Alternating Qualitative Parity Tree Automata.

Laureline Pinault[†], **supervisors:** Nathanaël Fijalkow and Olivier Serre[‡]

[†]ENS de Lyon, France
[‡]LIAFA, Paris, France

August 28, 2014

Abstract

Tree automata are powerful tools used to handle sets of infinite trees, widely needeed in program verification, since they provide a natural representation of branching-time systems. We study those tree automata equipped with an unusual acceptance semantics called qualitative (in order for the run to be accepted, almost all branches have to be accepting). In particular we study the links between alternating parity tree automata and non-deterministic ones (third and fourth part), and between Büchi (alternating or not) qualitative tree automata and co-Büchi ones (fith part).

Contents

1 Introduction

There are two kinds of programs: computational programs and reactive programs.

The *computational programs* run over an input in order to produce a final result on termination. For instance the program which takes an integer n as input and return $n!$, and the program which takes a graph as input and return a minimum spanning tree of the graph, are computational programs. They can be modeled as a black box with specification about input/output relations. Their correctness are expressed as Hoares triples.

The *reactive programs* maintain an ongoing interaction with their environments. Those programs whose main aim is to interact rather than compute are omnipresent: operating systems, drivers, CPUs, car controllers, ... They can be modeled by transition systems. They can also be modeled by behavioural trees: trees containing all possible behaviours (a behaviour is a sequence of system states). Those trees are generally infinite. Since their corectness is expressed as behavioural specifications, the latter modelization is useful for the program verification (the field where we study the corectness of the programs).

Example (Digital code)**.** We consider a digital code with three letters A, B and C, which accepts only the code ABA. This is a reactive program. Its transition system is:

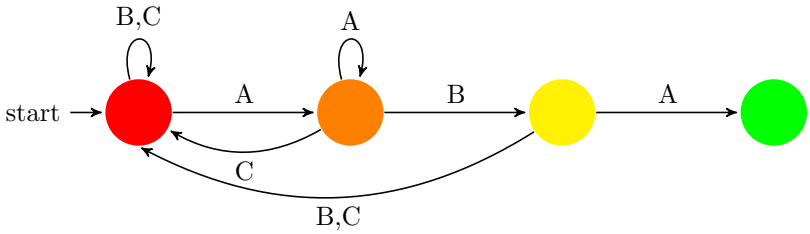


Figure 1: Transition system of the digicode system

Its behavioural tree is the unfolding of this transition system:

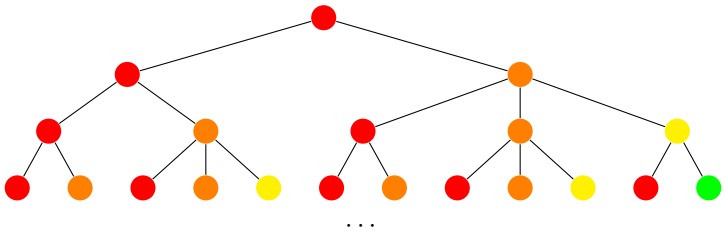


Figure 2: Behavioural tree of the digicode system

Our purpose is to develop tools for verification of reactive program. So for us a program will be assimilated to its behavioural tree. We restrict ourselves to bouded-branching behavioural trees. Those trees modelize most reactive programs. Since we only consider with bouded-branching trees, we can, without loss of generality, work on infinite binary trees.

Definition 1 (Infinite binary tree)**.** Let Σ be a finite alphabet. An infinite Σ-labelled binary tree (or simply a tree when Σ is clear from the context) is a map $t : \{0,1\}^* \mapsto \Sigma$.

In this setting, we shall refer to an element $n \in \{0,1\}^*$ as a node and to ε as the root. For a node n, we call $t(n)$ the label of n in t. A language of (infinite Σ-labelled binary) trees is a set of infinite Σ-labelled binary trees.

The behaviour specifications can be expressed in Monadic Second Order logic (MSO logic) or equivalently with tree automata. Indeed a language is MSO-definable if and only if it is recognizable by a tree automaton ([Rab69], for a proof of this result see [Tho97], theorem 6.19).

We will study tree automata. More precisely we are interested in studying a class of tree automata, called qualitative, which could be used for verification of programs where probabilistic choices or probabilistic interactions intervenes. More precisely we study in this report classical results and want to know wether they still hold for the qualitative tree automata. The second section is devoted to the formal definition of the automata and its numerous variants. The third and fourth section deals with the simulation theorem which stands that alternating and non-deterministic tree automata are equi-expressive. The fifth section presents some results around the parity hierarchy.

2 Alternating Qualitative Parity Tree Automata

What we will call a structure in what follows is an object containing positions which are labelling by a finite alphabet. For example, a word $u = u_0u_1...u_n$ is a structure whose position are indexed by integers between 0 and n. The label of the position $i \in [|0, n|]$ is u_i.

For a Σ-labelled binary tree the positions, indexed by elements of $\{0, 1\}^*$, are called nodes and are labelled by elements of Σ. If u and v are two nodes, we note $u < v$ the fact that u is a prefix of v.

An automaton works on objects with a given structure in order to decide if they fulfill a given property. The set of the objects which fulfill the tested property is called the language of the automaton. The objects studied can take various forms, like finite or infinite words, or even infinite binary trees, which we are studying here.

An automaton is defined by a tuple: $\mathcal{A} = (Q, \Sigma, \Delta, q_{in}, Acc)$ where:

- Q is a finite set of states.
- Σ is the finite alphabet labelling the input objects.
- Δ is a set of transitions.
- q_{in} is the initial state.
- Acc is the acceptance condition. Basically it is a subset of $Q^\star$ for finite structures, or Q^ω for infinite structures, but it can often be described by another way, such as a set of final states.

An automaton works as follow: it takes for input an object with a given structure and produces another object with the same structure but labelled with Q instead of Σ, called a run of the automaton. On this run, the acceptance condition Acc is easily checked.

Example. The following automaton recognize finite words wich start with a and end with b:

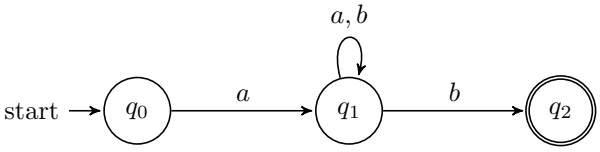


Figure 3: An automaton recognizing $a\Sigma^*b$

The garbage state is called q_r. There are two possible runs of this automaton over the word $aabab$: $q_0q_1q_1q_2q_rq_r$ and $q_0q_1q_1q_1q_1q_2$. The first one is rejecting and the second one is accepting.

Thus, for a tree automaton (*i.e.* an automaton recognizing languages of infinite binary trees), the run is also an infinite binary tree. In order to produce a run which is an infinite binary tree, we have $\Delta \subseteq Q \times \Sigma \times Q \times Q$: the transitions are of the form (q, a, q_0, q_1) where q, q_0, q_1 are states and a is a letter of the alphabet. When we are at a position u labelled by a in a state q (i.e. a labels the position u of the tree and q labels the position u of the run), if we choose the transition (q, a, q_0, q_1) we will label the position $u \cdot 0$ of the run by q_0 and the position $u \cdot 1$ of the run by q_1.

Definition 2 (Run of a tree automaton)**.** A *run* of the tree automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_{in}, Acc)$ over a Σ-labelled binary tree t is a Q-labelled binary tree ρ such that:
$\rho(\varepsilon) = q_{in}$ and $\forall u \in \{0,1\}^*, (\rho(u), t(u), \rho(u \cdot 0), \rho(u \cdot 1)) \in \Delta$.

Classically, a run is called accepting if for every branch b of the run, b is in Acc, and a tree is accepted by a tree automaton if there exists an accepting run of the automaton over the tree. But we can have some variations of tree automata by playing on three levels: the micro, the macro and the cosmo ones.

2.1 The Micro Level: Playing with the Acceptance Condition of Branches

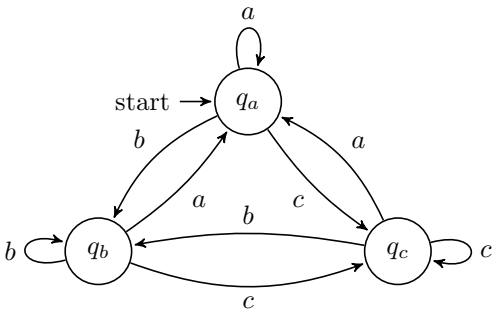
We discuss here the acceptance condition *Acc*. When we have an automaton working over finite words, the acceptance condition described by a set of final states suffices to encapsulate regular languages. But when we work with infinite structure, it is not that simple. In order to see what kind of conditions we can define over infinite structures, we will look at infinite word automata: we have an automaton which works over infinite words and produces a run which is an infinite word labelled with Q, the set of states.

The acceptance condition directly inherited from finite word automata is called the *Reachibility condition*: an infinite word is accepted if we reach a final state in the run. This condition is rather simple but it is not really adapted for infinite structures as it only characterizes finite prefixes of the word. Thus it is not possible with this acceptance condition to recognize languages such as "the word contains infinitely many a's".

A natural acceptance condition is to accept the run if we reach a final state infinitely often. This condition is called the *Büchi condition*. Unlike the previous one, the Büchi condition allows to easily test properties such as "the word contains infinitely many a's".
We can consider the dual condition as well: the run is accepted if we see only finitely often a final state. This condition is called the *co-Büchi condition*, and is described by a set of states in which we do not want to see infinitely often, also known as a set of forbidden states.

The *Parity condition* generalizes the Büchi and co-Büchi ones. Its functionment is the following: it associates an integer called colour to each state, and a run is accepted if the smallest colour seen infinitely often is even. Tha acceptance condition *Acc* is then described has a function $c : Q \to I \subseteq \mathbb{N}$, I being finite.

Example. $\Sigma = \{a, b, c\}$. We want to recognize the Σ-infinite words who have this property : *"If there are infinitely many a's then there are infinitely many b's"*. This is not possille with a Büchi or a co-Büchi automaton, but we can use this parity automaton:



$$\boxed{\begin{array}{l} c(q_a) = 1 \\ c(q_b) = 0 \\ c(q_c) = 2 \end{array}}$$

Figure 4: A parity automaton recognizing $(\Sigma^* a \Sigma^* b)^\omega + (\Sigma^* c)^\omega$

To retrieve the Büchi condition, we have to take $I = \{0, 1\}$ and colour the final states with the colour 0. To retrieve the co-Büchi condition, we have to take $I = \{1, 2\}$ and colour the forbidden states with the colour 1.

