that λ satisfies the property (Π), i.e. $\lambda \in \Pi$. Together with the fact that c_λ reaches the value l, we obtain that $[\![\mathcal{A}]\!]_{max}(W) \geq l$ – a contradiction. This shows that $\text{max-count}_{I,V} \preceq [\![\mathcal{A}]\!]_{max}$.

($\succeq$) Let k be any number and W be an associative word of the form (W) such that

$$[\![\mathcal{A}]\!]_{max}(W) \geq k. \tag{4}$$

We will show that

$$\text{max-count}_{I,V}(W) \geq k.$$

This is again split into two steps. Basing on the run of $\mathcal{A}$ on W, we construct a witness word $w \in W$. Then we show that the type of the witness is in V and that its support has at least k elements.

Construction of a witness By equation (4), there exists $\lambda = (c, M_f, M_p, U_s)$ such that the counter c_λ reaches value k at some moment and $\lambda \in \Pi$. Let m_p, m_f, u_s be elements which witness the property (Π) of λ.

There exists a sequence of neighboring edges $y_1, y_2, \ldots, y_k$ such that at each of them the counter c_λ is incremented and is not reset in between. Therefore, $y_1, \ldots, y_k$ are the last edges of sibling factors $f_1, \ldots, f_k$ of the same type M_f. For $i = 1, \ldots, k$, let x_i be the initial edge of the factor f_i and let $x_{k+1} = y_k$. Equation (1) implies that, for $i = 1, \ldots, k$,

$$W(x_i..x_{i+1}) = W(x_i..y_i) = M_f.$$

Let x_0 denote the first edge in W and let y_{k+1} denote the last edge in W. We need to label the positions of W with elements of M, to obtain a word $w \in W$. We will construct this labeling separately over each factor $f_1, \ldots, f_k$, and also separately over the segment $x_0..x_1$ of positions before the factor f_1 and the segment of positions $y_k..y_{k+1}$ after the factor f_k.

Labeling of the factors Since $m_f \in M_f$, this implies that for each $i = 1, \ldots, k$ there exists a labeling m^i of the factor $x_i..x_{i+1}$ by elements of M_f, such that the resulting word has type m_f. Since $m_f \in I$ and I is prime, each sequence m^i contains at least one element of I.

Labeling of the prefix There exists an M-labeling m^0 of the segment $x_0..x_1$, such that $\pi(m^0) = m_p$.

Labeling of the suffix Let $\beta = e(y_k)$. Since

$$u_s \in U_s = U_s(y_k) = U_\beta = M_{\beta+1} \cdot M_{\beta+2} \cdots M_n \cdot U_n,$$

there exists an M-labeling m^{k+1} of the factor $y_k..y_{k+1}$ and an element $u_n \in U_n$ such that $u_s = \text{type}(m^{k+1}) \cdot u_n$.

Putting together the labelings $m^0, m^1, \ldots, m^k$, we obtain our witness – a labeling w of the positions of W, and an additional element $u_n \in U_n$. Clearly, $w \in W$.

The type and support of the witness We verify that $\text{type}(w) \in V$:

$$\begin{aligned} \text{type}(m^0) \cdot \text{type}(m^1) \cdots \text{type}(m^k) \cdot (\text{type}(m^{k+1}) \cdot u_n) &= \\ m_p \cdot m_f \cdots m_f \cdot u_s &= \\ m_p \cdot m_f \cdot u_s &\in V. \end{aligned} \tag{5}$$

Moreover, $|\text{supp}_I(w)| \geq k$. We have thus shown that $\text{max-count}_{I,V}(W) \geq k$. Therefore, $\text{max-count}_{I,V} \geq [\![\mathcal{A}]\!]_{max}$. This ends the proof of the equivalence ($\simeq$) ◀

C.2 Back to the proof of Proposition 7

We now return to the proof of Proposition 7. Recall that we need to show that nested limsup automata recognize all languages of the form

$$\hat{\mathsf{U}}X\ \varphi(X) \qquad \textit{for } \varphi(X) \textit{ in } \text{WMSO}.$$

Fix a formula φ of WMSO with one free set-type variable. Without loss of generality, we assume that $\varphi(X)$ is false if X is not a chain, by considering the formula $\varphi(X) \wedge \mathit{chain}(X)$ instead of $\varphi(X)$ if necessary.

We consider the set U of tree φ-types, and the monoid M context φ-types, with the additional bit indicating whether the decoration is empty. The monoid M acts on the set

U. Note that M has a distinguished a prime ideal, corresponding to types of nonempty decorations.

Formally, these objects are defined as follows. For a (1-decorated) context p and tree t, let $\mu_p(t) = p \cdot t$. Recall that the mapping μ_p preserves φ-types. Therefore, μ_p lifts to a transformation of the set of φ-types, which is finite by Lemma 9. This transformation is called the *φ-context type* of the context p. Let S be the monoid of φ-types. Its neutral element is the context type of the empty context. The *augmented* φ-context type of p is the pair consisting of the φ-context type of p, and an additional bit set to 0 iff the decoration of p is empty. Therefore, the set of augmented φ-context types forms a submonoid M of the product $S \times \{0,1\}$, where $\{0,1\}$ is a monoid in which $x \cdot y = \max(x,y)$. Let I be the set of elements of M which are augmented with 1. Then I is a prime ideal in M.

Let U be the set of (1-decorated) φ-tree types. Then M acts in a natural way on U via the action of context types on U. Let $V \subseteq U$ be the set of types which satisfy φ. Let $\mathcal{A}_{I,V}$ denote the deterministic automaton constructed in Theorem 13. This automaton was assumed to run over finite associative sequences, but due to determinism it may also process infinite trees, as long as every finite rooted path of the tree is labeled by an associative sequence. We will construct a Nested limsup transducer (a $\mathscr{C}$-transducer, where $\mathscr{C}$ is the class of nested limsup automata) $\mathcal{T}_0$ which, for a given tree t, outputs precisely a labeled tree which can be processed by $\mathcal{A}_{I,V}$.

▶ **Lemma 16.** *There exists Nested limsup transducer $\mathcal{T}_0$ (not even using atomic limsup automata) such that nesting $\mathcal{A}_{I,V}$ on top of $\mathcal{T}_0$ results in a nested limsup automaton accepting precisely the trees which satisfy the formula $\hat{\mathsf{U}}X.\varphi(X)$.*

Sketch of proof. We outline the construction of $\mathcal{T}_0$. For any finite path π, which starts in the root of t, and ends in some vertex v of t, we factorize the tree t along the path π, by viewing t as $|\pi|$ contexts applied to the tree $t[v]$ rooted at v (see Figure 5). We then label

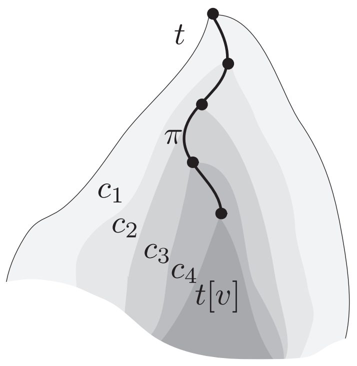


Figure 5 A tree factorized along a path π: $t = c_1 \cdot c_2 \cdot c_3 \cdot c_4 \cdot t[v]$

each vertex w along the path π by two sets – $\mu(w) \subseteq M$ and $\nu(w) \subseteq U$.

- $\mu(w)$ consists of those augmented context φ-types, which can be obtained by decorating the context corresponding to the vertex w,
- $\nu(w)$ consists of those tree φ-types, which can be obtained by decorating the subtree $t[w]$ of t rooted at w.

Clearly, the labels $\mu(w)$ and $\nu(w)$ do not depend on the choice of the finite path passing through w. Both labelings of t can be carried out by a Nested limsup Transducer which does not even use atomic lim sup automata.

The described labeling of the given path π can be viewed as an associative sequence over $(P(M), P(U))$. Therefore, the automaton $\mathcal{A}_{I,V}$ constructed in Theorem 13, after processing the labeled path π, stores in its counters an estimate of the value of max-count over this associative sequence. This value is nothing else as the size of the maximal witness contained in the path π (recall that we assumed that any witness is contained in some path). Thus, the nesting of the automaton $\mathcal{A}_{I,V}$ on top of $\mathcal{T}_0$ yields a nested limsup automaton, which recognizes whether t satisfies $\hat{\mathsf{U}}X.\varphi(X)$. ◀

Appendix Part II

From Nested Limsup Automata to Puzzles

D Reduction to puzzles

This section is devoted to a proof of the following result:

Proposition 3. *For every nested limsup automaton one can compute a puzzle that recognizes the same language.*

The essential difficulty. We begin with an example that describes the essential difficulty in the reduction. Suppose that a language K is recognized by an atomic limsup automaton $\mathcal{A}$. It is not difficult to define a nested limsup automaton that accepts the tree language

$$\mathsf{AG}K \overset{\text{def}}{=} \{t : \text{ every subtree of } t \text{ belongs to } K\}.$$

When proving the reduction to puzzles, we need to define a puzzle that recognizes the language $\mathsf{AG}K$. A natural automaton for $\mathsf{AG}K$ would be a kind of alternating automaton, which would spawn a new copy of $\mathcal{A}$ in every subtree. A run over an infinite tree would involve infinitely many copies, each one with its own counter. However, a puzzle is not an alternating automaton, and it only has a fixed finite number of counters. Therefore, in the reduction to puzzles we need a policy for reusing counters for new spawns of the automaton $\mathcal{A}$. This kind of policy can be seen as the essence of the reduction to puzzles.

D.1 Reduction to languages $\mathsf{AG}K$.

We begin the proof of the reduction to puzzles. We first show that languages of the form $\mathsf{AG}K$, as described in the example above, are the only kind of languages that need to be dealt with. The remainder of Section D is devoted to dealing with languages of the form $\mathsf{AG}K$, where either K or its complement is recognized by an atomic limsup automaton.

Let L be a language recognized by a nested limsup automaton $\mathcal{A}$ with input alphabet A. Let

$$\mathcal{A}_1, \ldots, \mathcal{A}_n$$

be the subautomata of $\mathcal{A}$ (the subautomata of $\mathcal{B}[\mathcal{B}_1, \ldots, \mathcal{B}_i]$ are $\mathcal{B}$ and all the subautomata of $\mathcal{B}_1, \ldots, \mathcal{B}_i$). For a tree t over alphabet A, let $\hat{t}$ be the tree over alphabet $A \times 2^n$ where the label of every node x is extended by a bit-vector that indicates which of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept the subtree of x. Define

$$\hat{L} = \{\hat{t} : t \text{ is accepted by } \mathcal{A}\} \subseteq \text{trees}(A \times 2^n).$$

Clearly, L is a projection of $\hat{L}$. By nondeterminism, languages recognized by puzzles are closed under projection. Therefore, it remains to prove that $\hat{L}$ is recognized by a puzzle.

▶ **Lemma 17.** *The language $\hat{L}$ is equal to a finite intersection of languages, where each intersected language is of one of the following kinds:*

1. *A regular tree language.*
2. $\mathsf{AG}K$*, where K or its complement is recognized by an atomic limsup automaton.*

Proof. Recall that $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are the subautomata of $\mathcal{A}$, with $\mathcal{A} = \mathcal{A}_1$.

