On the relationship between monadic and weak monadic second order logic on arbitrary trees, with applications to the mu-calculus

David Janin, Giacomo Lenzi

Fundamenta Informaticae (Submitted version) 1–19 1
IOS Press

On the relationship between monadic and weak monadic second order logic on arbitrary trees, with applications to the mu-calculus

David Janin
LaBRI
Université de Bordeaux I - ENSEIRB
351 cours de la libération,
F-33 405 Talence cedex
France
janin@labri.fr

Giacomo Lenzi
Dipartimento di Matematica
Università di Pisa
via Buonarroti 2
I-56127 Pisa
Italy
lenzi@mail.dm.unipi.it

Abstract. In 1970 [26], in *Weakly definable relations and special automata, Math. Log. and Found. of Set Theory, pp 1-23*, Rabin shows that a language is recognizable by a tree automaton with Büchi like infinitary condition if and only if it is definable as the projection of a weakly definable language.

In this paper, we refine this result characterizing such languages as those definable in the monadic Σ_2 level of the quantifier alternation depth hierarchy of monadic second order logic (MSO).

This new result also contributes to a better understanding of the relationship between the quantifier alternation depth of hierarchy of MSO and the fixpoint alternation depth hierarchy of the mu-calculus: it shows that the bisimulation invariant fragment of the monadic Σ_2 level equals the $\nu\mu$-level of the mu-calculus hierarchy.

Address for correspondence: David Janin, LaBRI, Université de Bordeaux I, 351, cours de la libération, 33 405, Talence cedex, France

1. Introduction

1.1. Motivation

In the 60's, Büchi and Rabin [4, 25] proved the decidability of monadic second order logic (MSO) on infinite words and infinite trees respectively. These two deep results of mathematical logic were obtained by means of automata theoretic characterizations of the expressive power of MSO on these structures. In the late 70's, it was realized [24] that infinite words or trees can be used as models of the behavior of computer systems. Consequently, the results of Büchi and Rabin, with the underlying theory of automata on infinite structures, became of direct interest to computer scientists. Since then, a considerable amount of research effort has been devoted to the development of the theory towards its application in computer science. This effort has produced a large number of deep and mathematically appealing results relating logic, automata and fixpoint calculi [29, 2, 9].

A central feature of this theory is that most specification languages (also called temporal logics [7]) that are used for describing and verifying the potential behaviors of programs do share the following properties:

1. they are fragment of MSO logic; the latter can even be seen as a (high level) assembly like language into which all these logics can be compiled,

2. they are invariant under bisimulation [23]; indeed, these logics talk about systems' behaviors and two bisimilar (models of) systems are most of the time (not to say always) considered to have the same behavior [18].

This remark motivates the study of the bisimulation invariant fragment of MSO.

Kozen's mu-calculus [15], an extension of modal logic by means of restricted set quantifiers (namely inductive and co-inductive definitions of sets), plays there a fundamental role. It is known since some time [22, 2] that, over trees (say on the binary tree), the mu-calculus is as expressive as MSO. Further studies show that, over arbitrary graphs, the mu-calculus just equals the bisimulation invariant fragment of MSO [14] and such a relationship is even richer than expected: as announced in [11], the first levels of the bisimulation invariant fragment of the monadic quantifier alternation depth hierarchy of MSO equal, one by one, the first levels of the fixpoint alternation hierarchy of the mu-calculus [3, 1]. More precisely, Van Benthem first shows that the bisimulation invariant fragment of first order logic (FO, the 0th level of the monadic hierarchy) equals modal logic (the 0th level of the mu-calculus hierarchy). And actually, this equality holds up to the level Σ_2 of the monadic hierarchy [11].

1.2. Main contribution

The purpose of this paper is to give a clear and complete presentation of the last level correspondence. Namely, we give here a complete proof of the following result:

Theorem 1.1. The bisimulation invariant fragment of the level Σ_2 of the monadic hierarchy equals the $\nu\mu$-level of the mu-calculus hierarchy.

In turn, this level is known to be as expressive as (modal) tree automata with Büchi conditions [22, 13].

This result refines Rabin's own logical characterization of Büchi definable properties of the binary tree as projections of weak MSO properties [26].

As a side result, we also prove and use the fact that the languages of trees definable by first-order formulas (FO-formulas) are boolean combination of topologically closed definable languages, i.e. boolean combination of languages of the μ-level of the mu-calculus hierarchy [12].

1.3. Related works

We shall also mention that, in the binary tree, the main result presented in this paper has already been announced by the second author [17] with quite a long and technical proof argument. Later a quite simpler argument, but for a slightly weaker result, has been given by Skurczynski [28]. This last result is weaker since it handles monadic Σ_2 formulas over the binary tree with FO kernels (called principal formulas) that are weaker than arbitrary FO-formulas as we are using here.

Still, in the following generalization of the binary case, we adopt several arguments from Skurczynski work and, quite distinctly, we handle FO-formulas by means of topological considerations obtaining thus a simple though presumably unknown yet automata theoretic bound on the expressive power of FO logic on trees.

At last, one shall observe that the monadic Σ_2 case does not follow (say by simple inductive argument) from the former characterization of the bisimulation invariant fragment of monadic Σ_1 [12]. In fact, the monadic Σ_1 kernel of a bisimulation invariant monadic Σ_2-formula is not necessarily invariant. And, over trees, the bisimulation invariant fragment of monadic Σ_1 is strictly weaker than full monadic Σ_1 as illustrated, for instance, by the formula $\exists x p(x)$ that is not bisimulation invariant.

1.4. Organization of the paper

The paper is organized as follows. In Section 2 we review standard notions about graphs, trees and bisimulation. The central notions of κ-expansions [14] that induce canonical representatives for all classes of bisimilar graphs is given. We also review the standard definition of the prefix topology on finitely branching trees.

In Section 3 we review the logics we use. In addition to first-order, monadic second order logic, and Kozen's *modal* mu-calculus, we also define the *counting* mu-calculus. This calculus is defined in a way similar to the mu-calculus except that one can use counting modalities instead of standard modalities. It relates with monadic second order logic over trees in the same way the modal mu-calculus relates with the bisimulation invariant fragment of monadic second order logic over trees.

In Section 4 we review various definitions of tree automata: alternating and non deterministic Büchi tree automata in their standard or weak versions. Closure properties and various technical expressiveness results about the class of languages definable by these automata are given. Definitions of closed or open automata (that characterize recognizable closed or open languages of trees) are also given.

The last section is devoted to the proof arguments of Theorem 1.1. More precisely, we successively show that every FO-formula over trees is equivalent to a weak non deterministic automaton. By projection, this shows that this holds as well for monadic Σ_1 formulas. Then, by complementation and simulation à la Muller and Schupp, we prove that every monadic Π_1 is equivalent to a Büchi automaton and thus, by projection again, every monadic Σ_2 formula as well. Then, the counter-saturation technique that has been developed in [14] can be applied in order to conclude the proof.

2. Graphs and trees

We review here the definition of transition systems, bisimulation equivalence and κ-expansion of transition systems which capture in some sense bisimulation equivalence. We also define a notion of counting bisimulation by adding a "local bijection" constraint to the notion of bisimulation. This new definition makes statements more uniform: counting bisimulation is the equivalence induced by unraveling as bisimulation is, in a sense, the equivalence induced by κ-expansions.

