Two-Way Alternating Automata and Finite Models

Mikołaj Bojańczyk

Uniwersytet Warszawski, Wydział MIM, Banacha 2, Warszawa,Polska

Abstract. A graph extension of two-way alternating automata on trees is considered. The following problem: „does a given automaton accept any *finite* graph?" is proven EXPTIME complete. Using this result, the decidability of the finite model problem for the modal μ-calculus with backward modalities is shown.

1 Introduction

Alternating tree automata were introduced Muller and Schupp [12]. In terms of expressibility these automata define the same class of languages as the simpler nondeterministic tree automata on the one hand and as the complex monadic second order theory on trees (S2S) on the other. Nevertheless, the formalism of alternating automata offers a good balance between logical manageability and computational complexity. Testing emptiness for alternating tree automata is far easier than the non-elementary procedures in S2S; on the other hand, closure under conjunction and negation is trivial for alternating automata and very difficult for nondeterministic automata (cf. the famous "complementation lemma" [13]).

An important variant of alternating tree automata are *alternating tree automata with the parity condition*, introduced by Emmerson and Jutla in [3]. The parity condition assigns to each of the finite number of states of the automaton a natural number and those runs of the automaton are considered accepting where the least number occurring infinitely often is even. The parity condition is of growing importance in automata theory, particularly in connection with games [6, 11].

Alternating automata are very closely connected to Kozen's modal μ-calculus [9]. Because of this close correspondence, the μ-calculus is a standard application for alternating automata [12, 2]. In the same spirit, the satisfiability problem for the propositional μ-calculus with backward modalities is proved decidable by Vardi [16] via a reduction to two-way alternating automata.

The $\mu-$calculus with *backwards modalities* augments the one-way μ-calculus with quantification over backward modalities, denoted by $\Diamond^-$ and $\Box^-$. A formula of the form $\Diamond^- \phi$ states that ϕ occurs in some predecessor of the current state, similarly for $\Box^-$. Analogously to the μ-calculus, a two-way automaton can, apart from the usual forward moves of one-way alternating automata, make backward moves [14].

Another application of the two-way automaton can be found in [5], where it is used to solve another satisfiability problem – this time for Guarded Fixed Point Logic, an extension of the Guarded Fragment [7] by fix-point operators.

There is an interesting common denominator to Guarded Fixed Point Logic, the μ-calculus with backward modalities and two-way alternating automata: none of them have a finite model property.

W say a logic has the *finite model* property if every satisfiable sentence is satisfiable in some finite structure. Modal logic and even the modal μ-calculus have the finite model property; the μ-calculus with backward modalities does not (consider the sentence $\nu X.(\Diamond X \wedge \mu Y.\Box^{-} Y)$). A similar situation occurs in the Guarded Fragment: the fix-point extension no longer has the finite model property, contrary to the "bare" Guarded Fragment [4] and some of its other extensions (most notably the Loosely Guarded Fragment [8]).

These observations give rise to the following decision problem: „Is a given sentence of the modal $\mu-$calculus with backward modalities (or guarded fixed point logic) satisfiable in some *finite* model?" While tackling this problem, we took the automata approach. However, for reasons sketched out below, it turns out that we need a new definition of two-way alternating automata.

Most modal logics have a bisimulation-invariance property and the two-way μ-calculus is no exception. In particular, a sentence of the two-way μ-calculus cannot distinguish between a model and its tree unraveling (technically, its two-way tree unraveling) and so every satisfiable sentence is satisfiable in a tree-like structure. Thus for the purpose of deciding satisfiability, one can constrain attention to tree models. This was the approach taken by Vardi; in fact his automata were alternating two-way automata on *infinite trees*.

As much as the tree model property is helpful in researching the satisfiability problem, things get more complicated where the finite model problem is concerned. The reason is that, unfortunately, finite models rarely turn out to be trees. There are finitely satisfiable sentences that have no finite tree models, for instance $\nu X.\Diamond X$. For this reason, while investigating the finite model problem we will consider automata on arbitrary graphs, not on trees. These automata correspond very closely to the μ-calculus and consequently also lack a finite model property.

The suitable example, presented in more detail in Example 1, is as follows. An alternating automaton is constructed that accepts a graph if every reachable state has a successor and every backward path is finite. From this follows that every graph accepted by this automaton must be infinite. A typical graph accepted by $\mathcal{A}$ has natural numbers as vertices, with edges representing the successor relation.