Consider one of the subautomata above, say $\mathcal{A}_i$. This subautomaton is of the form

$$\mathcal{B}_i[\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_k}],$$

where $\mathcal{B}_i$ is either a prefix automaton or an atomic limsup automaton, and the indexes $i_1, \ldots, i_k$ are from $\{1, \ldots, n\}$. The input alphabet of $\mathcal{B}_i$ consists of bit-vectors $\{0,1\}^k$, which stand for the results of the automata $\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_k}$. Let $\hat{\mathcal{B}}_i$ be the automaton with input alphabet $\{0,1\}^n$, which accepts a tree t if and only if $\mathcal{B}_i$ accepts the projection of the labels of t onto the coordinates $i_1, \ldots, i_k$. This automaton is almost identical to $\mathcal{B}_i$, in particular it is a prefix automaton, or an atomic limsup automaton. Define K_{i1} to be the language of trees $\hat{t}$ over alphabet $A \times \{0,1\}^n$ that satisfy the implication:

If the i-th bit of the label of the root in $\hat{t}$ is 1, then t is accepted by $\hat{\mathcal{B}}_i$.

Likewise, define K_{i0} to be the language

If the i-th bit of the label of the root in $\hat{t}$ is 0, then t is rejected by $\hat{\mathcal{B}}_i$.

If $\mathcal{B}_i$ is an atomic limsup automaton, then the language K_{i1} is recognized by an atomic limsup automaton, and the complement of the language K_{10} is recognized by an atomic limsup automaton. It is not difficult to see that

$$\hat{L} = K \cap \bigcap_{i \in \{1,\ldots,n\}} \mathsf{AG}K_{i0} \cap \mathsf{AG}K_{i1}$$

where K says that the root has 1 on the first-coordinate of the bit-vector in its label. Clearly K is a regular tree language. Also, when $\mathcal{B}_i$ is a prefix automaton, then both $\mathsf{AG}K_{i0}$ and $\mathsf{AG}K_{i1}$ are regular tree languages. This completes the proof of the lemma. ◀

▶ **Lemma 18.** *Languages recognized by puzzles are closed under intersection.*

Proof. The usual cartesian product construction works. The counters can be run in parallel, since puzzles allow multiple counters. The only nontrivial part is the parity condition, but parity tree automata are known to be closed under intersection. ◀

By the above two lemmas, it remains to show that every language of the two kinds used in Lemma 17 is recognized by a puzzle. The first kind is recognized by puzzles, because puzzles are equipped with a parity condition, which can be used to define any MSO-definable property. For the second kind, we prove the following lemmas.

▶ **Lemma 19.** *Let K be a language whose complement is recognized by an atomic limsup automaton. Then $\mathsf{AG}K$ is recognized by a puzzle.*

▶ **Lemma 20.** *Let K be a language recognized by an atomic limsup automaton. Then $\mathsf{AG}K$ is recognized by a puzzle.*

These lemmas are proved in Sections D.3 and D.4, respectively. First though, we introduce the concept of *tapes*, which is used in both proofs.

D.2 Tapes

In this section we describe tapes, which are used in the proofs of Lemmas 19 and 20. Some of the ideas, especially the definition of a *tape system*, are tree generalizations of what was called the *tape construction* in [2]. The notion of tapes, as presented here, is applied to any kind of automaton where the states are updated according to a top-down deterministic transition function

$$\delta : Q \times A \to Q^2.$$

This applies, in particular to atomic limsup automata. (When describing tapes, we ignore the additional component in the transition function of an atomic limsup automaton, which describes the counter operations.)

Configurations. Fix for the rest of Section D.2 a set of states Q and a transition function δ as above. Define a *configuration* to be a pair (x, q) where x is a tree node and q is a state. Given an input tree t, we can use the transition function δ, to define the left and right successor configurations for each configuration. A successor is a left or right successor configuration. One configuration is reachable from another if it is reachable via successor steps.

Tape. We define a tape (in a tree t) to be a partial function

$$T : 2^* \to Q$$

whose domain is connected, and which is consistent with the transition function δ. We often interpret T as the set of configurations

$$\{(x, T(x)) : T \text{ is defined on } x\},$$

so we can say that a configuration belongs to a tape, or take the union of two tapes (the result of a union of two tapes might no longer be a tape, but it is always a set of configurations). This interpretation is consistent with the set-theoretic definition of a function as a set of pairs (argument/value).

Tape systems. We say a tape S *merges* with a tape T if T contains a successor of a configuration of S. Let t be a tree. A *tape system* in t is a family of disjoint tapes $\mathcal{T}$, possibly infinite, such that for some ordering

$$\mathcal{T} = \{T_1, T_2, \ldots\},$$

the following *merge condition* holds: whenever T_i merges with T_j, then $j \leq i$.

One could think of a different definition of a tape system. In the different definition, which we will call a *generalized tape system*, a tape system is a family of disjoint tapes, possibly infinite, such that the merging relation is well-founded, i.e. it is impossible to find an infinite sequence of tapes $T_1, T_2, \ldots$ in the family such that T_i merges with T_{i+1}. It is easy to see that a tape system is also a generalized tape system, but not necessarily the other way round.

The following lemma states the key property of a tape system, which is a kind of compactness. We say a set of configurations T (which includes the case of tapes) is *covered* by a family of tapes $\mathcal{T}$ if every configuration of T belongs to some tape from $\mathcal{T}$.

▶ **Lemma 21.** *Let $\mathcal{T}$ be a tape system. Every tape covered by tapes from $\mathcal{T}$ is covered by a finite number of tapes from $\mathcal{T}$.*

Proof. Let $T_1, T_2, \ldots$ be an ordering of $\mathcal{T}$ as required by the definition of tape systems. Let S be a tape covered by $\mathcal{T}$. Map every configuration of S to the number of the (unique) tape from $\mathcal{T}$ that contains it. The mapping is non-increasing along paths in S, therefore it must have finite image. ◀

Observe that Lemma 21 fails for generalized tape systems. Because Lemma 21 is important for us, we no longer use generalized tape systems, and the reader can forget about them.

Existence of tape systems. The following straightforward lemma shows that tape systems exist.

▶ **Lemma 22.** *Let Σ be a set of configurations in a tree t, which is closed under successor configurations. Then there exists a tape system $\mathcal{T}$ that partitions Σ.*

Proof. Let $(x_1, q_1), (x_2, q_2)$ be an enumeration of configurations in Σ such that the sequence of node depths $|x_1|, |x_2|, \dots$ is non-decreasing. We define a sequence of disjoint tapes

$$T_1, T_2, \dots$$

by induction. In the definition, we maintain the invariant that $T_1 \cup \cdots \cup T_i$ is closed under successor configurations for every i.

Suppose we have already defined the tapes systems $T_1, \dots, T_{i-1}$. Let T be the set of configurations that are reachable from configuration (x_i, q_i). Define

$$T_i = T - (T_1 \cup \cdots \cup T_{i-1}).$$

Thanks to the invariant, the set of configurations T_i is connected, as the difference of two sets closed under successor configurations. In other words, T_i is a tape (possibly empty). Also, the invariant is maintained, because

$$T_1 \cup \cdots \cup T_i = T \cup T_1 \cup \cdots \cup T_{i-1}$$

is a set closed under successor configurations, as a union of two such sets, namely T and $T_1 \cup \cdots \cup T_{i-1}$.

We claim that the infinite sequence $T_1, T_2, \dots$ is a tape system. We only need to show that the merge condition holds. From the construction of the system, tape T_i can only merge with tapes $T_1, \dots, T_{i-1}$. ◀

Representing a tape system. Suppose that $\mathcal{T}$ is a family of disjoint tapes, which covers the case of tape systems. We represent this family as a tree

$$[\mathcal{T}] \in \text{trees}(Q \to (Q \cup \{\bot, \mathit{root}\}))$$

defined as follows. Let (x, q) be a configuration, and let T be the unique (possibly undefined) tape from $\mathcal{T}$ that contains the configuration (x, q). We define

$$[\mathcal{T}](x)(q) = \begin{cases} \bot & \text{if } T \text{ is undefined} \\ \mathit{root} & \text{if } T \text{ is defined and } x \text{ is the root of its domain.} \\ T(y) & \text{if } T \text{ is defined and contains the parent of } x \text{, call it } y. \end{cases}$$

In the last item, we treat T as a function from its domain to states. It is not difficult to see that the encoding $\mathcal{T} \mapsto [\mathcal{T}]$ is one-to-one.

We would like the following tree language to be regular:

$$\{t \otimes [\mathcal{T}] : \ \mathcal{T} \text{ is a tape system in } t\}.$$

($t \otimes [\mathcal{T}]$ denotes the label-wise merge of the trees t and $[\mathcal{T}]$.) It is not clear how to write an automaton recognizing the definition of a tape system; a naive construction would have to guess a linear ordering on the tapes of $\mathcal{T}$, which cannot be done by a finite state automaton.

Our approach is to recognize not all tape systems, but only a restricted class of tape systems, as defined below.

A tape system $\mathcal{T}$ is called *ancestral* if whenever a tape $S \in \mathcal{T}$ merges with a tape $T \in \mathcal{T}$, then the root node of the domain of T is an ancestor of, or equal to, the root node of the domain of S. Observe that the tape system defined in Lemma 22 is ancestral.

▶ **Lemma 23.** *The following tree language is regular.*

$$\{t \otimes [\mathcal{T}] : \ \mathcal{T} \textit{ is an ancestral tape system in } t\}$$

D.3 Puzzle for bounded counters

This section is devoted to showing Lemma 19, which says that for every language K whose complement is recognized by an atomic limsup automaton, the language $\mathsf{AG}K$ is recognized by a puzzle.

Fix, for the rest of Section D.3, an atomic limsup automaton $\mathcal{A}$ recognizing the complement of K. Let Q be its states and A its input alphabet.

Values of sets of configurations. We use the term *configuration path* for a sequence, possibly infinite, of configurations such that every element of the sequence is a successor of the previous element. To a finite configuration path

$$\pi = (x_1, q_1), \dots, (x_n, q_n)$$

we assign its value $\mathrm{val}(\pi) \in \mathbb{N}$ which is the maximal counter value seen when executing the counter operations of the automaton $\mathcal{A}$, assuming that the counter is initialized at 0. When T is a set of configurations, possibly infinite, we define $\mathrm{val}(T) \in \mathbb{N} \cup \{\infty\}$ to be the least upper bound on $\mathrm{val}(\pi)$, ranging over paths included in T.

Accessible configurations We use the term *initial configuration* for a configuration of the form (x, q_I), where q_I is the initial state of the automaton and x is any node of the tree. A configuration is called *accessible* if it is reachable from some initial configuration.

▶ **Lemma 24.** *Let t be a tree, and let $\mathcal{T}$ be a tape system for its accessible configurations. Then t belongs to $\mathsf{AG}K$ if and only if*

$$\mathrm{val}(T) < \infty \qquad \textit{holds for all } T \in \mathcal{T}.$$

Proof. By definition, a tree t belongs to $\mathsf{AG}K$ if and only if for every initial configuration (x, q_I), the set S of configurations reachable from (x, q) satisfies

$$\mathrm{val}(S) < \infty.$$

The "only if" implication in the statement of the lemma is straightforward. Suppose that T is a tape from $\mathcal{T}$. Then T is included in some set S as described above, and $\mathrm{val}(T) < \infty$ holds because the mapping val assigns smaller or equal numbers to smaller or equal sets of configurations.