There exist other acceptance conditions, such as the *Müller condition*, the *Rabin condition* and the *Street condition*. But the parity condition has enough expressive power to simulate any of them. For

this reason, we narrow us to the parity condition.

When we work with infinite trees, the run is an infinite tree and the acceptance condition *Acc* is used to test if a branch of the run is accepting. As a branch of an infinite tree is an infinite word, we work with exactly the same acceptance condition *Acc* to decide if a branch is accepting. Usually a run is said to be accepting if all the branches of the run are accepting. For example, one might want to test, if for all possible behaviours of ones program, the program returns infinitely often control to the user. Then a deterministic Büchi automaton (*i.e.* an automaton with a Büchi acceptance condition) labelling with a final state the nodes where the program returns control to the user, checks this property.

2.2 The Macro Level: Playing with the Acceptance Condition of Runs

Previously we said that for a run to be accepting, it classically had to have all its branches accepting. This is called the *classical semantics*. We can also think of a *qualitative semantics* where instead of requiring all of the branches to be accepted, it requires *almost* all of its branches to be accepted. This semantics is relevant in cases where the program has probabilistic choices or interracts with an environment with probilistic behaviours. Say we have for example a silly program which flips a coin; on head it stops, on tail it goes on. We do not want the program to stop infinitely often. But we do not want to verify this property on all possible behaviours: we want the probability of this happening to be zero.

We define below a probability measure over the sets of branches in order to define what "almost all" means.

Definition 3 (Cone)**.** Let t be a tree and u a node in the tree. We call *cone* of u and note by $cone(u)$ the set of branches which have u as prefix.

Definition 4 (Measure of a cone)**.** Let $u \in \{0,1\}^*$ be a node in a tree t. The measure of the cone of u is $\mu(cone(u)) = 2^{-|u|}$, where $|u|$ is the length of u[1].

From there, we use the Carathéodory Extension Theorem to define a unique measure on the Borel σ-field generated by the cones.

We say that a run ρ over a tree is *qualitatively accepting* if and only if $\mu(Acc(\rho)) = 1$, where $Acc(\rho)$ is the set of branches of ρ which belong to *Acc*. Usually a tree is qualitatively accepted by a tree automaton $\mathcal{A}$ if and only if there exists a qualitatively accepting run of $\mathcal{A}$ over this tree. We denote by $\mathcal{L}_{Qual}(\mathcal{A})$ or $\mathcal{L}^{=1}(\mathcal{A})$ the language of the trees qualitatively accepted by $\mathcal{A}$. When we work with the qualitative semantics we say that we work with qualitative automata.

We can also define the *positive semantics*: a run ρ over a tree is accepted if and only if $\mu(Acc(\rho)) > 0$. By analogy with the notation $\mathcal{L}^{=1}(\mathcal{A})$, we denote by $\mathcal{L}^{>0}(\mathcal{A})$ the language of the trees positively accepted by $\mathcal{A}$

This report mainly deals with classical results which can or can not be adapted in the qualitative semantics.

2.3 The Cosmo Level: Playing with the Acceptance Condition of Trees

When we work with deterministic automata there is only one possible run, so the question of the acceptance of a tree is not interesting: a tree is accepted by the automaton if and only if the run of the automaton over the tree is accepting. When there are several possible runs, we are used to working with non-determinism: we require that there exists an accepting run of the automaton over the tree. We could also imagine the dual: requiring that each possible run of the automaton over the tree be accepting. This is what we call a universal automaton.

While with non-deterministic automata we can choose the transition we want at each state, with universal automata we have to try all possible transitions. We can also imagine a mixed version: for some states we can choose the transition we want and for the others states we have to try all possible

[1]There exist other measures : for $p \in [0,1]$ and $u \in \{0,1\}^*$ we can define $\mu_p(cone(u)) = p^{|u|_0}(1-p)^{|u|_1}$. Then the recognized languages are not the same (reference ?). But we will not study this here.

transitions. Those automata are called *alternating automata*. There are several ways to describe alternation in an automaton. We describe here one of them, the one we will use from now on.

We distinguish two kinds of states: the ones where we can choose the transitions are called *existential states* and the others where we have to try all possible transitions are called *universal states*. Those two kinds of states form a partition of the set of states: $Q = Q_\exists \uplus Q_\forall$, $Q_\exists$ being the set of existential states and $Q_\forall$ being the set of universal states.

The easiest way to describe the acceptance of such an automaton is by playing a simple game, called the acceptance game. We have two players, Eve and Adam. Eve's goal is to get the tree accepted by the automaton, whereas Adam's goal is not to get the tree accepted. Thus, Eve is responsible for choosing a transition when we are in an existential state and Adam is responsible for choosing a transition when we are in a universal state. Given an automaton $\mathcal{A}$ and a tree t, the game is played as follows:

- We start in the initial state of $\mathcal{A}$, q_{in}. If the state is existential, Eve starts, otherwise Adam.
- When we are in an existential (resp. universal) state q in a node u of t, Eve (resp. Adam) chooses a transition $(q, t(u), q_0, q_1)$ in Δ.
- Once the transition has been choosen, Adam chooses the direction we will go to: if he chooses to explore the left subtree, we move to the node $u \cdot 0$, in the state q_0, otherwise we move to the node $u \cdot 1$ in the state q_1. Adam's choosing the path ensures that all branches are accepting, even if we only test one: if a branch was not accepted Adam could choose it.
- We go on like this. The succession of visited states is an infinite word. To know if the tree is accepted, all we have to do is check whether this infinite word belongs to *Acc* or not.

In what follows we formally define what we call a game in general, and then the specific game described above. We also explain how to adapt the latter to the qualitative semantics.

Definition 5 (Game)**.** A *game* $\mathcal{G}$ consists of an arena G (possibly infinite), an initial position v_0 and an acceptance or winning condition *Acc*. The arena G is an oriented graph (V, E) where the set of vertices V is partitioned between the vertices of the first player (we name her Eve) and the vertices of the second player (we name him Adam): $V = V_E \uplus V_A$. v_0 is the vertex of G where we start every play. *Acc* is the acceptance condition: a play can be described as an infinite word labelled by the vertices of G, so *Acc* is a subset of V^ω. Eve wins if this play belongs to *Acc*. Otherwise it's Adam. From now on *Acc* will be a parity condition[2].

Definition 6 (Strategy)**.** The strategy describes how the players play.

A *strategy* of Eve (resp. of Adam) is a function from the set of partial plays[3] ending in V_E (resp in V_A), to the set of all vertices V. This function is such that the image of a partial play $\lambda \cdot v$ is within v's neighbours[4].

The strategy of a player is said *positional* if it does not need to remember what has been played: the strategy of the player α becomes then a function from the vertices of the player α ($\alpha \in$ {Eve,Adam}), to the set of all vertices V, such that for all $v \in V_\alpha$, the image of v is in v's neighbours.

The strategy of a player α is said *winning* for the player α if and only if for all strategies of the other player, the play we get by playing accordingly to both strategies (the play is entirely determined by the strategies) is winning for α.

For a tree automaton $\mathcal{A}$ and a tree t we define the acceptance game $\mathcal{G}_{\mathcal{A},t}$ as follows:

- The arena $G_{\mathcal{A},t}$ is composed by two kinds of vertices: the ones where the players choose the next transition (an element of Δ) and the ones where Adam chooses the direction (left or right). The first ones have to contain the information of the node and the state we are in, so this is $\{0,1\}^* \times Q = \{0,1\}^* \times Q_\exists \uplus \{0,1\}^* \times Q_\forall$. The vertices in $\{0,1\}^* \times Q_\exists$ belong to Eve and the vertices $\{0,1\}^* \times Q_\forall$ belong to Adam. The second ones have to contain the information of

[2]For more details about acceptance conditions, see section 2.1

[3]A partial play is a prefix (a finite starting part) of a play

[4]The set of v's neighbours is the set $\{v' | (v, v') \in E\}$.

the node we are in and of the chosen transition, so this is $\{0,1\}^* \times \Delta$. These vertices are all owned by Adam. Edges are either $((n,q),(n,(q,t(n),q_0,q_1)))$ or $((n,(q,t(n),q_0,q_1)),(n\cdot 0,q_0)$ or $((n,(q,t(n),q_0,q_1)),(n\cdot 1,q_1)$, where $n \in \{0,1\}^*$ is a node and q,q_0,q_1 are states of the automaton.

- The initial position is the vertex (ε, q_{in}) where ε is the empty word (*i.e.* the root of t) and q_{in} is the initial state of $\mathcal{A}$.
- The winning condition *Acc* directly derives from the acceptance condition of $\mathcal{A}$. From a play in $\mathcal{G}_{\mathcal{A},t}$ we can naturally extract a sequence of states. The play is winning if and only if the sequence verifies the acceptance condition of $\mathcal{A}$.

The tree t is accepted by the automaton $\mathcal{A}$ if and only Eve has a winning strategy in the game $\mathcal{G}_{\mathcal{A},t}$.

Remark. If there is no universal state (resp. no existential state) we obtain a non-deterministic automaton (resp. a universal automaton). In both cases the acceptance defined with the game $\mathcal{G}_{\mathcal{A},t}$ coincides with the one defined by the runs.

We can also adapt the acceptance game for the qualitative semantics. Instead of having two players, we have three: Eve, Adam and an extra player Random. Eve and Adam choose the transitions as previously but Random chooses the directions.

Definition 7 (Stochastic game)**.** A *stochastic game* $\mathcal{G}^{=1}$ consists of an arena G (possibly infinite), an initial position v_0, a stochastic transition function δ and a winning condition *Acc*. The arena G is an oriented graph (V,E) where the set of vertices V is partitioned between the vertices of each player, Eve, Adam and Random: $V = V_E \uplus V_A \uplus V_R$. v_0 is the vertex of G where we start every play. δ is a function defined on V_R such that if $v \in V_R$, $\delta(v)$ is a probability distribution over $\{v' | (v,v') \in E\}$. $Acc \subseteq V^\omega$ is the acceptance condition.

