# Unambiguous Tree Languages Are Topologically Harder Than Deterministic Ones

Szczepan Hummel*
Institute of Informatics, University of Warsaw

The paper gives an example of a tree language $G$ that is recognised by an unambiguous parity automaton and is $\Sigma_1^1$-complete (analytic-complete) as a set in Cantor space. This already shows that the unambiguous languages are topologically more complex than the deterministic ones, that are all in $\Pi_1^1$.

Using set $G$ as a building block we construct an unambiguous language that is topologically harder than any countable boolean combination of $\Sigma_1^1$ and $\Pi_1^1$ sets. In particular the language is harder than any set in difference hierarchy of analytic sets considered by O. Finkel and P. Simonnet in the context of nondeterministic automata.

## Introduction

Topological complexity becomes more and more popular as a set complexity measure in theoretical computer science, especially in automata theory. Understanding how hard from the topological point of view are the languages recognised by a particular class of automata gives us more understanding of the power of those automata. It also gives us the access to very powerful and well developed tools coming from descriptive set theory.

One of remarkable uses of descriptive methods is the separation of classes of languages. Once we know the upper complexity bound for a given class, we can use it while showing that some languages do not belong to the class. To quote only recent applications of this method, it was used in [4] to show that deterministic max-automata are less expressive than nondeterministic ones. In [11], topological complexity methods were used to exclude a large class of automata as a potential automata model for MSO+U logic.

In this paper we address this complexity question for unambiguous automata on infinite binary trees, i.e. nondeterministic automata that have at most one accepting run on each tree. With a rise and development of models of automata that do not admit determinisation (infinite tree automata, register automata, BC-automata [5], etc.), the notions like unambiguity, and strong unambiguity, that can be seen as less restrictive variants of determinism, gain the importance. For the survey on forms of determinism see [9].

It is well known that deterministic parity tree automata recognise only sets in $\Pi_1^1$ topological class (coanalytic sets). On the other hand, nondeterministic automata recognise some sets that are neither analytic, nor coanalytic, but their expressive power is bounded by the second level of the projective hierarchy. By Rabin's complementation result ([17]), all nondeterministic languages are in $\Delta_2^1$ class, i.e. are both $\Sigma_2^1$ and $\Pi_2^1$.

It was shown by Niwiński and Walukiewicz in [15] (later described in [7] and [8]) that unambiguous automata do not recognise all nondeterministic languages. On the other hand, it is not hard to see that they are more expressive than deterministic automata. An example here might be language $UB$ of trees that have exactly one branch with infinitely many labels $a$.

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

M. Faella, A. Murano (Eds.): Games, Automata, Logics
and Formal Verification (GandALF 2012)
EPTCS 96, 2012, pp. 247–260, doi:10.4204/EPTCS.96.19

Language $UB$ is a $\Pi^1_1$ complete set. However, deterministic automata are also capable of recognising some $\Pi^1_1$ complete sets. Until now it was not known whether unambiguity introduces any hardness versus determinism from the topological viewpoint. Because $UB$ seems to be typical and close to the very definition of unambiguous automata, it has been widely believed that this example reflects the maximum power of unambiguity.

In this work, in Section 2, we show an example of unambiguous language $G$ that is $\Sigma^1_1$ complete, hence is not in $\Pi^1_1$. Then, in Section 3, using $G$, we construct another unambiguous language that is topologically harder than any set obtained from $\Sigma^1_1$ and $\Pi^1_1$ sets by countable boolean operations. As a consequence, the language itself is not such a boolean combination. At this level of granularity, this is the most that we can have in locating unambiguous class between deterministic and nondeterministic classes. To find out if there is a difference in topological complexity between nondeterministic and unambiguous languages we would have to refer to more precise complexity measures than just the projective hierarchy, e.g. to the finest one - the Wadge hierarchy.

In Section 4 we show that the example from Section 3 is actually strongly unambiguous, i.e. unambiguous together with its complement.

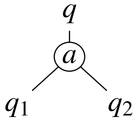
The full version of the paper with Appendix containing a proof of one general topological fact can be found online at: `http://www.mimuw.edu.pl/~shummel/unamb_topol.pdf`.

# 1 Preliminaries

Let $A$ be an arbitrary set of labels. An infinite binary tree over $A$ is a function $t : \{l,r\}^* \to A$. By $T_A$ we denote the set of all infinite binary trees over $A$. For $t \in T_A$ and $v \in \{l,r\}^*$, we use a notation $t_v$ for the subtree of $t$ rooted in $v$, i.e. $t_v(w) = t(v \cdot w)$.

## 1.1 Automata

A (nondeterministic) *parity tree automaton* over an alphabet $A$ consists of a finite set $Q$ of states, a transition relation $\delta \subseteq Q \times A \times Q \times Q$, an initial state $q_0 \in Q$, and a ranking function rank : $Q \to \omega$. We depict a transition $(q,a,q_1,q_2)$ as:



A run $\rho$ of the automaton on a tree $t$ is a labelling of the tree with states ($\rho : \{l,r\}^* \to Q$) consistent with the transition relation, i.e. for each node $v \in \{l,r\}^*$ the tuple $(\rho(v), t(v), \rho(vl), \rho(vr))$ belongs to $\delta$. We additionally require that the root of $t$ is labelled with an initial state $q_0$. The run is accepting if on each branch of the tree the highest rank occurring infinitely often in the run is even (the parity condition).

As usual, the language *recognised* by an automaton is the set of all trees on which the automaton has some accepting run.

The automaton is called (top-down) *deterministic* if its transition relation is in fact a function $\delta : Q \times A \to Q \times Q$. An important property of deterministic automata is that they have exactly one run on each input. A similar property gives a rise to consideration of a wider subclass of nondeterministic automata:

The automaton is *unambiguous* if it has at most one accepting run on each input. In other words, if it accepts a given tree, it can do it in only one way.