The paper [1] presented an incomplete solution to the finite model problem. The automata considered used the *Büchi acceptance condition* or, equivalently, the parity condition for the colors $\{0, 1\}$. The solution presented used a complicated technical pumping argument and the accompanying algorithm ran in doubly exponential time. Although an important step toward the solution of the general problem, the Büchi condition is insufficient for both the translations

from μ-calculus and the guarded logics which use the full power of the parity condition. This old result is improved here in two ways: we solve the finite model problem for the general parity condition, and we do this in singly exponential time, which turns out to be optimal:

Theorem 1
The finite model problem for alternating two-way automata with the full parity condition is EXPTIME complete

Here we present an outline of the proof. First of all, we immediately get rid of graphs and start working on trees, since tree automata are much easier to work with. A finite graph is represented as an infinite tree – its two-way tree unraveling. Of course, not all infinite trees represent finite graphs – for this a special condition, called *bounded signature*, must be satisfied. This way, the finite model problem is reduced to the emptiness problem for the tree language consisting of trees with bounded signature. Although this language is not regular itself, its emptiness is equivalent to the emptiness of an appropriate (effectively found) regular language.

2 The automaton

In this section we define our automata. Since our definition of acceptance uses games with the parity condition, we first briefly define them and recall some fundamental properties. A more detailed presentation can be found in [15].

2.1 Games with the parity condition

Games with the parity condition are powerful tool in the field of infinite tree automata. We take the game approach in defining the semantics of our two-way alternating automata.

Definition 2 (Parity condition game) A *game with the parity condition* is a tuple $\mathrm{G} = \langle V_0, V_1, E, v_0, \Omega \rangle$, where V_0 and V_1 are disjoint sets of *positions*, the function $\Omega : V = V_0 \cup V_1 \to \{0, \ldots, N\}$ is called the *coloring* function, $E \subseteq V \times V$ is the set of *edges*, and $v_0 \in V$ is some fixed *starting position*. We additionally assume that for every position $v \in V$, the set of outgoing edges $(v, w) \in E$ is finite.

The game is played by two players, 0 and 1, and consists of moving a token from one position in V to another along the edges of the graph. The first position is v_0. If the token is in a position from V_0, the player 0 makes the move, otherwise the second player decides. If at some point one of the players cannot make a move, she loses. Otherwise, the winner depends on the infinite sequence $v_0, v_1, \ldots$ of vertices visited in the game. This infinite play is winning for player 0 if the sequence $\Omega(v_0), \Omega(v_1), \ldots$ satisfies the parity condition defined below, otherwise the play is winning for player 1.

Definition 3 (Parity condition) An infinite sequence of natural numbers belonging to some finite set is said to satisfy the *parity condition* if the smallest number occurring infinitely often in the sequence is even.

The case when the set of natural numbers in question is $\{0,1\}$ is called the *Büchi condition*. The dual case of $\{1,2\}$ is called the *co-Büchi condition*.

A strategy for the player $i \in \{0,1\}$ is a mapping $s : V^* \times V_i \to V$ such that for each $v_0 v_1 \ldots v_j \in V^* V_i$, there is an edge in E from v_j to $s(v_0 v_1 \ldots v_j)$. We say a strategy is *memoryless* if $s(v_0 v_1 \ldots v_j)$ depends solely upon v_j. The concept of winning strategy is defined in the usual way. A very important theorem [3, 11], which will enable us to consider only memoryless strategies, says:

Theorem 4 (Memoryless determinacy theorem)
Every game with the parity condition is determined, *i. e. one of the players has a winning strategy. Moreover, the winner also has a memoryless winning strategy.*

2.2 Graphs and trees

In this paper, when speaking of graphs, we mean *labeled graphs with a starting position*. Such a graph is a tuple $G = \langle V, E, \Sigma, e, v_0 \rangle$, where V is the set of *vertices*, $E \subseteq V \times V$ is the set of *edges*, the *labeling* is a function $e : V \to \Sigma$ and $v_0 \in V$ is the *starting position*. We assume that the set Σ of *labels* is finite. Given a graph G, its set of vertices V, or *domain*, is denoted as $\mathrm{dom}(G)$. For $W \subset V$, the *induced* subgraph of G is the graph of domain W obtained by restricting the edges and labeling in G to W.