Because a transition system is simply a directed graph with a distinguished vertex called its root, we use the vocabulary of (directed) rooted graphs. In order to simplify statements and proofs, we consider only graphs built over a single binary relation symbol. All the results presented here can easily be generalized to (finitely) labeled directed graphs, i.e. graphs built over a finite set of binary relation symbols.

As trees are important when dealing with bisimulation, we also review some standard notation and definitions of trees.

2.1. Graphs and trees

Let $Prop$ be a finite set of unary predicate symbols and let E be a binary relation symbol. A *rooted graph*, simply called *graph* in the sequel, is a tuple:

$$M = \langle V^M, r^M, E^M, \{p^M\}_{p \in Prop} \rangle$$

with a set V^M of *vertices*, a *root* $r^M \in V^M$, a binary *successor relation* $E^M \subseteq V^M \times V^M$ and for each $p \in Prop$, a subset $p^M \subseteq V^M$.

We say that a vertex v is a successor of u when $(u, v) \in E^M$. The set of all successors of u is denoted by $Succ^M(u)$. We also use the notation $dom(M)$ for the domain V^M of the graph M.

Given a vertex $v \in V^M$, let $\lambda^M(v) \subseteq Prop$ be the set of predicate symbols defined by

$$\lambda^M(v) = \{p \in Prop : v \in p^M\}$$

and called the *color* of vertex v. In order to simplify the notation, we may use in the sequel the coloring function λ^M instead of the interpretation of each predicate symbol. In the sequel, we may also omit the superscript ^{M}, when there is no ambiguity, thus simply witting $M = \langle V, r, E, \lambda \rangle$ for such a graph.

A *directed path* in a M is a non empty finite or infinite word

$$w \in V^M.(V^M)^* \cup V^M.(V^M)^\omega$$

such that whenever $w = w_1.u.u'.w_2$ with $w_1 \in (V^M)^*$, $u \in V^M$, $u' \in V^M$ and $w_2 \in (V^M)^* \cup (V^M)^\omega$ one has $(u, u') \in E^M$. The length $|w|$ of a finite directed path w is defined as the number of occurrences of elements of V^M in w, i.e. when $w = u_0.u_1.\cdots.u_n$ then $|w| = n + 1$. In this case, we say u_0 is the source of w, u_n is the target of w and w is a directed path from u_0 to u_n.

Given any non zero cardinal κ, a *κ-indexed path* in M is defined similarly by adding elements of κ between vertices of a path. More precisely, a *κ-indexed path* in M is a non empty finite or infinite word

$$w \in V^M.(\kappa.V^M)^* \cup V^M.(\kappa.V^M)^\omega$$

such that the word $\pi_V(w)$ obtained from w by removing all elements of κ is a directed path. length, source or target of a κ-indexed path w are defined as the length, source or target of the path $\pi_V(w)$.

A tree is a graph M such that, for each vertex $v \in V^M$ there is one and only one path from r^M to v.

When M is a tree, the (directed) distance $d(u, v)$ between two vertices u and v is $n - 1$ when n is the length of the unique (directed) path from u to v or from v to u when it exists, ∞ otherwise. For every integer $h \geq 0$, we define the h-prefix $P_h(M, v)$ of a vertex v in M to be the (finite) tree rooted in v induced by the set of vertices of M at distance at most h from v. When v is simply the root sr^M we simply write $P_h(M)$.

If u is a vertex of a graph M, the *degree* of u is the number of its successors. The *degree* of a graph M is the supremum (be it finite or infinite) of the degrees of its vertices. A graph is called *finitely branching* if every vertex has finite degree. Observe that a finitely branching graph does not necessarily have finite degree.

On the set FBT of finitely branching tree, we consider the *prefix topology* defined by the set of basic open sets of the form :

$$\mathcal{O}_F = \{M \in FBT : \exists h \in \omega P_h(M) = F\}$$

where F is a finite tree. As shown in [12] for instance, this topology is Hausdorff, i.e. two trees that belong to the same open sets are equal (up to isomorphism). It is also metric as shown by taking, for instance the distance $d(M, N)$ between two trees M and N to be $d(M, N) = \min\{1/2^h : h \in \omega, P_h(M) = P_h(N)\}$ if it is defined of 1 otherwise. It even satisfies some weak form of compactness (again see [12]).

By extension, we say that a class $\mathcal{C}$ of arbitrary trees is closed (resp. open) when $\mathcal{C} \cap FBT$ is closed (resp. open). Observe that, on a topological point of view, this does not make much sense. However, in this paper, we essentially consider (restricted) classes of trees that are completely characterized by the finitely branching trees they contain. So this extension makes sense.

2.2. Bisimulation and models' expansions

Two graphs M and N are called *bisimilar* when there exists a relation $R \subseteq V^M \times V^N$, called a *bisimulation relation*, such that $(r^M, r^N) \in R$ and for every $(u, v) \in R$ and $p \in Prop$, $\lambda^M(u) = \lambda^N(v)$, and, whenever $(u, u') \in E^M$ for some u', then there exists v' such that $(v, v') \in E^N$ and $(u', v') \in R$, and whenever $(v, v') \in E^N$ for some v', then there exists u' such that $(u, u') \in E^M$ and $(u', v') \in R$.

If, in addition, for each $(u, v) \in R$, R establishes a bijection between $Succ(u)$ and $Succ(v)$, then we say that R is a *counting bisimulation*. In this case, we say that graph M and N are *counting bisimilar*.

The *κ-expansion* $T^\kappa(M)$ of a system M is defined as follows : let $V^{T^\kappa(M)}$ be the set of all finite κ-indexed paths of M with source r^M, the root $r^{T^\kappa(M)}$ equal r^M, the relation $E^{T^\kappa(M)}$ be the set of all pairs of the form $(w.u, w.u.k'.u') \in V^{T^\kappa(M)} \times V^{T^\kappa(M)}$ with $w \in (V^M.\kappa)^*$, u and $u' \in V^M$ and $k' \in \kappa$ such that $(u, u') \in E^M$. Moreover, let $p^{T^\kappa(M)}$ be the set of all κ-indexed paths of the form $w.u \in V^{T^\kappa(M)}$ with $w \in (V^M.\kappa)^*$ and $u \in p^M$.

When $\kappa = 1$, the κ-expansion of M, from now on denoted by $T(M)$, is nothing but what we usually call the *unraveling* of the graph M from its root r^M. In particular, vertices of $T(M)$ are all the finite paths from the root r^M in M.

Observe that for every cardinal κ and for every graph M, the κ-expansion $T^\kappa(M)$ of M is a tree. Moreover, the particular notion of 1-expansion (or unraveling) allows us to characterize trees in a very

simple way: a *tree* is a graph isomorphic to its unraveling.

The notion of unraveling (or 1-expansion) is related to counting bisimulation as follows.

Lemma 2.1. For any graphs M and N, M and N are counting bisimilar if and only if their unravelings $T(M)$ and $T(N)$ are isomorphic.

