Characterization of Logics Over Ranked Tree Languages

Thomas Place

LSV, ENS-Cachan, CNRS, INRIA
place@lsv.ens-cachan.fr

Abstract. We study the expressive power of the logics $EF + F^{-1}$, $\varDelta_2$ and boolean combinations of $\varSigma_1$ over ranked trees. In particular, we provide effective characterizations of those three logics using algebraic identities. Characterizations had already been obtained for those logics over unranked trees, but both the algebra and the proofs were dependant on the properties of the unranked structure and the problem remained open for ranked trees.

1 Introduction

Understanding the expressive power of logics over labeled trees is an important problem that can be found in many areas of Computer Science. In particular, a logic is said to have a decidable characterization if there exists a decision procedure for the following problem: given a regular language defined by its finite automaton, decide if it can be defined by a formula of the logic.

This type of problem is well known and has been extensively studied for word languages. Many word logics have been proven to have decidable characterizations using algebraic tools. Perhaps the most famous result is the characterization of $FO[<]$, the first-order logic with the order relation, which says that given a regular language L the three following properties are equivalent, L is definable by a first-order formula, L is star-free [8], the syntactic monoid of L is aperiodic [9]. Since the syntactic monoid of a regular language is a computable notion and aperiodicity a decidable property of monoids, the last property is actually a decidable characterization. This result demonstrates the importance and the relevance of the algebraic approach to obtain decidable characterizations. Most common word logics have been proven to have decidable characterizations using this approach.

Much less results are known for tree languages, while it has been known for a long time that regular tree languages are the languages definable in monadic second order logic with the ancestor relation, until recently very few results of this type were known. For example, giving a decidable characterization for the first-order logic with the ancestor relation remains an open problem. Decidable characterizations have been issued by Walukiewicz and Bojańczyk on ranked and unranked trees for some temporal logic in [4,5], on ranked and unranked trees by Benedikt and Segoufin for the first order logic with the successor relation in [1] and on ranked trees by Wilke for frontier testable languages in [11].

Recently, an algebraic formalism called Forest Algebras [5] was proposed for unranked trees. Using this formalism, decidable characterizations for several logics have been obtained. In [2], Bojańczyk, presents a characterization for $EF + F^{-1}$, the temporal logic with the ancestor and descendant modalities, in [7], Bojańczyk, Segoufin and Straubing present a characterization for boolean combinations of Σ_1, the logic of boolean combinations of purely existential first-order formulas, and in [3], Bojańczyk and Segoufin present a characterization for Δ_2, the languages definable by boolean combination of first order formulas with only one quantifier alternation.

Our aim in this paper is to present decidable characterizations for $EF+F^{-1}$, Δ_2 and Σ_1 over ranked trees using an algebraic formalism that is close to both Forest Algebras and the formalism used by Wilke in [11]. Both the statements and the proofs of the characterizations for unranked trees rely on the specific structure of unranked trees and have no obvious extension on ranked trees. For example, languages definable in $EF + F^{-1}$ are closed under bisimulation, the characterization of [2] reflects this property with an identity stating that those languages are closed under the action of duplicating a subtree within a tree, this property obviously does not make sense for ranked trees since each node has a fixed arity. Another simple example is the proof of the characterization of Σ_1, which uses the fact that there is a natural way to supress a node from an unranked tree to form a new unranked tree, once again the fixed arity of nodes on ranked trees makes this operation more delicate and technical. Therefore, since the arity of nodes cannot be expressed in those logics, giving a decidable characterization for ranked tree languages definable in them remained an open problem.

2 Notations

In this paper, we work with binary trees. Classic binary trees only have nodes of arity two or zero, meaning that every node has either zero or two sons, in the model we use, we also allow nodes of arity one (nodes that have exactly one son). Our motivation is that structures of arity one play a central role in our characterizations. However, this assumption is by no means restrictive since we can always suppose that the set of labels of arity one is empty.

Trees are defined over a finite alphabet (A, B, C), where A is a set of leaves symbols, B a set of unary symbols and C a set of binary symbols. The notion of tree is defined inductively as follows: Any $a \in A$ is a tree, if t is a tree bt is tree if $b \in B$ and if t and t' are trees, $c(t, t')$ is tree for $c \in C$. The notion of nodes of a tree is defined in the usual way. A tree t' is a subtree of a tree t if there is a node x of t such that t' is the tree we get by keeping only the nodes of t that are below x.