Given $k \in \mathcal{N}$, a *k-ary tree* is a special kind of graph whose domain is the set $[k]^*$ of finite sequences over $[k]$ and the edge set is $\{(v, v \cdot i) : v \in [k]^*, i \in [k]\}$. The starting position is the empty sequence ε and there are no restrictions on the labeling.

2.3 Nondeterministic automata

Definition 5 (Nondeterminitic parity automaton) A *nondeterministic parity automaton on k-ary trees* is the tuple:

$$\langle Q, q_0, \Sigma, \delta, \Omega \rangle$$

The finite set Σ is called the *alphabet*, Q is a finite set of *states*, $q_0 \in Q$ is called the initial state and the function $\Omega : Q \to \mathcal{N}$ assigns to each state q its *color* $\Omega(Q)$. The *transition function* δ assigns to each pair $(q, \sigma) \in Q \times \Sigma$ a set of *transitions* $\delta(q, \sigma) \subseteq Q^k$.

Given a k-ary tree $\langle [k]^*, E, \Sigma, e, \epsilon \rangle$, a *run* of the automaton is a function $\rho : [k]^* \to Q$ such that for each vertex $v \in [k]^*$,

$$\langle \rho(v \cdot 1), \ldots, \rho(v \cdot k) \rangle \in \delta(\rho(v), e(v))$$

A run ρ is *accepting* if $\rho(\epsilon) = q_0$ and on each infinite path $v_0, v_1, \ldots$, the sequence $\Omega(\rho(v_0)), \Omega(\rho(v_1)) \ldots$ satisfies the parity condition. An automaton *accepts* a tree if there exists an accepting run.

Theorem 6 (Rabin)
Regular tree languages are closed under boolean operations and set quantification

2.4 Alternating automata

Two-way alternating automata on infinite trees were studied by Vardi in [16] as a tool for deciding the satisfiability problem of the modal μ-calculus with backward modalities. As opposed to "normal" alternating automata, two-way automata can travel backwards across edges. For reasons explained in the introduction, when dealing with the finite model problem, we consider a graph version of the automata.

Definition 7 (Two-way alternating automaton) A *two-way alternating automaton* on Σ-labeled graphs is the tuple:

$$\langle Q_\exists, Q_\forall, q_0, \Sigma, \delta, \Omega \rangle$$

$Q_\exists, Q_\forall$ are disjoint finite sets of *states*, $q_0 \in Q = Q_\exists \cup Q_\forall$ is called the *starting state* and Ω is a function assigning to each state $q \in Q$ a natural number $\Omega(q)$ called the *color* of q. The *transition function* δ is of the form

$$\delta : Q \times \Sigma \to P(\{0, +, -\} \times Q)$$

Intuitively, a run of $\mathcal{A}$ over some graph G is based on a game between two players: $\exists$ and $\forall$. The automaton starts in state q_0 in the starting position of G. Afterwards, $\mathcal{A}$ moves around the graph, the player deciding which move to make depending on whether the current state is in $Q_\exists$ or $Q_\forall$. The set of possible moves depends on the value assigned by δ to current state of $\mathcal{A}$ and the label of the current position in G. A choice of the form $(q, +)$ (respectively $(q, -)$) means the automaton changes the state to q and must move to some position along a forward (respectively backward) edge. Choosing $(q, 0)$ does not change the position, only the state of the automaton. The winner is determined as in the parity game.

A precise definition is as follows. Given a labeled graph $G = \langle V, E, \Sigma, e, v_0 \rangle$ and a two-way alternating automaton $\mathcal{A} = \langle Q_\exists, Q_\forall, q_0, \Sigma, \delta, \Omega \rangle$, we define the game $\mathrm{G}(\mathcal{A}, G) = \langle V_0, V_1, E', v_0', \Omega' \rangle$:

- $V_0 = Q_\exists \times V$ and $V_1 = Q_\forall \times V$.
- $v_0' = (q_0, v_0)$.
- $((q, v)(q', v')) \in E'$ iff either:
  - $(0, q') \in \delta(q, e(v))$ and $v = v'$
  - $(-, q') \in \delta(q, e(v))$ and $(v', v) \in E$
  - $(+, q') \in \delta(q, e(v))$ and $(v, v') \in E$

– $\Omega'(q, v) = \Omega(q)$.

