

Characterizing EF over Infinite Trees and Modal Logic on Transitive Graphs

Balder ten Cate^{1*} & Alessandro Facchini^{2**}

¹ University of California, Santa Cruz

² Warsaw University

Abstract. We provide several effective equivalent characterizations of EF (the modal logic of the descendant relation) on arbitrary trees. More specifically, we prove that, for EF-bisimulation invariant properties of trees, being definable by an EF formula, being a Borel set, and being definable in weak monadic second order logic, all coincide. The proof builds upon a known algebraic characterization of EF for the case of *finitely branching* trees due to Bojańczyk and Idziaszek. We furthermore obtain characterizations of modal logic on transitive Kripke structures as a fragment of weak monadic second order logic and of the μ -calculus.

1 Introduction

Determining effective characterizations of logics on trees is an important problem in theoretical computer science, motivated by the desire to better understand the expressive power of logics such as first-order logic (FO) or the temporal logic CTL* on trees.

While for finite words this kind of problem is well-studied, the list of results for trees is much more frugal. The situation has improved a little in the last years, thanks to the successful use of *forest algebras* (see [9]). Notably, this formalism has been used for obtaining decidable characterizations for the classes of tree languages definable in EF + EX [8], EF + F⁻¹ [4,15], BC- $\Sigma_1(<)$ [6,15], $\Delta_2(\leq)$ [7,15], in FO with the child relation and in FO with counting quantifiers (modulo an integer) and the child relation [1]. This approach has then been extended in the case of the temporal logic EF on infinite but finitely branching trees by Bojańczyk and Idziaszek [5].

These results all demonstrate the importance of the algebraic approach to obtaining decidable characterizations of logics. In the case of infinite trees, it is natural to ask whether such logics also admit *topological* characterizations. Take for instance the logic EF, the modal logic of the descendant relation, which is a fragment of the modal μ -calculus. The formula $\mu x. \Box x$ of the modal μ -calculus, which defines the class of well-founded trees, is not equivalent to any EF-formula. This follows from results of Bojańczyk and Idziaszek [5] since the syntactic ω -forest algebra of the class of finitely branching trees satisfying $\mu x. \Box x$ fails to satisfy one of the algebraic equations that characterize the expressive power of EF on finitely branching trees. However, there is another explanation for the fact that $\mu x. \Box x$ is not equivalent to an EF-formula, which involves topology: the class of (arbitrarily branching) well-founded trees is not Borel,

* The first author was supported by the NSF grant IIS-0905276

** The second author is supported by a grant from the SNFS, n. PBLAP2-132006.

while EF-formulae can only define tree languages that are Borel. This raises the question whether EF, as a fragment of MSO, can be characterized by topological means.

In this work we give a positive answer to this question. We prove that a (not necessarily finitely branching) tree language is EF-definable if and only if it is invariant for EF-bisimilarity and Borel. Since EF is a fragment of *weak monadic-second order logic* (WMSO) and all WMSO-definable tree languages are Borel, we obtain as a corollary that EF is the EF-bisimulation invariant fragment of WMSO. All the proofs make crucial use of the results in [5]. Secondly, these characterizations are effective: given an MSO formula, one can effectively test whether it defines an EF-definable tree language. Thirdly, these characterizations show that for EF-bisimilar invariant languages WMSO is the Borel fragment of MSO. Since for all tree languages WMSO and MSO are incomparable (cf. Remark 1 in Subsection 2.2), this cannot be true in general. It is however an open question whether such a relative characterization of WMSO holds when restricted to *complete* binary trees (cf. [13]).

Theorem 1. *Let L be any MSO-definable tree language. The following are equivalent and decidable:*

- (1) L is EF-definable,
- (2) L is closed under EF-bisimulation and WMSO-definable,
- (3) L is closed under EF-bisimulation and Borel,
- (4) L is closed under EF-bisimulation, and for every L -idempotent context c , and for every forest f , $c(f)$ and $(c + cf)^\infty$ are L -equivalent.

Note that here, we consider the monadic second-order language with the child relation and without a horizontal sibling-order relation.

The fourth condition, which is essentially the condition that was used in [5] to characterize EF on finitely branching trees, involves some algebraic notions that we will introduce in Section 2. Thm. 1 does not hold for *finitely* branching trees. This is because on finitely branching trees, well-foundedness is equivalent to finiteness, which is Borel and closed under EF-bisimulation but not EF-definable.

From Thm. 1, we finally obtain the following effective characterizations of modal logic on transitive Kripke models (with a suitable definition of what it means for a class of pointed Kripke models to be Borel):

Theorem 2. *For every μ -formula ϕ , the following are equivalent and decidable:*

- (1) ϕ is equivalent on transitive Kripke models to a modal formula,
- (2) ϕ is equivalent on transitive Kripke models to a WMSO-formula,
- (3) the class of pointed transitive Kripke models satisfying ϕ is Borel.

Theorem 3. *For every WMSO-formula $\phi(x)$, the following are equivalent:*

- (1) $\phi(x)$ is equivalent on transitive Kripke models to a modal formula,
- (2) $\phi(x)$ is bisimulation invariant on transitive Kripke models.

Since EF is a fragment of first-order logic (with the descendant relation), Thm. 1, 2 and 3 imply that EF is the EF-bisimulation invariant fragment of FO on trees and that modal logic is the bisimulation invariant fragment of FO on transitive Kripke models. Both were known (cf. [11, Thm. 4.12]).

