Regular Languages of Infinite Trees That Are Boolean Combinations of Open Sets

Mikołaj Bojańczyk and Thomas Place*

University of Warsaw

Abstract. In this paper, we study boolean (not necessarily positive) combinations of open sets. In other words, we study positive boolean combinations of safety and reachability conditions. We give an algorithm, which inputs a regular language of infinite trees, and decides if the language is a boolean combination of open sets.

1 Introduction

In this paper, we work with infinite binary trees labeled by a finite alphabet. The set of trees can be interpreted as a compact metric space. The distance between two different trees is 2^{-n}, where n is the smallest depth where the two trees are different. In the topology induced by this distance, a set of trees L is open if for every tree $t \in L$, there is a finite prefix of t such that changing nodes outside the prefix does not affect membership in L. In other words, an open set is a reachability language. We are interested in understanding finite boolean combinations, not necessarily positive, of open sets. The main result of this paper is:

Theorem 1.1. *The following problem is EXPTIME complete. Given a nondeterministic parity automaton on infinite trees, decide if the recognized language is a boolean combination of open sets.*

In other words, this paper provides an effective characterization of boolean (not necessarily positive) combinations of open sets, within the class of regular languages of infinite trees.

A similar version of the problem, where one asks if L is simply an open set, and not a finite boolean combination of open sets, is significantly simpler. Here is the solution to this simpler problem which is folklore to the best of our knowledge. The key observation is that the topological closure of a tree language L is the set

$$closure(L) = \{t : \text{every finite prefix of } t \text{ can be extended to some tree in } L\}.$$

A language L is open if and only if its complement L^c satisfies $L^c = closure(L^c)$. Automata for both L^c and $closure(L^c)$ can be computed based on the automaton for L, and then one can test two regular languages for equality.

* Both authors supported by ERC Starting Grant "Sosna". A full version of this paper can be found at `www.mimuw.edu.pl/~bojan`.

A. Czumaj et al. (Eds.): ICALP 2012, Part II, LNCS 7392, pp. 104–115, 2012.

The difficulty in Theorem 1.1 is dealing with the boolean combinations.

Our approach to the problem uses forest algebra for infinite trees [4]. We intended to achieve two complementary goals: use the algebra to understand boolean combinations of open sets; and use boolean combinations of open sets to understand the algebra.

Goal 1: Understand Boolean Combinations of Open Sets. We believe that giving an effective characterization for a class $\mathscr{L}$ of regular languages can be the most mathematically rewarding thing that one can do with $\mathscr{L}$. The ostensible goal of an effective characterization – the algorithm deciding membership in $\mathscr{L}$ – is usually less interesting than the insight into the structure of $\mathscr{L}$ that is needed to get the algorithm. A famous example is the theorem of Schützenberger and McNaughton/Papert, which makes a beautiful connection between logic and algebra: a word language is definable in first-order logic if and only if its syntactic monoid is aperiodic [8,6].

We believe that our study of boolean combinations of open sets achieves this goal. We discover that this class of languages has a rich structure, which is much more complex in the case of infinite trees than in the case of infinite words. On our way to Theorem 1.1, we provide three conditions which are equivalent to being a boolean combination of open sets, see Theorem 5.3. Two of these conditions are stated in terms of games, and one is stated in terms of algebraic equations. We believe that each of these conditions are interesting in their own right.

Goal 2: Understand Algebra for Infinite Trees. The algebraic theory of languages of finite words is well studied, using monoids and semigroups, see the book by Straubing [9]. The algebraic theory of languages of infinite words is also well understood, see the book by Perrin and Pin [7]. There has been quite a lot of recent work on algebra for finite trees [2], but the theory is still not mature. Finally, the algebraic theory of infinite trees is very far from being understood, despite some work [4,1].

We believe that our study of boolean combinations of open sets has highlighted the kind of tools that might be important in the algebraic theory of infinite trees. An important theme is the use of games. As mentioned previously, we characterize boolean combinations of opens sets in terms of games, and a set of two identities. Even in the identities, there is a hidden game, which is played in the algebra. We see this as evidence that the algebraic theory of infinite trees will need to take games into account.