Definition 8 (Acceptance by the automaton) We say that the automaton $\mathcal{A}$ *accepts* a graph G under the strategy s if s is a winning strategy for player $\exists$ in the game $\mathrm{G}(\mathcal{A}, G)$. Such a strategy s is called *accepting*. We say $\mathcal{A}$ accepts graph G if there exists a strategy s such that $\mathcal{A}$ accepts G under s.

Note that without loss of generality we assume that accepting strategies are memoryless. We will conclude this section with an example of an automaton that accepts a graph, yet no finite one.

Example 1. Consider the following two-way automaton

$$\mathcal{A} = \langle \{q_x\}, \{q_y, q_z\}, q_y, \{a\}, \delta, \Omega \rangle$$

Let Ω be $\{(q_x, 0), (q_y, 0), (q_z, 1)\}$ and the transition function δ be:

$$\delta(q_x, a) = \{(+, q_y)\} \qquad \delta(q_y, a) = \{(0, q_x), (-, q_z)\} \qquad \delta(q_z, a) = \{(-, q_z)\}$$

We will examine the game $\mathrm{G}(\mathcal{A}, G)$, where $G = \langle \mathcal{N}, \{(n, n+1) : n \in \mathcal{N}\}, e, 0\} \rangle$, such that $e(n) = a$ for all $n \in \mathcal{N}$. Consider first the following example play. Since the starting state is q_y and the starting position is 0 , the play begins in $(q_y, 0)$. This is a game position for player $\forall$ – she has to choose a move from $\delta(q_y, a) = \{(0, q_x), (-, q_z)\}$, let's assume she picks $(0, q_x)$. This means we stay in the vertex 0 and now player $\exists$ has to choose from $\delta(q_x, a)$. There is only one possibility – namely $(+, q_y)$. This means $\exists$ has to choose a neighboring (in G) position along a forward edge. Again, $\exists$ has no choice, he has to choose 1; the game position is now $(q_y, 1)$. This goes on, through the game positions $(q_x, 1), (q_y, 2), (q_x, 2), \ldots$ until, say, we reach $(q_y, 10)$. Let's assume that this time player $\forall$ chooses $(-, q_z)$. Now it is her choice to choose a neighboring position in G, along a backward edge; she has to choose 9 – there is no other backward edge from 10. The play then goes on through game positions $(q_z, 9), (q_z, 8), \ldots, (q_z, 0)$ in which last position player $\forall$ loses for a lack of possible moves.

In the game $\mathrm{G}(\mathcal{A}, G)$ there are essentially two kinds of play: a finite play like the one above, where player $\exists$ wins, or an infinite one where player $\forall$ always chooses $(+, q_x)$ from $\delta(q_x, a)$. The play goes through positions $(q_y, 0)$, $(q_x, 0)$, $(q_y, 1)$, $(q_x, 1)$, ... The only color appearing infinitely often in this play is 0, thus, again, player $\exists$ wins.

So we see that in the game $\mathrm{G}(\mathcal{A}, G)$ player $\exists$ has a winning strategy, in other words, $\mathcal{A}$ accepts G. It can also be proven, that the automaton $\mathcal{A}$ accepts only graphs with an infinite forward path where no infinite backward path is ever reachable. In particular $\mathcal{A}$ accepts only infinite graphs.

3 The finite model problem

The example above is a motivation for the following problem: „does a given alternating two-way automaton accept some finite graph?" This is the problem – called the *finite model problem* – tackled in this section.

Our plan is as follows. In the next subsection we prove an auxiliary decidability result about certain graphs with red and green edges. This is then used to prove the decidability of the finite model problem.

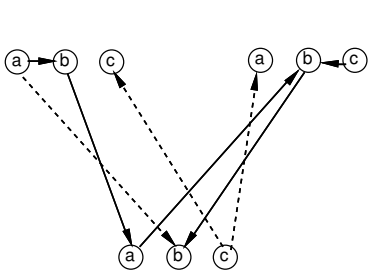
3.1 RG-graphs

An RG-graph is a directed graph with two types of edges – red ones and green ones. More formally, it is a tuple $\langle V, E_{\mathrm{r}}, E_{\mathrm{g}} \rangle$ where $E_{\mathrm{r}}, E_{\mathrm{g}} \subseteq V \times V$. RG-graphs have no labels.