The converse if implication follows from Lemma 21. Indeed, let (x, q_I) be an initial configuration, and let S be the configurations reachable from S. By the Lemma 21, S is covered by a finite number of tapes from $\mathcal{T}$. The result follows from the following straightforward observation: for any tapes $U_1, \dots, U_n$,

$$\mathrm{val}(U_1 \cup \dots \cup U_k) \quad \leq \quad \mathrm{val}(U_1) + \dots + \mathrm{val}(U_k).$$

◀

▶ **Lemma 25.** *The following language is recognized by a puzzle.*

$$\{[\mathcal{T}] : \ \mathrm{val}(T) < \infty \textit{ holds for every } T \in \mathcal{T}\}$$

Before we prove this lemma, we show how it completes the proof of Lemma 19, which is the goal of this Section D.3. Recall that we need to find a puzzle that recognizes the language $\mathsf{AG}K$. By Lemmas 22 and 24, a tree t belongs to $\mathsf{AG}K$ if and only if there exists an ancestral tape system $\mathcal{T}$ for its accessible configurations such that $\mathrm{val}(T) < \infty$ holds for every $T \in \mathcal{T}$. The second part of the equivalence is recognized by a puzzle thanks to Lemmas 23 and 25.

To finish Section D.3, we present the proof of Lemma 25. First, we state a simple coloring result, stated below.

▶ **Lemma 26.** *Let $\mathcal{T}$ be a family of disjoint tapes. There exists a coloring function $\tau : \mathcal{T} \to \{1, \ldots, |Q|\}$ such that every two tapes with the same color have disjoint domains.*

Proof. Let $\mathcal{F}$ be the family of factors that are domains of tapes from $\mathcal{T}$. (This is actually a multiset, since several tapes might have the equal domains.) Every node appears in at most $|Q|$ factors from $\mathcal{F}$. The family $\mathcal{F}$ can therefore be colored by at most $|Q|$ colors, in a top-down fashion. ◀

Proof of Lemma 25. The puzzle uses $|Q|$ counters, one for each of the colors from Lemma 26. The counters are acted upon according to the transitions of $\mathcal{A}$, and they are cut whenever a tape is finished. ◀

D.4 Puzzle for unbounded counters

This section is devoted to showing Lemma 20, which says that for every language K which is recognized by an atomic limsup automaton, the language $\mathsf{AG}K$ is recognized by a puzzle.

Fix, for the rest of Section D.4, an atomic limsup automaton $\mathcal{A}$ recognizing K. Let Q be its states and A its input alphabet.

We use the same definition of initial configuration as in the previous section. We say a configuration is unbounded if the run that begins in the configuration has unbounded counter values. A tree belongs to $\mathsf{AG}K$ if and only if every initial configuration is unbounded.

There are several differences with the proof from the previous section. Unlike for bounded counters, configurations with unbounded counters are not closed under successors (but they are closed under predecessors). The consequence is that it makes less sense to talk about a tape being unbounded. That is, a tape T might satisfy $\mathrm{val}(T) = \infty$, but some of its configurations might be bounded.

Witnesses. Let π be an infinite configuration path. We say that π is unbounded, if every configuration on π is unbounded. Every unbounded configuration is on some unbounded path, because every unbounded configuration has an unbounded successor.

▶ **Lemma 27.** *A tree t belongs to $\mathsf{AG}K$ if and only if there exists a family Π of disjoint unbounded configuration paths, such that every initial configuration can reach some configuration appearing in Π.*

Proof. We would like to underline that configuration paths, as a special case of tapes, are disjoint when they do not share configurations. They might share nodes in their domains.

The "if" implication is immediate, we focus on the "only if" implication. Suppose then that a tree t belongs to $\mathsf{AG}K$. We need to define the family Π from the statement of the lemma. Let $x_1, x_2, \ldots$ be an enumeration of all nodes. We define a sequence of unbounded configuration paths $\pi_1, \pi_2, \ldots$ such that for every i, the initial configuration in x_i can reach a configuration on one of the paths $\pi_1, \ldots, \pi_i$. We then take Π to be $\{\pi_1, \pi_2, \ldots\}$.

The definition of π_i is by induction. Suppose that $\pi_1, \ldots, \pi_{i-1}$ have already been defined. Consider the node x_i. Because t belongs to $\mathsf{AG}K$, the initial configuration in x_i, and therefore there is an unbounded configuration path π that is reachable from x_i. If π shares a configuration with some π_j for $j < i$, then we can use $\pi_i = \pi_j$. Otherwise, we define $\pi_i = \pi$. ◀

Let Π be a family of disjoint configuration paths. Because a configuration path is a special case of a tape, we can encode Π as a tree $[\Pi]$ using the encoding from Section D.2. The following lemma is shown using the same proof technique as for Lemma 25.

▶ **Lemma 28.** *The following language is recognized by a puzzle:*

$$\{t \otimes [\Pi] : \Pi \textit{ is a family of disjoint unbounded configuration paths in } t\}.$$

We now complete the proof of Lemma 20. Let M_1 be the language from Lemma 28. Let M_2 be the set of trees $t \otimes [\Pi]$ such that every initial configuration in the tree t can reach some configuration appearing in Π. The language M_2 is a regular language of infinite trees, and therefore it is recognized by a nondeterministic parity automaton, which is a special case of a puzzle. Since puzzles are closed under intersection, the language $M_1 \cap M_2$ is recognized by a puzzle. Finally, By Lemma 27, the language $\mathsf{AG}K$ is the projection of $M_1 \cap M_2$ onto the first coordinate. Since languages recognized by puzzles are closed under projection, it follows that $\mathsf{AG}K$ is recognized by a puzzle.

Appendix Part III

Deciding Emptiness of Puzzles

This part of the appendix is devoted to proving Theorem 5, which is restated below.

Theorem 5. *The following conditions are equivalent.*

1. *There is a limit accepting signature graph with fans in $\overline{\mathscr{P}_{\text{fin}}}$,*
2. *There is a limit accepting signature graph with fans in $\overline{\mathscr{P}_{\text{fin}}}$ with finitely many nodes,*
3. *The puzzle has an accepting run.*

Furthermore, given a puzzle one can decide if the conditions hold.

Here is a rough plan of our proof strategy.

- **Implication from 1 to 2**. The key point is that we can design an automaton model, closely resembling alternating automata on graphs, which recognizes limit accepting solution graphs. This automaton model, which is introduced in Section F, shares the following property with alternating automata on graphs: a nonempty automaton accepts a graph with finitely many nodes. Also, this graph can be unraveled into a regular tree. The corresponding section of the paper is Section F.
- **Implication from 2 to 3.** The key point is to get rid of the limits, and replace them by actual finite pieces of runs. The idea is of course to use finite pieces of runs from a sequence approximating the limit, but the implementation of this idea requires some technical effort. This is the part of the paper where we use the notion of bisimulation for metric spaces. The corresponding section of the paper is Section G.
- **Implication from 3 to 1.** The key point is to extract some limits from an arbitrary accepting run of a puzzle. To do this, we use a version of the Ramsey theorem adapted to metric spaces. The corresponding section of the paper is Section H.
- **Deciding if the conditions hold.** The key point in this item is to compute a finite representation of the set $\overline{\mathscr{P}_{\text{fin}}}$. The corresponding section of the paper is Section I.

We begin, however, with a discussion of multisets and their distance, which is presented in the next section.

E Multisets and their distance

In our proof, an important role will be played by multisets. A multiset over a set A is like a subset of A, but some elements can appear more than one time. We extend the notion of Hausdorff distance to closed multisets of elements of a compact metric space A.

Formally, a multiset M over a *domain* A is a mapping

$$M \colon A \to \overline{\mathbb{N}}.$$

The number $M(a)$ is the *multiplicity* of $a \in A$ in M. If the multiplicity of a is positive, then we say that a is an *element* of M. We also say that a has $M(a)$ *occurrences* in M. A multiset M is contained in a multiset N if $M \leq N$ as number-valued functions.

E.1 Metric

Assume that the domain A is a compact metric space. We consider the product metric over the set A^n, defined as

$$\text{distance}((a_1, \ldots, a_n), (b_1, \ldots, b_n)) = \max_{1 \leq i \leq n} \text{distance}(a_i, b_i).$$

Given a multiset M over A, define

$$\mathrm{tuples}_n(M) = \{(a_1, \ldots, a_n) : [\![a_1, \ldots, a_n]\!] \subseteq M\} \subseteq A^n$$

For example, when $M = [\![a, b, b]\!]$ then $\mathrm{tuples}_2(M)$ will contain the tuple (b, b) but not the tuple (a, a). We say a multiset is *closed* if $\mathrm{tuples}_n(M)$ is a closed subset of A^n, for every $n \in \mathbb{N}$, where A^n is equipped with the product metric.

We define the *multiset distance* between two multisets M, N as a discounted supremum over all $n \in \mathbb{N}$ of the Hausdorff distances between the sets $\mathrm{tuples}_n(M)$ and $\mathrm{tuples}_n(N)$:

$$\mathit{distance}(M, N) = \sup_{n\in\mathbb{N}} \frac{1}{n} \mathit{distance}(\mathrm{tuples}_n(M), \mathrm{tuples}_n(N)).$$

▶ **Lemma 29.** *Multiset distance is a metric on closed multisets. The resulting metric space is compact.*

Proof. First we verify that the above distance defines a metric over closed multisets. Symmetry is obvious. The triangle inequality follows from the triangle inequalities for the Hausdorff distances over each of the sets A^n. It therefore remains to show that if $\mathrm{distance}(M, N) = 0$ then $M = N$.

If $\mathrm{distance}(M, N) = 0$ then, by definition, $\mathrm{distance}(\mathrm{tuples}_n(M), \mathrm{tuples}_n(N)) = 0$ for all n. Since both M and N are closed, $\mathrm{tuples}_n(M)$ and $\mathrm{tuples}_n(N)$ are closed subsets of A^n. Because the Hausdorff distance is a distance over closed sets, it follows that $\mathrm{tuples}_n(M) = \mathrm{tuples}_n(N)$ for all n. In particular, if a appears in M at least n times, then it appears in N at least n times. Therefore, $M = N$.