2 Preliminaries and groundwork

2.1 The beauty of trees

Trees, forests, contexts. A colored directed graph g over a finite alphabet Σ , called a Σ -colored directed graph, is given by a triple $(V_g, \prec_g, \lambda_g)$ such that the domain V_g is a (possibly empty) set of nodes, $\prec_g \subseteq V_g \times V_g$ is a directed edge (or arrow) relation and $\lambda_g : V_g \rightarrow \Sigma$ is a coloring function. A node $x \in V_g$ is called a root if there is no node $y \in V_g$ such that $y \prec_g x$. A path from a node x to a node y is a sequence (x_1, \dots, x_n) of nodes such that $x_1 = x$, $x_n = y$ and for every $0 < i < n$, $x_i \prec_g x_{i+1}$. We say that a node x is reachable from a node y if there is a path from y to x . A pointed Σ -colored directed graph is a pair (g, s) where g is a Σ -colored directed graph and $s \in V_g$.

Unordered trees (*trees*, for short) will be colored directed graphs with a unique root (if the graph is not empty) and where every node is reachable along a unique path from the root. Given two nodes $x, y \in V_g$, we say that y is a son of x and x is a parent of y if $x \prec_g y$. In general, if there is a path of length at least 2 from x to y , then x is called an *ancestor* of y and y is called a *descendant* of x . A node that is not an ancestor of any node of a tree is called a *leaf*. A tree t is said to be *finitely branching* if for every node x , the set of sons of x is finite. The depth of a node x in a tree t is the length of the path from the root of t to x . Given a tree $t = (V_t, \prec_t, \lambda_t)$ and node $x \in V_t$, the subtree rooted in x is the tree $t.x = (V_{t.x}, \prec_t|_{V_{t.x}}, \lambda_t|_{V_{t.x}})$ where $V_{t.x}$ is the set of all reachable nodes of V_t from x , $\prec_t|_{V_{t.x}} = \prec_t \cap (V_{t.x} \times V_{t.x})$ and $\lambda_t|_{V_{t.x}} = \lambda_t \cap (V_{t.x} \times \Sigma)$. A tree is called *regular* if, up to isomorphism, it has only finitely many subtrees. The set of all trees over a finite alphabet Σ is denoted by T_Σ . A subset $L \subseteq T_\Sigma$ is called a Σ -*tree language*, or simply a tree language.

A forest over a finite set Σ is a colored directed graph over a finite alphabet Σ with possibly more than one root (and at least one if the domain is not empty) but such that any node is reached by a unique path from a single root. Given a forest f , and a node x of its domain, by $f.x$ we denote the subtree of f rooted in x . A context is a forest with a hole. Formally, a context over Σ is a forest over $\Sigma \cup \{\blacksquare\}$ where exactly one node is labeled by “ \blacksquare ”, and it is a leaf but not a root. As usual, we do not distinguish between two isomorphic trees or forests.

Operations on forests and contexts. We define two types of operations on forests and contexts: a (horizontal) *concatenation* operation, denoted by $+$, and a (vertical) *composition* operation, denoted by \cdot . Contrary to the concatenation operation, the composition operation is not commutative.

Given a sequence of forests $(f_i : i \in \alpha)$, with $\alpha \in \omega \cup \{\omega\}$, we want to concatenate these forests. Note that each forest can contain infinitely many rooted subtrees, and the length of the sequence itself, i.e., α , can be infinite. The concatenation of the forests f_i is defined as the forest $\sum_{i \in \alpha} f_i$ obtained by taking the disjoint union of all forests f_i . Since we identify isomorphic forests, this operation is commutative and associative.

We allow also to concatenate a sequence of forests where one of them is a context. Clearly the result of such a concatenation is a context.

Concerning vertical composition, we can compose a context c with a forest, resp. a context, f , and obtain as a result a forest, resp. a context, $c(f)$ simply by replacing the

hole node of c by f . More formally, let c be the context $(V_c, \prec_c, \lambda_c)$ and f be the forest $(V_f, \prec_f, \lambda_f)$. Without loss of generality, we can assume that $V_c \cap V_f = \emptyset$. Let $h \in V_c$ be the unique node labelled by “■” and $h_0 \in V_c$ be its unique parent, and let ρ_f be the set of roots of f . Then $c(f)$ is the forest defined by $((V_c \setminus \{h\}) \cup V_f, \prec', \lambda')$ where

- $\prec' |_{V_c \setminus \{h\}} = \prec_c |_{V_c \setminus \{h\}}$ and $\prec' |_{V_f} = \prec_f$, and for all $x \in \rho_f$, $h_0 \prec' x$,
- $\lambda' |_{V_c \setminus \{h\}} = \lambda_c |_{V_c \setminus \{h\}}$ and $\lambda' |_{V_f} = \lambda_f$.

For every pair of contexts c, c' and every forest f , we have that $c(c'(f)) = (c(c'))(f)$.

Note that, restricted to finitely (unordered) branching contexts and forests, the operations of concatenation and composition correspond to the ones in [5].

Myhill-Nerode equivalence. Given a tree language L , we want to define a notion of equivalence with respect to L , analogous to the well known Myhill-Nerode equivalence relation for finite words. Intuitively, two forests, or two contexts, are L -equivalent if they “behave the same” with respect to L . The definition we use, which we will now present, is essentially the one in [5] (cf. Remark 2).