Since in a parity automaton the set of states is finite, so is the set of used ranks (the image of the ranking function). We say that the automaton is of index $(\iota,\kappa) \in \omega \times \omega$ if $\iota \leq \mathrm{rank}(Q) \leq \kappa$. Since shifting all ranks by an even number does not change the language recognised by the automaton, it suffices to consider indices $(\iota,\kappa)$ for $\iota \in \{0,1\}$. We say that the language is of index $(\iota,\kappa)$ if there is an automaton of such index recognising this language. By identifying each index with the class of languages of this index we obtain the Rabin-Mostowski index hierarchy. Lines denote inclusion on the diagram of the hierarchy shown on Figure 1.

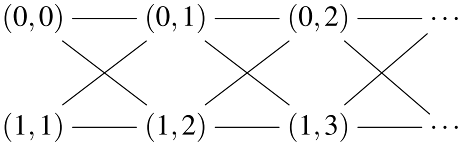


Figure 1: The Rabin-Mostowski index hierarchy.

The inclusions come immediately from the definition — only in some cases we need to shift ranks of corresponding automata by 2. Two classes (indices) at the same level of the hierarchy (levels are vertical on the diagram) are called *dual*, and by $\overline{(\iota,\kappa)}$ we denote the class (index) dual to $(\iota,\kappa)$.

Automata of index $(1,2)$ are called *Büchi automata*.

If nondeterministic parity automata are considered we talk of *nondeterministic index hierarchy*. Another often considered and very important variant is *alternating index hierarchy*. Since we will refer also to this hierarchy, we need to recall shortly the definition of alternating automata.

An *alternating parity tree automaton* is similar to the nondeterministic one with the exception that the set of states is partitioned into two parts $Q = Q_\exists \cup Q_\forall$. The semantics of such an automaton is defined by the game between two players $\exists$ and $\forall$. During a play in this game players construct a run of an automaton in a top-down manner. If a given node was labelled by the state from $Q_\exists$ during this construction then the next transition is chosen by Player $\exists$, otherwise it is chosen by Player $\forall$. The play is won by Player $\exists$ if in the constructed run on each branch the parity condition holds. A tree is accepted by the automaton if Player $\exists$ has a winning strategy in the game defined by this automaton on this tree.

The immediate, but important, fact concerning alternating automata in the context of index hierarchy is:

**Remark 1.1** *If a language is of alternating index $(\iota,\kappa)$ then its complement is of alternating index $\overline{(\iota,\kappa)}$.*

**Proof:**

It is enough to switch players and shift ranks by one in the automaton recognising a given language. □

The crucial fact about the hierarchies is that they are strict, i.e. all inclusions on Figure 1 are strict. This result for nondeterministic hierarchy is due to Niwiński [14]. For alternating hierarchy it was independently proven by Arnold [1] and Bradfield [6].

**Theorem 1.2 (Niwiński, Bradfield, Arnold)** *For infinite binary trees, both alternating and nondeterministic index hierarchies are strict.*

The languages recognised by nondeterministic (or equivalently alternating) tree automata are called *regular tree languages*. The languages recognised by deterministic (respectively unambiguous) automata are called deterministic (respectively unambiguous) tree languages.

## 1.2 Topology

For a fixed alphabet $A$, we treat $T_A$ as a topological space. A basic open set is obtained by fixing a finite prefix of trees (a finite tree starting in the root). Other open sets are obtained by taking arbitrary unions of basic open sets. If $A$ is finite, this topological space is homeomorphic (i.e. topologically isomorphic) to the Cantor space.

### 1.2.1 Projective hierarchy

The class $\mathrm{Bor}(X)$ of Borel sets in the topological space $X$ is the least class that:

- contains all open sets of $X$,
- is closed under complementation, and
- is closed under countable unions and intersections.

If a space is understood from the context the class is simply denoted by Bor.

The class of Borel sets is not closed under projection. Each set that is a projection of a Borel set is called *analytic*. The class of analytic sets is denoted by $\Sigma^1_1$. Formally[1]:

$$\Sigma^1_1(X) = \left\{P \subseteq X : \exists_{A\text{-finite}}\, \exists_{B\in \mathrm{Bor}(X\times T_A)}\; P{=}\pi_1(B)\right\}$$

The rest of the projective hierarchy is defined as follows:

$$\begin{array}{ll} \Pi^1_i & \text{consists of the complements of the sets from } \Sigma^1_i \\ \Sigma^1_{i+1} & \text{consists of the projections of the sets from } \Pi^1_i \end{array}$$

The sets from the class $\Pi^1_1$ are called *co-analytic*.

An important theorem of Souslin states that if a set is analytic and co-analytic, then it is Borel.

In the sequel we will also use two kinds of intermediate classes. The first kind is:

$$\Delta^1_i = \Sigma^1_i \cap \Pi^1_i$$

The second one is the $\sigma$-algebra generated by the sets at given level. Recall that the $\sigma$-algebra (also called $\sigma$-field) generated by a family $A$ is the closure of the family on countable union, countable intersection and complementation.

Figure 2 present the shape of the projective hierarchy. All the inclusions on the diagram are strict (apart from the leftmost equality coming from Souslin's Theorem). Proofs of all the facts mentioned in this section can be found in [12, Chapters 14 and 37].

### 1.2.2 Topological Complexity

A class of subsets of topological spaces is a *topological complexity class* if it is closed under preimages of continuous functions. In particular any class depicted on Figure 2 (if we do not fix any specific space) is a topological complexity class. Analogously to the complexity theory, there are the notions of *reductions* and *completeness*. Let $X$ and $Y$ be two topological spaces and let $K \subseteq X$ and $M \subseteq Y$. A continuous function $f : X \to Y$ is a *reduction* of $K$ to $M$ if $K{=}f^{-1}(M)$. In such case we say that $K$ is *Wadge-reducible* to $M$, or that $M$ is *topologically harder* (*more complex*) than $K$.