A set of trees L over an alphabet (A, B, C) is called a tree language. As usual a regular language is a language recognized by a finite bottom-up automata, all the languages we consider in this paper are regular.

A context over an alphabet (A, B, C) is a tree that has exactly one leaf labeled by a special symbol $*$ that we call the hole. Notice that there exists a natural composition operation on contexts, if we take two contexts p, q, we get a new context pq by replacing $*$ in p by q. Another natural operation is to attach a tree t to replace $*$, forming a new tree pt.

We also consider objects that we call bi-contexts, which are trees with a special subtree $c(*, *)$ with $c \in C$. Notice that we also have natural operations with this type of object, we can attach a bi-context below a context to get a new bi-context, or attach a tree to replace the left or right $*$ in a bi-context to get a context.

We are interested in three tree logics called $EF + F^{-1}$, Δ_2 and Σ_1. We see a tree as a logical structure, we take the set of its nodes as domain and consider unary predicates P_d for $d \in A, B, C$, which hold true in x if x is labeled by d, and a binary predicate for the ancestor relation $<$.

A formula φ of $EF + F^{-1}$ over an alphabet (A, B, C) is defined by the following grammar:

$$\varphi = P_d \; d \in A, B, C \mid EF\varphi \mid F^{-1}\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$$

Given a tree t and a node x of t, we say that $t, x \models \varphi$ if and only if:

- $\varphi = P_d$ and x is labeled by d
- $\varphi = EF\varphi'$ and x has a proper descendant x' such that $t, x' \models \varphi'$
- $\varphi = F^{-1}\varphi'$ and x has a proper ancestor x' such that $t, x' \models \varphi'$
- The semantics of the boolean operators are defined in the usual way

We say that $t \models \varphi$ if $t, x \models \varphi$ with x the root of t. A formula φ defines a tree language over (A, B, C) which is the set of trees t such that $t \models \varphi$.

Δ_2 is a restriction of the first-order logic on trees. A formula of Σ_2 is a first-order formula that can be written as the disjunctive and conjunctive combination of formulas of the form $\exists x_1 ... \exists x_n \forall y_1 ... \forall y_m \varphi(x_1, ..., x_n, y_1, ..., y_m)$ with φ quantifier free. A Π_2 formula is the negation of a Σ_2 formula. We say a language is in Δ_2 if it is definable by both a Σ_2 and a Π_2 formula.

Σ_1 is another restriction of the first-order logic on trees. We consider formulas that are boolean combinations of formulas of the form $\exists x_1 ... \exists x_n \varphi(x_1, ..., x_n)$ with φ quantifier free. A language is in Σ_1 if it is definable by such a formula.

Our main goal is to give a decidable characterization for those three logics on binary trees. More formaly, given $\mathcal{L}$ a tree logic, we want to be able to decide the following problem:

INPUT: A regular tree language L
PROBLEM: Can we define L with a formula of $\mathcal{L}$

Notice that the two logics $EF + F^{-1}$ and Δ_2 are related. On words they have the same expressive power [6, 10]. This result no longer holds on trees, for example if we fix a $a \in A$ and consider L the language of trees which two different leaves labeled by a, L is definable in Δ_2 but not in $EF + F^{-1}$. However, we will see they remain closely related.

For both words and unranked trees the characterizations of those logics are stated and proven using an algebraic formalism, we also define our own algebraic formalism. Our definition is intended to be close to the definition of Forest Algebras, the formalism used for unranked trees, in order to be able to follow and compare the ranked and unranked cases.

3 Binary Algebras

We defined three types of objects, trees, contexts and bi-contexts, our algebra reflects that by being constitued of three sets. The set corresponding to trees is actually very close to the set of states of an automaton and the set corresponding to bi-context very close to a transition function. However, we see them as algerbaic objets in order to stay close to the notions introduced for the characterizations over unranked trees.

We first give the formal definition of Binary Algebras, then move on with the definitions of morphisms of Binary Algebras, recognition of a language by a Binary Algebra and syntactic Binary Algebra of a language.