The crucial notion here is that of a *template*. There are two kinds of templates, *forest-templates* and *context-templates*. A forest-template for an alphabet Σ is a forest over the alphabet $\Sigma \cup \{\star\}$ in which one or more leafs (possibly infinitely many) are labeled \star , and no non-leaf node is labeled \star . Similarly, a context-template for Σ is a forest over the alphabet $\Sigma \cup \{\star\}$ in which one or more nodes (possibly infinitely many) are labeled “ \star ”. Intuitively, the occurrences of “ \star ” in a forest-template are placeholders for forests, and the occurrences of “ \star ” in a context-template are placeholders for contexts.

In what follows, we will make use of the operation of replacing a subtree with another forest. We will not give a formal definition of this operation. Just note that it can be defined in a straightforward manner by using horizontal concatenation and a generalization of the substitution operation used in defining vertical composition.

Thus, given a forest-template f over $\Sigma \cup \{\star\}$ and a forest g over Σ , we denote by $f[\star \mapsto g]$ the forest obtained by replacing every node labeled “ \star ” in f with the forest g . Similarly, given a context-template f over $\Sigma \cup \{\star\}$ and a context c over Σ , we denote by $f[\star \mapsto c]$ the forest obtained by replacing every subtree starting at a node labeled “ \star ” in f with the forest obtained by composing the context c with the (possibly empty) forest given by all the subtrees rooted at a child of the considered node labeled “ \star ”. In the special case where f is the infinite unary tree labeled by “ \star ”, then for every context c , $f[\star \mapsto c]$ will be denoted also by c^∞ .

We call a forest-template *guarded* if no root is labelled by “ \star ”. Analogously for context-templates. Let $L \subseteq T_\Sigma$ be a tree language over an alphabet Σ . We say that two contexts, c_1, c_2 , over Σ are L -equivalent (denoted by $c_1 \equiv_L c_2$) if for every guarded context-template t over alphabet $\Sigma \cup \{\star\}$, $t[\star \mapsto c_1] \in L$ iff $t[\star \mapsto c_2] \in L$. Similarly, two forests f_1, f_2 are L -equivalent (denoted by $f_1 \equiv_L f_2$) if for every guarded forest-template t over $\Sigma \cup \{\star\}$, $t[\star \mapsto f_1] \in L$ iff $t[\star \mapsto f_2] \in L$.

Note that since we are working with *tree* languages, we can safely ignore forests $t[\star \mapsto f]$ or $t[\star \mapsto c]$ that are not trees. Moreover, note that two trees (seen as forests) can be L -equivalent while they do not agree on membership in L .

Proposition 1. *L -equivalence is an equivalence relation (both on contexts and on forests).*

There is a natural generalization of forest-templates and context-templates, where the holes are marked. More precisely, multi-forest-templates and multi-context-templates for an alphabet Σ are forests over an extended alphabet $\Sigma \cup \{\star_1, \dots, \star_n\}$, satisfying the same conditions as forest-templates and context-templates. Given a multi-forest-template t over $\Sigma \cup \{\star_1, \dots, \star_n\}$ and forests f_1, \dots, f_n , the forest obtained by replacing every node labeled “ \star_i ” in t with the forest f_i , with $i = 1, \dots, n$ is denoted by $t[\star_1 \mapsto f_1, \dots, \star_n \mapsto f_n]$. Similarly for multi-context-templates.

The next lemma for L -equivalence and multi-forest-templates will be very useful:

Lemma 1. *Let L be any Σ -tree language and f any multi-forest-template over the extended alphabet $\Sigma \cup \{\star_1, \dots, \star_n\}$. Consider two finite sequences of Σ -forests (g_1, \dots, g_n) and (h_1, \dots, h_n) such that g_i and h_i are L -equivalent, for each $i = 1, \dots, n$. Then $f[\star_1 \mapsto g_1, \dots, \star_n \mapsto g_n]$ and $f[\star_1 \mapsto h_1, \dots, \star_n \mapsto h_n]$ are also L -equivalent.*

The previous lemma can analogously be shown to hold for multi-context templates.

From now on, when speaking about forest, templates we always mean *guarded* forest, analogously with context.

2.2 Monadic Second Order Logics

A colored directed graph g over an alphabet Σ can be viewed as a relational structure

$$M_g := \langle V_g, \prec_g, (P_a^g : a \in \Sigma) \rangle$$

where P_a^g are unary predicates with the interpretation $P_a^g = \{v \in V_g : \lambda_g(v) = a\}$. Colored directed graphs, and in particular forests, as relational structures can be described in first- or second-order logics with the child relation (\prec) plus unary predicates. In this paper we are interested in monadic second-order logic (MSO) and in weak monadic second-order logic (WMSO).

Let $\text{Var}_1 = \{x, y, \dots\}$ be a countable infinite set of first-order variables, and $\text{Var}_2 = \{X, Y, \dots\}$ a countable infinite set of monadic second-order variables. Given a finite alphabet Σ , the set of MSO-formulae over Σ is defined by the following grammar:

$$\phi ::= x \prec y \mid x = y \mid P_a(x) \mid X(x) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x\phi \mid \exists X\phi$$

with $a \in \Sigma, x \in \text{Var}_1, X \in \text{Var}_2$. A formula with no free variables is called a *sentence*.