Proof:
This fact follows immediately from the following observations: the functional relation from $V^{T(M)}$ to V^M that maps each finite path to its target is a counting bisimulation between $T(M)$ and M; and, over trees, a counting bisimulation relation is just an isomorphism. □

In a quite similar way, the notion of κ-expansion also gives in some sense canonical representatives of equivalence classes under bisimulation.

Lemma 2.2. (see [14])
For any infinite cardinal κ and for any graphs M and N of cardinality at most κ, M and N are bisimilar if and only if their κ-expansions $T^\kappa(M)$ and $T^\kappa(N)$ are isomorphic.

Proof:
Let R be a bisimulation relation between M and N and let cardinal κ be as above. Let R' be the relation between vertices of both $T^\kappa(M)$ and $T^\kappa(N)$ that relates any two κ-indexed paths of the same length whose targets belongs to R. Relation R' is a bisimulation relation. Moreover, provided κ is infinite (so that, with the help of the axiom of choice, $\kappa.\kappa = \kappa$) and big enough (actually not smaller than the degree of M and N), one can check that relation R' can be refined to become the relation of an isomorphism.

The converse is immediate since any graph M is bisimilar with its κ-expansion $T^\kappa(M)$. □

The assumption, in Lemma 2.2, that κ is infinite is essential. In fact, let M (resp. N) be the graph defined by a single edge (resp. two edges only) from the root, with all vertices labeled identically. They are bisimilar (and not counting bisimilar). However, for any finite cardinal κ distinct from zero, $T^\kappa(M)$ and $T^\kappa(N)$ are not isomorphic.

3. Logic and modal or counting mu-calculus

In this section we review the definition of first-order logic (FO) and monadic second-order logic (MSO), and the counting and modal propositional mu-calculus [15]. All logics are interpreted over graphs. Any graph M, as defined above, is a FO-structure on the vocabulary $\{r, E\} \cup Prop$ with r a constant symbol standing for the root, E a binary relation symbol and $Prop$ a set of unary relation symbols.

3.1. First-order and monadic second-order logic

Let $var = \{x, y, \cdots\}$ and $Var = \{X, Y, \cdots\}$ be disjoint sets of, respectively, first-order and monadic second-order variable symbols.

First-order logic over the vocabulary $\{r, E\} \cup Prop$ can be defined as follows. The set of FO-formulas is the smallest set containing the formulas $p(t)$, $t = t'$, $E(t, t')$, $X(t)$ for $p \in Prop$, $X \in Var$

and $t, t' \in var \cup \{r\}$, which is closed under negation $\neg$, disjunction $\vee$, conjunction $\wedge$ and existential $\exists$ and universal $\forall$ quantifications over FO variables.

The set of monadic second-order (MSO) formulas over the same vocabulary is the smallest set containing all FO-formulas and closed under negation $\neg$, disjunction $\vee$, conjunction $\wedge$ and existential $\exists$ and universal $\forall$ quantifications over FO and MSO variables.

We write $\varphi(x_1, \cdots, x_m, X_1, \cdots, X_n)$ for an MSO formula φ with free first-order variables among $\{x_1, \cdots, x_m\}$ and free set variables among $\{X_1, \cdots, X_n\}$. If M, v_1, ..., $v_m \in V^M$, and V_1, ..., $V_n \subseteq V^M$, we write

$$M \models \varphi(v_1, \cdots, v_m, V_1, \cdots, V_n)$$

to say that formula φ is true in M, or M satisfies φ, under the interpretation mapping each FO variable x_i to the vertex v_i and each set variable X_j to the set V_j. This satisfaction relation is defined in the standard way [27, 6].

A class $\mathcal{C}$ of graphs is called *MSO definable* when there exists a sentence $\varphi \in MSO$, i.e. an MSO formula with no free variable, such that $M \in \mathcal{C}$ if and only if $M \models \varphi$. A class $\mathcal{C}$ of transition systems is *bisimulation closed* (resp. *counting bisimulation closed*) if whenever $M \in \mathcal{C}$ and M' is bisimilar (resp. counting bisimilar) to M then $M' \in \mathcal{C}$. A sentence φ is *bisimulation invariant* (resp. *counting bisimulation invariant*) if the class of transition systems it defines is bisimulation closed (resp. counting bisimulation closed). Observe that bisimulation invariance implies counting bisimulation invariance since a counting bisimulation relation is a bisimulation relation.

Remark 3.1. Observe, by applying the characterization of bisimulation or counting bisimulation by κ-expansions (Lemmas 2.1 and 2.2), that if two bisimulation (or counting bisimulation) invariant formulas are equivalent, for all infinite κ, on κ-expansions of graphs (or trees), they are equivalent on arbitrary graphs.

Finally, the monadic quantifier alternation depth hierarchy is defined as follows. The first (or zeroth) level $\Sigma_0 = \Pi_0$ is defined as the set of all formulas of first order logic. Then, for each integer k, the level Σ_{k+1} (resp. level Π_{k+1}) is defined as the set of all formulas of the form $\exists X_1 \cdots \exists X_n \varphi$ with $\varphi \in \Pi_k$ (resp. $\forall X_1 \cdots \forall X_n \varphi$ with $\varphi \in \Sigma_k$). The bisimulation invariant (resp. unwinding invariant) fragment of the level Σ_k of $MSOL$ sentences is defined as the set of all bisimulation invariant (resp. unwinding invariant) sentences of Σ_k.

3.2. Modal and counting mu-calculus

The set of the modal μ-calculus formulas is the smallest set containing $Prop \cup Var$ which is closed under negation, disjunction and the following formation rules:

- if α is a formula then $\Diamond\, \alpha$ and $\Box\alpha$ are formulas,
- if α is a formula and X occurs only positively (i.e. under an even number of negations) in α then $\mu X.\alpha$ and $\nu X.\alpha$ are formulas.

The set of counting μ-calculus formulas is defined as above replacing standard modalities $\Diamond$ and $\Box$ by counting modalities $\Diamond_k$ and $\Box_k$ for any integer k.

A formula φ is said in *positive normal form* when negation only applies to atomic sub-formulas, i.e. constant or variable predicates of $Prop \cup Var$.

We use the same convention as for MSO with free variables, i.e. we denote by $\alpha(X_1, \cdots, X_n)$ a formula with free variables among $\{X_1, \cdots, X_n\}$. For convenience, we may also omit these free set variables in formula α considering implicitly that graphs have been built over the set of unary predicate symbols $Prop' = Prop \cup \{X_1, \cdots, X_n\}$. In the sequel, we call *fixed point formula* any formula of the modal or counting μ-calculus.

With each fixed point formula α, we associate a formula $\varphi_\alpha(x)$ of MSO with one free FO variable x and the same free set variables as α (implicitly added to the vocabulary) defined as follows. Let $p \in Pred$, α and β be fixed point formulas, X be a set variable, x and z be FO variables, and $\overline{z} = (z_1, \cdots, z_k)$ be a k-tuple of FO variables.