The strategies of the players are defined as previously. However since chance intervenes, what we are interested in is not anymore the winning strategies but the almost surely winning strategies. We say Eve has an almost surely winning strategy if for every strategy of Adam, if we play accordingly to both strategies Eve wins with probability 1. Indeed unlike the non-stochastic games, once we have fixed both strategies, many plays are possible, depending on chance. In order to define precisely what an almost surely winning condition is we have to define a measure over the sets of plays played accordingly to both strategies.

Definition 8 (Cone of a partial play)**.** Let λ be a partial play. We call *cone* of λ and note $cone(\lambda)$ the set of plays which have λ as prefix.

Definition 9 (Measure of a cone of a partial play)**.** Let σ be a strategy of Eve and τ be a strategy of Adam. We define inductively the measure $\mu^{\sigma,\tau}$ over the cones of partial plays: let λ be a partial play,

- If $\lambda = v$ where v is a vertex of the arena, then $\mu^{\sigma,\tau}(cone(\lambda)) = 1$ if $v = v_0$, the initial position and 0 otherwise.
- If $\lambda = \lambda' \cdot v$ where λ' is a partial play which ends in a state $v' \in V_E$ and v a vertex of the arena. Then $\mu^{\sigma,\tau}(cone(\lambda)) = \mu^{\sigma,\tau}(cone(\lambda'))$ if $v = \sigma(v')$ and 0 otherwise.
- If $\lambda = \lambda' \cdot v$ where λ' is a partial play which ends in a state $v' \in V_A$ and v a vertex of the arena. Then $\mu^{\sigma,\tau}(cone(\lambda)) = \mu^{\sigma,\tau}(cone(\lambda'))$ if $v = \tau(v')$ and 0 otherwise.
- If $\lambda = \lambda' \cdot v$ where λ' is a partial play which ends in a state $v' \in V_R$ and v a vertex of the arena. Then $\mu^{\sigma,\tau}(cone(\lambda)) = \mu^{\sigma,\tau}(cone(\lambda')) \cdot (\delta(v'))(v)$.

From there, we use the Carathéodory Extension Theorem to define a unique measure on the Borel σ-field generated by the cones of partial plays.

Definition 10 (Almost surely winning strategy)**.** Eve has an *almost surely winning strategy* σ if and only if for every strategy τ of Adam, $\mu^{\sigma,\tau}(Acc) = 1$.

For a tree automaton $\mathcal{A}$ and a tree t we define the acceptance game $\mathcal{G}^{=1}_{\mathcal{A},t}$ as follows:

- The arena $G_{\mathcal{A},t}$ is composed by two kinds of states: the ones where the players choose the next transition and the ones where Random chooses the direction. The first ones have to contain the information of the node and the state we are in, so this is $\{0,1\}^* \times Q = \{0,1\}^* \times Q_\exists \uplus \{0,1\}^* \times Q_\forall$. The vertices in $\{0,1\}^* \times Q_\exists$ belong to Eve and the vertices $\{0,1\}^* \times Q_\forall$ belong to Adam. The second ones have to contain the information of the node we are in and of the transition choosen, so this is $\{0,1\}^* \times \Delta$. These vertices are all owned by Random. Edges are either $((n,q),(n,(q,t(n),q_0,q_1)))$ or $((n,(q,t(n),q_0,q_1)),(n\cdot 0,q_0)$ or $((n,(q,t(n),q_0,q_1)),(n\cdot 1,q_1)$, where $n \in \{0,1\}^*$ is a node and q,q_0,q_1 are states of the automaton.
- The initial position is the vertex (ε, q_{in}) where ε is the empty word (*i.e.* the root of t) and q_{in} is the initial state of $\mathcal{A}$.
- δ associates the probability $1/2$ at each direction.
- The winning condition *Acc* directly derives from the acceptance condition of $\mathcal{A}$.

The tree t is qualitatively accepted by the automaton $\mathcal{A}$ if and only Eve has an almost surely winning strategy in the game $\mathcal{G}^{=1}_{\mathcal{A},t}$.

Remark. As for the classical semantics, if there is no universal state or no existential state the acceptance defined with the game $\mathcal{G}_{\mathcal{A},t}$ coincides with the one defined by the runs.

3 The Simulation theorem

When we work with automata, it is standard to consider an alternating version which generalizes non-deterministic automata we are used to working with. In certain cases the alternation enables indeed to recognize a broader class of languages, and in other cases it is not more powerful (but could give simpler tools, to show the complementation for example). Regarding the parity tree automata it is a well-known result that for the classical semantics alternating and non-deterministic automata are equi-expressive [MS95].

One of our main concerns was to know if the proof of this theorem was adjustable and more generally if the result still held for the qualitative semantics. In the first subsection we present the theorem and its proof for the classical semantics, and in the second subsection we exhibit where the classical construction fails in the qualitative case and give an argument which suggests that the theorem does not hold for the qualitative semantics.

3.1 For the Classical Semantics

The purpose of this section is to show the following theorem.

Theorem 3.1 (Simulation Theorem)**.** *[MS95] Let $\mathcal{A}$ be an alternating parity tree automaton. Then one can effectively construct a non-deterministic parity tree automaton $\mathcal{B}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.*

Proof. We do not give a complete proof of this classical result [MS95] but we rather describe the construction and exhibit the crucial arguments to later explain why it does not work for the qualitative semantics. For this purpose we rely on the presentation given in [FPS13].

The proof of the simulation theorem relies on two main results, the positionality of parity games and the determinization of parity automata over infinite words:

Lemma 3.1 (Positionality of parity games)**.** *Let $\mathcal{G}$ be a parity game. If the player α has a winning strategy in $\mathcal{G}$ then the player α has a positional winning strategy in $\mathcal{G}$. Moreover one can effectively compute this positional strategy.*[5]

Lemma 3.2. *[Determinization of parity automata over infinite words] Let $\mathcal{A}$ be a non-deterministic parity automaton over infinite words. Then one can effectively construct a deterministic parity automaton over infinite words $\mathcal{B}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.*[6]

[5] For a proof of this result see [GTW02], chapter 6.

[6] For a proof of this result see [Tho97], section 5.2

Let $\mathcal{A} = (Q_\exists, Q_\forall, \Sigma, \Delta_\mathcal{A}, q_{in}, Acc)$ be an alternating parity tree automaton. A tree t belongs to $\mathcal{L}(\mathcal{A})$ if and only if there exists σ, a strategy for Eve such that for every τ, strategy of Adam, Eve wins in $\mathcal{G}_{\mathcal{A},t}$. A strategy τ of Adam consists of two distinct parts: τ_1 to choose the transitions in the universal states and τ_2 to choose the path. From σ and τ_1 we can construct a run ρ_{σ,τ_1}. All the branches of this run have to be accepted in order for the tree to be accepted. To summarize: t is accepted if and only if $\exists\sigma\forall\tau_1\forall b \in \rho_{\sigma,\tau_1}, b \in Acc$.

The idea to transform this automaton into a non-deterministic one is to put Adam's choices of transitions in the acceptance condition: we will have a non-deterministic automaton $\mathcal{B}$ which accepts t if and only if $\exists\sigma\forall b \in \rho_\sigma, b \in Acc'$ where Acc' is a more complicated acceptance condition. So we will have $\mathcal{B} = \mathcal{D} \circ \mathcal{C}$, the synchronized product of $\mathcal{D}$ and $\mathcal{C}$, where $\mathcal{C}$ will be a non-deterministic automaton with states containing the information of Adam's choices of transitions and with a non-parity acceptance condition Acc' and $\mathcal{D}$ will be a deterministic automaton over infinite words which will transform Acc' into a parity condition.

- To construct $\mathcal{C}$: (rajouter des images si possible)

  $\mathcal{C} = (Q_\mathcal{C}, \Sigma, \Delta_\mathcal{C}, P_{in}, Acc')$, where:

  - $Q_\mathcal{C} = \mathcal{P}((Q_\exists \uplus Q_\forall) \times (Q_\exists \uplus Q_\forall))$: a state of $\mathcal{C}$ is a subset of pairs of states of $\mathcal{A}$. We name a state of $\mathcal{C}$. a tile. Let P be a tile. We name $\pi_1(P) = \{q \in Q_\exists \uplus Q_\forall | \exists q', (q, q') \in P\}$ and $\pi_2(P) = \{q \in Q_\exists \uplus Q_\forall | \exists q', (q', q) \in P\}$.
  - $\Delta_\mathcal{C}$ can be expressed as follows. Let P, P_1, P_2 be some tiles and $a \in \Sigma$. $(P, a, P_1, P_2) \in \Delta_\mathcal{C}$ if and only if we have all the following conditions:
    - $\pi_2(P) = \pi_1(P_1) = \pi_1(P_2)$.
    - $\forall q \in Q_\forall \cap \pi_2(P), (q, a, q_1, q_2) \in \Delta_\mathcal{A} \Leftrightarrow ((q, q_1) \in P_1 \wedge (q, q_2) \in P_2)$
    - $\forall q \in Q_\exists \cap \pi_2(P), \exists! q_1, (q, q_1) \in P_1 \wedge \exists! q_2, (q, q_2) \in P_2 \wedge (q, a, q_1, q_2) \in \Delta_\mathcal{A}$.
  - The initial tile is $P_{in} = \{(q_{in}, q_{in})\}$.
  - Acc' can be expressed as follows. Let $P_0P_1P_2P_3... \in Q_\mathcal{C}^\omega$. $P_0P_1P_2P_3... \in Acc' \Leftrightarrow (P_0 = P_{in}) \wedge (\forall q_0q_1q_2q_3... \in (Q_\exists \uplus Q_\forall)^\omega$ such that $\forall i \mathbb{N}(q_i, q_{i+1}) \in P_i$, we have $q_0q_1q_2q_3... \in Acc)$. Acc' is not a parity acceptance condition, so we need a second automaton $\mathcal{D}$ to transform Acc' into a parity acceptance condition. This automaton will work over infinite words.