The semantics of MSO-formulae is given in terms of valuations. Let g be a colored directed graph. We can identify a valuation val over g with a pair of functions $(\text{val}_1, \text{val}_2)$ where $\text{val}_1 : \text{Var}_1 \rightarrow V_g$ and $\text{val}_2 : \text{Var}_2 \rightarrow \wp(V_g)$ (we use \wp to denote powerset). Thus, a valuation for a (nonempty) colored directed graph g assigns to every first-order variable an element from the domain and to each second order variable a set of elements from the domain. A *weak* valuation is a valuation that assigns to each monadic second-order variable a finite set, i.e., such that $\text{val}_2 : \text{Var}_2 \rightarrow \wp_{\text{fin}}(V_g)$.

The meaning of a formula ϕ in a nonempty colored directed graph g relative to a valuation val for g is then defined in a standard way, using arbitrary valuations or using weak valuations. When we say that a formula is an MSO-formula, resp. a WMSO-formula, we mean that the formula is interpreted using arbitrary, resp. weak valuations.

Remark 1. The adjective “weak” is a bit misleading, since WMSO is in general not a fragment of MSO. Indeed, the class of finitely branching trees is not definable in MSO (because, as we will see, every MSO-formula that is satisfiable on trees is true of some finitely branching tree) but is defined by the WMSO-formula $\forall x \exists X \forall y (x \prec y \rightarrow Xy)$. The class of well-founded trees is definable in MSO but not in WMSO, as will follow from results we discuss below (in particular, from the fact that the class of well-founded trees is not Borel). On finitely branching trees, WMSO is strictly less expressive than MSO. This follows from the fact that on finitely branching trees “ X is a finite set” is expressed by the MSO-formula $\forall Y (\forall x (Yx \rightarrow Xx) \wedge \forall y (Yy \rightarrow \exists z (Yz \wedge y \prec z)) \rightarrow \neg \exists y (Yy))$ (“ X does not contain an infinite path”).

Given an MSO sentence ϕ over the alphabet Σ , the tree language defined by ϕ is the set of non empty trees:

$$L(\phi) = \{t \in T_\Sigma \setminus \{\emptyset\} : \phi \text{ is true in } M_t\}$$

Analogously for a WMSO-formula. We say that a tree language L is (W)MSO-definable if there is a (W)MSO-formula ϕ such that $L = L(\phi)$.

For $n \geq 1$, we denote by \equiv_n the equivalence relation that holds between two structures if they cannot be distinguished by an MSO-sentence of quantifier depth n . It is well known that, for each $n \geq 1$, over a finite signature, there are only finitely many \equiv_n -classes of structures. It follows that every structure can be completely described up to \equiv_n -equivalence by a single MSO-sentence of quantifier depth n . Furthermore, the equivalence relation \equiv_n can be characterized using a variant of Ehrenfeucht-Fraïssé games. These games are similar to the ones for first-order logic, except that there is a second type of move, where Spoiler selects a set of elements in one of the two structures and Duplicator responds with a corresponding set of elements in the other structure (in order for Duplicator to win the game, the resulting bijection should not only be a partial isomorphism with respect to the relations in the structures, but should also preserve membership in the chosen sets). As in the first-order case, Duplicator has a winning strategy in the n -round game ($n \geq 1$) iff the two structures satisfy the same MSO-formulae of quantifier depth at most n .

Proposition 2. *If L is an MSO-definable tree language, then there are only finitely many L -equivalence classes of forests and contexts. Moreover, every L -equivalence class (of forests and of contexts) has a finitely branching and regular member.*

Remark 2. It can be naturally asked what is the relation between the introduced notion of L -equivalence and the one originally introduced by Bojańczyk and Idziaszek. It follows by the previous Prop. 2 that if L -equivalence were defined using regular forest-templates and regular context-templates only, the result would be the same as L -equivalence the way we defined, assuming that L is a MSO definable tree language. This means that our definition essentially coincides with the one used in [5].

We still need a slightly more fine-grained version of Prop. 2. Let c be a context and f a forest. By a *forest built from c and f* , we will mean a forest g for which there exists a forest s such that replacing each non-leaf node in s by a copy of c and replacing each leaf-node of s by a copy of f yields g . We then say that s is the *skeleton* of g .

Proposition 3. *Let L be any MSO-definable tree language. Let c be a context, f a forest and g a forest built from c and f whose skeleton satisfies some MSO-sentence ψ . Then there is an L -equivalent forest g' built from c and f , whose skeleton is regular, finitely branching and satisfies ψ as well.*

2.3 The logic EF and EF-bisimulation

Fix an alphabet Σ . The set of formulae of EF over Σ is defined by the grammar

$$\phi ::= a \mid \phi \wedge \phi \mid \neg\phi \mid \text{EF}\phi \quad (a \in \Sigma)$$

The semantics of EF over non empty Σ -trees (where the distinguished node is the root of the tree) is defined inductively as usual by saying that every EF-formula $a \in \Sigma$ is true in trees with root label a and that an EF formula $\text{EF}\phi$ is true in trees that have a proper subtree where ϕ is true. For any EF formula ϕ , the class of trees where ϕ is true is denoted by $L(\phi)$. Given a tree language L , we say that L is EF-definable if there is an EF-formula ϕ such that $L = L(\phi)$. The following is well-known (see, e.g., [3]):

Proposition 4. *Every EF-definable tree language is also WMSO-definable (and, in fact, definable in first-order logic with the descendant relation).*