For a topological complexity class $\mathbf{K}$, a set $M$ is called $\mathbf{K}$-*hard* if any set $K \in \mathbf{K}$ is Wadge-reducible to $M$. We say that $M$ is $\mathbf{K}$-*complete* if additionally $M \in \mathbf{K}$.

[1] The choice of space $T_A$ at the second coordinate is not a commonly made choice in this definition, but it is best suited for our needs and the resulting notion is the same as in the standard definition.

$$\begin{tikzcd}
 & & \Pi^1_1 \arrow[dr, dash] & & & \Pi^1_2 \arrow[dr, dash] & & & \Pi^1_3 \arrow[dr, dash] & & \cdots \\
\mathrm{Bor} \arrow[r, equal] & \Delta^1_1 \arrow[ur, dash] \arrow[dr, dash] & & \sigma(\Sigma^1_1) \arrow[r, dash] & \Delta^1_2 \arrow[ur, dash] \arrow[dr, dash] & & \sigma(\Sigma^1_2) \arrow[r, dash] & \Delta^1_3 \arrow[ur, dash] \arrow[dr, dash] & & \sigma(\Sigma^1_3) & \cdots \\
 & & \Sigma^1_1 \arrow[ur, dash] & & & \Sigma^1_2 \arrow[ur, dash] & & & \Sigma^1_3 \arrow[ur, dash] & & \cdots
\end{tikzcd}$$

Figure 2: The projective hierarchy

## 1.3 Topological Complexity of Automata

We recall some basic facts binding automata and index hierarchies with topological hierarchies. We use these facts in further discussion.

**Theorem 1.3** *Each regular language of infinite trees is in* $\Delta^1_2$.

**Proof (sketch):**

For a fixed branch $\alpha$ of an infinite tree, the set of runs for which the parity condition on $\alpha$ holds, is Borel. Therefore, the set of accepting runs is a $\Pi^1_1$ set — "for all branches" corresponds to co-projection. Now, for a fixed nondeterministic parity automaton, the accepted trees are the ones for which there exists an accepting run, so the recognised language is $\Sigma^1_2$ as a projection of $\Pi^1_1$ set.

By Rabin's complementation lemma (see [17]), the complement of the language recognised by a nondeterministic automaton is also recognised by a nondeterministic automaton. If a language and its complement are both $\Sigma^1_2$ sets, they are in fact $\Delta^1_2$ sets. □

**Theorem 1.4** *Each language recognised by a deterministic parity tree automaton is in* $\Pi^1_1$.

**Proof (sketch):**

Each deterministic automaton defines a continuous function mapping a tree to the run on it. The set of accepted trees is, then, the inverse image of the set of accepting runs under a continuous function. The set of accepting runs is $\Pi^1_1$, then so is the recognised language. □

**Theorem 1.5** *Each language recognised by alternating parity automaton of index* $(1,2)$ *(resp.* $(0,1)$*) is in* $\Sigma^1_1$ *(resp.* $\Pi^1_1$*) topological class.*

**Proof:**

It was shown by Arnold and Niwiński [2] that each language of alternating index $(1,2)$ can be recognised by a nondeterministic automaton of index $(1,2)$ (a Büchi automaton). Rabin proved in [18] that each such language can be described by existential formula of monadic logic. This implies that they are analytic ($\Sigma^1_1$).

The fact for index $(0,1)$ comes from the duality — alternating $(0,1)$ automata recognise complements of sets recognised by alternating $(1,2)$ automata (see Remark 1.1). □

## 2 Analytic Complete Language

The result presented in this section was inspired by the unpublished work of Bilkowski on the decidability of Unambiguity Problem [3], and by the decidability result presented by Niwiński and Walukiewicz in [16]. Bilkowski has shown that the complement of a deterministic language is unambiguous if and only if the language is recognised by a thin automaton, i.e. by an automaton that has only countably many non-trivial paths in each accepting run. A path is trivial if, from some moment on, it is labelled only by all-accepting or all-rejecting states.

The deterministic automaton recognising the complement of language $G$ described below is thin in Bilkowski's sense, and has split property — the sufficient condition for the $\Pi^1_1$ hardness from the result of [16]. We do not give precise definitions, nor do we discuss the above results since the proofs in this article do not rely on them — they were only used while constructing the example presented in this section.

We call a branch of a binary tree over the alphabet $\{a,b\}$ *good* if:

1. it is labeled only with $a$'s,
2. it turns left infinitely many times.

Let:

$$G = \left\{ t \in T_{\{a,b\}} : t \text{ has a good branch} \right\}$$

First, we prove the crucial lemma:

**Lemma 2.1** *If an infinite binary tree over the alphabet $\{a,b\}$ has a good branch, then it has the left-most such branch, i.e. a good branch such that there is no good branch to the left.*

**Proof:**

Assume that a tree $t$ has a good branch. The construction of the left-most good branch goes as follows. We start from the root. If we have constructed the prefix of the branch up to the node $v$ we advance to the left descendant if there are good branches going through it. Otherwise we advance to the right descendant. Call the branch constructed by this procedure $\rho(t)$.

By the construction, it is clear that there is no good branch to the left from $\rho(t)$. Now we prove that $\rho(t)$ is good. Note that during the construction we maintain the invariant that there is a good branch going through a considered node. In particular, all nodes we have selected are labeled with $a$, therefore, we only need to verify property 2 from the definition to proof goodness of $\rho(t)$.

Assume that $\rho(t)$ turns left only finitely many times. Then there is a vertex $v$ on the branch $\rho(t)$ after which $\rho(t)$ turns only right. Let us take a good branch going through $v$, and call it $\sigma$. By the assumption, $\rho(t)$ is not good, so branches $\rho(t)$ and $\sigma$ diverge in some vertex $w$. Since $w$ is below $v$, $\rho(t)$ goes right from $w$ and $\sigma$ goes left. Since $\sigma$ is good, the construction should have selected the left descendant of $w$, but have selected the right one. That yields a contradiction, so $\rho(t)$ turns left infinitely many times. □

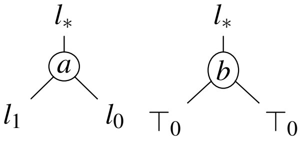
Now let $L = \overline{G}$.