- Atomic formulas :
  $\varphi_p(x) = p(x)$ and $\varphi_X = X(x)$,
- Boolean connectives :
  $\varphi_{\alpha\wedge\beta}(x) = \varphi_\alpha(x) \wedge \varphi_\beta(x)$, $\varphi_{\alpha\vee\beta}(x) = \varphi_\alpha(x) \vee \varphi_\beta(x)$
  and $\varphi_{\neg\alpha}(x) = \neg\varphi_\alpha(x)$
- Modalities :
  $\varphi_{\Diamond\alpha}(x) = \exists z\ E(x,z) \wedge \varphi_\alpha(z)$,
  $\varphi_{\Box\alpha}(x) = \forall z\ (E(x,z) \Rightarrow \varphi_\alpha(z))$
- Counting modalities :
  $\varphi_{\Diamond_k\alpha}(x) = \exists \overline{z}\ \mathit{diff}(\overline{z}) \wedge \bigwedge_{i\in[1,k]} E(x,z_i) \wedge \varphi_\alpha(z_i)$,
  $\varphi_{\Box_k\alpha}(x) = \forall \overline{z}\ ((\mathit{diff}(\overline{z}) \wedge \bigwedge_{i\in[1,k]} E(x,z_i)) \Rightarrow \bigvee_{i\in[1,k]} \varphi_\alpha(z_i))$,
- Fixed points :
  $\varphi_{\mu X.\alpha(X)}(x) = \forall X((\varphi_{\alpha(X)} \subseteq X) \Rightarrow X(x))$,
  $\varphi_{\nu X.\alpha(X)}(x) = \exists X(X \subseteq \varphi_{\alpha(X)}) \wedge X(x)$.

There, $\mathit{diff}(\overline{z})$ is the quantifier-free FO-formula stating that $z_i \neq z_j$ for all $i \neq j$, $\varphi_{\alpha(X)} \subseteq X$ is the MSO formula $\forall z(\varphi_{\alpha(X)}(z) \Rightarrow X(z))$, and, similarly, $X \subseteq \varphi_{\alpha(X)}$ is the MSO formula $\forall z(X(x) \Rightarrow \varphi_{\alpha(X)}(z))$.

For any fixed point formula α, we write $M \models \alpha$ when $M \models \varphi_\alpha(r)$. We say that an MSO sentence φ is equivalent to a fixed point formula α when $\models \varphi_\alpha(r) \Leftrightarrow \varphi$. Likewise, two fixed point formulas α and β are said to be equivalent when $\models \varphi_\alpha(r) \Leftrightarrow \varphi_\beta(r)$.

Lemma 3.1. Any fixed point formula is equivalent to a fixed point formula in positive normal form.

Observe that modalities $\Diamond$ and $\Diamond_1$ on the one hand, and modalities $\Box$ and $\Box_1$ on the other hand, have equal meaning. So the counting mu-calculus is an extension of the modal mu-calculus. This extension is proper. In fact, it follows from Lemma 3.2 below that, for instance, predicate $\Diamond_2\, p$ is not definable in the modal mu-calculus.

As is well known, the interpretation over a graph M of the predicate defined by $\mu X.\alpha(X)$ (resp. $\nu X.\alpha(X)$) is the least (resp. the greatest) solution of (the interpretation over $dom(M)$ of) the set equation defined by $X = \alpha(X)$.

Further analysis of the basic properties of the modal (resp. the counting) fixed point calculus shows that it does not distinguish bisimilar (resp. counting bisimilar) models.

Lemma 3.2. (Folklore)
For any modal (resp. counting) fixed point formula α, formula $\varphi_\alpha(r)$ is bisimulation invariant (resp. counting bisimulation invariant).

The following theorem shows that the bisimulation invariance (resp. counting bisimulation invariance) not only holds but even characterizes modal (resp. counting) fixed point calculi as fragments of MSO. In fact:

Theorem 3.1. (from Walukiewicz [30])
An MSO sentence is invariant under counting bisimulation if and only if it is equivalent to some counting mu-calculus formula.

and

Theorem 3.2. (Janin-Walukiewicz [14])
An MSO sentence is invariant under bisimulation if and only if it is equivalent to some modal mu-calculus formula.

Finally, let us mention that alternation of least and greatest fixed point constructions induces a hierarchy in modal or counting mu-calculus. More precisely, following Niwiński's definition, we define the fixpoint alternation depth hierarchies as follows.

The first level $N_0 = M_0$ is defined as the set of all (modal or counting) fixpoint free formulas with negation only applied to propositional constants of $Prop$. Then, for each integer k, the level N_{k+1} (resp. level M_{k+1}) is defined as the closure of $N_k \cup M_k$ under disjunction, conjunction, substitution – provided no free variable becomes bound during the substitution process – and greatest fixpoint construction (resp. least fixpoint construction).

In the modal case, the mu-calculus hierarchy has been proven infinite by Bradfield [3]. A similar result by Arnold [1] shows it is infinite as well in the counting case.

Actually, Arnold's result is stated for the modal mu-calculus on the binary tree. But this implies the more general statement above, as both counting and modal mu-calculus are equivalent on the binary tree, and the binary tree itself is definable in the counting mu-calculus by a formula with greatest fixed point constructions only.

The standard theory of mu-calculus [2] gives a first relationship between the mu-calculus and the monadic hierarchy. Namely:

Lemma 3.3. For every k, every counting modal mu calculus formula of class N_k is equivalent to a $MSOL$ formula of class Σ_k.

The other way around, various relationships are considered by the authors in [11].

In this paper we are mainly interested in the second level, N_2, of the Niwiński hierarchy. Observe that all formulas where all ν's precede all μ's belong to N_2 (actually N_2 is a bit larger than that, because of closure under substitution); for this reason, the level N_2 is sometimes called *$\nu\mu$-level*.

4. Tree Automata

In this section, we review the definitions and properties of various notion of tree automata. More precisely, we review the notion of alternating or non deterministic automata with Büchi condition possibly with the extra requirement of weakness as used by Rabin in his study of the weak monadic theory of the binary tree [26].

The automata are designed to read arbitrary trees as input, i.e. trees defined with a successor relation as opposed to the binary tree that is defined by means of two successor functions. It follows that the notion of transition we use is more involved than in the classical case. As a consequence, we need to proceed more cautiously when (re)defining non deterministic automata. A similar observation is made when studying tree-like structures [30].

4.1. Büchi alternating automata

Definition 4.1. (Alternating automata)
A *Büchi alternating automaton* is a tuple $\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$, where Q is a finite set of states, Σ is a finite set called the alphabet (in general $\Sigma = \mathcal{P}(Prop)$), $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \to FOL^+(Q)$ is the transition function, where $FOL^+(Q)$ is the set of all first order formulas with equality over one unary predicate P_q for any $q \in Q$, where every P_q occurs only positively. As a particular case, an automaton is called a *modal automaton* when, for any $q \in Q$, any $\sigma \in \Sigma$, $\delta(q, \sigma)$ does not contain the equality (nor disequality) predicate.

We say that an alternating automaton $\mathcal{A}$ is a *closed* automaton (resp. an *open* automaton) when $F = Q$ (resp. $F = \emptyset$).

The semantics of the alternating automaton $\mathcal{A}$ can be given by a game.