Following [5], we introduce a special bisimilarity game on forests, called the EF bisimulation game. We first define the game in the case of trees. Let t_0 and t_1 be two trees. The EF bisimulation game over t_0 and t_1 is played by two players: Bob and Anne. The game proceeds in rounds. At the beginning of each round, the state in the game is a pair of trees (t'_0, t'_1) . A round is played as follows. First if the root labels a_0, a_1 of t'_0, t'_1 are different, then Bob wins the whole game. Otherwise Bob selects one of the trees t'_i , for $i = 0, 1$, and its subtree s_i . Then Anne selects a subtree s_{1-i} in the other tree t'_{1-i} . The round is finished, and a new round is played with the state updated to (s_0, s_1) . If Anne can survive for infinitely many rounds in the EF bisimulation game on t_0 and t_1 , then we say that the trees t_0 and t_1 are EF-bisimilar.

Note that clearly if two trees are bisimilar in the standard way, they also are EF-bisimilar. The converse need not to be true. Consider for example the binary tree t on the alphabet $\{a, b\}$ where the only nodes labelled by a are the nodes 0^{2k+1} , with $k > 0$, and the tree t' on the alphabet $\{a, b\}$ where the only nodes labelled by a are the nodes 0^{2k} , with $k > 0$. The two trees are EF-bisimilar but not bisimilar.

A tree language L is called *invariant under EF-bisimulation* if it is impossible to find two trees, $t_0 \in L$ and $t_1 \notin L$ that are EF-bisimilar. From the previous remark on the interrelation between standard bisimilarity and EF bisimilarity, if two tree languages are EF-bisimilar they are also bisimilar, but the converse is in general not true.

This game is so designed that all tree languages defined by an EF formula are invariant under EF-bisimulation. Formally, we have that:

Proposition 5 ([5]). *Every EF-definable tree language is invariant under EF-bisimulation.*

The converse is false. A counter-example is the language of all well-founded trees over a fixed finite alphabet. This language is invariant under EF-bisimulation but it cannot be defined by an EF-formula, as follow from Thm. 1.

We say that a context c is *L -idempotent* if the composition $c(c)$ is L -equivalent to c .

Theorem 4. *An MSO definable tree language L can be defined by an EF formula iff*

- (1) *L is invariant under EF-bisimulation;*
- (2) *for every L -idempotent context c and for every forest f , $c(f) \equiv_L (c + c(f))^\infty$*

Moreover the previous two conditions are decidable.

Remark 3. This characterization of the logic EF was proved by Bojańczyk and Idziaszek in [5] for the case of finitely branching trees. It follows from Prop. 2 that the result holds for arbitrarily branching trees as well. Strictly speaking, the result in [5] is different as it characterizes definability by EF-formulae that are Boolean combination of formulae of the form $\text{EF}\phi$. However, as explained in [5], this is not an essential restriction. More precisely, call an EF-formula ψ a *EF-forest formula* if ψ is a Boolean combination of formulae of the form $\text{EF}\phi$. Then, one can prove that every EF-formula ϕ is logically equivalent to $\bigwedge_{a \in A} (a \rightarrow \phi_a)$, where each ϕ_a is an EF-forest formula. This means that a tree language L is EF-definable iff, for every $a \in A$, the language L_a is definable by an EF-forest formula, where a tree t is said to be in L_a iff the tree obtained from t by relabeling its root with a is in L . Thus, the equivalence relation \equiv_L in the previous theorem is, strictly speaking, the finite intersection of all \equiv_{L_a} .

We extend the notion of EF-bisimilarity to forests by saying that two forests f_1, f_2 are EF-bisimilar if the trees obtained from f_1 and f_2 by adding a “fresh” root, are EF-bisimilar. More precisely, let f_1 and f_2 be two forests over Σ . Let t_1 and t_2 be any pair of trees over $\Sigma \cup \{a\}$, $a \notin \Sigma$, such that: (i) each forest f_i is obtained from the corresponding tree t_i by removing the root node and (ii) $t_1(\varepsilon) = t_2(\varepsilon) = a$. Then we say that f_1 and f_2 are EF-bisimilar if the trees t_1 and t_2 are EF-bisimilar. Note that for forests f_1 and f_2 consisting of a single tree t_1 and t_2 respectively, saying that f_1 and f_2 are EF-bisimilar is not the same as saying that t_1 and t_2 are EF-bisimilar.

The following lemma, relating EF-bisimilarity to L -equivalence, will come in handy later on. Call two contexts EF-bisimilar, if they are bisimilar when viewed as forests over an alphabet containing an additional label “■”.

Lemma 2. *Let L be any EF-bisimulation-invariant tree language. Then every two EF-bisimilar forests are L -equivalent and every two EF-bisimilar contexts are L -equivalent.*

2.4 The topological complexity of tree languages

A topological space is a set X together with a collection τ of subsets of X (called the open sets) containing the empty set and X itself, such that τ is closed under arbitrary unions and finite intersections. Subsets of a topological space X can be classified according to their topological complexity, where the open sets and their complements (the closed sets) are considered to have the lowest complexity. In this context, the notion of a Borel set naturally arises. The set $\mathbf{Borel}(X)$ of Borel sets of a topological space (X, τ) is the smallest set that contains all open sets of X , and is closed under countable unions and complementation. The Borel sets of a topological space can be further classified into an infinite hierarchy, but this will be irrelevant for present purposes.