Organization of the Paper. In Section 2 we give our first characterization of boolean combination of open sets using games. Note that this characterization is not specific to trees and works for all topological spaces. Unfortunately, it is not effective. In Section 3, we provide basic definitions for trees and algebra. In Section 4, we make a sharper analysis of the non-effective characterization of Section 2 in the setting of trees. Finally, in Section 5 we state our effective characterization using algebraic identities.

2 A Game Characterization

We begin by studying boolean combinations of open sets in arbitrary topological spaces. Fix a topological space. In this paper, we are interested in the topological space of infinite trees, but the discussion in this section works for all spaces.

Let $X_1, \ldots, X_n$ be arbitrary subsets of the topological space. We define a game

$$\mathscr{H}(X_1, \ldots, X_n)$$

which is played by two players, called Alternator and Constrainer. The game is played in n rounds. At the beginning of each round $i \in \{1, \ldots, n\}$, there is an open set U_i. Initially, U_1 is the whole space. Round i of the game is played as follows.

- Alternator chooses a point $x_i \in U_i \cap X_i$. If there is no such point x_i, the game is interrupted and Constrainer wins immediately. Otherwise,
- Constrainer chooses an open set $U_{i+1} \ni x_i$, and the next round is played.

If Alternator manages to survive n rounds, then he wins.

A base for a topology is a set of 'base open sets' such that open sets are obtained as infinite unions of these sets. The following lemma shows that the rules of the game could be changed such that Constrainer can only pick base open sets.

Lemma 2.1. *Choose some base for the topology. If Constrainer has a winning strategy, then he has a winning strategy which uses base open sets for* $U_1, \ldots, U_n$.

Suppose that X is a set and $n \in \mathbb{N}$. We write $\mathscr{H}_{\in \notin}(X, n)$ for the game where Alternator needs to alternate n times between X and its complement, that is:

$$\mathscr{H}(X_1, \ldots, X_n) \qquad \text{where } X_i = \begin{cases} X & \text{when } i \text{ is odd} \\ \text{the complement of } X & \text{when } i \text{ is even} \end{cases}.$$

Example 2.2. Consider the space of real numbers, and let X be the rational numbers. Then for every n, Alternator wins the game $\mathscr{H}_{\in \notin}(X, n)$.

Example 2.3. In the real numbers, let X be the complement of $\{1/n : n \in \mathbb{N}\}$. Alternator wins $\mathscr{H}_{\in \notin}(X, 3)$. In the first round, Alternator plays $0 \in X$. In the second round, Alternator plays $1/n \notin X$ for some large n depending on Constrainer's move. In the third round, Alternator plays $1/n + \epsilon \in X$, for some small ϵ depending on Constrainer's move. Constrainer wins $\mathscr{H}_{\in \notin}(X, n)$ for $n \geq 4$.

Proposition 2.4. *The following conditions are equivalent for a set* X*:*

- X *is a finite boolean combination of open sets*
- *Constrainer wins the game* $\mathscr{H}_{\in \notin}(X, n)$ *for all but finitely many* n.

Refinement lemma. We now state a lemma, which shows that if the topological space is a metric space (this is the case of trees) Alternator's winning sets can be refined in an arbitrary finite way.

Lemma 2.5. *Assume the topological space is a metric space and let* $X_1, \ldots, X_n$ *be sets. For* $i \in \{1, \ldots, n\}$*, let* $\mathcal{Y}_i$ *a finite family of sets partitioning* X_i*. If Alternator wins*

$$\mathscr{H}(X_1, \ldots, X_n)$$

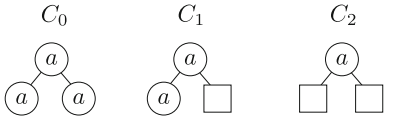
then there exist $Y_1 \in \mathcal{Y}_1, \ldots, Y_n \in \mathcal{Y}_n$ *such that Alternator wins*

$$\mathscr{H}(Y_1, \ldots, Y_n).$$

3 Preliminaries on Trees

Trees. We use possibly infinite trees where every node has zero or two children. For a finite alphabet A, we denote by H_A the set of infinite binary trees labeled over A. Notions of node, leaf, child, root, descendant are defined as usual. We write '$<$' the descendant relation (the smallest node being the root of the tree). If t is a tree and x a node of t, we write $t(x)$ the label of x in t, and $t|_x$ for the *subtree of* t *at* x.