Now we will verify that the set of closed multisets is compact. Assume that $M_1, M_2, \ldots$ is a sequence of closed multisets over A. Since for any $k \in \mathbb{N}$, the Hausdorff metric over A^k is compact, by using a diagonal argument, we may choose a subsequence $M_{n_1}, M_{n_2}, \ldots$ such that for any fixed $k \in \mathbb{N}$, the sequence

$$\mathrm{tuples}_k(M_{n_1}), \mathrm{tuples}_k(M_{n_2}), \ldots$$

of closed subsets of A^k is convergent to some closed subset L_k of A^k. From the sequence of sets $L_1, L_2, \ldots$ it is straightforward to construct a multiset M such that for all $k \in \mathbb{N}$,

$$\mathrm{tuples}_k(M) = L_k.$$

Then, M is a closed multiset and the sequence $M_{n_1}, M_{n_2}, \ldots$ converges to M. ◀

E.2 Union and partitions

If M and N are two multisets over some domain A, trated as mappings

$$M, N : A \to \overline{\mathbb{N}},$$

then we define their *union* as the mapping $M \cup N$, where

$$(M \cup N)(a) = M(a) + N(a).$$

A (finite) *partition* of M is a representation of M as a multiset union

$$M = M_1 \cup M_2 \cup \ldots \cup M_n.$$

▶ **Proposition 30.** *Let A be a compact metric space, and let $\mathbb{M}_c(A)$ denote the set of closed multisets over A. The binary multiset union mapping*

$$\cup\colon \mathbb{M}_c(A)^2 \to \mathbb{M}_c(A)$$

is continuous and open.

Recall that an open mapping is a mapping which maps open sets to open sets.

We omit the proof of the above proposition. It follows via abstract-nonsense topological arguments from a similar fact for the Hausdorff distance for closed sets.

▶ **Lemma 31.** *Let X be a closed multiset in a metric space, partitioned as*

$$X = X_1 \cup \cdots \cup X_n.$$

For every positive $\varepsilon \in \mathbb{R}$ there exists $\delta \in \mathbb{R}$ such that for any set Y with

$$\text{distance}(X, Y) < \delta,$$

there exists a partition

$$Y = Y_1 \cup \cdots \cup Y_n$$

such that

$$\text{distance}(X_1, Y_1), \ldots, \text{distance}(X_n, Y_n) < \varepsilon.$$

Proof. It follows from the previous proposition that the n-ary union

$$\cup\colon \mathbb{M}_c(A)^n \to \mathbb{M}_c(A)$$

is continuous and open. Let us denote this function f.

The argument is purely topological. Let

$$\bar{X} = (X_1, X_2, \ldots, X_n) \in \mathbb{M}_c(A)^n.$$

Then, $f(\bar{X}) = X$. Let $B_\varepsilon \subseteq \mathbb{M}_c(A)^n$ denote the open ball of radius ε around $\bar{X}$. Since B_ε is an open set and the map f is open, it follows that the image $f(B_\varepsilon)$ is an open subset of $\mathbb{M}_c(A)$. Moreover, it obviously contains $f(\bar{X}) = X$. Hence, there is some radius δ, such that the open ball $B'_\delta \subseteq \mathbb{M}_c(A)$ of radius δ around X is contained in $f(B_\varepsilon)$:

$$B'_\delta \subseteq f(B_\varepsilon).$$

In particular, if distance$(X, Y) < \delta$, then $Y \in f(B_\varepsilon)$. Hence, $Y = f(\bar{Y})$ for some $\bar{Y}$ such that

$$\text{distance}(\bar{Y}, \bar{X}) < \varepsilon.$$

◀

F Alternating automata with fan condition

In this section we prove the implication from 1 to 2 in Theorem 5. The proof uses an automaton model which can be used to recognize limit accepting signature graphs. The model is called *alternating automata with fan condition*. It accepts edge labeled multigraphs, whose edges are labeled by a potentially infinite set A, and with a distinguished initial node. Such an automaton $\mathcal{A}$ has two, nearly orthogonal mechanisms for testing if an input multigraph G is accepted: one talking about the outcome of the parity game over the associated arena game($\mathcal{A}, G$) (defined below, in a completely standard way), and the second condition which specifies which fans are allowed in G. The only interaction between these conditions is that in the parity game, the labels of the chosen transitions are required to match the labels of the input graph, and the fan condition specifies the multiset of labels leaving from each vertex of the input graph.

Definition. An *alternating automaton with fan condition* or *fan alternating automaton* is given by the following ingredients:

- A possibly *infinite input* alphabet A.
- A finite set of *states* Q, partitioned into $Q = Q_\forall \cup Q_\exists$,
- An *initial state* $q_I \in Q$,
- A set of *transitions* $\delta \subseteq Q \times (A \cup \{\epsilon\}) \times Q$, where $\epsilon \notin A$ is used for describing *ϵ-transitions*
- A *parity acceptance condition* $\Omega : Q \to \mathbb{N}$.
- A *fan condition*, which is a possibly infinite family $\mathscr{P}$ of multisets over A.

Semantics. Let $\mathcal{A}$ be an automaton as defined above, and consider an edge labeled multigraph G. We first define a parity game, denoted by game($\mathcal{A}, G$). The definition is standard, and does not mention the fan condition.

The game game($\mathcal{A}, G$) is played by two players $\forall$ and $\exists$. The positions in the arena are pairs (p, v) where p is a state of the automaton, and v is a node in the graph G. A position belongs to the player who controls the state in the position. Edges in the arena are of the form

$$(p, v) \to (q, w),$$

such that either $p = q$ and $(p, \epsilon, q) \in \delta$, or there is a σ-labeled edge from v to w in G and $(p, \sigma, q) \in \delta$. The accepting condition is the parity condition, as given by the mapping Ω from the automaton.

Let $\mathcal{A}$ be an alternating automaton $\mathcal{A}$ with fan condition $\mathscr{P}$, and let G be a multigraph with edges labeled by A. Then $\mathcal{A}$ *accepts* G from node v if the following conditions hold.

- Player $\exists$ has a winning strategy in game($\mathcal{A}, G$), from the position (q_I, v),
- For every vertex x in G, the fan of x belongs to the family $\mathscr{P}$.

Recognizing limit accepting signature graphs. The automaton model is designed to recognize limit accepting signature graphs with fans in $\overline{\mathscr{P}_{\text{fin}}}$, as stated in the following straightforward lemma:

▶ **Lemma 32.** *There is an alternating automaton $\mathcal{A}$ with fan condition $\mathscr{P}_{\text{lim}}$ which accepts precisely the set of limit accepting signature graphs with fans in $\overline{\mathscr{P}_{\text{fin}}}$.*

▶ Remark. We skip the description of the alternating automaton $\mathcal{A}$. Note however, that the automaton $\mathcal{A}$ can be designed in such a way that it ignores the actual label of each edge of the input graph, but only cares about its *path type* – a finite information defined as follows.

The *path type* of a path signature σ, denoted path-type(σ) is specified by the following components.

- The source and target states of σ,
- The set of states appearing in σ,

and for each counter $c \in C$,

- A bit indicating if σ cuts c,
- A bit indicating if σ resets c,
- A bit indicating if σ increments c,
- A bit indicating if σ contains an ω on counter c.

In the end, we will be interested in the decision problem of testing a fan alternating automaton for emptiness. It is not immediately clear how formalize the decision problem for fan alternating automata, because such an automaton has an infinite description. We discuss this issue in the following section.

F.1 Decidability issues

In this section, we show how fan alternating automata can be finitely presented, so that they can be input by an algorithm for emptiness. The idea is to convert any fan alternating automaton $\mathcal{A}$ into a fan alternating automaton $[\mathcal{A}]$ with a finite input alphabet, but with equivalent emptiness. We use the simple observation that as far as game$(\mathcal{A}, G)$ is concerned, the actual label σ of the edge in G is irrelevant – what matters, is the transition relation which σ induces in $\mathcal{A}$.

The abstraction of an automaton. Let $\mathcal{A}$ be a fan alternating automaton. We define a new automaton, which we call $[\mathcal{A}]$. This automaton $[\mathcal{A}]$ inherits from $\mathcal{A}$ the states Q, the partition $Q = Q_\exists \cup Q_\forall$, the initial state q_I, the set of ϵ-transitions, the parity acceptance condition Ω. The only differences between $\mathcal{A}$ and $[\mathcal{A}]$ are on the input alphabet, the set of edge transitions, and the fan condition.

First, for a label $\sigma \in A$, we define the following set.

$$\text{type}(\sigma) \overset{\text{def}}{=} \{(p, q) : \ (p, \sigma, q) \in \delta\} \in P(Q \times Q).$$

The input alphabet of $[\mathcal{A}]$ is the finite set

$$[A] = P(Q \times Q).$$

Observe that the input alphabet of $[\mathcal{A}]$ does not depend on the input alphabet of $\mathcal{A}$. The transition relation $[\delta]$ in $[\mathcal{A}]$ is defined in a tautological way:

$$[\delta] \overset{\text{def}}{=} \{(p, X, q) : \ X \in [A], \ (p, q) \in X\}.$$

We now define the fan condition in $[\mathcal{A}]$. Let A be the input alphabet of $\mathcal{A}$. If Σ is a multiset over A, then we may consider the multiset image of Σ under the mapping type: $A \to [A]$, denoted type(Σ).

If N is a multiset and $n \in \mathbb{N}$ a number, then we say that N has threshold n if no element in N has a multiplicity larger than n. For a multiset M and number $n \in \mathbb{N}$, we write $\text{trim}_n(M)$ for the largest multiset which is contained in M and has threshold n.

For example, if M contains two occurrences of an element a and ∞-many occurrences of an element b, then $\text{trim}_5(M)$ contains two occurrences of a and 5 occurrences of b.

The fan condition $[\mathscr{P}]$ of $[\mathcal{A}]$ is defined as follows. A multiset M over $[A]$ belongs to $[\mathscr{P}]$ if and only if there is some $\Sigma \in \mathscr{P}$ such that

$$\text{trim}_n(M) = \text{trim}_n(\text{type}(\Sigma)),$$

where $n = |Q|$ is the number of states of $\mathcal{A}$.

This completes the definition of the automaton $[\mathcal{A}]$. Note that the fan condition $[\mathscr{P}]$ is determined by the following family of of multisets over $[A]$:

$$\{\text{trim}_n(\text{type}(\Sigma)) : \quad \Sigma \in \mathscr{P}\}.$$

Since $[A]$ and n are finite, there are only finitely many possible multisets with multiplicities up to n, and so $[\mathscr{P}]$ has a finite description. Since all the remaining ingredients of $[\mathcal{A}]$ are finite, the abstraction of an automaton can be used as an input to a decision procedure.

▶ **Proposition 33.** *$\mathcal{A}$ is nonempty if and only if $[\mathcal{A}]$ is nonempty.*

Proof. The left-to-right implication is straightforward, and does not depend on the chosen threshold $n = |Q|$. We first present this implication. For a graph G accepted by $\mathcal{A}$, we define a graph $[G]$ accepted by $[\mathcal{A}]$ in an obvious way: it has the same vertices as G, and the edge labels are transformed via the mapping $\text{type}: A \to [A]$. Then, $\text{game}(\mathcal{A}, G)$ is virtually identical to $\text{game}([\mathcal{A}], [G])$ – the only difference is that the labels of G no longer come from an infinite alphabet, but directly describe the transition relation in $\mathcal{A}$. Therefore, the player $\exists$ either wins in both games, or looses in both games.

It remains to show that $[G]$ satisfies the fan condition $[\mathscr{P}]$. However, if x is a vertex, then the fan of x in $[G]$ is the image of the fan of x in G under the mapping type, and by assumption, this fan belongs to $\mathscr{P}$. It follows from the definition that the fan of x in $[G]$ belongs to $[\mathscr{P}]$.