- To construct $\mathcal{D}$:

  $\mathcal{D}$ is a deterministic parity automaton that works over infinite words labelled by $Q_\mathcal{C}$ and recognizes Acc'. The construction of $\mathcal{D}$ is carried out in two steps :

  - We construct a non-deterministic parity automaton that recognizes $\overline{Acc'}$.

    $$\overline{Acc'} = \{P_0P_1P_2P_3...|(\exists q_0q_1q_2q_3..., (\forall i \in \mathbb{N}, (q_i, q_{i+1}) \in P_i) \wedge (q_0q_1q_2q_3... \notin Acc)) \vee (P_0 \neq P_{in})\}$$

    To do this : if $P_0 \neq P_{in}$ we accept, if $P_0 = P_{in}$ we guess the states $q_0, q_1, q_2, q_3, ...$ by non-determinism and we check that $q_0q_1q_2q_3... \notin Acc$, which is a parity condition. Since automata working over infinite words are closed under complementation, we obtain by complementation, a non-deterministic parity automaton which recognizes Acc'.
  - We use the lemma 3.2 to get a deterministic parity automaton recognizing Acc'.

The positionality result is used to show that this construction works. Indeed, in the construction we "deplaced the choices of Adam" in the new acceptance condition. And we first construct the run thanks to Eve's strategy and the check the new acceptance condition with the automaton $\mathcal{D}$. If we had not the positionality result, it would mean that she would have needed the knowledge of what have been played before (partly by Adam) in order to play. Then the transformation would not have worked.

□

Remark. If we make the construction presented above for a universal automaton (*i.e.* an alternating automaton with no existential state), we obtain a deterministic automaton. It means that, compared to the non-deterministic, the universal class is not very expressive.

3.2 For the Qualitative Semantics

As we just saw the two key ingredients in the previous proof are a positionality result for Eve and a determinisation one. *A priori* the determinisation result does not intervene in a different way for the qualitative semantics. And since there also exists a positionality result for Büchi stochastic games (see [FPS13], section 3) we could hope that the Simulation Theorem still holds for the qualitative semantics, at least for Büchi condition. However, we will see in the first part that the construction we made for the classical semantics does not work anymore and in a second part that, even if there existed such a theorem, it would not be effective, at least for the co-Büchi condition.

3.2.1 Where the classical construction fails

For the classical semantics we expressed the acceptance of a tree t by an alternating parity tree automaton $\mathcal{A}$ as : $\exists\sigma\forall\tau_1\forall b, b \in Acc$ where σ is an Eve's strategy, τ_1 is an Adam's strategy for the choices of transitions and b a branch of the run defined by σ and τ_1. And we transformed this condition into : $\exists\sigma\forall b, b \in Acc'$ where Acc' "contains the condition $\forall\tau_1$". When we change to the qualitative semantics the acceptance of a tree t by a qualitative alternating parity tree automaton $\mathcal{A}$ can be expressed as : $\exists\sigma\forall\tau_1\forall^{=1}b, b \in Acc$ where σ, τ_1, b are the same objects than previously and where $\forall^{=1}$ means "for almost all" and is formally defined by $\mu_{\rho_{\sigma,\tau_1}}(Acc) = 1$. Unfortunately we can not transform it into a condition which looks like : $\exists\sigma\forall^{=1}b, b \in Acc'$ where Acc' "contains the condition $\forall\tau_1$" because we can not exchange the quantifiers $\forall$ and $\forall^{=1}$ as we exchanged the two quantifiers $\forall$ for the classical semantics.

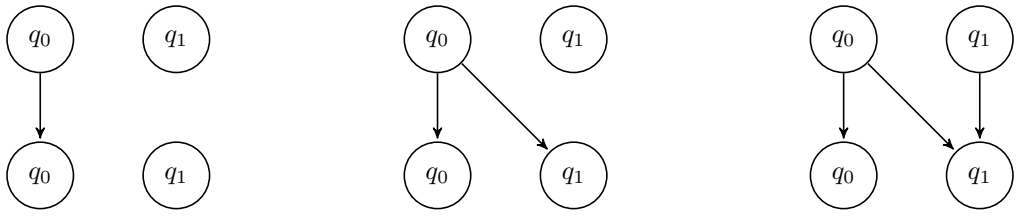
In this subsection we show that due to these quantifiers the construction does not work anymore, through two counter-examples : the first one shows that if we apply ingenuously the construction, the acceptance condition Acc' is too strong, and the second one shows that if we weaken, even a little bit, this condition it becomes too weak. Both counter-examples are qualitative universal parity automata, which allows an easier construction (the construction leads to a deterministic automaton).

Counter-Example 1. The alphabet consists of only one letter. The automaton has two states : q_0 and q_1, both universal; q_0 is the initial state. The goal is to reach q_1. Thus it is a Büchi tree automaton[7]. At each position, Adam can choose to put q_0 on the left and q_1 on the right or the opposite. Hence a strategy of Adam comes down to choosing the unique branch labelled with only q_0. Since for all Adam's strategy there is only one branch which does not satisfy the acceptance condition, the unique tree is qualitatively accepted by the automaton[8]. But if we apply the construction of the automaton $\mathcal{C}$ described in Section 3.1. The deterministic automaton obtained does not recognize anything :

Let $\mathcal{A}$ be the automaton described above.

$$\mathcal{A} = (Q, \Sigma, \Delta, q_{in}, F) = (\{q_0; q_1\}, \varnothing, \{(q_0, q_0, q_1); ((q_0, q_1, q_0); (q_1, q_1, q_1)\}, q_0, \{q_1\})$$

There is only three of the tiles which will be used for the construction: $\pi_{in} = \pi_1 = \{(q_0, q_0)\}$, $\pi_2 = \{(q_0, q_0), (q_0, q_1)\}$, and $\pi_3 = \{(q_0, q_0), (q_0, q_1), (q_1, q_1)\}$ (see Figure 5).



(a) $\pi_1 = \pi_{in}$ (b) π_2 (c) π_3

Figure 5: The tiles used in the construction of $\mathcal{C}$

The transition function of $\mathcal{C}$ is:

$$\delta : \begin{cases} \pi_1 & \mapsto & \pi_3, \pi_3 \\ \pi_3 & \mapsto & \pi_{10}, \pi_{10} \\ \pi_{10} & \mapsto & \pi_{10}, \pi_{10} \end{cases}$$

[7] All reachibility automata are Büchi automata : we only have to loop on a final state at infinity when we reach one to get a Büchi automaton.

[8] A single branch has a measure null so the complementary has measure 1.

In the run of the $\mathcal{C}$ over the sole tree (depicted in Figure 6), all branches are the same and they all contain a rejecting path (one of them is depicted in red in Figure 6).

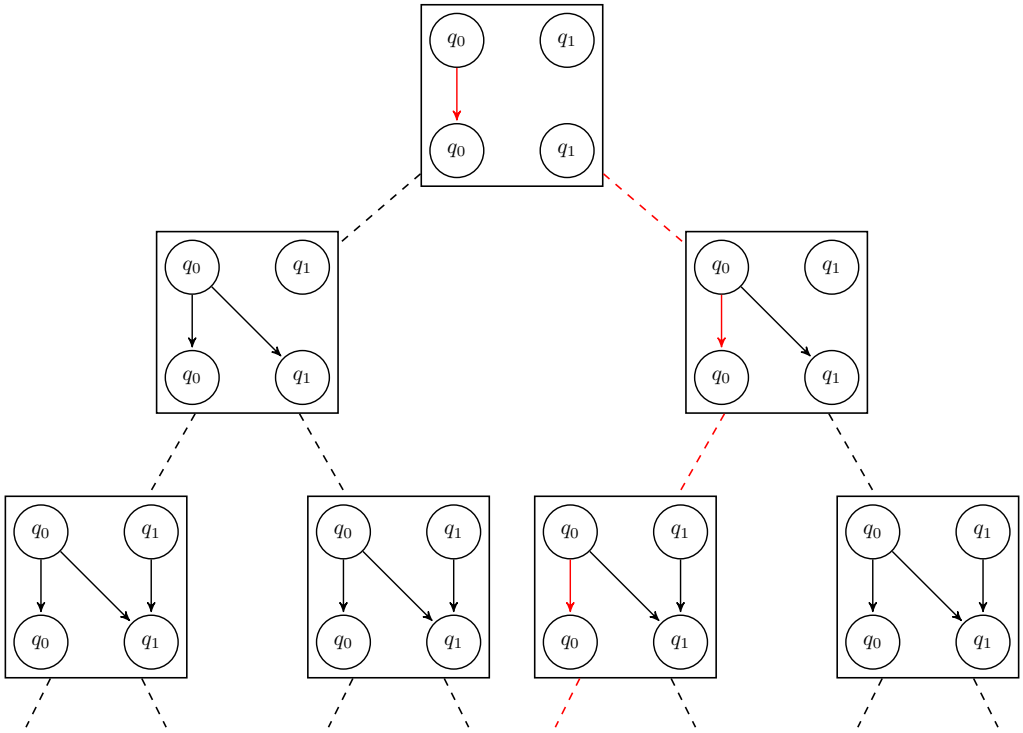


Figure 6: The run of the automaton $\mathcal{C}$

Counter-Example 2. Since we saw with the previous counter-example that the acceptance condition Acc' was too strong, we will try to weaken it a little bit: instead of wanted that all paths of a branch satisfy Acc, we want that *almost all* paths of a branch satify Acc.

Let $P_0P_1P_2P_3... \in Q_{\mathcal{C}}^{\omega}$. $P_0P_1P_2P_3... \in Acc' \Leftrightarrow (P_0 = P_{in}) \wedge (\forall^{=1} q_0q_1q_2q_3... \in (Q_{\exists} \uplus Q_{\forall})^{\omega}$ such that $\forall i \mathbb{N}(q_i, q_{i+1}) \in P_i$, we have $q_0q_1q_2q_3... \in Acc$).

We do not explicit formally the counter example for the construction with this new acceptance condition Acc', but only give the idea of it.

The alphabet is $\{a, b\}$. The principle of the automaton is the following: in order to win,Adam has to guess the labelling of the tree. The functionment of the automaton is very simple: while we read the letter Adam guessed, we go on; as soon as we read another letter, we go in a final state and loop on it for eternity. Thus this is a co-Büchi automaton. Adam has a unique winning strategy for every tree: the strategy which guess the right labelling. It means that every tree is rejected (the goal of Adam is to get the tree rejected).

However, when we construct the automaton $\mathcal{C}$ with the weak acceptance condition, all trees are accepted. Indeed, in the run of $\mathcal{C}$ over a tree, each branch contains only one rejecting path; the one which corresponds t the labelling of this precise branch.