**Proposition 2.2** *Language $G$ is recognised by an unambiguous automaton and its complement $L$ is recognised by a deterministic automaton.*

**Proof:**

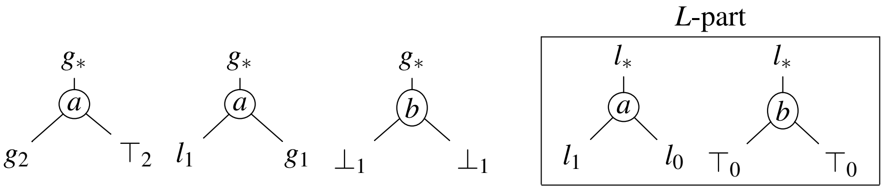
Thanks to Lemma 2.1, to prove unambiguity of $G$ it is enough to show an automaton that guesses the left-most good branch and verifies correctness of the guess. The idea is that automaton goes along a branch labelled with $a$'s, proving that the branch turns left infinitely many times, and proving that everything that diverges to the left from the branch does not have a good branch (i.e. belongs to $L$), and not caring what happens to the right from the branch.

Let us then start with constructing a deterministic automaton $\mathscr{L}$ recognising $L$.It has 3 states: states $l_0$ and $l_1$ (subscript indicates rank) occur on paths that have had only $a$'s so far, and track turns to the left; state $\top_0$ is all-accepting (i.e. self-looping with rank 0). Initial state is $l_1$ and the transitions are as follows:



The automaton uses ranks $\{0,1\}$. Note that, for given tree $t$, the run of $\mathscr{L}$ on $t$ has a branch with infinitely many ranks 1 on it if and only if there is an $a$-labelled branch turning left infinitely often in $t$. Therefore the automaton recognises $L$.

The automaton $\mathscr{G}$ for language $G$ uses $\mathscr{L}$ as a component. It has 7 states and uses ranks $0,1,2$. States $g_1$ and $g_2$ (again, subscript indicates rank) are used to track the branch; states of $\mathscr{L}$ — to prove non-existence of a good branch in a subtree; state $\top_2$ is all-accepting, and state $\bot_1$ is all-rejecting. The initial state is $g_1$. The automaton uses the following transitions:



It is not hard to see that presented automaton implements described idea, therefore accepts if and only if given tree has a good branch. It is unambiguous, because it only can accept by labelling the left-most good branch with $g$ states. □

**Remark 2.3** *Automaton $\mathscr{L}$ is of index $(0,1)$, and $\mathscr{G}$ is of index $(0,2)$.*

*Note that automaton $\mathscr{G}$ can be transformed into an equivalent one of index $(1,3)$, by making L-part use ranks $\{2,3\}$ instead of $\{0,1\}$.*

**Proposition 2.4** *Set $G$ is $\Sigma^1_1$ complete.*

**Proof:**

To prove the hardness we continuously reduce the set $IF$ of $\omega$-branching trees with an infinite branch to our set $G$. Set $IF$ is a well known $\Sigma^1_1$-complete subset of the space $Tr$ of trees on $\omega$, i.e. prefix-closed subsets of $\omega^*$ (see e.g. [12, Theorem 27.1]). The topology on $Tr$ is similar to the one on $T_A$: a basic open set is obtained by fixing some finite part of trees. E.g. for some integer $n$, we fix what nodes out of $\{1,...,n\}^{\leq n}$ belong and what do not belong to all trees in a set.

We construct a reducing function $f : Tr \to T_{\{a,b\}}$. Fix a tree $t \in Tr$. Put labels $a$ to the root and all right descendant nodes in the tree $f(t)$. For each node $n_1 n_2 n_3 \dots n_k$ of tree $t$ we put label $a$ to the node $r^{n_1} l r^{n_2} l r^{n_3} l \dots r^{n_k} l$ in $f(t)$. Remaining left descendant nodes obtain label $b$.

Note that $f(t)$ has an $a$-labelled branch that turns left infinitely many times (i.e. a good branch) if and only if $t$ has an infinite branch. Therefore

$$f(t) \in G \iff t \in IF$$

So $f$ indeed reduces $IF$ to $G$.

Function $f$ is continuous, because the labels at $n$'th level of the tree $f(t)$ are determined by the finite part of a tree $t$, namely the part in $\{1,...,n\}^{\leq n}$.

The upper topological complexity bound of set $G$ comes from Proposition 2.7, that will be proven later, and from Theorem 1.5. □

Therefore, we have proven the following:

**Theorem 2.5** *There is a $\Sigma^1_1$ complete language of infinite trees, that is recognised by an unambiguous parity automaton.*

Recall the theorem that is stated as Corollary 4.14 in [10]:

**Theorem 2.6 (Finkel, Simonnet)** *A tree language recognised by an unambiguous Büchi automaton is Borel.*

Thanks to this theorem we know that no unambiguous Büchi automaton recognises language $G$. Proposition 2.4 implies that $G$ is not a $\Pi^1_1$ set, therefore, by Theorem 1.5, it cannot be recognised by any (even alternating) automaton of index $(0,1)$. As a result we obtain that the use of 3 priorities is necessary for unambiguous automaton to recognise $G$. On the other hand observe that:

**Proposition 2.7** *Language G is recognised by a nondeterministic Büchi automaton.*

**Proof:**

It suffices to remove $L$-part from the unambiguous automaton $\mathscr{G}$ presented in the proof of Proposition 2.2, replacing $l_1$ with $\top_2$ in other transitions to obtain needed nondeterministic automaton. The only purpose of that part was to make sure that we select left-most good branch. We do not need this if we do not care about the number of accepting runs. □

From this observation we obtain that the result by Finkel and Simonnet is tight in the sense that the following strengthening of Theorem 2.6 does NOT hold: "An unambiguous language that is recognised by some (possibly ambiguous) nondeterministic Büchi automaton, is Borel". Let us state it as follows:

**Corollary 2.8** *There is a language of non-Borel topological complexity that is on one hand unambiguous, and on the other hand Büchi.*

# 3 Beyond Boolean Combinations

In this section we construct an unambiguous tree language that is topologically harder than any set in $\sigma(\Sigma^1_1)$. For that we need to prove that the class of the sets that reduce to the constructed language contains all analytic sets, and is closed under complementation and countable unions.