The topology we will consider here on trees is the *prefix topology*, where the open sets are, intuitively, those sets of trees for which membership of a tree is witnessed by

a finite-depth prefix of the tree. Thus, for example the set of trees containing a node labeled a is an open set, but the complement is not.

Formally, for a Σ -tree t and a natural number $n \geq 1$, the *depth- n prefix* of t , denoted by $t^{(n)}$, is the Σ -tree obtained by restricting the domain of t to all the elements of V_t of depth at most n . We say that two trees t, t' are *equivalent up to depth n* if $t^{(n)} = t'^{(n)}$. We call a set X of trees open if for each $t \in X$ there is a natural number $n \geq 1$ such that for all trees t' , if t and t' are equivalent up to depth n then $t' \in X$. The reader may verify that this indeed yields a topological space. When we say that a set of trees is Borel, we will mean that it is Borel with respect to this topology.

By induction on the structure of a formula, it is easy to verify that:

Proposition 6. *Every WMSO-definable set of trees is Borel.*

A tree is *well-founded* if it has no infinite branch. It is well-known that the tree language of all well-founded trees is an example of a tree language that is *not* Borel.

Given two topological spaces X and Y , a function $F : X \rightarrow Y$ is continuous if for every open set $B \subseteq Y$, $F^{-1}(B) \subseteq X$ is also open. If $F : X \rightarrow Y$ is a continuous function, and $A \subseteq X$ and $B \subseteq Y$ are sets such that $F^{-1}(B) = A$, then we write $A \leq_W B$, and we say that A continuously reduces to B . The relation \leq_W is clearly a pre-order. Intuitively, $A \leq_W B$ means that A is topologically no more complex than B . In particular, we have the following well known fact:

Proposition 7. *If X and Y are topological spaces $A \subseteq X$, $B \subseteq Y$ are sets, such that $A \leq_W B$, then $B \in \mathbf{Borel}(Y)$ implies $A \in \mathbf{Borel}(X)$.*

Given a depth- n prefix $t^{(n)}$ over Σ , by $t^{(n)} \cdot T_\Sigma$ we denote the set of all trees over Σ extending $t^{(n)}$. The next proposition, determining a sufficient condition for a function in order to be continuous, will be very useful.

Proposition 8. *Let Σ and Σ' be two finite sets, and F be a function from T_Σ into $T_{\Sigma'}$. If for every depth- n prefix $t^{(n)}$ over Σ there exists a depth- n prefix $t'^{(n)}$ over Σ' such that $F(t^{(n)} \cdot T_\Sigma) \subseteq t'^{(n)} \cdot T_{\Sigma'}$, then F is continuous.*

3 Main Result

We now prove the equivalence of the given characterizations of the logic EF on trees.

Theorem 1. Let L be any MSO-definable tree language. The following are equivalent and decidable:

- (1) L is EF-definable,
- (2) L is closed under EF-bisimulation and WMSO-definable,
- (3) L is closed under EF-bisimulation and Borel,
- (4) L is closed under EF-bisimulation, and for every L -idempotent context c , and for every forest f , $c(f)$ and $(c + cf)^\infty$ are L -equivalent.

Proof. The proof proceeds in a round-robin fashion. Decidability follows from Thm. 4. (1) \Rightarrow (2) follows by applying Prop. 5 and Prop. 4, (2) \Rightarrow (3) by Prop. 6, while (4) \Rightarrow (1) is given by Thm. 4. The proof of (3) \Rightarrow (4) is by contraposition. Suppose there is an idempotent context c and a forest t such that $c(t)$ and $(c + c(t))^\infty$ are not L -equivalent. There are two cases:

- (a) for some forest-template e , $e[\star \mapsto c(t)] \in L$ and $e[\star \mapsto (c + c(t))^\infty] \notin L$,
 (b) for some forest-template e , $e[\star \mapsto c(t)] \notin L$ and $e[\star \mapsto (c + c(t))^\infty] \in L$.

We will show that both these two possibilities imply that L is not Borel, by giving a continuous reduction of the non-Borel set WF or its complement to L . Recall that WF is the set of all well-founded trees over an alphabet consisting of a single letter a .

For any tree t' over $\{a\}$, let $\widehat{t'}$ be the forest obtained as follows: if t' is the empty tree then $\widehat{t'}$ is ct , otherwise $\widehat{t'}$ is obtained by replacing every leaf node of t' by the forest ct , and replacing every non-leaf node by the context $(c + ct)$. Finally, let $F(t')$ be the tree $e(\widehat{t'})$, i.e., the result of replacing every \star in e by $\widehat{t'}$. Because for every k , the first k levels of t' determines the first k levels of $F(t')$, by Prop. 8 F is continuous.

Claim (i) if $t' \in \text{WF}$, then $\widehat{t'}$ is L -equivalent to ct

Claim (ii) if $t' \notin \text{WF}$, then $\widehat{t'}$ is L -equivalent to $(c + ct)^\infty$

This implies that L is not Borel. Indeed, suppose case (a) holds. From the above claim, we obtain that $t' \in \text{WF}$ iff $F(t') \in L$. This means that $F^{-1}(L) = \text{WF}$, proving that $\text{WF} \leq_W L$. Analogously, if case (b) holds, from the previous claim we obtain that $t' \in \text{WF}^{\complement}$ iff $F(t') \in L$, where WF^{\complement} is the complement of WF. Thus $F^{-1}(L) = \text{WF}^{\complement}$ and therefore $\text{WF}^{\complement} \leq_W L$. It remains only to prove the above claim.