Multicontexts. A *multicontext* is a tree with some distinguished unlabeled leaves called its *ports*. The number of ports is called the *arity*, there might be infinitely many ports. Given a multicontext C and a valuation η which maps ports to trees, we write $C[\eta]$ for the tree obtained by replacing each port x by the tree $\eta(x)$. A tree $C[\eta]$ is said to *extend* the multicontext C, conversely the multicontext C is said to be a *prefix* of the tree $C[\eta]$. The set of all trees extending a multicontext C is denoted by $C[*]$. The following picture shows three multicontexts, with arities $0, 1$ and 2, with the ports depicted by squares.



The multicontext C_0 is a tree, and $C_0[*]$ is $\{C_0\}$. The multicontext C_1 is a prefix of every tree where the root label is a, and the left child of the root is a leaf with label a. Finally, C_2 is a prefix of every tree with root label a. We are mostly interested in finite prefixes, which are multicontexts where every path ends in a leaf, which is either a port or a normal leaf.

Contexts. A *context* is a multicontext with exactly one port. We write V_A the set of contexts over A. We write C, D for contexts. Given two contexts C, D, we write $C \cdot D$ for the context obtained by replacing the port of C with D. One can verify that $\cdot$ is associative, therefore, $(V_A, \cdot)$ is a monoid (with the empty context, denoted by $\Box$, as neutral element). V_A also acts on H_A, with $C \cdot t$ defined as the tree obtained by replacing the port of C with t. Finally, we write C^∞ for the infinite tree $C \cdot C \cdot C \cdots$.

3.1 Tree Languages and Algebra

We are mainly concerned with regular languages of infinite trees. This is the class of languages of infinite trees that is recognized by nondeterministic parity automata; or equivalently recognized by alternating parity automata; or equivalently can be defined in monadic second-order logic. See [5].

Recall that our goal is to decide if a given regular language L of infinite trees is a boolean combination of open sets. It will be important for us to work with a canonical representation of L. As our canonical representation, we use equivalence classes of trees and contexts with respect to a natural Myhill-Nerode style equivalence, see below.

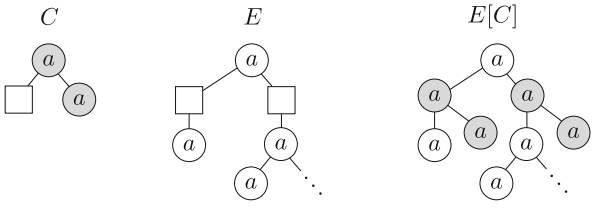
The equivalence classes form a kind of algebra, which is similar to the forest algebra for infinite trees from [4]. The similarity is that both algebras represent infinite trees. The difference is that the algebra in [4] represents finitely branching unranked trees, while the algebra in this paper represents binary trees. We do not use unranked trees because for unranked finitely branching trees, the topological space is not compact. This is because there is no converging subsequence in a sequence of trees where the n-th tree has n children of the root.

Myhill-Nerode Equivalence. Fix L a language of trees over an alphabet A. We define two Myhill-Nerode equivalence relations: one for trees and one for contexts.

Let C be a multicontext, possibly with infinitely many ports. For a tree t, we write $C[t]$ for the tree obtained by putting t in all ports of C. In the Myhill-Nerode equivalence for trees, we say trees t and t' are L-equivalent if

$$C[t] \in L \Leftrightarrow C[t'] \in L \qquad \text{for every multicontext } C.$$

To give a similar definition for contexts, we use a variant of multicontexts where the ports can be substituted by contexts and not trees. Such a multicontext is called a *context environment*. Formally speaking, a context environment is defined like a multicontext, except that the ports have exactly one child (instead of being leaves). Given a context environment E and a context C, we write $E[C]$ for the tree obtained by substituting C for every port of E, as in the following picture:



We define two contexts C and C' to be L-equivalent if

$$E[C] \in L \Leftrightarrow E[C'] \in L \qquad \text{for every context environment } E.$$