A Binary Algebra is a tuple (H, V, W) where H and W are sets and V is a monoid, we note $\cdot$ its operation. The idea is that each set represents one of the three objects we defined, H represents trees, V contexts and W bi-contexts. Several operations are defined on this tuple, each one reflecting an operation we described on trees, contexts and bi-contexts. The operations of contexts over trees and bi-contexts are reflected by actions of V on H and W. An action of a monoid V on H is function $f: V \times H \to H$ such that $f(v \cdot v', h) = f(v, f(v', h))$. We abusively write both actions $\cdot$ $(f(v, h) = v \cdot h)$, we also want those actions to be faithful, meaning that for each v the function $h \to f(v, h)$ must be different. Finally, we reflect the bi-context tree operations by having two operations $\diamond_l$ and $\diamond_r$, from $W \times H$ onto V, such that $(w \diamond_l h) \cdot h' = (w \diamond_r h') \cdot h$ (the order in which we attach trees under bi-contexts is not important). Given this property we will sometimes write $w(h, h')$ instead of $(w \diamond_l h) \cdot h'$.

We use the usual notion of morphism: A morphism of binary algebras $\alpha : (H_1, V_1, W_1) \to (H_2, V_2, W_2)$ is actually composed of three applications $\beta : H_1 \to H_2$, $\gamma : V_1 \to V_2$ and $\delta : W_1 \to W_2$, such that γ is a morphism and $\gamma(v)\beta(h) = \beta(vh)$, $\gamma(v)\delta(h) = \delta(vh)$, $\delta(w) \diamond_l \beta(h) = \gamma(v \diamond_l h)$ and $\delta(w) \diamond_r \beta(h) = \gamma(v \diamond_r h)$.

Given an alphabet (A, B, C), the Free Binary Algebra $(A, B, C)^{\Delta}$ is the binary algebra $(H_{\Delta}, V_{\Delta}, W_{\Delta})$ such that H_{Δ} is the set of trees that can be built using (A, B, C), V_{Δ} is the set of contexts that can be built using (A, B, C) and W_{Δ} is the set of bi-contexts that can be built using (A, B, C). The operation $\cdot$ is the natural context operation. $w \diamond_l h$ $(w \diamond_r h)$ is the context obtained by filling the left (right) hole of w with h.

We say a tree language L is recognized by a Binary Algebra (H, V, W) if and only if there is a morphism $\alpha : (A, B, C)^{\Delta} \to (H, V, W)$ and a subset G of H such that $L = \alpha^{-1}(G)$. It is simple to show that a tree language is regular if and only if it is recognized by a finite binary algebra.

Given a tree language we define an equivalence relation on H_Δ as follows $t \sim_L t'$ if and only if for all context $p \in V_\Delta$, pt is in L if and only if pt' is in L. We extend this equivalence on V_Δ as follows $p \sim_L p'$ if and only if for all trees $t \in H_\Delta$, pt is in L if and only if $p't$ is in L. Finally we extend it on W_Δ as follows $q \sim_L q'$ if and only if for all pairs of trees $t, t' \in H_\Delta$, $q(t,t')$ is in L if and only if $q'(t,t')$ is in L. Those relations define a congruence over $(A,B,C)^\Delta$ and we call the quotient, the syntactic binary algebra of L. The syntactic binary algebra of a regular language is a finite object that we can compute from the automata of the language.

In this paper we only consider finite binary algebras. Given any finite monoid V there is a computable number $\omega(V)$ such that for every element v of V, v^ω is an idempotent ($v^\omega = v^\omega v^\omega$). We write this number ω when V is understood from the context.

4 Links between Δ_2 and $EF + F^{-1}$

Recall that we said that over words the logics $F + F^{-1}$ and Δ_2 have the same expressive power and that while this property is no longer true on trees, both the statements and proofs of the characterizations remained closely related. In this section we describe a subclass of regular languages that contains both the languages definable in $EF + F^{-1}$ and Δ_2, and verifies those common properties.