We first introduce some terminology. Clearly, $\widehat{t'}$ is a forest that is built from the context c and the tree t . Let us denote the skeleton of $\widehat{t'}$ by s . Suppose that s is not well-founded. If x is a node in the domain of s such that the subtree $s.x$ is well-founded and there is no ancestor y of x such that $s.y$ is also well-founded, then we call the subforest of $\widehat{t'}$ built up from c and t whose skeleton is $s.x$ a *maximal well-founded subforest* of $\widehat{t'}$.

We first prove claim (i). Suppose that $t' \in \text{WF}$. Then, clearly, s is well-founded. By Prop. 3, $\widehat{t'}$ is L -equivalent to a finitely branching, regular forest t'' built up from c and t whose skeleton s'' is well-founded (recall that well-foundedness is MSO-definable). Since every infinite finitely branching tree has an infinite branch, we then know that the skeleton s'' is in fact finite. Let k be the length of the longest branch of s'' . It is easy to see that s'' is EF-bisimilar to a tree that has a single path whose labels (read from the root to the leaf) form the word $c^{k+1}t$. For convenience, we denote this tree itself by $c^{k+1}t$. Analogously, it is not hard to see that t'' is EF-bisimilar to the forest built up from c and t whose skeleton is $c^{k+1}t$, i.e., the forest $c^{k+1}t$. Since EF-bisimilarity implies L -equivalence (Lemma 2), $\widehat{t'}$ is L -equivalent to $c^{k+1}t$. And since c is an L -idempotent context, we have that $\widehat{t'}$ is in fact L -equivalent to ct .

Next, we prove claim (ii). Suppose that $t' \notin \text{WF}$. Clearly, s is non well-founded. By Prop. 3, $\widehat{t'}$ is L -equivalent to a regular (finitely-branching) forest t'' built up from c and t whose skeleton s'' is non well-founded (recall that non well-foundedness is MSO-definable). Because $\widehat{t'}$ is regular, there are only finitely many maximal well-founded subforests of $\widehat{t'}$ built up from c and t . Thus since we know that every well-founded forest built from c and t is L -equivalent to ct , we can take all maximal well-founded subforests of $\widehat{t'}$ built up from c and t and replace each of these subforests by ct . Here, we use the substitution lemma (Lemma 1). Furthermore, because EF-bisimilarity implies L -equivalence (Lemma 2), if the resulting forest contains several copies of ct next to each other as siblings, they can be collapsed into one. This yields a new forest that can be viewed as a forest built up from the context $c + ct$ alone, whose skeleton has no leaves. But any such forest is EF-bisimilar, hence L -equivalent, to the forest $(c + ct)^\infty$. \square

4 Eliminating recursion from μ -formulae on transitive structures

Recall that a Kripke model is a non-empty $\wp(\text{Prop})$ -colored directed graph, where Prop is some set of proposition letters, and that a *pointed* Kripke model is a Kripke model with a distinguished state. We may assume that Prop is finite. Kripke models are more general than trees: they can contain cycles, there is not necessarily a unique root, and a node may satisfy any number of proposition letters.

We refer to [10] for the syntax and semantics of the modal μ -calculus. The class of all pointed models of a μ -formula ϕ is denoted by $\|\phi\|$, whereas the class of all *transitive* pointed models of ϕ is denoted by $\|\phi\|^{\text{tr}}$. Notice that the logic EF can be seen as modal logic interpreted on the transitive closure of trees.

On arbitrary Kripke models, modal logic corresponds to the bisimulation invariant fragment of first-order logic [2], whereas the modal μ -calculus is the bisimulation invariant fragment of MSO [12]. This means that on arbitrary Kripke models, a μ -formula is equivalent to a modal formula iff it is equivalent to a FO formula. Moreover, by a result of Martin Otto [14], this is decidable. We prove an analogous effective characterization for transitive models:

Theorem 2. For every μ -formula ϕ , the following are equivalent and decidable:

- (1) ϕ is equivalent on transitive Kripke models to a modal formula,
- (2) ϕ is equivalent on transitive Kripke models to a WMSO-formula,
- (3) the class of pointed transitive Kripke models satisfying ϕ is Borel.

For a μ -formula ϕ , let $(\phi)_\mu^*$ be the μ -formula obtained from ϕ by replacing every subformula of the form $\diamond\phi$ by $\mu x. \diamond(x \vee \phi)$ where x is a fresh variable. For a WMSO formula ϕ , let $(\phi)_{\text{WMSO}}^*$ be the WMSO formula obtained from ϕ by replacing each subformula of the form $x \prec y$ by the WMSO formula that says that the node y is reachable from x along \prec . It is easy to see that the following holds :

Lemma 3. For every pointed Kripke model (g, s)

- (1) for every μ -formula ϕ , we have that $(g, s) \in \|(\phi)_\mu^*\|$ iff $(g^{\text{tr}}, s) \in \|\phi\|^{\text{tr}}$
- (2) for every WMSO-formula ϕ , we have that $(\phi)_{\text{WMSO}}^*$ is true in g iff ϕ is true in g^{tr} where by g^{tr} we denote the transitive closure of g .