The Algebra. We write H_L for the equivalence classes of trees with respect to L, and V_L for the equivalence classes of contexts with respect to L. We write:

$$\alpha : (H_A, V_A) \to (H_L, V_L)$$

for the two-sorted function which maps trees and contexts to their L-equivalence classes; this function is called the *syntactic morphism of* L. We use the name *tree type* for elements of H_L and *context type* for elements of V_L. It is not difficult to show that both H_L and V_L are finite when L is regular. The syntactic morphism can be computed based on a nondeterministic tree automaton recognizing L, in exponential time [4]. Finiteness of H_L and V_L is necessary but not sufficient for regularity, for instance both H_L and V_L are finite for any language defined in the logic MSO+U [3].

Lemma 3.1. *The following operations respect L-equivalence.*

1. *For every multicontext D, the operation: $t \mapsto D[t]$.*
2. *For every context environment E, the operation: $C \mapsto E[C]$.*
3. *The composition of contexts $(C_1, C_2) \mapsto C_1 \cdot C_2$.*
4. *Substituting a tree in the port of a context: $(C, t) \mapsto C \cdot t$.*
5. *Infinite iteration of a context: $C \mapsto C^\infty$.*
6. *For every letter a, the operations $t \mapsto a(t, \Box)$ and $t \mapsto a(\Box, t)$.*

It follows that the above operations can be applied to elements of the syntactic algebra.

Idempotents. Given any finite monoid V, there is (folklore) a number $\omega(V)$ (denoted by ω when V is understood from the context) such that for each element v of V, v^ω is an idempotent: $v^\omega = v^\omega v^\omega$.

4 Boolean Combinations of Open Sets of Trees

As mentioned in the introduction, we use prefix topology on trees, which yields a topology identical to that of the Cantor space. In this topology, a *base open set* is defined to be any set $C[*]$, where C is a *finite* multicontext. Open sets are defined to be arbitrary unions of base open sets. This topology is the same as the topology generated by a distance, which says that trees are at distance 2^{-n} where n is the smallest depth where the two trees differ. This paper is about finite boolean combination of open sets. Typical boolean combinations of open sets include

- Trees over alphabet $\{a, b, c\}$ which contain at least one a and no b's.
- Trees over alphabet $\{a, b\}$ which contain two or five a's.

Languages, which are not boolean combinations of open sets include

- Trees over alphabet $\{a, b\}$ with finitely many a's.
- Trees over alphabet $\{a, b\}$ with a finite and even number of a's.

Let us revisit the game from Proposition 2.4 in the case of trees. In this special case, points are trees. By Lemma 2.1, we may assume that Constrainer uses base open sets, which are finite multicontexts. The game begins with the whole space, which corresponds to the empty multicontext. In each round, Alternator chooses a tree that extends the current multicontext, and then Constrainer chooses a finite multicontext that is a prefix of the tree chosen by Alternator.

Example 4.1. Consider an alphabet $\{a, b\}$ and the language L="infinitely many a's". It is not difficult to see that Alternator can win the game $\mathscr{H}_{\in \notin}(L, n)$ for every $n \in \mathbb{N}$. This is because every finite multicontext can be extended to a tree with finitely many a's, or to a tree with infinitely many a's. By Proposition 2.4, L is not a boolean combination of open sets.

Proposition 2.4 helps us understand finite boolean combinations of open sets, but it is not an effective characterization. To be effective, we should be able to decide if Alternator wins $\mathscr{H}_{\in \notin}(L, n)$ for every n. The following simple lemma shows how to decide the winner for a given n.