Definition 4.2. (Model-checking game)
For any tree $T = \langle V, r, E, \lambda \rangle$, we defined a game $G(\mathcal{A}, T)$ played by two players, whom we call Marker and Selector.

Positions in the game are pairs of $Q \times V$. The initial position is (q_0, r).

From a position (q, v), where q is a state of $\mathcal{A}$ and v is a vertex of T:

1. the Marker chooses a "marking relation" $m \subseteq Q \times Succ(v)$ that satisfies the local constraints expressed via the transition function, i.e. when m is seen as the (flat) first order structure $m = \langle Succ(v), \{q^m\}_{q \in Q} \rangle$ over the vocabulary Q with, for each $q \in Q$, $q^m = \{v' \in Succ(v) : (q, v') \in m\}$, one must have $m \models \delta(q, \lambda(v))$,

2. and the Selector chooses a pair $(q', w) \in m$ and this pair becomes the new position.

If either player cannot move, the other wins. Otherwise, we have an infinite play; if along the play some state of F occurs infinitely often, then Marker wins; otherwise Selector wins.

The automaton $\mathcal{A}$ *accepts* the tree T if the Marker has a winning strategy in $G(\mathcal{A}, T)$. The language *defined* by $\mathcal{A}$ is the set of all trees accepted by $\mathcal{A}$.

Lemma 4.1. Every alternating Büchi automaton is equivalent, over arbitrary trees, to a counting mu-calculus formula of class N_2. If, moreover, the automaton is a closed (resp. open) automaton, then it is equivalent to a formula of class N_1 (resp. M_1).

Proof:
Let $\mathcal{A} = \langle Q, \Sigma, q_0, F, \delta \rangle$ be an alternating Büchi automaton. Following the standard techniques of the mu-calculus theory [2], we essentially have to translate the transition specification δ of the automaton into counting formulas in order to build an equivalent counting mu-calculus formula and this can be done by standard argument in logic.

More precisely, for every $q \in Q$ and every $a \in \Sigma$, the formula $\delta(q, a)$ is a positive first order formula (with equality and disequality) over unary predicates. By known results about first order logic [5], every positive first order formula over unary predicates can be rewritten as a positive boolean combination of $|b| \geq k$, and $|\neg b| < k$, where b is a positive boolean combination of predicates, $|b|$ is the cardinal of the elements satisfying b, and k is a non negative integer.

Now, the fact that the successors of a vertex x in a tree verify $|b| \geq k$ amounts to say that x verifies $\langle\rangle_k\ b$, and that its successors verify $|\neg b| < k$ amounts to say that x verifies $\Box_k b$. Hence, the fact that the successors of a vertex verify $\delta(q, a)$ is equivalent to a counting modal formula $\alpha(q, a)$, positive in the predicates $P_{q'}$ for all $q' \in Q$.

Let a range over the alphabet of the automaton. Let us view the states of the automaton as fixpoint variables. Following the general theory of fixpoint calculi [2] to the mu-calculus given by all the counting modalities $\langle\rangle_k$ and $\Box_k$, the automaton is equivalent to the component q_0 of a system of "least and greatest fixpoint equations" consisting of two consecutive, finite sequences of equations: a first sequence of greatest fixpoint equations of the form $q =_\nu \bigvee_a a \wedge \alpha(q, a)$ $(q \in F)$ and a second sequence of least fixpoint equation of the form $q =_\mu \bigvee_a a \wedge \alpha(q, a)$ $(q \notin F)$. Then, it is known that such a system of equations can be transformed (by "sequencing" equations as defined in [2] definition 1.4.9, page 32) into a single counting mu-calculus formula of class N_2.

As a particular case, if the automaton $\mathcal{A}$ is a closed (resp. open) automaton then the mu-calculus formula obtained in such a way belongs to class N_1 (resp. M_1). □

4.2. Büchi non deterministic automata

Intuitively, we can think of a position (q, v) of the model-checking game above as a copy of the automaton which is running on vertex v in state q. In general, it may occurs that several distinct copies of the automaton are running in different states on the same vertex v. This possibility is a major source of difficulties and complexity. The notion of functional runs and the related notion of non deterministic automata (or, more accurately, non alternating automata) is a way to handle these difficulties.

Definition 4.3. (Functionnal acceptance)
With the same notation as above, a *(functional) run* of $\mathcal{A}$ on T is a (partial) function $\rho : V^T \to Q$, from the vertices of T to Q, such that $\rho(r^T) = q_0$ and, for each vertex v in the domain of ρ, the marking induced by ρ on the successors of v verifies the transition specification $\delta(\rho(v), \lambda^T(v))$.

A run ρ is called *accepting* if along all infinite paths $v_0.v_1.\cdots.v_i.\cdots$ of vertices in the domain of ρ there is some state in F that occurs infinitely often in the sequence of states $\rho(v_0).\rho(v_1).\cdots.\rho(v_i).\cdots$.

A tree T is *functionally accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on T. The language of trees *functionally defined* by $\mathcal{A}$ is the set of all trees functionally accepted by $\mathcal{A}$.

Observe that every tree functionally accepted is accepted in the usual sense. In fact, every accepting functional run gives a winning strategy for marker. But the converse is false in general. For instance, consider an automaton on one letter a and three states, q_0 (initial), q_1 and q_2 (accepting), and the rules $\delta(q_1, a) = \delta(q_2, a) = true$ and $\delta(q_0, a) = \exists x, y.q_1(x) \wedge q_2(y)$. In fact, the tree consisting of the root and one successor is accepted by this automaton, but not functionally accepted. Indeed, the transition specification $\delta(q_0, a)$ does require that two copies of the automaton in state q_1 and q_2 are sent on the same successor of the root. For non deterministic automata, as defined here, the equivalence does hold.

Definition 4.4. (Non deterministic automata)
Let us define *basic disjunctive formula* as any FO-formula of the form

$$\exists x_1, \cdots, x_n \mathit{diff}(x_1, \cdots, x_n) \wedge \bigwedge_{i\in[1,n]} p_i(x_i) \wedge \forall y \mathit{diff}(x_1, \cdots, x_n, y) \rightarrow \bigvee_{j\in[1,m]} q_j(y)$$

with $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ two lists (possibly with repetitions) of states interpreted as unary predicates. Let us also define *disjunctive formulas* as any finite disjunction of basic disjunctive formulas.

Following [13], we say that an alternating automaton is *non deterministic* (or, more accurately, *non alternating*) if its transition formulas are disjunctive formulas.

We observe first that automata acceptance simplifies with non deterministic automata.

Lemma 4.2. For non deterministic automata, functional acceptance and ordinary acceptance coincide.

Proof:
Let D be a disjunctive formula and let FUN be the first order formula saying that every element satisfies at most one predicate q for exactly one $q \in Q$. If we view models as markings, that is, sets of pairs, one can observe that every model of D includes a model of $D \wedge FUN$.