We now define what it means for a class of models to be “Borel”. Given a pointed model (g, s) , its *set of tree companions* is the set $T(g, s)$ of pointed models (t, r) such that (g, s) is bisimilar (in the usual sense) with (t, r) , and the underlying directed graph of t is a tree with root r . For a set P of pointed models, $T(P) = \bigcup_{(g,s) \in P} T(g, s)$. We say that the set of pointed transitive models of a μ -formula ϕ is Borel iff $T(\|(\phi)_\mu^*\|)$, viewed as a tree language over $\wp(\text{Prop})$, is Borel. We can finally proceed to show how Thm. 2 follows from Thm. 1.

Proof of Theorem 2. The implication (1) \Rightarrow (2) is easily verified. Thus, if we verify that (2) \Rightarrow (3) and (3) \Rightarrow (1) we are done. For the first implication we reason as follows. Suppose that ϕ is equivalent on transitive Kripke models to a WMSO-formula γ . Then by Lemma 3, $(\phi)_\mu^*$ is equivalent on $\wp(\text{Prop})$ -trees to the WMSO-formula $(\gamma)_{\text{WMSO}}^*$. By Prop. 6 we obtain that $T(\|(\phi)_\mu^*\|)$ is a Borel set. For the second implication, first

recall that for every EF-formula φ , there is a modal formula ψ such that for every tree t over Prop, φ is true over t iff $(t, r) \in \|\psi\|_{\mu}^*$, where r is the root of t . Suppose that $\|\phi\|^{\text{tr}}$ is Borel. By definition, this means that $T(\|(\phi)_{\mu}^*\|)$ is a Borel set. But on the one hand we have that the set $T(\|(\phi)_{\mu}^*\|)$ is the set of tree models of the μ -formula $(\phi)_{\mu}^*$, meaning that it is MSO-definable. This set is clearly closed under EF-bisimulation. By Theorem 1, $T(\|(\phi)_{\mu}^*\|)$ is EF-definable, meaning that, by applying Lemma 3, there is modal formula ψ such that $\|\psi\|^{\text{tr}} = \|\phi\|^{\text{tr}}$. \square

Theorem 3. For every WMSO-formula $\phi(x)$, the following are equivalent:

- (1) $\phi(x)$ is equivalent on transitive Kripke models to a modal formula,
- (2) $\phi(x)$ is bisimulation invariant on transitive Kripke models.

Proof. (1) \Rightarrow (2) is clear. In order to prove (2) \Rightarrow (1), we reason as follows. Let $\phi(x) \in \text{WMSO}$ be bisimulation invariant on transitive models. Then $\|(\phi(x))_{\text{WMSO}}^*\|$ is closed under bisimulation. By [12], there is a μ -formula ψ equivalent to $(\phi(x))_{\text{WMSO}}^*$ on all models, meaning that $\phi(x)$ and ψ are equivalent on transitive models. By applying Thm. 2 we conclude the proof. \square

References

1. BENEDIKT M., SEGOUFFIN L.: Regular Tree Languages Definable in FO and in FO_{mod} . *ACM Trans. on Computational Logic*: 11(1) (2009)
2. VAN BENTHEM, J.: *Modal Correspondence Theory*. PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam (1976)
3. BLACKBURN P., DE RIJKE M., VENEMA Y.: *Modal Logic*. Cambridge Univ. Press (2001)
4. BOJAŃCZYK M.: Two-Way Unary Temporal Logic over Trees. *LICS'07*: 121–130 (2007)
5. BOJAŃCZYK M., IDZIASZEK T.: Algebra for Infinite Forests with an Application to the Temporal Logic EF. *CONCUR'09*: 131–145 (2009)
6. BOJAŃCZYK M., SEGOUFFIN L., STRAUBING H.: Piecewise Testable Tree Languages. *LICS'08*: 442–451 (2008)
7. BOJAŃCZYK M., SEGOUFFIN L.: Tree Languages Defined in First-Order Logic With One Quantifier Alternation. *ICALP'08*: 233–245 (2008)
8. BOJAŃCZYK M., WALUKIEWICZ I.: Characterizing EF and EX Tree Logics. *Theoretical Computer Science* 358 (2-3): 255–272 (2006)
9. BOJAŃCZYK M., WALUKIEWICZ I.: Forest Algebras. In: *Automata and Logic: History and Perspectives*: 107–132, Amsterdam University Press (2007)
10. BRADFIELD, J., STIRLING C.: Modal Logic and Mu-Calculi. In: J. Bergstra, et al.: *Handbook of Process Algebra* Elsevier, North-Holland: 293–332 (2001)
11. DAWAR A., OTTO M.: Modal Characterisation Theorems over Special Classes of Frames. *Ann. Pure Appl. Logic* 161(1): 1–42 (2009)
12. JANIN D., WALUKIEWICZ I.: On the Expressive Completeness of the Propositional μ -Calculus with Respect to Monadic Second Order Logic. *CONCUR'96*: 263–277 (1996)
13. MURLAK F.: Weak Index vs Borel Rank. *STACS'08*: 573–584 (2008)
14. OTTO M.: Eliminating Recursion in the μ -Calculus. *STACS'99*: 531–540 (1999)
15. PLACE T.: Characterization of Logics Over Ranked Tree Languages. *CSL'08*: 401–415 (2008)
16. WALUKIEWICZ I.: Monadic Second-Order Logic on Tree-Like Structures. *Theoretical Computer Science* 275(1-2): 311–346 (2002)