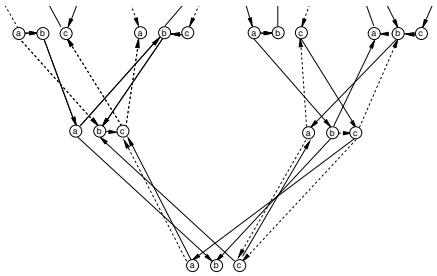
Definition 9 Assuming that for $i \in \mathbb{Z}$, $(v_i, v_{i+1}) \in E_{\mathrm{r}} \cup E_{\mathrm{g}}$:

- A *finite path* is a sequence $v_0 v_1 \ldots v_n$.
- A *backward infinite path* is a sequence $\ldots v_{-1} v_0$.
- A *forward infinite path* is a sequence $v_0 v_1 \ldots$
- A *path* is any one of the above.

Given a set X, an RG-graph over $\{\varepsilon, 0, \ldots, m\} \times X$ is (m, X)-*V-shaped* iff for all edges $((i,x),(j,y))$, either $i = \varepsilon$ or $j = \varepsilon$ or $i = j$. By VG^m_X we denote the set of (m, X)-V-shaped RG-graphs.



A 2-V-shaped graph over $\{a, b, c\}$



$\overline{t}$ for some tree over $\mathrm{VG}^2_{\{a,b,c\}}$

Given an m-ary tree t (which is not an RG-graph) whose vertices are labeled by elements of VG^m_X, the RG-graph $\overline{t}$ is defined as follows. The vertex set of $\overline{t}$ is $\mathrm{dom}(t) \times X$. If there is an edge from (i,x) to (j,y) in the label of v in t for $i, j \in \{\varepsilon, 0, \ldots, m\}$, then there is an edge from $(v \cdot i, x)$ to $(v \cdot j, y)$ of the same color in the graph $\overline{t}$. Note that $\overline{t}$ is not a tree.

The trees in the lemmas in this subsection are m-ary trees labeled by VG^m_X.

Definition 10 A tree t is *path bounded* iff there is a finite bound on the number of red edges in paths in $\overline{t}$.

Definition 11 A tree t is *path finite* iff there is no (infinite) path with an infinite number of red edges in $\overline{t}$

Given an $m \in \mathcal{N}$, a path is m-*acyclic* if it visits each vertex at most m times. Given $n \in \mathcal{N}$, we say a path π in a labeled tree is n-*upward* if there exist paths $\pi_1, \ldots, \pi_{n+1}$ and vertices $v_1 \leq \ldots \leq v_n$ such that:

- $\pi = \pi_1 v_1 \pi_2 v_2 \dots \pi_n v_n \pi_{n+1}$
- All v_i have the same label.
- π contains a red edge for some $i \in \{2, \dots, n\}$.

A path is n-vertical if it is n-upward or the reverse path is. The rather easy proof of the following lemma is omitted:

Lemma 1. *For every $m, n \in \mathcal{N}$ and every labeled tree t, there is a constant $M_t^{m,n}$ such that every m-acyclic path in t containing more than $M_t^{m,n}$ red edges is n-vertical.*

A labeled tree t is *regular* if it contains a finite number of nonisomorphic subtrees. The following observation is crucial in our proof:

Lemma 2. *A regular tree is path bounded iff it is path finite.*

Proof The left to right part is obvious. For the other direction, assuming that a regular tree t is not path bounded we will prove it is not path finite.

Let $\tau_1, \dots, \tau_n$ be the isomorphism types of t. Let t' be the $\{\tau_1, \dots, \tau_n\} \times \mathrm{VG}_X^m$-labeled tree obtained from t by additionally labeling each vertex with its isomorphism type. Take some path π in $\overline{t}$ with more than $M_{t'}^{|X|,|X|}$ edges. We can assume that π is acyclic, otherwise the proof would be done. Consider its projection (by erasing the X component) onto the tree t'. Obviously, this projection is $|X|$-acyclic and, using Lemma 1, we can find two comparable vertices v and v' in t' such that for some $x \in X$, (v, x) and (v', x) are linked in $\overline{t}$ by a path π' containing a red edge.