Hence, applying this facts along plays, if Marker has a winning strategy in the model checking game $G(\mathcal{A}, T)$ of a nondeterministic automaton $\mathcal{A}$ on a tree T, then it has one where in each vertex, exactly one copy of the automaton is sent in each successor. And the latter strategy gives an accepting functional run. □

Non deterministic automata can even be seen as some normal form for automata "restricted to" functional acceptance. And, conversely, one can use the more flexible syntax of alternating automata with functional acceptance instead of that of non deterministic automata. More precisely:

Lemma 4.3. For every alternating automaton $\mathcal{A}$ there is a nondeterministic automaton $\mathcal{A}'$, such that the tree language functionally defined by $\mathcal{A}$ equals the language defined by $\mathcal{A}'$.

Proof:
Let Q be the state set of $\mathcal{A}$. From the analysis of first order logic over unary predicates given in [5] it

follows that, for every first order formula $\varphi \in FOL(Q^+)$, there is a disjunctive formula D such that $D \wedge FUN \Leftrightarrow \varphi \wedge FUN$. Let $\mathcal{A}'$ be the automaton obtained from $\mathcal{A}$ by replacing φ with D. Then $\mathcal{A}'$ is non deterministic, and the languages functionally defined by $\mathcal{A}$ and $\mathcal{A}'$ coincide. But by the previous lemma, the language functionally defined by $\mathcal{A}'$ coincides with the language defined by $\mathcal{A}'$ in the usual sense. □

The next property of non deterministic automaton is an essential feature in many proofs in automata theory.

Lemma 4.4. The tree languages definable by non deterministic automata are closed under projection.

Proof:
It is enough to prove the same closure result for automata with functional acceptance. Let $\mathcal{A} = \langle Q, \Sigma_1 \times \Sigma_2, q_0, F, \delta \rangle$ be an automaton on an alphabet which is a Cartesian product of two sets Σ_1 and Σ_2. Let $\pi_1 : \Sigma_1 \times \Sigma_2 \to \Sigma_1$ be the first projection associated to the Cartesian product. If L is the language functionally defined by $\mathcal{A}$, an automaton which defines functionally $\pi_1(L)$ is $\mathcal{B} = \langle Q, \Sigma_1, q_0, F, \delta' \rangle$ where $\delta'(q, \sigma) = \bigvee_{\tau \in \Sigma_2} \delta(q, (\sigma, \tau))$. In fact, a function ρ is a run of $\mathcal{A}$ on a tree T if and only if it is a run of $\mathcal{B}$ on the tree $\pi_1(T)$. So, $\mathcal{A}$ accepts functionally T if and only if $\mathcal{B}$ accepts functionally $\pi_1(T)$. □

4.3. Weak alternating automata

Weak automata are obtained from Büchi automata by restricting the structure of automata. They play a fundamental role in the analysis of the expressive power of FO and monadic Σ_1 on trees. They have been used as a characterization of weak MSO (or the alternation free mu-calculus) on trees [20, 19, 16].

Definition 4.5. A Büchi alternating automaton $\mathcal{A} = \langle Q, \Sigma_1 \times \Sigma_2, q_0, F, \delta \rangle$ is called *weak automaton* if there is a partially ordered set I and a partition of Q, $Q = \bigcup_{i \in I} Q_i$, such that F is a union of some of the Q_i's, and for every $q \in Q_i$, $\delta(q, \sigma) \in FOL^+(\bigcup_{j \leq i} Q_j)$: this means that along plays, the index decreases, or remains the same.

The sets Q_i included in F are called the accepting components of the automaton. The other Q_i's are called rejecting components.

Lemma 4.5. The tree languages definable by weak alternating automata are closed under complementation.

Proof:
Let $\mathcal{A} = \langle Q = \bigcup_{i \in I} Q_i, \Sigma, q_0, F, \delta \rangle$ be a weak alternating automaton. We define the automaton $\mathcal{B}$ to be the automaton

$$\mathcal{B} = \langle Q = \bigcup_{i \in I} Q_i, \Sigma, q_0, Q \setminus F, \delta^d \rangle$$

where $\delta^d(q, a)$ is the dual formula of $\delta(q, a)$: $\delta^d(q, a) = \neg\delta(q, a)[\neg q / q : q \in Q]$.

Observe that automaton $\mathcal{B}$ is a weak automaton. In general, the winning condition of the dual of a Büchi automaton shall be a co-Büchi condition, i.e. the dual condition of the Büchi condition. However, the weakness assumption makes Büchi and co-Büchi condition equivalent.

More precisely, in the model-checking game, a play is winning for the Marker with the Büchi condition when infinitely many accepting states occur. By duality, a play is winning for the Marker with the co-Büchi condition when finitely many non accepting states occur. In general, this makes that Büchi automaton are not closed under complement. However, under the assumption of weakness, there cannot be an infinite alternation of accepting and rejecting states in the model-checking games, so Büchi or co-Büchi criteria *are* equivalent.

It is then a classical feature of alternating automata, following for instance [21], that the automaton $\mathcal{B}$ defined above do recognize the complement of $L(\mathcal{A})$.

More precisely, one can check that for any tree T, any winning strategy for Marker in the game $G(\mathcal{A}, T)$ induces a winning strategy for Selector in the game $G(\mathcal{B}, T)$ and vice versa, any winning strategy for Selector in the game $G(\mathcal{A}, T)$ induces a winning strategy for Marker in the game $G(\mathcal{B}, T)$. Since model-checking games are determined, this conclude the proof. □

Lemma 4.6. The tree languages definable by weak non deterministic automata are closed under union and intersection.

Proof:
By Lemma 4.2 it is enough to show the same for weak automata with functional acceptance.

Now the idea is that to define the union (resp. the intersection) of two automata by means of a product. More precisely, given two weak automata $\mathcal{A} = \langle Q = \bigcup_{i\in I} Q_i, \Sigma, q_0, F, \delta\rangle$ and $\mathcal{B} = \langle Q' = \bigcup_{j\in J} Q'_j, \Sigma, q'_0, F', \delta'\rangle$. we define the automaton $\mathcal{C} = \langle Q\times Q', \Sigma, (q_0, q'_0), F'', \delta''\rangle$ where $F'' = F\times Q' \cup Q\times F'$ for the union (resp. $F'' = F\times F'$ for the intersection), and, given π_1 and π_2 the projections of the Cartesian product $Q\times Q'$, the transition function defined in such a way that, for each $(q, q') \in Q\times Q'$, each $a \in \Sigma$, a marking m satisfies $\delta''((q, q'), \sigma)$ if the projected marking $\pi_1(m)$ satisfies $\delta(q, \sigma)$ and the projected marking $\pi_2(m)$ satisfies $\delta'(q', \sigma)$.

The automaton $\mathcal{C}$ does functionally recognize the union (resp. the intersection) of the languages functionally recognized by $\mathcal{B}$ and $\mathcal{B}$. Moreover, it is a weak automaton with $Q'' = \bigcup_{(i,j)\in I\times J} Q_i \times Q'_j$ with $I \times J$ ordered with the product order. □

Lemma 4.7. The tree languages definable by weak non deterministic automata are closed under projection.

Proof:
The construction given in the proof of Lemma 4.4 for non deterministic automata just applies similarly to weak non deterministic automata. □

Finally, adapting Muller and Schupp's simulation theorem [21]:

Theorem 4.1. (Simulation)
Any alternating weak (resp. closed or open) automaton $\mathcal{A}$ is equivalent to a non deterministic Büchi (resp. non deterministic closed or open) automaton $\mathcal{A}_n$.