3.2.2 The Emptiness Problem for Qualitative Alternating co-Büchi Tree Automata

The emptiness problem is undecidable for qualitative alternating co-Büchi automata. Yet it is decidable for qualitative non-deterministic parity automata [FPS13]. Consequently if there exists a simulation theorem for qualitative alternating co-Büchi automata, it is not effective.

This section is dedicated to the proof of the undecidability of the emptiness problem for qualitative universal co-Büchi automata[9]. In this purpose we will use probabilistic automaton working over infinite words.

[9]Since universal automata are a subclass of altern automata, it means that the emptiness problem is undecidable for qualitative alternating co-Büchi automata.

A probabilistic automaton is an automaton whose each transition has a probability to happen. Consequently the sets of run has a probability to happen, and we can define a measure on them. The acceptance of a word or tree is defined by this measure. For a probabilistic automaton $\mathcal{A}$, we often consider $\mathcal{L}^{=1}(\mathcal{A})$ or $\mathcal{L}^{>0}(\mathcal{A})$ or $\mathcal{L}^{\geq 1/2}(\mathcal{A})$.

Definition 11 (Emptiness Problem)**.** Given an automaton $\mathcal{A}$, the emptiness problem for $\mathcal{A}$ is to know whether $\mathcal{L}(\mathcal{A})$ is empty or not.

Lemma 3.3. *The emptiness problem for probabilistic co-Büchi automata working over infinite words is undecidable : given a probabilistic automaton over infinite words $\mathcal{B}$, it is undecidable to know wether $\mathcal{L}^{=1}_{co-Büchi}(\mathcal{B})$ is empty or not.*

Proof. This result is a direct consequence of the results of [BGB12].

Let $\mathcal{A}$ a probabilistic Büchi automaton working over infinite words.

The problem to know whether $\mathcal{L}^{>0}_{Büchi}(\mathcal{A})$ is empty or not is undecidable (see [BGB12], theorem 7.2).

There exist a probabilistic Büchi automaton working over infinite words $\mathcal{B}$ such that $\mathcal{L}^{>0}_{Büchi}(\mathcal{B}) = \overline{\mathcal{L}^{>0}_{Büchi}(\mathcal{A})}$ (see [BGB12]).

Let w a word. $w \in \mathcal{L}^{>0}_{Büchi}(\mathcal{A}) \Leftrightarrow w \notin \mathcal{L}^{>0}_{Büchi}(\mathcal{B}) \Leftrightarrow w \in \mathcal{L}^{=1}_{co-Büchi}(\mathcal{B})$. Indeed if there is not a set of accepted which has a positive measure it means that almost all branches are not accepted (i.e. accepted for the dual condition).

Therefore the problem to know whether $\mathcal{L}^{=1}_{co-Büchi}(\mathcal{B})$ is empty or not is undecidable. □

Proposition 3.1. *The emptiness problem is undecidable for qualitative universal co-Büchi tree automata.*

Proof. Following the proof given in [FPS13], lemma 2, we reduce this problem to the emptiness problem for probabilistic co-Büchi automata working over infinite words, which is undecidable (see Lemma 3.3).

Let $\mathcal{B}$ a probabilistic co-Büchi automaton working over infinite words. We want to construct $\mathcal{A}$, an qualitative universal co-Büchi tree automaton such that $\mathcal{L}(\mathcal{A}) = \varnothing \Leftrightarrow \mathcal{L}(\mathcal{B}) = \varnothing$

- To contruct $\mathcal{A}$:

  We can suppose, without any loss of generality, that $\mathcal{B}$ is such that each transition has a probability $1/2$ to happen.

  We construct $\mathcal{A}$ as follows: the set of states and the acceptance condition are the same. The set of transitions $\Delta_{\mathcal{A}} = \cup\{(q,a,q_1,q_2);(q,a,q_2,q_1)\}$ for all $\{(q,a,q_1);(q,a,q_2)\} \subseteq \Delta_{\mathcal{B}}$.

- To show $\mathcal{L}(\mathcal{A}) = \varnothing \Leftrightarrow \mathcal{L}(\mathcal{B}) = \varnothing$:

  $\Rightarrow$: $\mathcal{L}(\mathcal{B}) \neq \varnothing \Rightarrow \exists w \in \mathcal{L}(\mathcal{B}) \Rightarrow t_w \in \mathcal{L}(\mathcal{A})$ where t_w is the tree where all branches are labelled by w.

  $\Leftarrow$: $\mathcal{L}(\mathcal{A}) \neq \varnothing \Rightarrow \exists t \in \mathcal{L}(\mathcal{A}) \Rightarrow \forall \psi$,pure strategy of Adam (i.e. non-randomized strategy), Adam loses in $\mathcal{G}^{=1}_{\mathcal{A},t}$. Yet chance can not help to win in the Markov Decision Process [Gim09] (here $\mathcal{G}^{=1}_{\mathcal{A},t}$ is a Markov Decision Process because Eve does not play). Thus Adam loses in $\mathcal{G}^{=1}_{\mathcal{A},t}$ with the randomized strategy 1/2 - 1/2 (the strategy which at the vertex (n,q) such that $t(n) = a$ and $\Delta_{\mathcal{A}}$ contains the transitions (q,a,q_1,q_2) et (q,a,q_2,q_1) associates the vertex (n,q,q_1,q_2) with probability $1/2$ and the vertex (n,q,q_1,q_2) with probability $1/2$).

  The automaton muni provided with this strategy can be seen as a qualitative probabilistic co-Büchi tree automaton $\mathcal{C}$. The qualitative probabilistic co-Büchi tree automaton $\mathcal{C}'$ which has $(q,a,q_1,q_1,1/2)$ and $(q,a,q_2,q_2,1/2)$ as transitions instead of $(q,a,q_1,q_2,1/2)$ and $(q,a,q_2,q_1,1/2)$ recognizes the same language that $\mathcal{C}$. Therefore $\exists t \in \mathcal{L}(\mathcal{A}) \Rightarrow t \in \mathcal{L}(\mathcal{C}')$. Moreover there exist a reduction from the emptiness problem for probabilistic co-Büchi automata working over infinite words to the emptiness problem for qualitative probabilistic co-Büchi tree automaton which states exactly that $\mathcal{L}(\mathcal{C}') \neq \varnothing \Rightarrow \mathcal{L}(\mathcal{B}) \neq \varnothing$. ([CHS14], theorem 48).

□

4 Qualitative Pumping lemmas

In spite of the counter-examples of the section 3.2.1, we did not show that qualitative alternating parity tree automata are strictly more expressive than qualitative non-deterministic parity tree automata, because the counter-examples were pathological cases where the recognized languages are either empty or the set of all possible trees. So we would like to find a purely qualitative alternating language, *i.e.* a language recognized by a qualitative alternating parity tree automaton but not by an qualitative non-deterministic parity tree automaton.

For this purpose, after some unseccessful tries, it was interesting to try to characterize the languages recognized by qualitative alternating parity tree automata. A way to characterize the languages recognized by automata is the pumping lemmas. In the first subsection we present an existential pumping lemma. In the second subsection we present another pumping lemma we called "pumping on a branch".

4.1 Existential Pumping lemma

There exists an existential pumping lemma for qualitative non-deterministic parity tree automata ([CHS14], Lemma 13). In this section we present it and then show that it still holds for qualitative alternating parity tree automata.

Definition 12 (Pointed Tree)**.** Let t be a tree and $u \in \{0,1\}^*$ be a node. A pair $\Delta = (t, u)$ is called a *pointed tree.*

With a pointed tree $\Delta_1 = (t_1, u_1)$ and a tree t_2, we associate a new tree, $\Delta_1 \cdot t_2$, by plugging t_2 in t_1 instead of the subtree rooted at u_1 . Formally, $\Delta_1 \cdot t_2(u) = t_1(u)$ if u_1 is not a prefix of u and $\Delta_1 \cdot t_2(u) = t_2(u')$ if $u = u_1u'$ for some $u' \in \{0,1\}^*$. We can also define the product of two pointed trees $\Delta_1 = (t_1, u_1)$ and $\Delta_2 = (t_2, u_2)$ by letting $\Delta_1 \cdot \Delta_2 = (\Delta_1 \cdot t_2, u_1u_2)$. Finally, with a pointed tree $\Delta = (t, u)$, we associate a tree Δ^ω by taking an infinite iteration of the product: $\Delta^\omega(v) = t(v')$ where v' is the shortest word such that $v = u^k v'$ for some $k > 0$.

Lemma 4.1 (Existential Pumping Lemma for Qualitative Non Deterministic Automata)**.** *Let $\mathcal{A}$ be an n-states non-deterministic qualitative tree automaton, t be a tree in $\mathcal{L}_{Qual}(\mathcal{A})$ and u be a node of depth greater than n. Then there exist three pointed trees Δ_1, Δ_2 and Δ_3 such that $t = \Delta_1 \cdot \Delta_2 \cdot \Delta_3 \cdot t[u]$ and $\Delta_1 \cdot \Delta_2^\omega \in \mathcal{L}_{Qual}(\mathcal{A})$.*

Proof. We do not give the proof of this result but only the idea of the proof, in order to underline the needed adjustments for the proof for the alternating case. The complete proof can be found in [CHS14], lemma 13.

The idea of the proof is that, since $\mathcal{A}$ is finite, in an accepting run there exist, on the branch of u, before u, two nodes labelled by the same state q. From there we define the pointed trees as shows the Figure 7. This construction works because we only add a countable number of sets of rejecting branches, each of those sets having a measure null. □

Lemma 4.2 (Existential Pumping Lemma for Qualitative Alternating Automata)**.** *Let $\mathcal{A}$ be an n-states alternating qualitative tree automaton, t be a tree in $\mathcal{L}_{Qual}(\mathcal{A})$ and u be a node of depth greater than 2^n. Then there exist three pointed trees Δ_1, Δ_2 and Δ_3 such that $t = \Delta_1 \cdot \Delta_2 \cdot \Delta_3 \cdot t[u]$ and $\Delta_1 \cdot \Delta_2^\omega \in \mathcal{L}_{Qual}(\mathcal{A})$.*

Proof. Even if we can not think in terms of accepting runs for alternating automata, the idea is similar. If we fix an Eve's strategy, we can define a kind of run: we label the nodes of the run by the set of possible states at this node (the possible states depend on Adam's strategy). We use the nodes where those sets are the same to define the pointed trees.