For the right-to-left implication, we proceed similarly. However, we need the following lemma. Let $[G]$ be a graph accepted by $[\mathcal{A}]$, and x its vertex. We say that the fan $[\Sigma]$ of x is *liftable*, if there is a $\Sigma \in \mathscr{P}$ such that

$$\text{type}(\Sigma) = [\Sigma].$$

▶ **Lemma 34.** *If $[\mathcal{A}]$ accepts some multigraph $[G]$, it also accepts a multigraph with liftable fans.*

Proof. We need to replace every fan in $[G]$ by a liftable fan. This may require removing edges, or adding edges with a specified label. Removing edges is more difficult, as they may be required by the winning strategy of the player $\exists$.

Suppose that a graph $[G]$ is accepted by $[\mathcal{A}]$. Then, there exists a positional winning strategy for $\exists$ in $\text{game}([\mathcal{A}], [G])$, which is a function

$$S: Q_\exists \times \text{nodes}([G]) \to Q \times \text{edges}([G]).$$

For a node x we say an outgoing edge e is *important* if

$$S(p, x) = (q, e) \qquad \text{for some } p \in Q_\exists \text{ and } q \in Q.$$

Clearly, there are at most $|Q_\exists|$ important edges for any given node x.

Let x be a node of $[G]$. Let $[\Sigma]$ be the fan of x. By assumption that $[\Sigma] \in [\mathscr{P}]$, there exists some Σ such that for $n = |Q|$,

$$\text{trim}_n([\Sigma]) = \text{trim}_n(\text{type}(\Sigma)).$$

We will alter the edges leaving from x, so that the fan $[\Sigma]$ is liftable to Σ.

Let $[\sigma]$ be a label in $[\Sigma]$, and let $[\Sigma]_{[\sigma]}$ denote the multiset of all occurrences of $[\sigma]$ in $[\Sigma]$. Similarly, let $\Sigma_{[\sigma]}$ be the multiset of all elements in Σ which have type $[\sigma]$. $[\Sigma]$ is liftable to Σ, if for every $[\sigma]$, the size of $[\Sigma]_{[\sigma]}$ is equal to the size of $\Sigma_{[\sigma]}$. In general, however, equality might not hold. We consider the three following possibilities.

$[\Sigma]_{[\sigma]}, \Sigma_{[\sigma]}$ are of the same size In this case, no edges with label $[\sigma]$ need to be added, since there is a bijection between edges leaving x with label $[\sigma]$, and the elements in Σ with type $[\sigma]$.

Note that in the remaining cases, when $|[\Sigma]_{[\sigma]}| \neq |\Sigma_{[\sigma]}|$, from the assumption on Σ it follows that both $[\Sigma]_{[\sigma]}$ and $\Sigma_{[\sigma]}$ have at least $|Q|$ elements.

$[\Sigma]_{[\sigma]}$ is larger than $\Sigma_{[\sigma]}$ Suppose that the label $[\sigma]$ has multiplicity in $[\Sigma]$ larger than $|\Sigma_{[\sigma]}|$. Then, this multiplicity is larger than $|Q|$. In particular, x has some outgoing edge e with label $[\sigma]$ which is not important. It follows that the edge e can be removed from the graph $[G]$, maintaining the fact that player $\exists$ wins in game($[\mathcal{A}], [G]$). This way, we may remove outgoing edges from x as long as it has more than $|\Sigma_{[\sigma]}|$ outgoing edges with label $[\sigma]$.

Since the fan condition $[\mathscr{P}]$ does not care about multiplicities of edges above the threshold $|Q|$, it follows that the obtained graph still has fans in $[\mathscr{P}]$, and therefore is accepted by $[\mathcal{A}]$.

$[\Sigma]_{[\sigma]}$ is smaller than $\Sigma_{[\sigma]}$ Then all we need to do is to increase the multiplicity of some edge leaving from x with label $[\sigma]$, so that altogether there are $|\Sigma_{[\sigma]}|$ edges leaving from x with label $[\sigma]$. This operation does not affect the parity game. Similarly as in the previous case, the obtained graph still satisfies the fan condition. Hence, the obtained graph is accepted by $[\mathcal{A}\ ]$.

After performing the above operation with each label $[\sigma]$, the fan of x becomes liftable. We repeat the process for every node in $[G]$. ◀

We return to the proof of the right-to-left implication of the proposition. Let $[G]$ be a multigraph accepted by $[\mathcal{A}]$, with liftable fans. We construct the graph G similarly as in the left-to-right implication, this time lifting the labels of the edges from the alphabet $[\Sigma]$ to the alphabet Σ, in a way which is consistent with types. Since $[G]$ is liftable, this can be done so that the fans belong to $\mathscr{P}$. Moreover, such an operation does not alter the parity game. Therefore, the obtained labeled multigraph G is accepted by $\mathcal{A}$. ◀

Emptiness for fan alternating automata. In this section, we prove that emptiness is decidable for fan alternating automata.

▶ **Proposition 35.** *The following problem is decidable:*

- *Input: a fan alternating automaton $\mathcal{A}$, given by $[\mathcal{A}]$.*
- *Question: Is $\mathcal{A}$ nonempty?*

Furthermore, if $\mathcal{A}$ is nonempty then it accepts a graph with finitely many nodes.

Proof. One can write an MSO formula $\varphi_{\mathcal{A}}$ over edge labeled trees, such that $\varphi_{\mathcal{A}}$ is satisfied exactly in the trees accepted by $[\mathcal{A}]$. Satisfiability for MSO formulas is decidable by Rabin's theorem. Furthermore, every satisfiable formula has a model, which is a regular tree. Finally, the translation of models from $[\mathcal{A}]$ to $\mathcal{A}$, as described in the proof of Proposition 33, preserves regularity of trees. From a regular tree, one can use bisimulation to obtain a graph with finitely many nodes. ◀

Proposition 35 combined with Lemma 32, yields the implication from 1 to 2 in Theorem 5. Also, Proposition 35 combined with Lemma 32 opens the way for the decidability part in Theorem 5, as stated in the following lemma.

▶ **Lemma 36.** *Deciding if there exists a limit accepting signature graph with fans in $\overline{\mathscr{P}_{\text{fin}}}$ reduces to computing, for a given value $n \in \mathbb{N}$, of the following family of multisets of path types with threshold n:*

$$\{\text{trim}_n(\text{path-type}(\Sigma)) : \ \Sigma \in \overline{\mathscr{P}_{\text{fin}}}\}. \tag{1}$$

Proof. Let $\mathcal{A}$ be the automaton described in Lemma 32. As observed in the remark following the lemma, in the automaton $\mathcal{A}$, for any path signature $\sigma \in A$, $\text{type}(\sigma)$ is determined by $\text{path-type}(\sigma)$. It follows that the family $[\overline{\mathscr{P}_{\text{fin}}}]$ can be computed from the family (3), where n is the number of states of $\mathcal{A}$. ◀

Computing the family (3) is the subject of Section I.

G From a limit accepting signature graph to an accepting run

In this section, we prove implication from 2 to 3 in Theorem 5. Let G be a limit accepting signature graph with fans in $\mathscr{P}_{\text{lim}}$, which has finitely many nodes, as in condition 2 of Theorem 5. Our goal is to find an accepting run of the puzzle, as stated in condition 3 of Theorem 5.

The general idea is quite natural: we unravel the graph G, and as we go deeper in the unfolding, we replace fans from $\mathscr{P}_{\text{lim}}$ by closer and closer approximations from $\mathscr{P}_{\text{fin}}$. However, in order to present the precise construction, we need to resolve a number of technical details.

Unravelling First, we perform a straightforward unravelling of the graph G with finitely many nodes, obtaining a limit accepting signature graph T such that:

- T is *acyclic*, so that no path visits the same node twice,
- T is *rooted*, so that there is a distinguished *root node*, from which all other vertices are reachable,
- T is *finitely branching*, so that every node has only finitely many (immediate) successors (note that a node might have infinitely many outgoing edges, due to parallel edges);
- Finally, the fans of T form a finite subset of $\mathscr{P}_{\text{lim}}$.

The construction of T is proceeds as expected. Let x be the root node of G. The nodes in T are sequences $x_1 \cdots x_n \in \text{nodes}(G)^+$ subject to $x_1 = x$, and such that there is some edge in G from x_i to x_{i+1} for $i = 1, \dots, n-1$. The edges are defined by

$$\{(x_1 \cdots x_n, \sigma, x_1 \cdots x_n x_{n+1}) : \ (x_n, \sigma, x_{n+1}) \text{ is an edge in } G\}.$$

The graph T is finitely branching because G has finitely many nodes. Clearly, T is a limit accepting signature graph. Also, it is easy to see that the fans of T are a subset of the fans of G, in particular the fans of T are a finite subset of $\mathscr{P}_{\text{lim}}$.

Fix the graph T satisfying the above conditions for the rest of this section. What remains to be done is to replace the fans in T by closer and closer approximations from $\mathscr{P}$, and then substitute them by actual factors, finally obtaining a run of of the puzzle. To describe these technical details, we use a notion of bisimulation that is adapted to converging sequences, which we now describe.

G.1 Converging bisimulation

Converging bisimulation. Let Σ be a metric space, possibly infinite. Consider a graph with edge labels from Σ. We define an infinite game that is played by players Spoiler and Duplicator. A parameter of the game is a sequence

$$r : \mathbb{N} \to \mathbb{R}_{>0}$$

of positive real numbers, which is called the *convergence rate.* We are interested in the case when r tends to 0. We will also sometimes mention the degenerated convergence rate, which is constantly equal to 0.

The game is played in rounds. At the beginning of each round, there is a pair of nodes v_0, v_1. One round is played as follows. First, Spoiler chooses $i \in \{0,1\}$ and an edge e_i that leaves node v_i. Then, Duplicator responds with an edge e_{1-i} that leaves node v_{1-i}. Duplicator's response has to be such that the labels of the two edges e_0, e_1 are at distance at most $r(n)$, where n is the number of the current round. If Duplicator cannot find such an edge, the game is terminated and Spoiler wins. Otherwise, the game proceeds to a new round and the new nodes being the targets of e_0, e_1. If the game lasts forever, Duplicator wins.

We say that two nodes v_0, v_1 are *r-bisimilar* if Duplicator wins the game with initial nodes v_0, v_1 and convergence rate r. The nodes are *converging bisimilar* if they are r-bisimilar for some rate that converges to 0. Converging bisimilarity is an equivalence relation. If the nodes are from different graphs, we implicitly work in the disjoint union of the graphs.

The classical notion of bisimulation is recovered as a special case of this one, in at least two ways. One way is to use a degenerate rate that is constantly equal to 0. We call this 0-bisimilarity. Another way is to use the discrete metric on the set of edge labels, which gives distance 1 to every two distinct edge labels. Then, r-bisimilarity is the same as bisimilarity for any convergence rate that uses values smaller than 1.

G.2 Acyclic graphs

In this section, we show how to approximate acyclic signature graphs with fans in $\mathscr{P}_{\text{lim}}$, by acyclic signature graphs with fans in $\mathscr{P}_{\text{fin}}$.

▶ **Lemma 37.** *Let T be a signature graph with fans in $\mathscr{P}_{\text{lim}}$, which is finitely branching, acyclic and rooted. Let r be a non-increasing convergence rate. There is a signature graph T' with fans in $\mathscr{P}_{\text{fin}}$, and the same nodes as T, such that for every node x, x in T is r-bisimilar to x in T'.*

Proof. By the statement of the lemma, the nodes of T' are the same as the nodes of T, so we only need to define the edges.