Proof:
Over binary trees, this result follows from Muller and Schupp's construction [21]. For the modal mu-calculus, it follows from the construction given in [13]. It relies on standard techniques presented in a

quite general setting in [10, 2]. It has also been proved again in the modal case with slightly different techniques in [16].

Here, we essentially explicit the construction. The proof of its correctness is left as a (standard) exercise.

Let $\mathcal{A} = \langle Q = \bigcup_{i \in I} Q_i, \Sigma, q_0, F, \delta \rangle$ be a weak alternating automaton. We define the Büchi automaton $\mathcal{A}' = \langle Q', \Sigma, q'_0, F', \delta' \rangle$ as follows:

1. $Q' = \mathcal{P}(Q \times \{0,1\} - F \times \{0\})$,
2. $q'_0 = \{(q_0, 1)\}$,
3. $F' = \mathcal{P}(Q \times \{1\})$,
4. for each $P \in Q'$, each $a \in \Sigma$, a marking $m' \subseteq Q' \times V$ on the set of predicate Q' satisfies $\delta'(P, a)$ when, for each $(q, x) \in P$, there is a marking $m_1 \subseteq Q \times V$ such that:
   (a) $m_1 \models \delta(q, a)$ (local constraint satisfied),
   (b) for each $(q_1, v_1) \in m_1$, one has $((q_1, y), v_1) \in m'$ such that:
      i. $y = 1$ when $q_1 \in F$
      ii. $y = 0$ or $y = x$ when $q_1 \notin F$, depending on wether $P \in F'$ (global reset of the winning condition) or $P \notin F'$ (local updating of the winning condition),

We claim that a tree is accepted by the alternating automaton $\mathcal{A}$ if and only if it is functionally accepted by the automaton $\mathcal{A}'$. Then, applying Lemma 4.3, on can build out of $\mathcal{A}'$ a non deterministic automaton that recognizes the language of trees functionally accepted by $\mathcal{A}'$.

Observe that this construction holds starting from an arbitrary alternating Büchi automaton $\mathcal{A}$. In general, even if automaton $\mathcal{A}$ is a weak automaton, automaton $\mathcal{A}'$ is not a weak automaton. However, starting from a closed or open (weak) automaton (with $F = Q$ or $F = \emptyset$ respectively) the construction simplifies and the resulted automaton can be defined as a closed or open (weak) automaton (with $F' = Q'$ or $F' = \emptyset$ respectively).

More precisely, when $F = \emptyset$, observe that all reachable state from q'_0 (but the initial state q'_0 that only appear initially) belong to $\mathcal{P}(Q \times \{0\})$ so we can restrict Q' to be the set $\mathcal{P}(Q \times \{0\})$ with $q'_0 = (q_0, 0)$ and $F' = \emptyset$ without changing the language of trees functionally recognized by $\mathcal{A}'$. Quite similarly, when $F = Q$, all reachable states from q'_0 belongs to $\mathcal{P}(Q \times \{1\})$. This means we can restrict Q' to be the set $\mathcal{P}(Q \times \{1\})$ with $F' = Q'$ without changing the language of trees functionally recognized by $\mathcal{A}'$.

□

5. The main results

In this section, we prove Theorem 1.1 by a series of lemmas that easily follows from the results presented in the previous sections.

Lemma 5.1. (From FO to non deterministic weak)
On arbitrary trees, every first order formula is equivalent to a non deterministic weak automaton.

Proof:
In order to do so, applying Gaifman theorem, we first show that FO-formulas define languages of trees that are finite boolean closure of closed languages in the sense of the prefix topology. Then, in turn, classical results of automata theory ensure that these languages are definable by means of non deterministic weak automata.

Let d be a positive integer. A FO-formula $\varphi(x)$ with a single free variable x is called *basic d-local* when all quantifications in $\varphi(x)$ are relativized to vertices at distance at most d from x, i.e. vertices reachable from x by a undirected path of length at most d.

Theorem 5.1. (Gaifman [8])
Let φ be a FO-formula on trees. There exist $d \geq 0$ such that φ is equivalent to a finite boolean combination of formulas of the form

$$\exists x_1 \cdots x_n \bigwedge_{i \neq j} d(x_i, x_j) > d \wedge \varphi_i(x_i)$$

where $\varphi_1(x), \ldots, \varphi_n(x)$ are basic d-local formulas and $d(x_i, x_j) > d$ means that there is no undirected path between x_i and x_j of length smaller than or equal to d.

Then we have:

Corollary 5.1. Every FO-definable language of tree is a finite boolean combination of closed languages.

Proof:
The negation of a formula φ of the form

$$\varphi \equiv \exists x_1 \cdots x_n \bigwedge_{i \neq j} d(x_i, x_j) > d \wedge \varphi_i(x_i)$$

defines a closed language. In fact, assume there is a sequence of trees $\{T_n\}_{n \in \omega}$ that converges towards a limit T. If $T_n \models \neg\varphi$ for all $n \in \omega$ then $T \models \neg\varphi$. Otherwise, if $T \models \varphi$ there is a finite depth h such that the satisfiability of φ is witnessed by vertices that belong to the h-prefix of T. Since the sequence $\{T_n\}_{n \in \omega}$ converges towards T, there is also an N such that for all $n \geq N$, T_n and T have isomorphic h-prefix hence $T_n \models \varphi$ which contradicts the hypothesis. □

Now, we have:

Lemma 5.2. FO-definable closed languages are definable by means of finite closed non deterministic automata. And, similarly, FO-definable open languages are definable by finite open non deterministic automata.

Proof:
The case of closed languages is proved in [12]. By complementation (Lemma 4.5), this also shows that FO-definable open languages are recognizable by means of open alternating automata. But then, the Simulation Theorem 4.1 shows that these languages are then recognizable by means of open non deterministic automata. □

Since closed and open non deterministic automata are special case of non deterministic weak automata this, by applying also Lemma 4.6, concludes the proof of Lemma 5.1. □

Remark 5.1. We could have given an explicit construction of such an automaton. In fact, from Gaifman normal formal form, it is quite an easy exercise. Remind however that the satisfiability problem for a FO-formula on tree is non elementary. Hence, the automaton translation of a FO-formula is also non elementary.

Then we prove:

Lemma 5.3. (From Π_1 to weak)
On arbitrary trees, every monadic Π_1 formula is equivalent to a weak automaton.

Proof:
Since weak non deterministic automata are closed under projection (Lemma 4.7) we know, by applying Lemma 5.1 that the languages definable by monadic Σ_1 formula are recognizable by means of alternating weak automaton. Then, applying Lemma 4.5 concludes the proof. □

Now we have:

Lemma 5.4. (From Σ_2 to Büchi)
On arbitrary trees, every monadic Σ_2 formula is equivalent to a Büchi automaton.

Proof:
By applying Lemma 5.3, every Π_1 formula is equivalent to a weak alternating automaton. So, by applying Simulation Theorem 4.1, it is also equivalent to a non deterministic Büchi automaton. Now, closure under projection (Lemma 4.4) concludes the proof. □

We prove then the analogous of Theorem 1.1 for counting bisimulation.