Let σ be an almost-surely winning strategy for Eve in the game $\mathcal{G}^{=1}_{\mathcal{A},t}$ played over t.

For each node v such that $v < u$, we consider the set $P(v)$ of all possible states for this node when we play accordingly to the strategy σ.

- For ε, $P(\varepsilon) = \{q_{in}\}$ where q_{in} is the initial state of $\mathcal{A}$.
- For v prefix of u and direct successor of w, $v = w \cdot 1$ for instance,
  $P(v) = \{q \mid \exists q' \in Q_\exists, \sigma(w, q') = q\} \cup \{q \mid \exists q' \in Q_\forall, \exists q'', (q', t(w), q'', q) \in \Delta\}$.

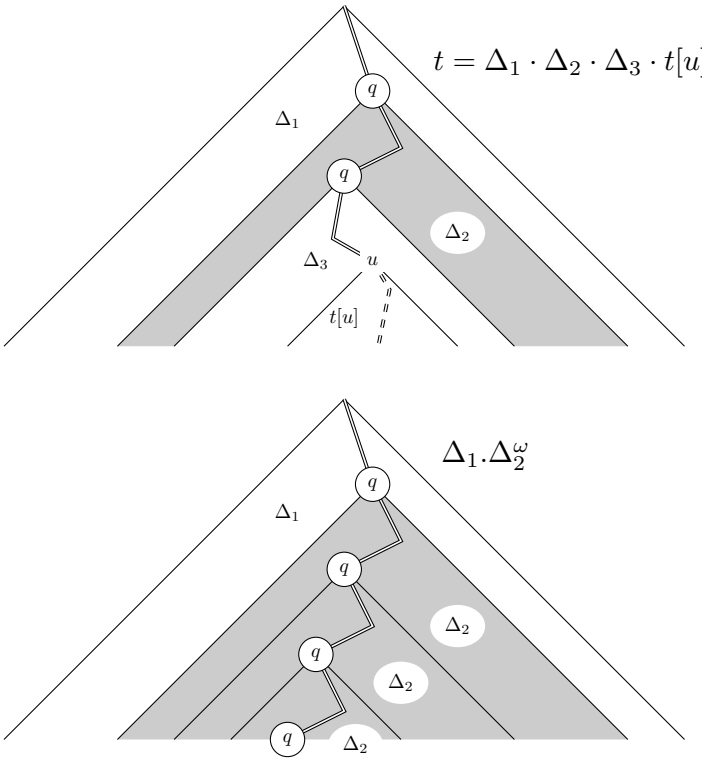


Figure 7: Existential pumping Lemma

As there are n different states in $\mathcal{A}$, there are 2^n possibilities for the sets of states, so there are two nodes, let say u_1 and u_2, such that $u_1 < u_2 < u$ (we name v the node such that $u_2 = u_1 v$) and $P(u_1) = P(u_2)$. We subsequently define the pointed trees as follows:

- $\Delta_1 = (t, u_1)$
- $\Delta_2 = (t, u_2)$
- $\Delta_3 = (t, u)$

Let $t^\omega = \Delta_1 \cdot \Delta_2^\omega$ and σ^ω be the strategy of Eve obtained by inferring σ on t^ω. σ^ω is defined as follows. For a node w:

- Either $w \in \Delta_1$, and then $\forall q \in Q_\exists$, $\sigma^\omega(w, q) = \sigma(w, q)$.
- Or w can be written $u_1 v^k v'$ where $k \in \mathbb{N}$ and $u_1 v' \in \Delta_2$, and then $\forall q \in Q_\exists$, $\sigma^\omega(w, q) = \sigma(u_1 v', q)$.

We want to prove that σ^ω is an almost-surely winning strategy for Eve in the game $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$ played over t^ω, i.e. that $\forall \tau^\omega$, strategy of Adam in $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$, $\mu^{\sigma^\omega,\tau^\omega}(Acc_G) = 1$ where Acc_G is the acceptance condition of the games $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$ and $\mathcal{G}^{=1}_{\mathcal{A},t}$.

In order to achieve this goal, we partition the set of plays played in $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$. As a plays is played on a branch of t^ω, we partition the set of branches of t^ω and this partition will induce a partition over the set of plays played in $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$.

We name $B_0 = \{u_1 v^\omega\}$ and $B_1 = \Delta_1 \cdot \Delta_2$. Then, for all v' of length j such that $u_1 v' < u_2$, and for all $k \in \mathbb{N}$, let $B_{|v|^{k-1}+j+1} = cone(u_1 v^k v' \alpha)$ where $\alpha \in \{0, 1\}$ is such that $u_1 v^k v' \bar{\alpha} < u_1 v^\omega$ (and we name $B'_j = cone(u_1 v')$). We have $\biguplus_{i\in\mathbb{N}} B_i$ a partition of the set of branches of t^ω which naturally induces $\biguplus_{i\in\mathbb{N}} C_i$, a partition of the plays played accordingly to σ^ω and τ^ω in $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$ (we also infer some C'_j from the B'_j). So $\mu^{\sigma^\omega,\tau^\omega}(Acc_G) = \sum_{i\in\mathbb{N}} \mu^{\sigma^\omega,\tau^\omega}(Acc_G \cap C_i)$. Then:

- $\mu^{\sigma^\omega,\tau^\omega}(C_0) = 0 \Rightarrow \mu^{\sigma^\omega,\tau^\omega}(Acc_G \cap C_0) = 0 = \mu^{\sigma^\omega,\tau^\omega}(C_0)$.

- C_1 is a set of plays which can be played in $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$ and also in $\mathcal{G}^{=1}_{\mathcal{A},t}$, therefore $\sigma^\omega|_{C_i} = \sigma|_{C_i}$ and $\exists\tau$, strategy of Adam in $\mathcal{G}^{=1}_{\mathcal{A},t}$ such that $\sigma^\omega|_{C_i} = \sigma|_{C_i}$. Thereby $\mu^{\sigma^\omega,\tau^\omega}(Acc_G \cap C_1) = \mu^{\sigma,\tau}(Acc_G \cap C_1) = \mu^{\sigma,\tau}(C_1) = \mu^{\sigma^\omega,\tau^\omega}(C_1)$.

- For $i \geq 2$, C_i is of the form $C_{|v|^{k-1}+j-1}$ where $k \in \mathbb{N}$ and j is the length of v' such that $u_1v' < u_2$. A play played accordingly to σ^ω and τ^ω on a branch of C_i begins on u_1v^kv'. We look at the state reached in the nodes u_1v^k and u_1v^kv', let say q. As the sets of possible states were the same in u_1 and u_2, they are the same in u_1 and u_1v^k. Thus there is a set of strategies of Adam in $\mathcal{G}^{=1}_{\mathcal{A},t}$ such that if we play along a branch beginning with u_1, we come in u_1 with the state q. Among them, there is a strategy τ such that τ^ω behaves on B_i, from the node u_1v^k on, as τ behaves on B'_j, from the node u_1 on.

  Let us assume that $\mu^{\sigma^\omega,\tau^\omega}(C_i \backslash (Acc_G \cap C_i)) \neq 0$. Since Acc_G is prefix closed, it means that $\mu^{\sigma,\tau}(C'_j \backslash (Acc_G \cap C'_j)) \neq 0$, which is absurd because σ is almost-surely winning in $\mathcal{G}^{=1}_{\mathcal{A},t}$. Consequently $\mu^{\sigma^\omega,\tau^\omega}(C_i \backslash (Acc_G \cap C_i)) = 0 \Rightarrow \mu^{\sigma^\omega,\tau^\omega}(Acc_G \cap C_i) = \mu^{\sigma^\omega,\tau^\omega}(C_i)$.

Therefore we have $\mu^{\sigma^\omega,\tau^\omega}(Acc_G) = \sum_{i\in\mathbb{N}} \mu^{\sigma^\omega,\tau^\omega}(C_i) = 1$ because $\biguplus_{i\in\mathbb{N}} C_i$ is a partition of the plays of $\mathcal{G}^{=1}_{\mathcal{A},t^\omega}$. □

Remark. When we pump, in addition of the copies of the branches, we add another branch (the one we pump on). This latter is *not accepting*. This is why this pumping lemma does not work for the classical acceptance. Yet there exists a weak version of this pumping lemma which still stands for the classical acceptance:

Let $\mathcal{A}$ be an n-states non-deterministic tree automaton, t be a tree in $\mathcal{L}_{Qual}(\mathcal{A})$ and u be a node of depth greater than n. Then there exist three pointed trees Δ_1, Δ_2 and Δ_3 such that $t = \Delta_1 \cdot \Delta_2 \cdot \Delta_3 \cdot t[u]$ and $\forall k \in \mathbb{N}, \Delta_1 \cdot \Delta_2^k \cdot t[u] \in \mathcal{L}(\mathcal{A})$.

4.2 Pumping on a Branch

We present here another pumping lemma which stands for both non-deterministic and alternating qualitative parity tree automata. This pumping has not the standard existential form of the pumping lemmas. It was found while trying to show that $\mathcal{L}_a = \{t | \exists u, t(u) = a\}$ was not a alternating qualitative language (see Corollary 1).

Lemma 4.3. *[Pumping on a branch] Let $\mathcal{L}$ be a qualitative alternating parity tree language (i.e. a language recognized by a qualitative alternating parity tree automaton). Let $(t_i)_{i\in\mathbb{N}}$ a sequence of trees in $\mathcal{L}$ such that $\exists b = (b_i)_{i\in\mathbb{N}} \in \{0;1\}^\omega$, a branch such that $\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, m \geq n \Rightarrow$ the pointed trees $(t_n, b_0b_1...b_n)$ and $(t_m, b_0b_1..$ are the same. This sequence converges: call its limit t_∞ (the limit is defined by prefix convergence).*

Then $t_\infty \in \mathcal{L}$.

Proof. Let $\mathcal{L}$ be a qualitative alternating parity tree language. Let $\mathcal{A}$ be the qualitative alternating parity tree automaton which recognizes $\mathcal{L}$. Let $(t_i)_{i\in\mathbb{N}}$ a sequence of trees in $\mathcal{L}$ such that $\exists b = (b_i)_{i\in\mathbb{N}} \in \{0;1\}^\omega$, a branch such that $\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, m \geq n \Rightarrow (t_n, b_0b_1...b_n) = (t_m, b_0b_1...b_n)$. $(t_i)_{i\in\mathbb{N}}$ converges to t_∞.