Lemma 4.2. *Given regular tree languages $L_1, \ldots, L_n$, one can decide who wins $\mathscr{H}(L_1, \ldots, L_n)$. In particular, given L and n, one can decide who wins $\mathscr{H}_{\in \notin}(L, n)$.*

Proof. The statement "Alternator wins the game $\mathscr{H}(L_1, \ldots, L_n)$" can be formalized in monadic second-order logic on the complete binary tree, by a formula which can be computed based on the languages $L_1, \ldots, L_n$. Therefore, the winner can be decided using Rabin's theorem. □

The above lemma gives a semi-algorithm for deciding if a regular language is a finite boolean combination of open sets. For $n = 1, 2, \ldots$, use Lemma 4.2 to compute the winner of $\mathscr{H}_{\in \notin}(L, n)$. If Constrainer wins the game for some n, then he also wins the game for $n + 1, n + 2, \ldots$ and therefore the algorithm can terminate and declare that the L is a finite boolean combination of open sets. If the language is *not* a finite boolean combination of open sets, then the algorithm does not terminate.

Observe that even when the algorithm *does* terminate, it does multiple calls to Rabin's theorem, which has non-elementary complexity.

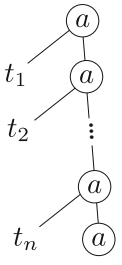
The main contribution of this paper is a finer analysis of the problem, which yields an algorithm (not a semi-algorithm) deciding if a tree language is a finite boolean combination of open sets.

4.1 The Infinite Game

Proposition 2.4 can be rephrased as: a language L is *not* a finite boolean combination of open sets if and only if player Alternator can win $\mathscr{H}_{\in \notin}(L, n)$ for arbitrarily large n. One could imagine a variant of the game, more difficult for Alternator, where infinitely many rounds have to be played. Call the infinite variant $\mathscr{H}_{\in \notin}(L, \infty)$. In Example 2.2, which is about rational numbers, Alternator can win the infinite game.

It is clear that if Alternator wins $\mathscr{H}_{\in\notin}(X,\infty)$, then he also wins $\mathscr{H}_{\in\notin}(X,n)$ for every n. We show a counterexample for the converse implication, which is a regular tree language. This counterexample language necessarily uses trees, because the converse implication holds for regular languages of infinite words.

The counterexample language L is the set of trees over $\{a,b\}$ of the form:



such that n is some natural number, and for each $i \in \{1,\ldots,n\}$, the tree t_i is either finite, or contains no b nodes. Observe that in a tree from L, the rightmost branch is necessarily finite.

Fact 1. *Alternator loses $\mathscr{H}_{\in\notin}(L,\infty)$, but wins $\mathscr{H}_{\in\notin}(L,n)$ for every n.*

5 The Effective Characterization

In this section, we present the main result of the paper.

The Context Game. So far we have worked with a game $\mathscr{H}(L_1,\ldots,L_n)$, for tree languages $L_1,\ldots,L_n$. We define a similar game for languages of contexts. Recall that contexts are defined as a special case of trees, with an additional port label that appears in exactly one leaf. From the distance on trees, we get a distance on contexts. This yields the definition of a game for a sequence $K_1,\ldots,K_n$ of context languages. To avoid confusion between trees and contexts, we denote the context game by $\mathscr{V}(K_1,\ldots,K_n)$.

Games on Types. Consider a language L and its syntactic morphism

$$\alpha_L : (H_A, L_A) \to (H_L, V_L).$$

Recall that by definition of the syntactic morphism, a type $h \in H_L$ is actually equal to the set of trees $\alpha_L^{-1}(h)$. Therefore, it makes sense to talk about the game $\mathscr{H}(h_1,\ldots,h_n)$ for a sequence of tree types. Likewise for context types. Define

$$\mathscr{H}_L \stackrel{\text{def}}{=} \{(h_1,\ldots,h_n) \in (H_L)^n : \ n \in \mathbb{N} \text{ and Alternator wins } \mathscr{H}(h_1,\ldots,h_n)\}$$
$$\mathscr{V}_L \stackrel{\text{def}}{=} \{(v_1,\ldots,v_n) \in (V_L)^n : \ n \in \mathbb{N} \text{ and Alternator wins } \mathscr{V}(v_1,\ldots,v_n)\}$$

A comment on notation is in order here. The sets $\mathscr{H}_L$ and $\mathscr{V}_L$ contain words, over alphabets H_L and V_L, respectively. Usually when dealing with words, one omits

the brackets and commas, and writes abc instead of (a, b, c). When the alphabet is V_L, this leads to ambiguity, since the expression vwu can be interpreted as: 1) a word with a single letter obtained by multiplying v, w, u in the context monoid V_L; or 2) a three-letter word over the alphabet V_L. These two interpretations should not be confused, so we write $(v_1, \ldots, v_n)$ for n-letter words over the alphabet V_L. For the sake of uniformity, we also write $(h_1, \ldots, h_n)$ for n-letter words in over the alphabet H_L, although there is no risk of ambiguity here.