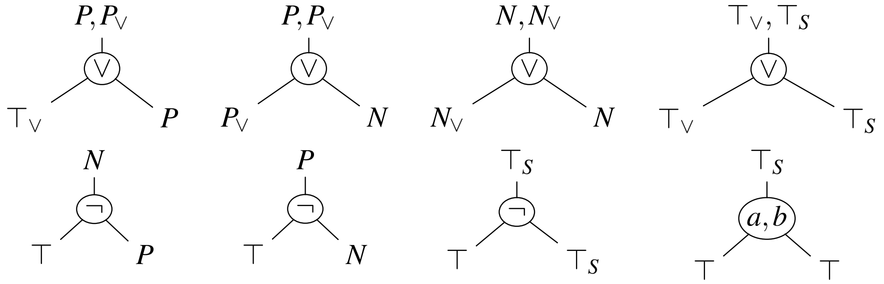
The construction of the language harder than boolean combinations of analytic sets goes through an automaton. The automaton, we will call it $\mathscr{C}$, uses as building blocks:

1. the unambiguous automaton $\mathscr{G}$ recognising language $G$ from Section 2,
2. the deterministic automaton $\mathscr{L}$ recognising $L := \overline{G}$.

Since we will work with larger alphabet than just $\{a,b\}$, we modify the automata (and recognised languages), such that they treat all letters except of $a$ like $b$. After this modification, the languages recognised by the automata are complements even in case of larger alphabet.

The idea is that automaton $\mathscr{C}$ expects the part of a tree near the root to be shaped as a formula defining some set in the $\sigma$-algebra $\sigma(\Sigma^1_1)$. Countable union is represented by the branch turning only left, where right descending subtrees correspond to subformulas. Complementation nodes in a way disregard the left descendants. In nodes corresponding to the atoms of the formula (analytic sets) the automaton expects subtrees from language $G$. The details follow.

The automaton $\mathscr{C}$ works over the alphabet $A = \{a, b, \vee, \neg\}$. Apart from the states of the automata $\mathscr{G}$ and $\mathscr{L}$, it uses states: $N$, $P$, $N_\vee$, $P_\vee$, $\top_\vee$, $\top_S$, $\top$ ($P$ stands for 'Positive', $N$ for 'Negative'; $\top_\vee$ and $\top_S$ serve only verifying the shape of the formula). The initial state is $P$. The transitions of $\mathscr{C}$ are as follows:



where $\top$ is an all-accepting state. Additionally, when the automaton encounters letter $a$ or $b$ in state $P$, it starts to act as automaton $\mathscr{G}$. If it encounters one of these letters in state $N$, it starts to act as automaton $\mathscr{L}$.

States $N$, $P$, $P_\vee$, $\top_S$ have rank 1, states $N_\vee$, $\top_\vee$, $\top$ rank 0.

Note that automaton $\mathscr{C}$ is of index $(0,2)$ — it is because of ranks used by $\mathscr{G}$.

We will use the notation $\mathscr{A}_q$ for the automaton $\mathscr{A}$ modified in such a way, that $q$ becomes an initial state. Let us prove the following:

**Proposition 3.1** *For each set $S \subseteq X$ obtained from analytic sets by countable boolean operations the following holds:*

> *there is a continuous function $f : X \to T_A$, that simultaneously reduces $S$ to $L(\mathscr{C})$, and $\overline{S}$ to $L(\mathscr{C}_N)$, i.e.:*
>
> 1. *$\mathscr{C}$ accepts $f(x)$ iff $x \in S$,*
> 2. *$\mathscr{C}_N$ accepts $f(x)$ iff $x \notin S$.*
>    *Additionally we require:*
> 3. *For all $x$, $\mathscr{C}_{\top_S}$ accepts $f(x)$.*
>
> $(*)$

**Proof:**

We need to prove that the class of sets for which the property $(*)$ holds contains all analytic sets, and is closed under complementation and countable unions.

**(analytic sets)** Take an arbitrary analytic set $S \subseteq X$. Since $G$ is analytic-complete, there exists a continuous function $f : X \to T_{\{a,b\}}$ reducing $S$ to $G$. We will show that the same function is actually a needed simultaneous reduction to $L(\mathscr{C})$ and $L(\mathscr{C}_N)$.

Note that among the trees over the alphabet $\{a, b\}$, $\mathscr{C}$ accepts exactly the trees from $G$. Therefore:

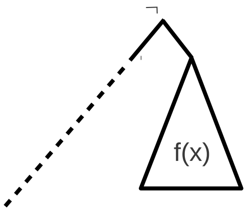
$$f(x) \in L(\mathscr{C}) \iff f(x) \in G \iff x \in S$$

Now we have to show that $\mathscr{C}$ accepts $f(x)$ from state $N$ if and only if $x \notin S$. Recall that from state $N$, on trees over $\{a,b\}$, $\mathscr{C}$ acts exactly like automaton $\mathscr{L}$. Since $\mathscr{L}$ recognises the complement of $G$, we obtain the statement.

From state $\top_S$ each tree with $a$ or $b$ in the root is accepted, so condition 3 also holds for $f$.