We want to show that $\mathcal{A}$ recognizes t_∞. To do this, we will contruct, from the strategies σ_i which enable Eve to win almost surely in $\mathcal{G}^{=1}_{\mathcal{A},t_i}$, an almost surely winning strategy σ_∞ for Eve in $\mathcal{G}^{=1}_{\mathcal{A},t_\infty}$.

- To construct σ_∞:

  Let $\mathcal{A} = (Q_\exists, Q_\forall, \Sigma = \{a;b\}, \Delta, q_{in}, Acc)$ where Acc is a parity acceptance condition.

  Then we can denote $\mathcal{G}^{=1}_{\mathcal{A},t_i} = (V_\exists, V_\forall, V^i_\Delta, \delta^i, (\epsilon, q_{in}), Acc)$ where $V_\exists = \{0;1\}^* \times Q_\exists$ and $V_\forall = \{0;1\}^* \times Q_\forall$. Indeed, only the set of vertices V^i_Δ (and its stochastic transition function δ^i) depend on the labelling of the tree and then on i. And we have: $\forall i \in \mathbb{N}$, $\sigma_i : V^*V_\exists \to V^i_\Delta$ (because in the arena $G^{=1}_{\mathcal{A},t_i}$, all the edges from $V_\exists$ go to V^i_Δ).

  We partition the set of branches of t_∞ on $\biguplus_{i=0}^\infty B_i \uplus \{b\}$ where $B_i = b_0b_1...b_i\{0;1\}^\omega = Cone(b_0b_1...b_i)$ (see Figure 8). We will define $\sigma_\infty : V^*V_\exists \to V^\infty_\Delta$ successively on the B_i's (*i.e.* on the existential nodes which are prefix of a branch in B_i). Indeed, the union of all these nodes covers the whole tree. We will note $\sigma|_{B_i}$ for $\sigma|_{(\{0^{i'}, i' \leq i\} \times Q_\exists) \cup ((0^i 1\{0;1\}^*) \times (Q_\exists))}$.

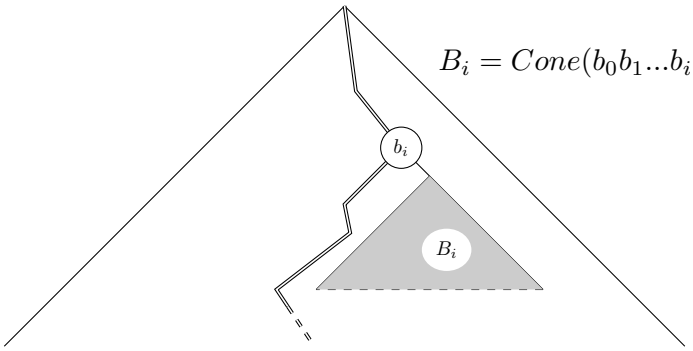


Figure 8: B_i

Suppose we defined σ_∞ for $(B_i)_{i<j}$, for some j. We want to define $\sigma_\infty|_{B_j}$. For this we work from a sequence obtained after successive extractions and gluings from $(t_k)_{k\in\mathbb{N}}$. At the j^{th} step, we name this sequence $(t_k^j)_{k\in\mathbb{N}}$ and $(\sigma_k^j)_{k\in\mathbb{N}}$ the associated sequence of strategies. We will start by building $(t_k^{j+1})_{k\in\mathbb{N}}$. (For the 0^{th}, we do the same but from the initial sequences, without extractions).

We look at $(t_k^j)_{k>j}$. In each of those trees, t_k, B_j has the same labelling. Thus the associated strategies $\sigma_k|_{B_j}$ do not depend on k. $\mathcal{V}(\{(b_0b_1...b_j, q), q \in Q_\exists\})$ is finite because it is a subset of $\{(b_0b_1...b_j, q), q \in Q_\exists\} \times Q \times Q$, which is finite. Besides, when she plays with the strategy σ_k, Eve chooses for each element $(b_0b_1...b_j, q)$ such that $q \in Q_\exists$, an element of $\mathcal{V}(\{(b_0b_1...b_j, q), q \in Q_\exists\})$. Therefore, among the $(\sigma_k^j)_{k>j}$, there exists an infinity of strategies $(\sigma_k^{j+1})_{k>j}$ such that $\forall k > j, \sigma_k^{j+1}|_{\{(b_0b_1...b_j,q),q\in Q_\exists\}} = \sigma_{j+1}^{j+1}|_{\{(b_0b_1...b_j,q),q\in Q_\exists\}}$. We note $(t_k^{j+1})_{k>j}$ the associated tree sequence and we glue it with $(t_k^j)_{k\leq j}$ to get $(t_k^{j+1})_{k\in\mathbb{N}}$.

Then we define $\sigma_\infty|_{B_j} = \sigma_{j+1}^{j+1}|_{B_j}$. We can define it this way because, thanks to the successive extractions made at the steps $i < j$, we are certain that $\sigma_{j+1}^{j+1}|_{\{b_0b_1...b_{j'},j'<j\}\times Q_\exists}$ is consistent with the strategies previously defined at the previous, on those nodes.

- To show that σ_∞ is almost surely winning for Eve in $\mathcal{G}_{\mathcal{A},t_\infty}$:

  We want to show that $\forall\tau_\infty$, an Adam's strategy in $\mathcal{G}_{\mathcal{A},t_b}$, $\mu^{\sigma_\infty,\tau_\infty}(Acc) = 1$.

  $\mu^{\sigma_\infty,\tau_\infty}(Acc) = \sum_{i=0}^{\infty} \mu^{\sigma_\infty,\tau_\infty}(Acc \cap B_i)$[10] because the plays played on a single branch have a measure equal to 0. Moreover, $\forall i$:

  There exists τ_{i+1}, an Adam's strategy for the acceptance game of t_{i+1}^{i+1} such that $\tau_{i+1}|_{B_i} = \tau_\infty|_{B_i}$. This simply comes from the fact that the labelling of B_i is the same in t_{i+1} and in t_∞). And, by construction $\sigma_{i+1}|_{B_i} = \sigma_\infty|_{B_i}$. So:

  $\mu^{\sigma_\infty,\tau_\infty}(Acc \cap B_i) = \mu^{\sigma_\infty|_{B_i},\tau_\infty|_{B_i}}(Acc \cap B_i) = \mu^{\sigma_{i+1}^{i+1}|_{B_i},\tau_{i+1}^{i+1}|_{B_i}}(Acc \cap B_i) = \mu^{\sigma_{i+1}^{i+1},\tau_{i+1}^{i+1}}(Acc \cap B_i) = \mu^{\sigma_{i+1}^{i+1},\tau_{i+1}^{i+1}}(B_i) = \mu^{\sigma_\infty,\tau_\infty}(B_i)$

  Thus we have: $\mu^{\sigma_\infty,\tau_\infty}(Acc) = \sum_{i=0}^{\infty} \mu^{\sigma_\infty,\tau_\infty}(Acc \cap B_i) = \sum_{i=0}^{\infty} \mu^{\sigma_\infty,\tau_\infty}(Acc \cap B_i) = \mu^{\sigma_\infty,\tau_\infty}(\cup_{i=0}^{\infty} B_i) = \mu^{\sigma_\infty,\tau_\infty}(\mathcal{P}(\Omega)\backslash\mathcal{P}(0^\omega)) = 1$

  where for some set A, we denote by $\mathcal{P}(A)$ the set of plays played on branches of A.

□

Corollary 1. *The qualitative alternating parity tree automata and the non-qualitative alternating parity tree automata are incomparable.*

[10] Actually B_i is a subset of branches while Acc is a subset of plays played in $\mathcal{G}_{\mathcal{A},t_\infty}$ but the partition of the branches of t_∞, $(B_i)_{i\in\mathbb{N}}$ can be canonically transposed to a partition of the plays played in $\mathcal{G}_{\mathcal{A},t_\infty}$ because each play is played on a branch. We still note $(B_i)_{i\in\mathbb{N}}$ this partition to simplify.

Proof. We show that there exists a non-qualitative alternating parity tree language which cannot be recognized by a qualitative alternating parity tree automaton: $\mathcal{L}_a = \{t | \exists u, t(u) = a\}$ is not a qualitative alternating language.

Indeed by contradiction, assume that $\mathcal{L}_a$ is qualitative alternating. $\forall i \in \mathbb{N}$, we define t_i as follows:

$$t_i : \left\{ \begin{array}{lcl} 0^i & \mapsto & a \\ u \neq 0^i & \mapsto & b \end{array} \right.$$

$(t_i)_{i\in\mathbb{N}}$ is a sequence of trees which belong to $\mathcal{L}_a$ and satisfy the condition of the Lemma 4.3. $(t_i)_{i\in\mathbb{N}}$ converges to t_b, the tree labelled only with b's. Hence $t_b \in \mathcal{L}_a$, which is absurd. Therefore $\mathcal{L}_a$ is not qualitative alternating.

For the other sens, the result is already true for the qualitative non-deterministic automata and the non-qualitative non deterministic automata (equal to non-qualitative alternating, see the Simulation theorem, section 3).

□

Corollary 2 (Non-Complementation)**.** *The set of the qualitative alternating parity tree languages is not closed under complementation.*

Proof. We will show that there exists a qualitative alternating parity tree language $\mathcal{L}$ such that $\overline{\mathcal{L}}$ is not qualitative alternating.

Let $\mathcal{L} = \{t |$ the set of branches containing a finite number of b's has measure $1\}$. A qualitative deterministic co-Büchi automaton which puts a forbidden state when we read a b, recognizes $\mathcal{L}$. $\mathcal{L}$ is qualitative deterministic and so qualitative alternating.

$\overline{\mathcal{L}} = \{t |$ the set of branches containing a infinite number of b's has a measure strictly greater than $0\}$. By contradiction, assume that $\overline{\mathcal{L}}$ is qualitative alternating. $\forall i \in \mathbb{N}$, we define t_i as follows:

$$t_i : \left\{ \begin{array}{lcl} 0^i \cdot \{0;1\}^* & \mapsto & b \\ u \neq 0^i & \mapsto & a \end{array} \right.$$

$(t_i)_{i\in\mathbb{N}}$ is a sequence of trees which belong to $\overline{\mathcal{L}}$ and satisfy the condition of the lemma 4.3. $(t_i)_{i\in\mathbb{N}}$ converges to t_a, the tree labelling only with a's. Hence $t_a \in \overline{\mathcal{L}}$, which is absurd. Therefore $\overline{\mathcal{L}}$ is not qualitative alternating.