Since v and v' a roots of isomorphic trees (because they have the same label in t'), we can pile infinitely many copies of π' on each other obtaining a path that is either forward or backward infinite. □

The reason why we consider path finite trees is that they are regular:

Lemma 3. *There is a co-Büchi automaton recognizing the path finite trees.*

Proof We will show a Büchi automaton recognizing the complement of this language. Our automaton needs to check if there is a path in $\overline{t}$ containing infinitely many red edges. The automaton accepts if either:

- There is a cycle $(v, x), \dots, (v, x)$ containing a red edge.
- There exists an infinite acyclic path $v_0, v_1, \dots$ in the tree t labeled by pairs $(x_0, x_0'), (x_1, x_1'), \dots \in X \times X$ such that:
  - There is an edge in $\overline{t}$ between (v_i, x_i') and (v_{i+1}, x_{i+1}).
  - There is a path (perhaps of zero length) in $\overline{t}$ linking (v_i, x_i) with (v_i, x_i').
  - The edges and cycles mentioned above are directed accordingly to create either an infinite forward path or an infinite backward path.
  - This path has infinitely many red edges.

Both of these conditions can be checked by a nondeterministic Büchi automaton similar to the one constructed in [16]. □

The proof of the following can be found in [15]:

Theorem 12 (Rabin)
Every nonempty regular tree language contains a regular tree

Theorem 13
For a regular tree language L, the following conditions are equivalent:

1. *L contains some path bounded tree.*
2. *L contains some regular path bounded tree.*
3. *L contains some regular path finite tree.*
4. *L contains some path finite tree.*

Proof Since, by Lemma 3, the set of path finite trees in L is regular, $3 \Leftrightarrow 4$ follows from Theorem 12. On the other hand, the equivalence $2 \Leftrightarrow 3$ follows from Lemma 2.

The implication $2 \Rightarrow 1$ is obvious, for the other direction more care is needed because the set of path bounded trees is not regular. For a given M, however, it can be shown that the language L_M of trees where each path has at most M red vertices is regular. If L contains a path bounded tree, then L_M is nonempty for some M. Thus L_M contains a regular tree, which gives us 2. □

Using Lemma 3 and Theorem 13 we obtain:

Lemma 4. *Given a regular language L over VG_X^m, it is decidable whether there is a path bounded tree in L.*

Assume the language L is recognizable by a Büchi nondeterministic automaton. Because emptiness for co-Büchi and Büchi automata is polynomial and the automaton in Lemma 3 is of exponential size with respect to m and $|X|$, the time taken by such a procedure is exponential in $|X| \cdot m$ and polynomial in the size of the automaton recognizing L.

3.2 Signatures and induced RG-graphs

We are now set to show the decidability of the finite model problem. We use Theorem 15, which represents finite graphs as trees satisfying a certain condition. This condition, in turn, can be expressed using bounded paths and, using the results from the previous section, the finite model problem is proved decidable.

Consider an alternating two-way automaton $\mathcal{A} = \langle Q_\exists, Q_\forall, q_0, \Sigma, \delta, \Omega \rangle$. Fix a graph G with vertices V and a strategy s for the player $\exists$ in the game $\mathrm{G}(\mathcal{A}, G)$. For an odd color $i \in \Omega(Q)$ assumed by the acceptance condition we will define the relations $\to_i, \Rightarrow_i \subseteq (Q \times V)^2$ and the partial function $\mathrm{Sig}_i : Q \times V \to \mathcal{N} \cup \{\infty\}$ as follows:

- $(q_1, v_1) \to_i (q_2, v_2)$ if $\Omega(q_1) \geq i$ and in the game $\mathrm{G}(\mathcal{A}, G)$ there exists a play compliant with s with a move from (q_1, v_1) to (q_2, v_2). In other words, $\mathcal{A}$ can go from the position v_1 and state q_1 to the position v_2 and state q_2 in one move: either any move by $\forall$ or the single move by $\exists$ compliant with s. The transitive closure of this relation is denoted as $\to_i^*$. Note that $(q_1, v_1) \to_i (q_2, v_2)$ implies the position (q_1, v_1) is reachable in $\mathrm{G}(\mathcal{A}, G)$ under s.

- $(q_1, v_1) \Rightarrow_i (q_2, v_2)$ if $(q_1, v_1) \rightarrow_i (q_2, v_2)$ and $\Omega(q_1) = i$.
- $\mathrm{Sig}_i(q, v)$ is defined only for states q such that $\Omega(q) \geq i$ and equals the upper bound on the $n \in \mathcal{N}$ such that for some $q_1, q_1', v_1, v_1' \ldots q_n, q_n', v_n, v_n'$:

$$(q, v) \rightarrow_i^* (q_1, v_1) \Rightarrow_i (q_1', v_1') \rightarrow_i^* (q_2, v_2) \Rightarrow_i \ldots \rightarrow_i^* (q_n, v_n) \Rightarrow_i (q_n', v_n')$$

Intuitively, given a strategy s for $\exists$, $\mathrm{Sig}_i(q, v)$ tells us how many "bad" states of odd color i can be visited by the automaton starting from (q, v) before they are either annulled by a state of color less than i or no more states of color i can be visited. This can be encoded by the very local relations $\rightarrow_i, \Rightarrow_i$ which say what are the possible moves between neighboring vertices of the graph.