Definition 1. *We say a Binary Algebra (H, V, W) belongs to TDA if it verifies the following identities:*

$$\forall u, v \in V \qquad (uv)^\omega = (uv)^\omega u (uv)^\omega \tag{1}$$

$$\forall h \in H\ \forall w \in W \qquad w \diamond_l h = w \diamond_r h \tag{2}$$

$$\begin{gathered}\forall w, w' \in W\ \forall h, h' \in H \quad \textit{and for } e = ((w \diamond_l h)(w' \diamond_l h'))^\omega \\ e = e(w \diamond_l h')(w' \diamond_l h)e\end{gathered} \tag{3}$$

Notice that over words, the languages definable in $F + F^{-1}$ are exactly the languages whose syntactic monoid verifies Equation (1) (called the DA equation), therefore, this definition is indeed an extension of the word case. Also notice that while (1) and (2) are equations that have clear equivalents on Forest Algebras that are stated in the characterizations of Δ_2 and $EF + F^{-1}$ over unranked trees, (3) is a new equation. Over unranked trees, the properties of Forest Algebras would allow us to derive a similar result from (1), in our case, its statement is needed in order to prove properties that are not consequences of (1). Finally (2) means that $\diamond_r = \diamond_l$, therefore, from now on, we only write $\diamond$.

We will prove our characterizations by induction on orders on binary algebras that are significant because of the properties of TDA, we define them here and state those properties. We define two orders, one on H and one on V, we begin with the order on H.

Given a binary algebra (H, V, W) and $h, h' \in H$, we say that $h \sqsubseteq h'$ if and only if there exists $v \in V$ such that $h = vh'$. We write $h \sim h'$ if and only if

$h \sqsubseteq h'$ and $h' \sqsubseteq h$. We show that $\sim$-classes of a binary algebra of TDA verify useful properties.

Definition 2. *Given $h \in H$ we call $stab(h)$ a tuple of three sets:*

$$stab_H(h) = \{g \in H \mid \exists w \in W \quad w(h,g) \sim h\} \; stab_V(h) = \{v \in V \mid vh \sim h\}$$
$$stab_W(h) = \{w \in W \mid \exists g \in H \quad w(h,g) \sim h\}$$

The following proposition shows that in Binary Algebras of TDA, $stab(h)$ depends only on the $\sim$-class of h. Therefore, we will sometimes write $stab(G)$ if G is a $\sim$-class. This result makes $\sqsubseteq$ a relevant order to use for the inductions in the proofs of our characterizations.

Proposition 1. *If (H,V,W) verifies the TDA identities, the following properties hold:*

- $h \sim h' \Rightarrow stab(h) = stab(h')$.
- $stab_V(h)$ *is a submonoid of* V.
- $stab_W(h) \diamond stab_H(h) \subseteq stab_V(h)$
- $stab_V(h) stab_W(h) \subseteq stab_W(h)$

This proposition is proven using classical algebraic techniques. Notice that Equation (3) is used in order to prove the third item.

The order $\sqsubseteq$ has a second property that we will use in our proofs, there is a single minimal class of $\sim$ relatively to $\sqsubseteq$. We call it H_{min}.

Lemma 1. *Given (H,V,W) a Binary Algebra, there is a minimal $\sim$-class in H relatively to $\sqsubseteq$ and if $(H,V,W) \in TDA$, for all minimal h, there exists $u_h \in V$ such that $\forall h' \in H \; u_h h' = h$.*

Proof. Let $g \in H_{min}$. We write $H = \{h_1, ..., h_n\}$. We take $w \in W$. Consider:

$$h_{min} = ((w \diamond h_1)...(w \diamond h_n)^{\omega} g$$

$h_{min} \in H_{min}$, hence, we have $g = v h_{min}$. We take:

$$u_g = v((w \diamond h_1)...(w \diamond h_n)^{\omega} w(*, ((w \diamond h_1)...(w \diamond h_n))^{\omega} g)$$

Thanks to equation (1) we have $\forall h \; u_g h = g$. $\square$

Over V we use the classic Green relation $\mathcal{R}$ on monoids, $v \leq_{\mathcal{R}} v'$ if and only if $\exists u \in V \mid v = v'u$. $\mathcal{R}$-classes verify properties that are very similar to the ones verified by $\sim$-classes, if (H,V,W) belongs to TDA.