**(closure under the complement)** Take any set $S \subseteq X$ for which $(*)$ holds, and let $f : X \to T_A$ be an appropriate reducing function. We construct a reduction $\overline{f} : X \to T_A$ of $\overline{S}$ to $L(\mathscr{C})$ by putting:

$\overline{f}(x) :=$

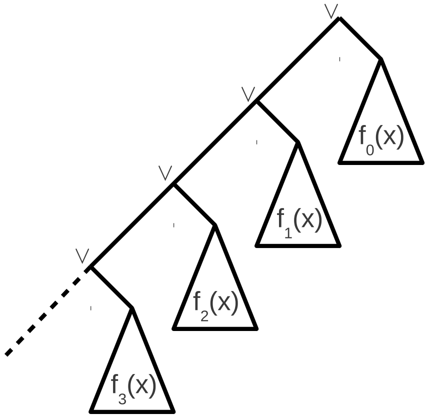


(the rest of the leftmost branch does not matter)

By the inductive assumption and by the shape of the transitions over letter $\neg$, we obtain that $\overline{f}(x)$ is accepted from state $P$ if and only if $f(x)$ is accepted from state $N$, that $\overline{f}(x)$ is accepted from state $N$ if and only if $f(x)$ is accepted from state $P$, and that $\overline{f}(x)$ is accepted from state $\top_S$ if and only if $f(x)$ is accepted from state $\top_S$.

**(closure under unions)** Take a set $S \subseteq X$ such that $S = \bigcup_{i=0}^{\infty} S_i$, where each $S_i$ has property $(*)$. For each $i$, let $f_i : X \to T_A$ be an appropriate reduction as in $(*)$ for set $S_i$. We build a reduction for set $S$ as follows:

$f(x) :=$



Function $f$ is clearly continuous. We now prove that it is a needed simultaneous reduction.

Let $x \in S$, and let $i_0$ be the least $i$ such that $x \in S_i$. Then, from the inductive assumption: $f_{i_0}(x)$ is accepted from state $P$; for each $i < i_0$, $f_i(x)$ is accepted from state $N$; for each $i > i_0$, $f_i(x)$ is accepted from state $\top_S$. Then, by the shape of the transitions over label $\vee$, $f(x)$ is accepted from state $P$.

Let now $x \notin S$. Then $x \notin S_i$ for each $i$. Therefore, for each $i$, $f_i(x)$ is accepted from state $N$. Then $f(x)$ is accepted from state $N$, by the shape of transitions over $\vee$.

If each $f_i(x)$ is accepted from state $\top_S$, then by the shape of the transitions from $\top_S$ and $\top_\vee$ over $\vee$, $f(x)$ is also accepted from state $\top_S$. □

In particular we have proven:

**Corollary 3.2** *Each set from $\sigma(\Sigma_1^1)$ continuously reduces to $L(\mathscr{C})$.*

We use one more fact from descriptive set theory. It can be proven using Theorems 1E.3, 1D.2, 1D.3 from [13] and the fact that $\sigma(\Sigma_1^1)$ is closed under the complement (the details can be found in Appendix of the full version of this paper).

**Proposition 3.3** *For each $n \geq 1$, there is no $\sigma(\Sigma^1_n)$-complete set.*

**Corollary 3.4** *Set $L(\mathscr{C})$ is in $\Delta^1_2 \setminus \sigma(\Sigma^1_1)$.*

Let now $W \subseteq T_A$ be the language of trees corresponding to improperly shaped formulas, i.e. the set of trees that:

1. contain a branch labelled only with $\vee$ and $\neg$, that turns right infinitely many times, and turns right after each occurrence of $\neg$ (formula is not well-founded),

or

2. contain a path from the root that turns right after each occurrence of $\neg$, ends in a left descendant node $v$ that is labelled with something different than $\vee$, a parent of $v$ is labelled with $\vee$, and all nodes above $v$ are labelled with $\vee$ or $\neg$.

**Lemma 3.5** *Sets $L(\mathscr{C})$, $L(\mathscr{C}_N)$ and $W$ constitute a partition of $T_A$. In particular $L(\mathscr{C})$ and $L(\mathscr{C}_N)$ are disjoint.*

**Proof:**

First we show that if $t \in W$, then $t$ is not accepted from neither of states $P$, $N$. Assume first, that $t$ has a branch $\rho$ as in point 1 of the definition of $W$. Consider a run of the automaton starting from $P$ or $N$. Branch $\rho$ is labelled with states $N$, $P$, $N_\vee$, $P_\vee$, $\top_\vee$, $\top_S$ (ranks 0 and 1) in this run, and after each turn to the right there is state $N$, $P$, or $\top_S$ (each of rank 1). The highest rank occurring infinitely often on branch $\rho$ is 1, so the run is not accepting. Now assume that $t$ has a path as in point 2 of the definition of $W$. In a run starting from $P$ or $N$, the node $v$, as a left descendant of a node labelled with $\vee$, always obtains a state label of $N_\vee$, $P_\vee$, or $\top_\vee$. Since from these states there are only transitions over letter $\vee$ and $v$ is labelled with something different, the run gets stuck.

Let us now consider well-shaped trees, i.e. trees outside $W$. Recall that prefixes of such trees represent well-founded formulas. Using this fact we define a rank on such trees. For a tree $t$, $\text{rank}(t)$ is an ordinal defined by the rules:

- If the root of $t$ is labelled with $a$ or $b$, then $\text{rank}(t) = 0$.
- If the root is labelled with $\neg$, then $\text{rank}(t) = \text{rank}(t_r) + 1$, where $t_r$ is a subtree of $t$ rooted in right descendant node of the root.
- If the whole left-most branch is labelled with $\vee$, then $\text{rank}(t) = sup\{\text{rank}(t_{l^n r}) + 1 : n \geq 0\}$, where $t_{l^n r}$ are subtrees diverging from the path to the right.

Note that the above set of rules allows to define the rank for all well-shaped trees.