□

Remark. Since the qualitative non-deterministic tree parity langages are not closed under complementation we expected this result. It is not a good news because it mean that they do not form a boolean algebra and so there is no interesting logic equivalent to these automata (there can not be a logic including negation which is equivalent).

5 Mostowski Parity Hierarchy

When using a parity condition, each state of the automaton is labelled by a integer - called colour - belonging to a finite interval $[i; j]$. A run is accepting if the smallest colour seen infinitely often is even. A language is said to be i, j-feasible if there exists a parity automaton using the interval of priorities $[i; j]$ accepting this language. Of course, the language does not change if we shift all priorities by steps of 2 or -2 so we can restrict ourselves to $i = 0$ or $i = 1$. It is also clear that the bigger the interval $[i; j]$ is, the more tree languages are i, j-feasible. Mostowski first studiedt this parameter [Mos85], and the corresponding ladder-shaped hierarchy (depicted in Figure 9 for tree automata) is named after him.

This Mostowski hierarchy exists in different variants according to the nature of the transition relation used by the automaton. We study in this section the first level of this hierarchy parity tree automata: the Büchi and co-Büchi conditions.

5.1 For the Classical Semantics

For the classical acceptance, the hierarchy is strict for all standard models of automata: deterministic [Wag77], non-deterministic [Niw86], and alternating [Arn99].

Here we will develop the well-known fact that the Büchi and the co-Büchi conditions are incomparable for tree automata.

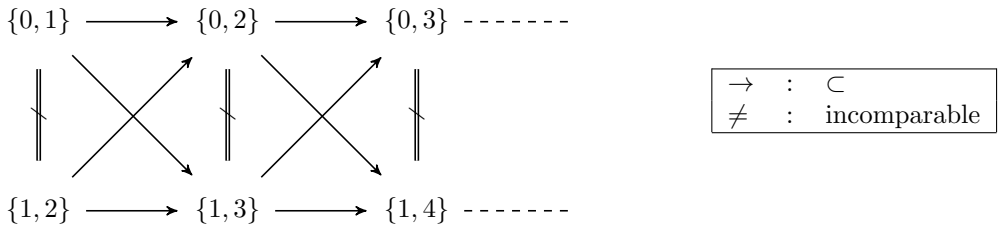


Figure 9: The Mostowski parity hierarchy.

Proposition 5.1. *The tree languages 1,2-feasible are not included in the tree languages 0,1-feasible.*

Proof. We will show that $\mathcal{L}_b = \{t | \forall B \in t, \exists^{<\infty} b \text{ on } B\}$, the set of trees whose all branches contain only finitely many b's, is 1,2-feasible but not 0,1-feasible.

This language is 1,2-feasible: when we read a b we go to a forbidden state.

By contradiction, assume that $\mathcal{L}_b$ is 0,1-feasible. There exists a Büchi tree automaton $\mathcal{A}$ n states that recognizes $\mathcal{L}_b$. We consider the tree t such that $t(u) = b$ if $\exists k, (0^*1)^{k \leq n}$ and $t(u) = a$ otherwise. The tree t is recognized by $\mathcal{A}$: there exists ρ an accepting run of $\mathcal{A}$ over t. We look at the branch 0^ω. It is accepting, so there exists a final state on this branch, at the node u_0. We turn right. the branch $u_0 10^\omega$ is accepting, so there is a final state at the position u_1. We do the same $n-1$ more times. We have $n+1$ final states, at the positions $u_0, u_1, ...u_n$. And $\mathcal{A}$ has only n states, so $\exists i < j \in \{0, 1, ..., n\}, \rho(u_i) = \rho(u_j) = q_f$ where q_f is a final state of $\mathcal{A}$. We pump on those state, in the run.

We define two pointed trees: $\Delta_1 = (t, u_i)$ and $\Delta_2 = (t[u_i], u_j)$. We have $t = \Delta_1 \cdot \Delta_2 \cdot t[u_j]$. The tree $\Delta_1 \cdot \Delta_2^\omega$ is recognized by $\mathcal{A}$. Indeed the only branches we add compared to t are either copies of branches in Δ_2 (only the prefixes differ) or the branch we pumped on. The copies of branches of Δ_2 are accepting because t is recognized by $\mathcal{A}$. The branch we pumped on is accepting because we pumped on infinite states, and so there is an infinity of them on the branch. And this branch contains infinitely many b's. So the tree $\Delta_1 \cdot \Delta_2^\omega$ should not be recognized. So this is absurd: $\mathcal{L}_b$ is not 0,1-feasible.

□

5.2 For the Qualitative Semantics

We do not know if the Mostowski parity hierarchy is still applicable for the qualitative semantics.

As we have seen along this report, for the qualitative semantics there also exist some results that differ for qualitative Büchi tree automata and qualitative co-Büchi tree automata. In fact the emptiness problem is decidable for qualitative Büchi tree automata [FPS13] and undecidable for qualitative co-Büchi tree automata (see section 3.2.2 for more details). This leads to think that qualitative Büchi tree languages and qualitative co-Büchi tree languages are indeed incomparable, and that perhaps the hierarchy still stands. We study here the possible incomparability of Büchi and co-Büchi conditions. Indeed if the Büchi and co-Büchi conditions were comparable, there would be no effective construction to transform one in another.

As for the classical semantics, we would like to show that the qualitative tree languages 1,2-feasible are not included in the qualitative tree languages 0,1-feasible, at least for non-deterministic automata. The qualitative version of $\mathcal{L}_b$, we named $\mathcal{L}'_b$ seems a good candidate, because of the following property:

Proposition 5.2. *If there exists a qualitative non-deterministic Büchi automaton $\mathcal{A}$ which recognizes $\mathcal{L}'_b = \{t | \forall^{=1} B \in t, \exists^{<\infty} b \text{ on } B\}$; then for all qualitative non-deterministic co-Büchi automaton $\mathcal{B}$, there exists a qualitative non-deterministic Büchi automaton $\mathcal{C}$ which recognize $\mathcal{L}(\mathcal{B})$.*

Proof. We do not give the entire proof which is rather simple, but only the principle.

$\mathcal{C}$ works as follows: it guesses an accepting run of $\mathcal{B}$, labels the tree by the colours given by this run, and lets $\mathcal{A}$ verify that there is a finite number of 1 on almost all branches (*i.e.* that the co-Büchi condition is satisfied). □

In order to prove that $\mathcal{L}'_b$ was not 0,1-feasible by a qualitative non-deterministic automaton, we tried to follow the same idea than for the classical acceptance. The tree we pumped for the classical acceptance is a counter-example to the non-deterministic automaton which would wait, until it guesses a node from where there is not any b left. In fact, in the example developed in the proof of the result for the classical semantics, for every node u in the branch 0^ω, there is a node below which is labbeled by a b: the node $u \cdot 1$.

In the same idea we found an counter-example to the qualitative non-deterministic automaton which would wait, until it guesses there is not any b left. This is the tree t such that: $t(u) = b$ if $u = 0, 1^*1^{f(1)}$ or $0, 1^*1^{f(1)}0, 1^*1^{f(2)}$ or ... and $t(u) = a$ otherwise, where f is a function which grows exponentially. This tree seems to be a good counter-example, but we did not success in showing that it indeed is.

6 Conclusion

We tried during this internship to draw the landscape of the qualitative parity tree automata class. We know that for the classical acceptance the universal and alternating parity tree automata can respectively be simulated by deterministic and non-deterministic parity tree automata. One of the main purpose was to know whether it was still true for the qualitative acceptance.

We have neither proved that it was the case nor that it was not. Yet we gave arguments which lead to think that, for the qualitative acceptance, the alternating parity tree automata are more expressive. While trying to find an example of an alternating language which would not be non-deterministic, we also characterize the qualitative tree language through two pumping lemmas.

Another element to define in the cartography of the qualitative tree languages was to define the frontier between the Büchi and co-Büchi automata. We did not deepen this subject, still we gave some element of answer. To follow this idea it could be interesting to try to characterize the Büchi $\cap$ co-Büchi tree languages.

In doing so, we found some other interesting results. The major one is the non-complementation of alternating parity tree automata. The alternation is often an easy tool to complement automata (we transform the existential states into universal ones and the universal states into existential ones, and dualize the acceptance condition). So this result has been a disappointment. It makes the qualitative class as it is useless. To make the qualitative class useful, we have to extend it (the positive measure could be a trail, but brings its problem, for the union and the intersection for example), or to restrict it.

Acknowledgments

I would like to thank my supervisors, Olivier Serre and Nathanaël Fijalkow, who made themselves available, to explain to me some results and to listen to mines. Thanks to them, it has been a pleasure to discover the world of computer science research.

I also would like to thank Olivier Serre and and all the anonymous reviewers for their thorough reviews and very useful comments that helped improve the clarity and the relevance of this report.

References

[Arn99] André Arnold. The μ-calculus alternation-depth hierarchy is strict on binary trees. 33(4/5):329–340, 1999.

[BGB12] Christel Baier, Marcus Größer, and Nathalie Bertrand. Probabilistic ω-automata. *Journal of the ACM*, 59(1):1, 2012.

[CHS14] Arnaud Carayol, Axel Haddad, and Olivier Serre. Randomisation in automata on infinite trees. 2014. 37 pages. To appear.

[FPS13] Nathanaël Fijalkow, Sophie Pinchinat, and Olivier Serre. Emptiness of alternating tree automata using games with imperfect information. volume 24 of *LIPIcs*, pages 299–311. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

[Gim09] Hugo Gimbert. Randomized Strategies are Useless in Markov Decision Processes. July 2009.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500, 2002.

[Mos85] Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation theory*, volume 208, pages 157–168. 1985.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata. *Theoretical Computer Science*, 141(1&2):69–107, 1995.

[Niw86] Damian Niwiński. On fixed-point clones (extended abstract). volume 226, pages 464–473, 1986.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, volume III, pages 389–455. 1997.

[Wag77] Klaus W. Wagner. Eine topologische charakterisierung einiger klassen regulärer folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 13(9):473–487, 1977.