Definition 3. *Given $v \in V$ we call $stab(v)$ a tuple of three sets:*

$$stab_H(v) = \{g \in H \mid \exists w \in W \quad v(w \diamond g) \; \mathcal{R} \; v\} \; stab_V(v) = \{u \in V \mid vu \; \mathcal{R} \; v\}$$
$$stab_W(v) = \{w \in W \mid \exists g \in H \quad v(w \diamond g) \; \mathcal{R} \; v\}$$

Again, we show in the following proposition that in Binary Algebras of TDA, $stab(v)$ depends only on the $\mathcal{R}$-class of v. Therefore, we will sometimes write $stab(U)$ if U is a $\mathcal{R}$-class. The order $\leq_{\mathcal{R}}$ is also relevant to be used for the inductions in the proofs of our characterizations.

Proposition 2. *If (H, V, W) verifies the TDA identities, the following properties hold:*

- $v \mathcal{R} v' \Rightarrow stab(v) = stab(v')$.
- $stab_V(v)$ *is a submonoid of* V.
- $stab_W(v) \diamond stab_H(v) \subseteq stab_V(v)$
- $stab_V(v) stab_W(v) \subseteq stab_W(v)$

5 Characterization of $EF + F^{-1}$

In this section we give a decidable characterization for $EF + F^{-1}$. One of the identities we state in our characterization uses a relation over V. This relation can be seen as a more powerful binary variant of a relation used in [2] over Forest Algebras. In [2], the relation compared two contexts, from a context a smaller context can be built by suppressing subtrees that are off the path from the root to the hole. This notion does not have any obvious extension to binary trees since we cannot supress subtrees in a binary tree without changing the arity of a node. We use an alternate relation, instead of suppressing subtrees, we ask that all the subtrees that are off the path from the root to the hole in the smaller context can be found in the same path in the bigger context.

Definition 4. *Given $u, v \in V$, we say that $u \dashv v$ if and only if we can write:*

- $u = (u_0 \diamond h_0)...(u_n \diamond h_n)$
- $v = (u_0 \diamond g_0)...(u_n \diamond g_n)$
- $\{h_0, ..., h_n\} \subseteq \{g_0, ..., g_n\}$

With $u_0, ..., u_n \in W$ and $g_0, ..., g_n \in H$.

Theorem 1. *Let L be a regular tree language on an alphabet (A, B, C), L is definable in $EF + F^{-1}$ if and only if its syntactic binary algebra belongs to TDA and verifies the two following identities:*

$$\begin{gathered}\forall w \in W \ \forall h, g \in H \\ (w \diamond h)^\omega g = (w \diamond h)^\omega w((w \diamond h)^\omega g, (w \diamond h)^\omega g)\end{gathered} \tag{4}$$

$$\begin{gathered}\forall u_1, u_2, v_1, v_2 \in V \quad \text{such that } u_1 \dashv u_2 \text{ and } v_1 \dashv v_2 \\ (u_1 v_1)^\omega (u_2 v_2)^\omega = (u_1 v_1)^\omega v_2 (u_2 v_2)^\omega\end{gathered} \tag{5}$$

Notice that identities (2), (3), (4) and (5) are sufficient, and that identity (1) is redundant since it a consequence of (5) when $u_1 = u_2$ and $v_1 = v_2$.

The characterization proposed in [2] for unranked tree languages shares some similarities with our definition. This characterization can be seen as divided in two parts, an horizontal one and a vertical one. Horizontally, it states closure under bisimulation, while we still state commutativity with (2), as we said closure under duplication of subtrees has no sense on binary trees, we replace it with (4). We will use (4) to solve in an alternate way the cases where closure under duplication of subtrees is used on unranked trees.

Vertically, the characterization of [2] used two identities, the classic DA equation (1), and another equation similar to our equation (5). Since (1) is a consequence of (5), (5) replaces those two equations and plays a similar role in the proof. Recall however, that we had to redifine the relation $\dashv$, which will lead to differents uses of this equation. Finally, we need to state a last equation (3) which is used for proving the propreties of the *stab* sets in the TDA section. This equation is needed in order to solve problems related to the lack of malleability of the binary tree structure.