Fact 2. *Both $\mathscr{H}_L$ and $\mathscr{V}_L$ are regular languages of finite words.*

Proof. Both languages are closed under removing letters. Every language closed under removing letters is regular, by Higman's lemma. □

The above fact is amusing, but useless, because it does not say how to compute automa for $\mathscr{H}_L$ and $\mathscr{V}_L$ as a function of the language L.[1] If we are not interested in efficiency, membership in $\mathscr{H}_L$ can be decided with Lemma 4.2. The same kind of algorithm works for $\mathscr{V}_L$. Later on, we give a more efficient algorithm.

$(h_1, \ldots, h_n) \in \mathscr{H}_L$	implies	$(C[h_1], \ldots, C[h_n]) \in \mathscr{H}_L$
$(v_1, \ldots, v_n) \in \mathscr{V}_L$	implies	$(E[v_1], \ldots, E[v_n]) \in \mathscr{H}_L$
$(v_1, \ldots, v_n), (w_1, \ldots, w_n) \in \mathscr{V}_L$	implies	$(v_1 w_1, \ldots, v_n w_n) \in \mathscr{V}_L$
$(v_1, \ldots, v_n) \in \mathscr{V}_L, (h_1, \ldots, h_n) \in \mathscr{H}_L$	implies	$(v_1 h_1, \ldots, v_n h_n) \in \mathscr{H}_L$
$(v_1, \ldots, v_n) \in \mathscr{V}_L$	implies	$(v_1^\infty, \ldots, v_n^\infty) \in \mathscr{H}_L$
$(h_1, \ldots, h_n) \in \mathscr{H}_L$	implies	$(a[\Box, h_1], \ldots, a[\Box, h_n]) \in \mathscr{V}_L$
$(h_1, \ldots, h_n) \in \mathscr{H}_L$	implies	$(a[h_1, \Box], \ldots, a[h_n, \Box]) \in \mathscr{V}_L$

Table 1. Closure properties of $\mathscr{H}_L$ and $\mathscr{V}_L$. C is a multicontext, E is a context environment, and a is a letter.

Lemma 5.1. *The sets $\mathscr{H}_L$ and $\mathscr{V}_L$ satisfy the closure properties in Table 1.*

Notice the similarity of Table 1 with the operations in Lemma 3.1. Another way of stating Lemma 5.1 is that for every $n \in \mathbb{N}$, $(\mathscr{H}_L, \mathscr{V}_L)$ restricted to sequences of length n is a subalgebra of the the n-fold power of the syntactic algebra (H_L, V_L).

We define the *alternation* of a word to be its length, after iteratively eliminating letters that are identical to their predecessors. The alternation of *abaabbb* is 4. We say that a set of words has *unbounded alternation* if it contains words with arbitrarily large alternation.

Proposition 5.2. *For a regular language L of infinite trees, the following conditions are equivalent.*

- *Alternator wins the game $\mathscr{H}_{\in \notin}(L, n)$ for infinitely many n.*
- *The set $\mathscr{H}_L$ has unbounded alternation.*

[1] To the best of our knowledge it is possible, although unlikely, that computing $\mathscr{H}_L$ and $\mathscr{V}_L$ is undecidable.

Proof. We begin with the top-down implication. We show that if Alternator wins the game $\mathscr{H}_{\in \notin}(L, n)$, then $\mathscr{H}_L$ contains a word of length n where every two consecutive letters are different. Suppose then that Alternator wins $\mathscr{H}_{\in \notin}(L, n)$, which means that he wins $\mathscr{H}(L_1, \ldots, L_n)$ where L_i is L for odd-numbered rounds and its complement for even-numbered rounds. Both L and its complement can be partitioned into tree types. By Lemma 2.5, Alternator wins $\mathscr{H}(h_1, \ldots, h_n)$ for some sequence of types, such that h_i is included in L or its complement, depending on the parity of i. In particular, the consecutive types are different.