Consider a node x in T, and let $|x|$ denote the distance of x from the root. Let $\Sigma \in \mathscr{P}_{\text{lim}}$ be the fan of x, and let $x_1, \ldots, x_n$ be its successors. (We use the assumption on finite branching here.) For $i \in \{1, \ldots, n\}$, let Σ_i be the multiset of labels on edges that connect x to x_i. The multisets $\Sigma_1, \ldots, \Sigma_n$ form a partition of the fan Σ. Choose ε to be the value $r(|x|)$ assigned by the rate r to the distance of x from the root. Apply Lemma 31 to the partition $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_n$ and ε, yielding a number δ. By definition of $\mathscr{P}_{\text{lim}}$, we know that there is some factor profile

$$\Sigma' \in \mathscr{P}_{\text{fin}} \qquad \text{such that} \qquad \text{distance}(\Sigma', \Sigma) < \delta.$$

By Lemma 31, there is some partition

$$\Sigma' = \Sigma'_1 \cup \cdots \cup \Sigma'_n \qquad \text{such that distance}(\Sigma_i, \Sigma'_i) < r(|x|), \text{ for all } i \in \{1, \dots, n\}.$$

We define the edges leaving x in T' as follows. For each $i = 1, \dots, n$ and each occurrence of a label σ in the multiset Σ'_i, we create an edge from x to x_i labeled by σ. By definition, the fan of x in T' is $\Sigma' \in \mathscr{P}_{\text{fin}}$.

The root of the graph T' is r-bisimilar to the root of T: Duplicator's strategy is to have the same node in both graphs. The same holds for other nodes, because the rate is non-increasing. ◀

G.3 An intermediate acyclic signature graph

Recall that we have fixed an acyclic limit signature graph T, obtained from unravelling a limit signature graph G with finitely many nodes. Our goal in Section G is to find an accepting run of the puzzle, as stated in condition 3 of Theorem 5.

As an intermediate step, we will consider the rooted acyclic signature graph T', obtained by applying Lemma 37 to the signature graph T, with a suitable convergence rate r. We begin by defining r. We then show that the graph T' has certain good properties.

The convergence rate. In order to apply Lemma 37 to T, we need to specify some convergence rate which will be appropriate for our needs later on.

▶ **Lemma 38.** *For every $n \in \mathbb{N}$, there exists a distance $\varepsilon_n \in \mathbb{R}$ such that for every path signatures at distance at most ε_n, the path signatures agree on the following information:*

1. *The source and target states.*
2. *The maximal rank, under the parity condition, that appears in the path signature.*
3. *The set of counters that are increment/cut/reset at least once.*
4. *For every counter c, the value of the counter, counted up to threshold n.*

Proof. All of the above properties are regular properties of path signatures. ◀

Let the sequence $\varepsilon_1, \varepsilon_2, \dots$ be as in Lemma 38. Without loss of generality we assume that the sequence is decreasing. The convergence is defined by $r(n) = \varepsilon_n$.

Applying Lemma 37. Apply Lemma 37 to the acyclic signature graph T and the convergence rate r defined above. Let T' be the resulting signature graph, which has fans in $\mathscr{P}_{\text{fin}}$. The following lemmas prove some good properties of T'.

The depth of a node in a rooted acyclic graph is its distance from the root. The notions of depth coincide in T and T', so we can simply write $|x|$ to indicate the depth, without indicating which of the graphs T or T' we have in mind.

▶ **Lemma 39.** *Let π be a path in T, with source and target nodes x, y. There exists a path π' in T' with the same source and target nodes x, y, and such that*

$$\text{distance}(\text{signature}(\pi), \text{signature}(\pi')) \leq \varepsilon_{|x|}.$$

Proof. By bisimilarity. ◀

▶ **Lemma 40.** *Every path in T' satisfies the parity condition.*

Proof. Consider an infinite path π' in T' that begins in the root and visits edges $e'_1, e'_2, \ldots$. Apply Lemma 39 to each edge e'_n, yielding an edge e_n with

$$\text{distance}(\text{signature}(e_n), \text{signature}(e'_n)) \leq \varepsilon_n \leq \varepsilon_1.$$

By Lemma 38, it follows that for every n, the maximal ranks, under the parity condition, are the same for edges e_n in T and e'_n in T'. Since every path in T satisfies the parity condition, it follows that the path $\pi = e_1 e_2 \ldots$ satisfies the parity condition, and therefore so does π'. ◀

Let G be a signature graph, e.g. $G = T$ or $G = T'$, x a node in G, and c a counter. We define

$$\text{cutzone}(G, x, c) = \{y \in \text{nodes}(G) : \text{there exists a path from } x \text{ to } y \text{ that does not cut } c\}.$$

▶ **Lemma 41.** *Let x be a node T, or equivalently in T'. For every counter c,*

$$\text{cutzone}(T, x, c) = \text{cutzone}(T', x, c) \qquad \textit{and} \qquad c \in \mathsf{U}(T, x) \iff c \in \mathsf{U}(T', x)$$

Proof. By items 3 and 1 of Lemma 38, and r-bisimilarity. ◀

▶ **Lemma 42.** *Let x be a node in T'. For every counter $c \in \mathsf{U}(T', x)$, and for every $N \in \mathbb{N}$, some path leaving x in T' has value at least N on counter c.*

Proof. Let x, c, N be as in the statement of the lemma. We need to find some path in T' leaving x has value at least N on counter c. Because c belongs to $\mathsf{U}(T', x) = \mathsf{U}(T, x)$ and T is limit accepting, the set

$$\text{cutzone}(T, x, c)$$

contains arbitrarily deep nodes that have an outgoing edge with ω on counter c. Pick a node $y \in \text{cutzone}(T, x, c)$ that has depth at least N, and such that some edge leaving y is labeled by a signature, call it σ, which has ω on counter c. By Lemma 39, there must be some edge leaving y in T', which is labeled by a signature, call it σ', such that

$$\text{distance}(\sigma, \sigma') \leq r(|y|) \leq \varepsilon_N.$$

By item 4 of Lemma 38, it follows that the value of σ' on counter c is at least N. By Lemma 41, the path from x to y in T' that does not cut counter c. ◀

▶ **Lemma 43.** *Let x be a node in T'. For every counter $c \notin \mathsf{U}(T', x)$, there is some $N \in \mathbb{N}$ such that all paths leaving x in T' have value at most N on counter c.*

Before we prove Lemma 43, we show a sublemma. This sublemma uses the fact that T originates from a graph with finitely many nodes.

▶ **Lemma 44.** *Let x and c be as in the assumptions of Lemma 43. There is a bound $M \in \mathbb{N}$ such that every edge, in T', that leaves a node from* $\text{cutzone}(T', x, c)$ *has value at most M on counter c.*

Proof. Because T is a limit accepting and $c \notin \mathsf{U}(T, x)$, it follows that there are finitely many nodes $y \in \text{cutzone}(T, x, c)$ that have an outgoing edge, in T, whose label has ω on counter c. (We use the assumption on T being finitely branching, and König's lemma.) Let Y be the set of these nodes y. For a node y in T, let Σ_y be the fan of y. Let Σ be the union of all fans, ranging over nodes in

$$\mathrm{cutzone}(T, x, c) - Y.$$

Because the tree T is obtained from unfolding a graph with finitely many nodes, there are finitely many possible fans. In particular, Σ is a finite union of fans. Each fan in T is a closed set (in the topological sense), as it belongs to $\mathscr{P}_{\lim}$. Therefore Σ is closed as a finite union of closed sets. By construction, Σ does not contain any path signature that has ω on counter c. Because Σ is compact, it follows that there is constant $K \in \mathbb{N}$ such that every path signature in Σ has value at most K.

Consider now a signature σ' that labels an edge in T', which leaves a node $y \in \mathrm{cutzone}(T', x, c)$ that has depth at least K, and which does not belong to Y. Using Lemma 39, we show that σ' has value at most K on counter c. Finally, we are left with the edges that leave nodes $y \in Y$, or nodes in $\mathrm{cutzone}(T', x, c)$ at depth at most K. But there are finitely many such edges, and therefore they have some maximal value on counter c. We choose M to be bigger than K and this maximal value. ◀

(of Lemma 43). Let $|G|$ be the number of nodes in the signature graph, which was used to define T. There are two possible cases for every node x, concerning paths in the tree T:

- Some path in $\mathrm{cutzone}(T, x, c)$ increments c infinitely often.
- Every finite path in $\mathrm{cutzone}(T, x, c)$, either resets c or passes through at most $|G|$ distinct edges where c is incremented.

Because counter c from the statement of the lemma belongs to $\mathsf{U}(T, x)$, and because T is limit accepting, the second case must hold. By bisimilarity, the second case must also hold in the tree T'. By applying Lemma 44, we see that every path in T' in $\mathrm{cutzone}(T', x, c)$ has value at most $K \cdot M$. ◀

G.4 A run of the puzzle

In this section, we complete the proof of implication from 2 to 3 in Theorem 5, by constructing an accepting run t of the puzzle.

Suppose that t is a run of the puzzle, and $\mathcal{F}$ is a family of finite factors that partitions the nodes of t. We define a signature tree $t/\mathcal{F}$ as follows. The nodes of t are the factors from $\mathcal{F}$. The tree structure is inherited from the factorization: in $t/\mathcal{F}$ there is an edge from a factor $F \in \mathcal{F}$ to a factor $G \in \mathcal{F}$ if the root of G is a port of F. The label of this edge is the signature of the path, in t, that goes from the root of F to the root of F.

▶ **Lemma 45.** *There exists a run t of the puzzle, together with a family $\mathcal{F}$ of finite factors that partitions the nodes of t, such that T' is 0-bisimilar to $t/\mathcal{F}$.*

Proof. We create a run t of the puzzle in the natural way, by replacing every node x of T' by a factor whose signature is the fan of x in T'. This can be done, because the fans in T' are from $\mathscr{P}_{\mathrm{fin}}$. ◀

▶ **Lemma 46.** *The run t is accepting.*

Proof. The parity condition is satisfied thanks to Lemma 40. Consider now a node x in t. Suppose first that x is the root of a factor $F \in \mathcal{F}$. Using bisimilarity, and Lemmas 42 and 43, we show that for every counter, $c \in \mathsf{U}_x$ if and only if there paths leaving x with unbounded value on counter c. For a node x that is not the root of a factor $F \in \mathcal{F}$, we use the sanity condition. ◀

H From an accepting run to a limit accepting signature graph

In this section, we prove implication from 3 to 1 in Theorem 5. Let ρ be an accepting run of the puzzle. We fix ρ for the rest of Section H. Our goal is to produce a limit accepting signature graph with fans in $\mathscr{P}_{\lim}$.

▶ **Lemma 47.** *Let x be a node of ρ, and let $\varepsilon \in \mathbb{R}_+$. There is a fan $\Sigma \in \mathscr{P}_{\lim}$ and a finite factor F in ρ with root x such that*

$$\text{distance}(\Sigma, \text{signature}(F)) \leq \varepsilon$$

and for every counter c, c belongs to $\mathsf{U}(\rho, x)$ if and only if some path signature in Σ has value ω on counter c.