We will illustrate the concept of signature using Example 1. Recall that we were dealing with the graph $G = \langle \mathcal{N}, \{(n, n+1) : n \in \mathcal{N}\}, e, 0\}\rangle$. Consider the winning strategy for the player $\exists$ described in the example, call it s. We will define the signature for this strategy. There is only one odd color in the acceptance condition – so only the relations $\rightarrow_1$, $\Rightarrow_1$ and the function Sig_1 need be defined.

Consider first the state q_z. Since $\delta(q_z, a) = \{(-1, q_z)\}$, only backward moves to q_z are possible and we obtain that $(q_z, v) \rightarrow_1 (q, v')$ iff $q' = q_z$ and $v' = v - 1$. Since q_x, q_y are of color 0, $(q, v) \rightarrow_1 (q', v')$ does not hold for $q \in \{q_x, q_y\}$. Moreover, in this particular example, $\rightarrow_1$ is the same relation as $\Rightarrow_1$, since there are no states of color greater than 1.

Now we turn to the signature $\mathrm{Sig}_1(q, v)$, which is only defined if $q = q_z$. Given a vertex v, it is easy to see that at most v steps can be made using $\Rightarrow_1$:

$$(q_z, v) \Rightarrow_1 (q_z, v-1) \Rightarrow_1, \ldots, \Rightarrow_1 (q_z, 0)$$

This shows that $\mathrm{Sig}_1(q_z, v)$ is v for all vertices $v \in \mathcal{N}$ of the example graph.

Definition 14 A *two-way k-ary tree* is a graph whose domain is k^* and where all edges are exclusively either (v, vi) or (vi, v) for $v \in k^*$ and $i \leq k$.

In particular, every k-ary tree is a two-way k-ary tree.

We say an automaton accepts a graph G with a *bounded signature* if there is some strategy s and bound $M \in \mathcal{N}$ such that $\mathrm{Sig}_i(q, v) \leq M$ for all q, v and odd i. The following was proved in [1] for automata with the Büchi acceptance condition, but the proof for the general parity condition is the same:

Theorem 15 (Bounded signature theorem)
An alternating two-way automaton accepts some finite graph iff it accepts some two-way $|Q|$-ary tree with a bounded signature.

Given an automaton $\mathcal{A}$ with states Q and m colors in the acceptance condition, let $\mathrm{VG}(\mathcal{A})$ stand for $\mathrm{VG}^{|Q|}_{Q \times [m]}$. Note that $|\mathrm{VG}(\mathcal{A})|$ is exponential in $|Q|$. This alphabet will be used to encode the relations $\rightarrow_i$ and $\Rightarrow_i$ in the tree itself.

Definition 16 Given a Σ-labeled two-way $|Q|$-ary tree t and a memoryless strategy s for the player $\exists$, the *induced tree* $\mathrm{T}_{t,s}$ is the unique $\mathrm{VG}(\mathcal{A})$-labeled tree such that $\overline{\mathrm{T}_{t,s}}$ has:

- A red edge from (v,q,i) to (v',q',i) iff $(q,v) \Rightarrow_i (q',v')$.
- A green edge from (v,q,i) to (v',q',i) iff $(q,v) \rightarrow_i (q',v')$.

The construction of $\mathrm{T}_{t,s}$ yields:

Lemma 5. *The signature of t,s is bounded iff $\mathrm{T}_{t,s}$ is path bounded.*

To prove the next lemma, we need to encode strategies for $\exists$ in trees. Note that in a graph G, a memoryless strategy in the game $\mathrm{G}(\mathcal{A},G)$ is a function $s : Q \times \mathrm{dom}(G) \rightarrow Q \times \mathrm{dom}(G)$ such that for $s(q,v) = (q',v')$, either $v = v'$ or $(v,v') \in E \cup E^{-1}$. If the graph in question is a two-way tree, we can use a different format:

$$\overline{s} : \mathrm{dom}(t) \rightarrow Q \rightarrow Q \times \{-1, \epsilon, 1, \ldots, k\}$$

Let us denote the finite set $Q \rightarrow Q \times \{-1, \epsilon, 1, \ldots, k\}$ by $\mathrm{S}_{\mathcal{A}}$. Given a strategy s over the tree t, the *encoding* of s is the relevant mapping $\overline{s} : \mathrm{dom}(t) \rightarrow \mathrm{S}_{\mathcal{A}}$. This allows us to encode strategies within a tree using an alphabet whose size is exponential in $|Q|$.