Now we prove by the transfinite induction on $\text{rank}(t)$ that each well-shaped tree $t$ is either accepted from state $P$ or from $N$, but not from both.

If $\text{rank}(t) = 0$, the tree has $a$ or $b$ in the root. It is, then, accepted from state $P$ if and only if $t \in G$, and is accepted from state $N$ if and only if $t \in L$. Since $L$ is the complement of $G$, we are done with this case.

If $\text{rank}(t) > 0$, then the root is labelled with $\vee$ or $\neg$. If it is labelled with $\neg$, then $t$ is accepted from state $P$ if and only if $t_r$ is accepted from state $N$, and vice versa. Since $t_r$ is of lower rank than $t$, it suffices to use the inductive assumption for $t_r$.

If $t$ has label $\vee$ in the root, then, since it is well-shaped, it has label $\vee$ at whole left-most branch. All subtrees diverging to the right from the left-most branch have lower rank then $t$, so we can use the inductive assumption for them. Then:

either at least one of the subtrees is accepted from state $P$ and not from $N$ — in this case $t$ is accepted from $P$, but not from $N$,

or each of them is accepted from state $N$ and not from $P$ — then tree $t$ is accepted from state $N$, but not from $P$.

□

**Proposition 3.6** *Automaton $\mathscr{C}$ is unambiguous.*

**Proof:**

The only nondeterminism in automaton $\mathscr{C}$ occurs in state $P$ (or $P_\vee$) and label $\vee$. Two transitions are possible from this configuration — one assigns state $P$ to the right child, the other one assigns state $N$ to the right child. By Lemma 3.5, only one of the transitions can be used in an accepting run from a given node in a given tree. □

Recall that automaton $\mathscr{C}$ is of index $(0,2)$. Since $L(\mathscr{C})$ is neither a $\Sigma^1_1$ set, nor a $\Pi^1_1$ set, by Theorem 1.5, we have:

**Remark 3.7** *Language $L(\mathscr{C})$ cannot be recognised by any alternating automaton using* $2$ *ranks (neither* $(0,1)$*, nor* $(1,2)$*).*

## 4 The Example Is Strongly Unambiguous

A language $M$ is called *strongly unambiguous* if both $M$ and $\overline{M}$ are recognised by unambiguous automata. Because of the apparent asymmetry of the class of unambiguous languages, strong unambiguity seems to be an important notion. Some arguments for its importance were given by Colcombet in [9].

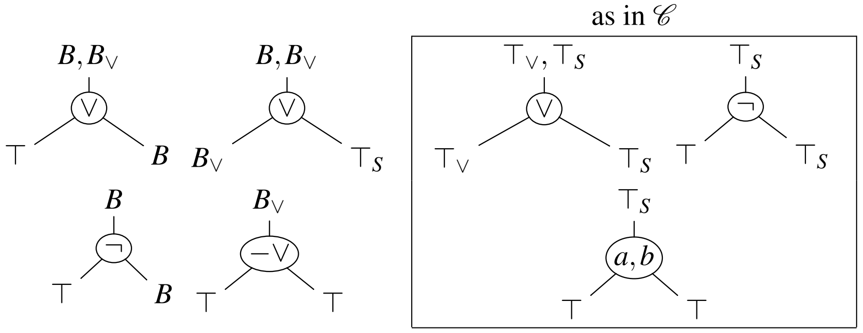
In this section we prove that the example from Section 3 is actually strongly unambiguous.

**Proposition 4.1** *The complement of language $L(\mathscr{C})$ is recognised by an unambiguous tree automaton.*

**Proof:**

Recall that, by Lemma 3.5, $\overline{L(\mathscr{C})}$ is the disjoint union of $W$ and $L(\mathscr{C}_N)$. Since the class of unambiguous languages is closed under disjoint unions, and $\mathscr{C}_N$ is an unambiguous automaton, we only need to show how to recognise incorrectly shaped trees unambiguously. It can be done by finding the rightmost incorrect branch. The proof that the rightmost incorrect branch exists is analogous to the proof of Lemma 2.1.

The automaton uses states $B$, $B_\vee$, $\top_S$, $\top_\vee$, $\top$. States $B$ and $B_\vee$ are used to track the branch and $\top_S$ serves showing non-existence of incorrect branch in a subtree. State $B$ is of rank 2; $\top_S$ and $B_\vee$ of rank 1; the remaining states are of rank 0. The initial state is $B$. The transitions are as follows:



where $-\vee$ is any label different than $\vee$.

Each run of the automaton has exactly one branch that is labelled (whole or up to some node) with states $B$ and $B_\vee$. The ranks of the branch fulfill the parity condition if a node with label different than $\vee$ occurs as a left descendant of a $\vee$-node — then the branch is as in point 2 of the definition of set $W$; or if it turns right infinitely many times and is whole labelled with $\vee$ or $\neg$ — then it reflects point 1 of the definition of set $W$. The subtrees diverging to the right from the branch are well shaped — as in automaton $\mathscr{C}$, $\top_S$ guarantees this. Therefore, the automaton is unambiguous — the right-most incorrect branch has to be selected. □

The following theorem summarises the topological results of this paper:

**Theorem 4.2** *There is a strongly unambiguous tree language that is not in $\sigma(\Sigma^1_1)$ class.*

# 5 Related Work

In [10] the authors consider the difference hierarchy of analytic sets. They show a sequence of regular tree languages hard for the levels $D_\alpha(\Sigma^1_1)$ of the hierarchy, for $\alpha < \omega^\omega$. Then, on page 10, they note that the languages (even the first one in the sequence) are not recognised by unambiguous tree automata.

The base for the construction of Finkel and Simonnet is the set:

$$\{t \in T_{\{0,1\}} : \text{there is a branch in } t \text{ with infinitely many labels } 1\}$$

If we look at the construction there, we see that we can use unambiguous set $G$ from this paper as a base, and hardness results still hold. Actually the proofs remain exactly the same, since they only use $\Sigma^1_1$-hardness of the basic set.