We now do the bottom-up implication. Suppose that $\mathscr{H}_L$ has unbounded alternation. Since $\mathscr{H}_L$ is closed under removing letters, there must be some $g, h \in H_L$ such that $\mathscr{H}_L$ contains all the words

$$(g, h), (g, h, g, h), (g, h, g, h, g, h), \ldots. \tag{1}$$

Since g and h are different elements of the syntactic algebra, it follows that there must be some multicontext C such that the tree type $C[g]$ is contained in L, while the tree type $C[h]$ is disjoint with L. By applying Lemma 5.1 to (1), we conclude $\mathscr{H}_L$ contains all the words

$$\big(C[g], C[h]\big), \big(C[g], C[h], C[g], C[h]\big), \big(C[g], C[h], C[g], C[h], C[g], C[h]\big), \ldots.$$

It follows that Alternator can alternate arbitrarily long between the language L and its complement. □

The Main Theorem. So far, we have proved Propositions 2.4 and 5.2, which characterize finite boolean combinations of open sets in terms of games. Neither of these game characterizations is effective. We now add a final characterization, which uses identities and is effective.

Theorem 5.3 (Main Theorem). *For a regular language L of infinite trees, the following conditions are equivalent.*

1. *L is a finite boolean combination of open sets.*
2. *Constrainer wins the game $\mathscr{H}_{\in \notin}(L, n)$ for all but finitely many n.*
3. *The set $\mathscr{H}_L$ has bounded alternation.*
4. *The following identities are satisfied.*

$$u^\omega w w^\omega = u^\omega v w^\omega = u^\omega u w^\omega \qquad \text{if } (u, v, w) \in \mathscr{V}_L \text{ or } (w, v, u) \in \mathscr{V}_L \tag{2}$$

$$(u_2 w_2^\omega v)^\omega u_1 w_1^\infty = (u_2 w_2^\omega v)^\infty \qquad \text{if } (u_1, u_2) \in \mathscr{V}_L \text{ and } (w_1, w_2) \in \mathscr{V}_L \tag{3}$$

Theorem 5.3 implies Theorem 1.1 from the introduction, which says that one can decide if the language recognized by a nondeterministic parity automaton on infinite trees, is a boolean combination of open sets. Indeed: there are finitely many sequences of length two and three in $\mathscr{V}_L$. These sequences can be computed using Lemma 4.2. It is then sufficient to test if the identities from item 4 are valid by checking all combinations. A more detailed proof, together with the EXPTIME completeness, can be found in the full version of the paper.

In Propositions 2.4 and 5.2, we have shown that conditions 1, 2 and 3 in Theorem 5.3 are equivalent. It remains to show that conditions 3 and 4 are equivalent. The proof of the implication from 4 to 3 is the technical core of this paper, and is in the full version. Below we prove the much simpler converse implication, which at least can serve to illustrate the identities.

Implication from 3 to 4. We prove the contrapositive: if one of the identities (2) or (3) is violated, then $\mathscr{H}_L$ has unbounded alternation.

Suppose first that (2) is violated. We will show that $\mathscr{V}_L$ has unbounded alternation. This is enough, by the following lemma.

Lemma 5.4. *If $\mathscr{V}_L$ has unbounded alternation, then so does $\mathscr{H}_L$.*

The assumption that (2) is violated says that there are $u, v, w \in V_L$ such that

$$(u, v, w) \in \mathscr{V}_L \qquad \text{or} \qquad (w, v, u) \in \mathscr{V}_L,$$