Proof. Let $F_1 \subseteq F_2 \subseteq \ldots$ be a sequence of factors, such that each has root x and every descendant of x eventually belongs to some F_n. By compactness, one can choose the sequence so that there is some limit

$$\Sigma = \lim_{n\to\infty} \text{signature}(F_n).$$

We choose this limit to be Σ from the statement of the lemma, and we choose F to be F_n for n sufficiently large. To finish the proof of the lemma, we use observe that, by the definition of the distances on factor signatures and path signatures, the following conditions are equivalent:

- Σ contains some path signature with value ω on counter c.
- For every $n \in \mathbb{N}$, there is some path leaving x that has value at least n on counter c.

The second condition is equivalent to $c \in \mathsf{U}(\rho, x)$. ◀

Choose ε to be ε_1, as stated in Lemma 38. Fix this choice for the rest of this section. Apply Lemma 47 to every node $x \in \text{nodes}(\rho)$ and to ε, yielding

$$\{\Sigma_x\}_{x\in\text{nodes}(\rho)} \qquad \text{and} \qquad \{F_x\}_{x\in\text{nodes}(\rho)}. \tag{1}$$

▶ **Lemma 48.** *There exists a signature graph G, and a mapping*

$$f : \text{nodes}(G) \to \text{nodes}(\rho)$$

with the following properties

1. *The root of G is mapped by f to the root of ρ.*
2. *For every node $x \in \text{nodes}(G)$, the fan of x in G is $\Sigma_{f(x)}$, as defined in (1).*
3. *For every finite path π in G, there exists a finite path in ρ, call it $f(\pi)$ such that:*
   - *The source node of π is mapped by f to the source node of $f(\pi)$, likewise for target.*
   - *The distance between the signatures of π and $f(\pi)$ is at most ε.*

Proof. The graph G is going to be rooted and acyclic, so it makes sense to talk about the depth of a node in G. We define the nodes and edges of G by induction on their depth, together with the mapping f. In the induction base, we begin with a root node in G, which is mapped by f to the root of ρ. This guarantees condition 1 in the statement of the lemma.

Suppose that we have already defined a node x in G, and the value $f(x)$. In the definition of the children of x, and of the function f, we will use the fan $\Sigma_{f(x)}$ and the corresponding factor $F_{f(x)}$. For every path signature $\sigma \in \Sigma_{f(x)}$, we create a new child of x, call it x_σ, which is connected to x by an edge with label σ. This guarantees that the fan of x in G is equal to $\Sigma_{f(x)}$, and therefore satisfies condition 2 in the statement of the lemma. By the assumption on the signature $F_{f(x)}$ being a close approximation of $\Sigma_{f(x)}$, we know that there is a port, call it y_σ, in the factor $F_{f(x)}$, such that signature of the path from $f(x)$ to y_σ is at distance at most ε from σ. We define $f(x_\sigma)$ to be y_σ. This guarantees condition 3 in the statement of the lemma. ◀

Let G be the signature graph from the above lemma. By condition 2, all fans in G are from $\mathscr{P}_{\text{lim}}$. In the rest of Section H, we show that G is limit accepting, thus finishing the implication from 3 to 1 in Theorem 5. Fix G for the rest of the section. Before proving that G satisfies the four conditions of a limit accepting signature graph, we state a lemma.

▶ **Lemma 49.** *Let x be a node in G and c a counter. The following conditions are equivalent*

- $c \in \mathsf{U}(G, x)$*;*
- $c \in \mathsf{U}(\rho, f(x))$*;*
- *The fan of x in G contains an edge with ω on counter c.*

Proof. Let π be any path leaving x in G. By definition of path signatures, the path signature of π stores the state in x. By condition 3 of Lemma 48, there is a path $f(\pi)$ in ρ that leaves $f(x)$, whose signature satisfies

$$\text{distance}(\text{signature}(\pi), \text{signature}(f(\pi))) \leq \varepsilon.$$

By choice of ε, the path signatures are close enough to ensure that they have the same source state. It follows that the state in node $f(x)$ in the accepting run ρ is the same as the state in node x in G, and therefore also the first two conditions in the statement of the lemma are equivalent.

By definition of G, the fan of x is $\Sigma_{f(x)}$. By Lemma 47, it follows that condition 3 is equivalent to condition 2. ◀

We now show that G satisfies the four conditions of a limit accepting signature graph.

Condition 1 Condition 1 of the definition of a limit accepting signature graph says that the root node is labeled by the initial state of the puzzle. The state in the root of G is the same as in the root of ρ, and both are initial.

Condition 2 Condition 2 of the definition of a limit accepting signature graph says that all nodes are reachable from the root node. This is how the graph G was defined.

Condition 3 Condition 3 of the definition of a limit accepting signature graph says that the parity condition is satisfied on every infinite path. Let π be an infinite path in G. Let $k \in \mathbb{N}$ be the maximal rank, under the parity acceptance condition, which appears infinitely often in π. We want to prove that k is even. Decompose π into finite paths

$$\pi = \pi_0\pi_1\pi_2 \ldots$$

so that k is the maximal parity rank that is seen in the paths $\pi_1, \pi_2, \ldots$. Apply the function f, as defined in item (3) of Lemma 48, to these finite paths, yielding a path

$$f(\pi) = f(\pi_0)f(\pi_1)f(\pi_2) \ldots \tag{2}$$

in the accepting run ρ. For every n, the distance between the signatures of the paths π_n and $f(\pi_n)$ is at most ε. By choice of ε, we know that the maximal parity rank appearing in π_n and $f(\pi_n)$ is the same, in particular it is k for $n \geq 1$. Since ρ is an accepting run, it follows that k is even.

Condition 4 Condition 4 of the definition of a limit accepting signature graph says that for every node x and counter c, counter c belongs to $\mathsf{U}(G, x)$ if and only if

(4a) There is an infinite path from x, such that every prefix of the path can be extended to a finite path that does not cut c, and reaches a node whose fan contains an edge with ω on counter c; or

(4b) There is an infinite path from x, which does not cut c, resets it finitely often, and increments it infinitely often.

We will show that the constructed graph G has an even stronger property:

> If $c \in \mathsf{U}(G, x)$, then condition (4a) holds, and if $c \notin \mathsf{U}(G, x)$, then neither condition (4a) nor condition (4b) holds.

▶ Remark. The above, stronger property could have been used instead of the Condition 4 in a more restrictive definition of a limit accepting signature; however, it is more complicated to state than the chosen definition, and we do not need such a strong property in the proofs of the implications from 1 to 2, nor from 2 to 3 in Theorem 5.

- In the top down implication we do not use condition (4b). We simply prove that $c \in \mathsf{U}(G, x)$ implies condition (4a). Let x be a node in G, and c a counter in $\mathsf{U}(G, x)$. We build an infinite sequence of nodes

  $$x = x_1, x_2, \ldots \in \text{nodes}(G)$$

  such that for every n, counter c belongs to $\mathsf{U}(G, x_n)$ and there is an edge in G from x_n to x_{n+1} that does not cut counter c. If we construct such a sequence, then we have proved condition (4a), because by Lemma 49, the fan of every node on the path contains an edge whose signature has value ω on counter c.
  The nodes on the path are defined by induction. The induction base for $n = 1$ is immediate. Suppose that we have constructed the sequence up to node x_n. Consider the factor $F_{f(x_n)}$ in the run ρ. Because

  $$c \in \mathsf{U}(G, x_n) = \mathsf{U}(\rho, f(x_n)),$$

  there must be some port y of the factor $F_{f(x_n)}$, such that the path from $f(x_n)$ to y does not cut counter c, and $c \in \mathsf{U}(\rho, y)$. Let q be the sate in node y of ρ. Because

  $$\text{distance}(\text{signature}(F_{f(x_n)}), \Sigma_{f(x_n)}) \leq \varepsilon$$

  then there must be some $\sigma \in \Sigma_{f(x_n)}$ that does not cut counter c and reaches state q. By construction, the fan of x_n in G is $\Sigma_{f(x_n)}$. Choose x_{n+1} to be any child of x_n that is the target of an edge with label σ.
- We now prove that (4a) implies that $c \in \mathsf{U}(G, x)$. If there is an infinite path as in (4a), then in particular there is a finite path π in G that begins in x and reaches a node, call it y, whose fan contains a path signature with ω on counter c. By Lemma 49, it follows that $c \in \mathsf{U}(G, y)$. Consider the path $f(\pi)$, which goes in ρ from $f(x)$ to $f(y)$. Because the signatures of π and $f(\pi)$ are close, it follows that the path $f(\pi)$ does not cut counter c. Since ρ is an accepting run, and $c \in \mathsf{U}(\rho, f(y))$, it follows that also $c \in \mathsf{U}(\rho, f(x))$. Therefore, by Lemma 49, $c \in \mathsf{U}(G, x)$.

- We now prove that (4b) implies that $c \in \mathsf{U}(G,x)$. Suppose that π is an infinite path in G that begins in x, never cuts counter c, resets c finitely often and increments it infinitely often. Using the same argument as for the parity condition, we prove that there is such a path in ρ that begins in $f(x)$. Since ρ is an accepting run, it follows that c belongs to $\mathsf{U}(\rho, f(x))$, which is the same set as $\mathsf{U}(G,x)$.

This completes the implication from 3 to 1 in Theorem 5.

I Deciding if there exists a limit accepting signature graph

In this section, we prove the last part of Theorem 5, which says that given a puzzle $\mathcal{P}$, one can decide if there is a limit accepting signature graph with fans in $\overline{\mathscr{P}_{\text{fin}}}$.

Recall that by Lemma 36, the above decision problem reduces to computing the following family of multisets of path types for a given threshold r:

$$[\overline{\mathscr{P}_{\text{fin}}}]_r \overset{\text{def}}{=} \{\text{trim}_r(\text{path-type}(\Sigma)) : \ \Sigma \in \overline{\mathscr{P}_{\text{fin}}}\}. \tag{3}$$

The rest of this section is devoted to proving the following proposition.

▶ **Proposition 50.** *For any given $r \in \mathbb{N}$, one can compute the set $[\overline{\mathscr{P}_{\text{fin}}}]_r$.*

We will use a result of Colcombet and Loeding [10] which we now recall. They use the notion of a *cost automaton* over finite trees. Such automata, apart from accepting or rejecting an input tree, assign a natural number to any accepted tree. More precisely, a *B-automaton* is a nondeterministic tree automaton, which is additionally equipped with counters which can be incremented, reset, checked or left unchanged (the sequence of counter operations is in a bottom-up fashion). For a B-automaton $\mathcal{A}$, the *value* of a run over a tree is the maximal checked value of any counter, and the value of a tree is the infimum of the values of all accepting runs over the tree.

The authors consider the *domination relation* for functions defined over sets of finite trees. This is the same relation as in Definition 12.

▶ **Theorem 51** (Theorem 13 in [10]). *Let $\mathcal{A}$ and $\mathcal{B}$ be two B-automata, and let $[\![\mathcal{A}]\!]$ and $[\![\mathcal{B}]\!]$ be the functions over the set of finite trees which they compute. It is decidable whether $[\![\mathcal{A}]\!]$ dominates $[\![\mathcal{B}]\!]$.*

We introduce some auxiliary notions needed in the proof of Proposition 50.