Lemma 6. *The set of all induced trees is a regular tree language*

Proof It is easy to find a nondeterministic Büchi automaton that, given a tree t labeled by $\mathrm{VG}(\mathcal{A})$, guesses a labeling $\sigma : \mathrm{dom}(t) \rightarrow \Sigma$ and an encoding $\overline{s} : \mathrm{dom}(t) \rightarrow \mathrm{S}_{\mathcal{A}}$ and verifies that t is indeed the induced tree for σ and the strategy s. The state space of this automaton is exponential on $|Q|$. □

Corollary 17 The finite model problem is decidable in time exponential in $|Q|$.

Proof By Theorem 15, $\mathcal{A}$ accepts some finite graph iff $\mathcal{A}$ accepts some tree t with a bounded signature. This, by Lemma 5 is equivalent to the existence of an induced path bounded tree. Since the language of induced trees is regular by Lemma 6, we can use Lemma 4 to show the decidability. □

We prove the lower bound by a reduction from the emptiness problem for *one-way automata*. Such an automaton is a two-way automaton where δ does not use the transitions $\{-\} \times Q$. This is the original alternating automaton [12, 3]. In the one-way case, one can prove using Rabin's Theorem 12 that every nonempty one-way alternating automaton accepts some finite graph. This means that for one-way automata, the emptiness and finite model problems are equivalent. Since the emptiness for PDL, which is EXPTIME hard [10], can be reduced to the emptiness problem for alternating one-way automata, the finite model problem for one-way automata is also EXPTIME hard. The one-way automata being a special case of two-way automata, we obtain:

Theorem 18
The finite model problem for alternating two-way automata is EXPTIME complete

Without going into any details, we only state the application of our result to the μ-calculus. Readers interested in the translation should refer to [16, 1].

Theorem 19
The finite model problem for the propositional modal μ-calculus with backward modalities is decidable in EXPTIME.

Before we conclude, we state one possible extension of this work. Two-way alternating automata are also used in the paper [5] to decide the satisfiability of formulas of the so-called Guarded Fragment with fixed points. It is possible that our result can be applied to solving the open problem of whether the finite model property for formulas of the Guarded Fragment with fixed points is decidable.

References

1. Mikołaj Bojańczyk. The finite graph problem for two-way alternating automata. In *FOSSACS 2001*, volume 2030 of *LNCS*, pages 88–103, 2001.
2. A. Saoudi D. E. Muller and P. E. Shupp. Weak alternating automata give a simple explanation why most temporal and dynamic logics are decidable in exponential time. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science*, pages 422–427, 1988.
3. E. A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. 32th IEEE Symposium on Foundations of Computer Science*, pages 368–377, 1991.
4. E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 1999.
5. E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proceedings 14th IEEE Symp. on Logic in Computer Science*, pages 45–54, 1999.
6. Y. Gurevich and L. Harrington. Automata, trees and games. In *Proc. 14th. Ann. ACM Symp. on the Theory of Computing*, pages 60–65, 1982.
7. J. van Benthem H. Andreka and I. Nemeti. Modal logics and bounded fragments of predicate logic. *Journal of Philosophical Logic*, pages 217–274, 1998.
8. I. Hodkinson. Loosely guarded fragment has finite model property. *J. Symbolic Logic*, to appear.
9. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
10. R. Ladner M. Fischer. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.
11. A. Mostowski. Games with forbidden positions. Technical report, University of Gdańsk, 1991.
12. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
13. M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141, 1969.
14. G. Slutzki. Alternating tree automata. *Theoretical Computer Science*, 41:305–318, 1985.
15. Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Language Theory, III*, pages 389–455. Springer, 1997.
16. M. Vardi. Reasoning about the past with two-way automata. In *vol. 1443 LNCS*, pages 628–641, 1998.