Theorem 5.2. The counting bisimulation invariant fragment of monadic Σ_2 equals the $\nu\mu$-level of the counting mu-calculus.

Proof:
Since any $\nu\mu$-formula of the counting mu-calculus is equivalent to a (counting bisimulation invariant) monadic Σ_2 formula, it is sufficient to prove the converse.

Let φ be a monadic Σ_2 formula counting bisimulation invariant.

First, by Lemma 5.4, we know that, over trees, formula φ is equivalent to a Büchi automaton. By applying Lemma 4.1 it is thus equivalent, over trees, to a $\nu\mu$-formula α of the counting mu-calculus.

Since both φ and α are counting bisimulation invariant, by Remark 3.1, we conclude that they are equivalent on arbitrary models. □

For bisimulation invariance, we have:

Lemma 5.5. (Saturation)
For arbitrary infinite cardinal κ, on κ-expansions of trees, every monadic Σ_2 formula is equivalent to a modal Büchi automaton.

Proof:
Let φ be a monadic Σ_2 formula. By Lemma 5.4, over κ-expansions, formula φ is equivalent to a Büchi automaton. But, following [14], on κ-expansions, Büchi automaton are equivalent to modal Büchi automata. □

So we conclude then the proof of Theorem 1.1 that is restated below for the sake of clarity.

Theorem 5.3. The bisimulation invariant fragment of monadic Σ_2 equals the $\nu\mu$-level of the modal mu-calculus.

Proof:
Since any modal $\nu\mu$-formula is equivalent to a (bisimulation invariant) monadic Σ_2 formula, it is sufficient to prove the converse.

Let φ be a monadic Σ_2 formula bisimulation invariant.

First, by Lemma 5.5, we know that, over κ-expansions of trees, formula φ is equivalent to a modal Büchi automaton. By applying Theorem 4.1 it is thus equivalent, over trees, to a $\nu\mu$-formula α of the modal mu-calculus. Then, by Remark 3.1 they are equivalent on arbitrary models. □

References

[1] Arnold, A.: The mu-calculus alternation-depth hierarchy over the binary tree is strict, *Theoretical Informatics and Applications*, **33**, 1999, 329–339.

[2] Arnold, A., Niwiński, D.: *Rudiments of mu-calculus*, Studies in Logic and the Foundations of Mathematics vol. 146, North–Holland, 2001.

[3] Bradfield, J.: The modal mu-calculus alternation hierarchy is strict, *Theoretical Comp. Science*, **195**, 1998, 133–153.

[4] Büchi, J. R.: On a decision method in restricted second order arithmetic, *Proceedings of the international congress on Logic, Methodology and Philosophy of Science 1960*, Stanford University Press, 1962.

[5] Courcelle, B.: The Monadic Second Order Logic Of Graph I: Recognizable sets of finite graphs, *Inf. and Comp.*, **85**, 1990, 12–75.

[6] Ebbinghaus, H., Flum, J., Thomas, W.: *Mathematical Logic*, Undergraduate Texts in Mathematics, Springer-verlag, 1984.

[7] Emerson, E.: Temporal and modal logic, in: *Handbook of Theor. Comp. Science (vol. B)* (J. Van Leeuwen, Ed.), Elsevier, 1990, 995–1072.

[8] Gaifman, H.: Modeling Concurrency By Partial Orders and Nonlinear Transition Systems, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354, 1988.

[9] Grädel, E., Thomas, W., Wilke, T., Eds.: *Automata, Logics and Infinite Games*, LNCS Tutorial 2500, Springer, 2002.

[10] Janin, D.: Automata, tableaus and a reduction theorem for fixpoint calculi in arbitrary complete lattices, *IEEE Symp. on Logic in Computer Science (LICS)*, 1997.

[11] Janin, D., Lenzi, G.: Relating Levels of the Mu-calculus Hierarchy and Levels of the Monadic Hierarchy, *IEEE Symp. on Logic in Computer Science (LICS)*, IEEE Computer Society, 2001.

[12] Janin, D., Lenzi, G.: On the logical definability of topologically closed recognizable languages of infinite trees, *Computing and Informatics*, **21**, 2002, 185–203.

[13] Janin, D., Walukiewicz, I.: Automata for the modal mu-calculus and related results, *Mathematical Found. of Comp. Science (MFCS)*, LNCS 969, 1995.

[14] Janin, D., Walukiewicz, I.: On the expressive completeness of the modal mu-calculus with respect to monadic second order logic, *Conf. on Concurrency Theory (CONCUR'96)*, LNCS 1119, 1996.

[15] Kozen, D.: Results on The Propositional μ-calculus, *Theoretical Comp. Science*, **27**, 1983, 333–354.

[16] Kupferman, O., Vardi, M.: $\Pi_2 \cap \Sigma_2 = AFMC$, *Int. Call. on Aut., Lang. and Programming (ICALP)*, LNCS 2719, 2003.

[17] Lenzi, G.: A new logical characterization of Büchi automata, *Symp. on Theor. Aspects of Computer Science (STACS)*, LNCS 2010, 2001.

[18] Milner, R.: *Communication and concurrency*, Prentice-Hall, 1989.

[19] Mostowski, A.: Hierarchies of weak automata on weak monadic formulas, *Theoretical Comp. Science*, **83**, 1991, 323–335.

[20] Muller, D. E., Saoudi, A., Schupp, P. E.: Alternating automata, the weak monadic theory of trees and its complexity, *Theoretical Comp. Science*, **97**, 1992, 233–244.

[21] Muller, D. E., Schupp, P. E.: Alternating automata on infinite trees, *Theoretical Comp. Science*, **54**, 1987, 267–276.

[22] Niwiński, D.: Fixed Points vs. Infinite Generation, *Proc. 3rd. IEEE LICS*, 1988.

[23] Park, D.: Concurrency and automata on infinite sequences, *5th GI Conf. on Theoret. Comput. Sci.*, LNCS 104, Karlsruhe, 1981.

[24] Pnueli, A.: The Temporal Logic of Programs, *18th Symposium on Foundations of Computer Science*, 1977.

[25] Rabin, M. O.: Decidability of second order theories and automata on infinite trees, *Trans. Amer. Math. Soc.*, **141**, 1969, 1–35.

[26] Rabin, M. O.: Weakly definable relations and special automata, *Mathematical Logic and Foundation of Set Theory*, North Holland, 1970.

[27] Shoenfield, J. R.: *Mathematical Logic*, Addison-Wesley, Reading, Mass., 1967.

[28] Skurczyński, J.: A characterization of Büchi tree automata, *Information Processing Letters*, **81**, 2002, 29–33.

[29] Thomas, W.: Automata on Infinite Objects, in: *Handbook of Theor. Comp. Science (vol. B)* (J. Van Leeuwen, Ed.), Elsevier, 1990, 133–191.

[30] Walukiewicz, I.: Monadic second order logic on tree-like structures, *Symp. on Theor. Aspects of Computer Science (STACS)*, 1996, LNCS 1046. Full version in *Information and Computation* 164 (2001) pp. 234-263.