Now we note that each automaton built during the construction is unambiguous if we use unambiguous automaton $\mathscr{G}$ recognising basic set $G$ and unambiguous automaton $\mathscr{L}$ recognising its complement (i.e. set $L$ from Section 2) as basic building blocks. Indeed, for given tree $t$, the automaton described in the proof of Lemma 4.5 in [10] always selects the path corresponding to the smallest ordinal $\omega^{n-1} \cdot a_{n-1} + \omega^{n-2} \cdot a_{n-2} + \cdots + \omega \cdot a_1 + a_0$ for which the tree $t_{l^{a_{n-1}} r l^{a_{n-2}} r \ldots r l^{a_0}}$ belongs to $G$, and proves that all paths corresponding to less ordinals end up in something that does not belong to $G$ (i.e. belongs to $L$). Unambiguous automaton $\mathscr{G}$ is used to verify belonging to $G$ and $\mathscr{L}$ is used to verify belonging to $L$.

As a result, using example languages $G$ and $L$ from this paper and the construction from the paper [10] by Finkel and Simonnet, we get a sequence of unambiguous languages hard for the classes $D_\alpha(\Sigma^1_1)$, for $\alpha < \omega^\omega$. We give this fact here only as a note, because all the languages in the sequence reduce to the language from Section 3. This is because each set at any countable level of the difference hierarchy of analytic sets is, by the definition, in $\sigma(\Sigma^1_1)$.

We add a note by an anonymous reviewer. The game tree language $W_{(0,2)}$ (one of the languages used in [1] to show strictness of alternating index hierarchy) does not belong to the class $\sigma(\Sigma^1_1)$, which is an answer to the question asked in the last paragraph of [10]. Indeed, language $L(C)$ is of index $(0,2)$ and thus is Wadge-reducible to language $W_{(0,2)}$ by the result cited in [10, Lemma 5.2].

# Acknowledgements

The author wants to thank Michał Skrzypczak and Henryk Michalewski for their support and advice in some topological proofs. Many thanks also go to Damian Niwiński and the anonymous reviewers for very valuable comments that have led to a significant improvement of the paper.

## References

[1] André Arnold (1999): *The $\mu$-calculus alternation-depth hierarchy is strict on binary trees*. ITA 33(4/5), pp. 329–340. Available at `http://dx.doi.org/10.1051/ita:1999121`.

[2] André Arnold & Damian Niwiński (1992): *Fixed point characterization of weak monadic logic definable sets of trees*. In: Tree Automata and Languages, pp. 159–188.

[3] Marcin Bilkowski (2010): personal communication.

[4] Mikołaj Bojańczyk (2011): *Weak MSO with the Unbounding Quantifier*. Theory Comput. Syst. 48(3), pp. 554–576. Available at `http://dx.doi.org/10.1007/s00224-010-9279-2`.

[5] Mikolaj Bojańczyk & Thomas Colcombet (2006): *Bounds in $\omega$-Regularity*. In: LICS, pp. 285–296. Available at `http://doi.ieeecomputersociety.org/10.1109/LICS.2006.17`.

[6] Julian C. Bradfield (1999): *Fixpoint alternation: Arithmetic, transition systems, and the binary tree*. ITA 33(4/5), pp. 341–356. Available at `http://dx.doi.org/10.1051/ita:1999122`.

[7] Arnaud Carayol & Christof Löding (2007): *MSO on the Infinite Binary Tree: Choice and Order*. In: CSL, pp. 161–176. Available at `http://dx.doi.org/10.1007/978-3-540-74915-8_15`.

[8] Arnaud Carayol, Christof Löding, Damian Niwiński & Igor Walukiewicz (2010): *Choice functions and well-orderings over the infinite binary tree*. Central European Journal of Mathematics 8, pp. 662–682. Available at `http://dx.doi.org/10.2478/s11533-010-0046-z`.

[9] Thomas Colcombet (2012): *Forms of Determinism for Automata (Invited Talk)*. In: STACS, pp. 1–23. Available at `http://dx.doi.org/10.4230/LIPIcs.STACS.2012.1`.

[10] Olivier Finkel & Pierre Simonnet (2009): *On Recognizable Tree Languages Beyond the Borel Hierarchy*. Fundamenta Informaticae 95(2-3), pp. 287–303. Available at `http://dx.doi.org/10.3233/FI-2009-151`.

[11] Szczepan Hummel & Michał Skrzypczak (2012): *The Topological Complexity of MSO+U and Related Automata Models*. Fundamenta Informaticae 119(1), pp. 87–111.

[12] Alexander S. Kechris (1995): *Classical Descriptive Set Theory*. Graduate Texts in Mathematics 156, Springer-Verlag.

[13] Yiannis N. Moschovakis (2009): *Descriptive Set Theory: Second Edition*. Mathematical Surveys and Monographs 155, American Mathematical Society.

[14] Damian Niwiński (1986): *On Fixed-Point Clones (Extended Abstract)*. In: ICALP, pp. 464–473. Available at `http://dx.doi.org/10.1007/3-540-16761-7_96`.

[15] Damian Niwiński & Igor Walukiewicz (1996): *Ambiguity problem for automata on infinite trees*. Unpublished note.

[16] Damian Niwiński & Igor Walukiewicz (2003): *A gap property of deterministic tree languages*. Theor. Comput. Sci. 1(303), pp. 215–231. Available at `http://dx.doi.org/10.1016/S0304-3975(02)00452-8`.

[17] Michael O. Rabin (1969): *Decidability of Second-Order Theories and Automata on Infinite Trees*. Transactions of the AMS 141, pp. 1–23.

[18] Michael O. Rabin (1970): *Weakly Definable Relations and Special Automata*. Mathematical Logic and Foundations of Set Theory, pp. 1–23.