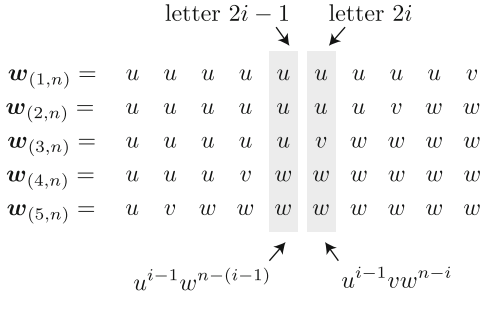
but the three context types $u^\omega w w^\omega$, $u^\omega v w^\omega$ and $u^\omega u w^\omega$ are not all equal. If the three context types are not equal, then second one must be different from either the first one or the third one. We only do the proof for the case when $(u, v, w) \in \mathscr{V}$ and when $u^\omega w w^\omega \neq u^\omega v w^\omega$. For nonzero $n \in \mathbb{N}$ and $i \in \{1, \ldots, n\}$, define

$$\boldsymbol{w}_{(i,n)} = (\overbrace{u, \ldots, u}^{2n-2i+1 \text{ times}}, v, \overbrace{w, \ldots, w}^{2(i-1) \text{ times}}) \in (H_L)^{2n}.$$

This word is obtained from (u, v, w) by duplicating some letters, and therefore it belongs to $\mathscr{V}_L$. For some n, consider the words

$$\boldsymbol{w}_{(1,n)}, \ldots, \boldsymbol{w}_{(n,n)} \in \mathscr{V}_L.$$

These are n words of length $2n$. Let us multiply all these words coordinate-wise, yielding a word $\boldsymbol{w}$, also of length $2n$, which is depicted in the following picture:



Choose some k, and take $n = k \cdot \omega + 1$, and $i \in \{\omega, 2\omega, \ldots, (k-1) \cdot \omega\}$. Consider letters $2i + 1$ and $2i + 2$ in the word $\boldsymbol{w}$, which are

$$u^i w^{n-i} = u^\omega w w^\omega \qquad u^i v w^{n-i-1} = u^\omega v w^\omega.$$

By assumption, these letters are different, and therefore the word $\boldsymbol{w}$ has alternation at least k. Because k was chosen arbitrarily, it follows that $\mathscr{V}_L$ has unbounded alternation.

Consider now the case when (3) is violated. This means $\mathscr{V}_L$ contains pairs (u_1, u_2) and (w_1, w_2) such that for some $v \in V_L$,

$$e^\infty \neq e u_1 w_1^\infty \qquad \text{for } e \stackrel{\text{def}}{=} (u_2 w_2^\omega v)^\omega.$$

Lemma 5.5. *Let u_1, u_2, w_1, w_2 and e be as above. If $(h_1, \ldots, h_n) \in \mathscr{H}_L$, then*

$$(e^\infty, eh_1, \ldots, eh_n) \in \mathscr{H}_L \tag{4}$$

$$(eu_1 w_1^\infty, eh_1, \ldots, eh_n) \in \mathscr{H}_L \tag{5}$$

Using the lemma, one shows by induction that for every n, the sequence which alternates n times between e^∞ and $e^\infty u_1 w_1^\infty$ belongs to $\mathscr{H}_L$. This completes the proof of the implication from 3 to 4, and also of Theorem 5.3.

6 Conclusion

We have proved an effective characterization of boolean combination of open sets. We hope that the characterization sheds some light on the nature of such languages. Also, we hope that the technical tools we developed, involving both algebra and games, will be useful in future work on regular languages of infinite trees. One class of particular interest is Weak-MSO (i.e. MSO where set quantification is restricted to finite sets). This class can be characterized by wreath products of boolean combinations of open sets.

References

1. Blumensath, A.: An algebraic proof of Rabin's theorem (unpublished manuscript)
2. Bojańczyk, M.: Algebra for trees. In: Handbook of Automata Theory. European Mathematical Society Publishing House (to appear)
3. Bojańczyk, M.: A Bounding Quantifier. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 41–55. Springer, Heidelberg (2004)
4. Bojańczyk, M., Idziaszek, T.: Algebra for Infinite Forests with an Application to the Temporal Logic EF. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 131–145. Springer, Heidelberg (2009)
5. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS, vol. 2500. Springer, Heidelberg (2002)
6. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press (1971)
7. Perrin, D., Pin, J.-É.: Infinite Words. Elsevier (2004)
8. Schützenberger, M.P.: On finite monoids having only trivial subgroups. Information and Control 8 (1965)
9. Straubing, H.: Finite Automata, Formal Languages, and Circuit Complexity. Birkhäuser (1994)