Let a *path type* be a specification as described in the remark following Lemma 32, i.e. a path type $[\sigma]$ contains:

- A source and a target state,
- A set of appearing states,

and for each counter $c \in C$,

- A bit indicating if the counter c is cut,
- A bit indicating if the counter c is reset,
- A bit indicating if the counter c is incremented,
- A bit indicating if the counter c contains an ω.

Fix a path type $[\sigma]$. For a path signature $\sigma \in \widehat{\Lambda}^*$, we say that σ *matches qualitatively* the type $[\sigma]$ if all the information, except for containment of ω, agrees in path-type(σ) and in $[\sigma]$. We also specify how to measure quantitatively how well σ matches the type $[\sigma]$. We define the *excess* and *lack* of σ with respect to $[\sigma]$ as follows.

$$\mathit{excess}_{[\sigma]}(\sigma) = \max_{c\in C}\{\mathrm{val}(\sigma, c) : \ [\sigma] \text{ does not indicate that counter } c \text{ contains an } \omega\}$$
$$\mathit{lack}_{[\sigma]}(\sigma) = \min_{c\in C}\{\mathrm{val}(\sigma, c) : \ [\sigma] \text{ indicates that counter } c \text{ contains an } \omega\}$$

Intuitively, a path signature σ matches quantitatively the path type $[\sigma]$ if it has a small excess and large lack. The following lemma follows easily from the definitions.

▶ **Lemma 52.** *Let σ be a path signature and $[\sigma]$ a path type. Then, the following conditions are equivalent.*

1. *The path type of σ is $[\sigma]$,*
2. *The three conditions are satisfied*
   - *σ matches qualitatively $[\sigma]$,*
   - *$\mathit{excess}_{[\sigma]}(\sigma) < \omega$,*
   - *$\mathit{lack}_{[\sigma]}(\sigma) = \omega$.*

▶ **Lemma 53.** *There is a B-automaton over finite words over the alphabet Λ, which can test whether an input word σ matches a given path type $[\sigma]$, and compute its excess or lack with respect to $[\sigma]$.*

Proof. It is straightforward to construct the automaton which computes the excess – such an automaton can be made deterministic, and has one counter per each $c \in C$, and computes the value $\mathrm{val}(\sigma, c)$ using its built-in max operation.

Computing the lack is slightly more complicated, since the automaton must compute a minimum over values $\mathrm{val}(\sigma, c)$, which in turn are defined using the max operation. However, for this we use the built-in feature of nondeterministic B-automata, that the value computed by a is the minimum over all accepting runs. ◀

We now return to the proof of Proposition 50, that the set $[\overline{\mathscr{P}_{\mathrm{fin}}}]_r$ is computable.

Proof. Let $[\Sigma]$ be a multiset of path types with threshold r, i.e. every element appears at most r times in $[\Sigma]$. For the rest of this section we fix $[\Sigma]$. We describe how to decide using B-automata whether

$$[\Sigma] \in [\overline{\mathscr{P}_{\mathrm{fin}}}]_r.$$

The automata $\mathcal{A}$ and $\mathcal{B}$. We define two B-automata $\mathcal{A}$ and $\mathcal{B}$ which work over binary trees, whose inner nodes are labeled by elements of the transition relation δ of our puzzle $\mathcal{P}$, and leaf nodes are labeled by path types. We assume that the leaf nodes have no siblings, i.e. parents of leaf nodes are unary nodes. The automata $\mathcal{A}$ and $\mathcal{B}$ will be equipped with the same set of counters C as the puzzle.

The automaton $\mathcal{A}$ is a B-automaton. It is constructed as a deterministic bottom-up tree automaton, which accepts a tree t if:

- the labeling of the tree is consistent with the transition relation of the puzzle $\mathcal{P}$,
- the multiset of leaf labels of t is equal to $[\Sigma]$, up to threshold r (i.e. trim_r of both multisets is equal); and
- on each path π whose leaf node is labeled by a path type $[\sigma]$, the signature of the path π matches qualitatively the path type $[\sigma]$.

Moreover, we design $\mathcal{A}$ so that

$$[\![\mathcal{A}]\!](t) = \max_{\pi} \mathit{excess}_{[\sigma]}(\sigma),$$

where π ranges through all root-to-leaf paths of t, σ denotes the path signature of π and $[\sigma]$ denotes its leaf label.

The automaton $\mathcal{B}$ is similar. It is a B-automaton which can be described just as $\mathcal{A}$, with the difference that $\mathcal{B}$ computes on each path the lack rather than the excess:

$$[\![\mathcal{B}]\!](t) = \min_{\pi} \mathit{lack}_{[\sigma]}(\sigma),$$

Note that a formal definition of $\mathcal{B}$ is slightly more complicated than the definition of $\mathcal{A}$, since $\mathcal{B}$ needs to use nondeterminism in order to compute the minimum. Note that $\mathcal{A}$ and $\mathcal{B}$ both accept the same regular language of finite trees.

The following lemma, together with Theorem 51, prove Proposition 50. ◀

▶ **Lemma 54.** *The following conditions are equivalent.*

1. $[\Sigma] \in [\overline{\mathscr{P}_{\mathrm{fin}}}]_r$
2. *Over the regular language of trees accepted by $\mathcal{A}$ and $\mathcal{B}$, the function $[\![\mathcal{A}]\!]$ computed by $\mathcal{A}$ is strictly dominated by the function $[\![\mathcal{B}]\!]$ computed by $\mathcal{B}$.*

(We say that f is strictly dominated by g if f is dominated by g but not vice-versa.)

We will show the bottom-up implication in the lemma; the other one is similar. Assume that the second condition holds. This implies that there exists a sequence of trees $t_1, t_2, \ldots$ which are accepted by $\mathcal{A}$ and $\mathcal{B}$, and such that:

- there is a bound $m \in \mathbb{N}$ such that for all $n \in \mathbb{N}$

$$[\![\mathcal{A}]\!](t_n) < m$$

- for each $n \in \mathbb{N}$,

$$[\![\mathcal{B}]\!](t_n) \geq n.$$

Let Σ_n denote the signature of the tree t_n, treated as a factor, with leaf nodes replaced by ports. Then $\Sigma_n \in \mathscr{P}_{\mathrm{fin}}$. By compactness of the set $\overline{\mathscr{P}_{\mathrm{fin}}}$, we may assume that the sequence $t_1, t_2, \ldots$ is such that

$$\lim_{n\to\infty} \Sigma_n = \Sigma \quad \text{for some} \quad \Sigma \in \overline{\mathscr{P}_{\mathrm{fin}}}.$$

To complete the bottom-up implication in Lemma 54, it suffices to prove the following sublemma.

▶ **Lemma 55.** *Let Σ be as describe above. Then*

$$\mathrm{trim}_r(\text{path-type}(\Sigma)) = [\Sigma], \tag{4}$$

In particular, $[\Sigma] \in [\overline{\mathscr{P}_{\mathrm{fin}}}]_r$.

Proof. Let $n \in \mathbb{N}$. The mapping which maps root-to-leaf paths in t_n to path signatures in Σ_n is a multiset bijection. Therefore, we may unambiguously define a multiset mapping

$$\lim \text{path-type}\colon \Sigma_n \to [\Sigma],$$

defined so that for a root-to-leaf path π which ends with a leaf node labeled by $[\sigma]$,

$$\lim \text{path-type}(\sigma) = [\sigma].$$

Note that if the above holds for $\sigma \in \Sigma_n$ and $[\sigma]$, then by the acceptance condition of $\mathcal{A}$, σ matches $[\sigma]$ qualitatively. Moreover, if $\lim \text{path-type}(\Sigma_n)$ denotes the multiset image of the mapping $\lim \text{path-type}$, then

$$\text{trim}_r(\lim \text{path-type}(\Sigma_n)) = [\Sigma]. \tag{5}$$

This also follows from the acceptance condition of $\mathcal{A}$.

For $n \in \mathbb{N}$ and $[\sigma] \in [\Sigma]$, let

$$\Sigma_n^{[\sigma]} = \{\sigma \in \Sigma_n : \lim \text{path-type}(\sigma) = [\sigma]\}$$

denote the multiset of elements of Σ_n which have $\lim \text{path-type}$ equal to $[\sigma]$. Then, the multiset Σ_n is partitioned into finitely many multisets:

$$\Sigma_n = \bigcup_{[\sigma]\in[\Sigma]} \Sigma_n^{[\sigma]}.$$

Without loss of generality, by compactness, we may assume that for each $[\sigma] \in [\Sigma]$,

$$\lim_{n\to\infty} \Sigma_n^{[\sigma]} = \Sigma^{[\sigma]} \tag{6}$$

for some multiset $\Sigma^{[\sigma]}$. Moreover, by continuity of multiset union, it follows that

$$\Sigma = \bigcup_{[\sigma]\in[\Sigma]} \Sigma^{[\sigma]}.$$

▶ **Claim 1.** *For each $\sigma \in \Sigma^{[\sigma]}$,*

$$\text{path-type}(\sigma) = [\sigma]$$

Let $\sigma \in \Sigma^{[\sigma]}$. Then, by (6), the path signature σ can be approximated by some element of $\Sigma_n^{[\sigma]}$, i.e. for each $n \in \mathbb{N}$ there exists $\sigma_n \in \Sigma_n^{[\sigma]}$ such that

$$\lim_{n\to\infty} \sigma_n = \sigma.$$

In particular, σ matches qualitatively $[\sigma]$, since this is a regular property, and is satisfied by each σ_n.

Note that by the definition of the functions computed by $\mathcal{A}$ and $\mathcal{B}$, and the assumption on t_n, if $\sigma_n \in \Sigma_n^{[\sigma]}$, then for some $m \in \mathbb{N}$ independent of n,

$$\mathit{excess}_{[\sigma]}(\sigma) < m \tag{7}$$
$$\mathit{lack}_{[\sigma]}(\sigma) \geq n. \tag{8}$$

Since the properties (7) and (8) are regular properties (for the fixed m, and each $n \in \mathbb{N}$), it follows that σ satisfies (7) and (8) for every $n \in \mathbb{N}$. Therefore,

$$\mathit{excess}_{[\sigma]}(\sigma) < m$$
$$\mathit{lack}_{[\sigma]}(\sigma) = \omega.$$

By Lemma 52, this proves that the path type of σ is $[\sigma]$, proving the claim.

Note that by (5) the size of the multiset $\Sigma_n^{[\sigma]}$ matches, up to threshold r, the number of occurrences of $[\sigma]$ in $[\Sigma]$. By the convergence (6), the same applies to the multiset $\Sigma^{[\sigma]}$. Together with the above claim, this implies that

$$\mathrm{trim}_r(\text{path-type}(\Sigma)) = [\Sigma].$$

This proves Lemma 55, ending the bottom-up implication of Lemma 54. The other implication is similar.

Hence, deciding whether $[\Sigma] \in [\overline{\mathscr{P}_{\mathrm{fin}}}]_r$ reduces to the problem of domination for B-automata over finite trees. This problem is decidable by [10]. This proves Proposition 50. ◀