Before we prove this proposition, we prove that it fulfills our decidability goal, given a regular language we compute it syntactic binary algebra and then, since the $\dashv$ relation is computable via a fix point algorithm, we can decide whether it satisifes the identities or not, which decides our problem.

We proceed with the proof, we prove both directions using an Ehrenfeucht Fraïssé approach. A k rounds Ehrenfeucht Fraïssé game on two trees t and t' is played as follows. There are two players called Spoiler and Duplicator with two pebbles each, when the game begins, each player has a pebble on the root of t or t'. A round is played as follows, with a pebble at position x_1 on t and at position x_2 on t':

- Spoiler chooses one of the two trees, say t, and he chooses a node y_1 which is either a proper descendant or a proper ancestor of x_1 and puts his pebble on it.
- Duplicator chooses y_2 in t' with the same label as y_1 and which is a proper descendant of x_2 if y_1 was a proper descendant of x_1 and a proper ancestor of x_2 if y_1 was a proper ancestor of x_1. If she can't play Spoiler wins.
- Both players take back their pebble in x_1, x_2 and they move on to the next round with y_1, y_2 playing the role of x_1, x_2.

If Duplicator can survive k rounds she is declared the winner and we write $t \cong_k t'$. We state a Lemma that links the notion of definability in $EF + F^{-1}$ with the notion of game we just defined, this Lemma is proven using classical Ehrenfeucht Fraïssé techniques. The rank of a formula is its nesting depth of modalities.

Lemma 2. *We have $t \cong_k t'$ if and only if t and t' satisfy the same $EF + F^{-1}$ formulas of rank k.*

The easier direction of Theorem 1 is proven using classic Ehrenfeucht Fraïssé techniques, if a language L is definable in $EF+F^{-1}$, the syntactic binary algebra of L must verify the identities of Theorem 1.

We prove the hardest direction of Theorem 1, if L is a language whose syntactic algebra verifies (2), (3), (4) and (5), it is definable in $EF + F^{-1}$. Let

(H, V, W) be the syntactic binary algebra of L and α the corresponding morphism, we suppose that $\forall h \in H\ \exists a \in A$ such that $\alpha(a) = h$ (notice that this assumption is not restrictive as we will not consider the size of A in the proof). Given a subset X of H we say that a tree t is X-trimmed if and only if the only subtrees of t that have their image under α in X are leaves. Instead of directly proving the proposition we prove a slightly more general Proposition.

Proposition 3. *Let X be a union of $\sim$-classes of H and $v \in V$. There exists $k \in \mathbb{N}$ such that for all X-trimmed trees t and t', we have:*

$$t \cong_k t' \quad \Rightarrow \quad v\alpha(t) = v\alpha(t')$$

This proposition is very similar to the one proved in [2], the main difference in the statement being our usage of Ehrenfeucht Fraïssé games. We also use a similar proof structure to prove it, the inner proofs however, are different because of our new equations and of the constraints related to the ranked tree structure.

We first show that Theorem 1 is a consequence of this Proposition 3. Take $X = \emptyset$ and $v = 1_V$ (the neutral element of V) we get:

$$\exists k \text{ such that } \forall t, t' \quad t \cong_k t' \quad \Rightarrow \quad \alpha(t) = \alpha(t')$$

It means that for some k, L is the union of classes of the Ehrenfeucht Fraïssé game. Since the classes of the Ehrenfeucht Fraïssé game are definable in $EF + F^{-1}$ (see Lemma 2), it follows that L is definable in $EF + F^{-1}$. The rest of this section is devoted to the proof of Proposition 3. We show that it holds for:

$$k \geq (2^{2(|B|+|C|)})(2^{2dp(v)})(4 \times 2^{2|H-X|})((2^{2|V|})^{|H|+1})(3 \times 2^{2|H|})$$

We proceed by induction on the four following parameters:

1. $|H|$
2. The number of $\sim$-classes left in $H - X$
3. $|B| + |C|$
4. The number of $\mathcal{R}$-classes left below v

We consider three cases. First, we suppose that there are at least two $\sim$-classes in $H - X$, we will decrease the first and second induction parameters to conclude by induction. In the second case, we suppose that there are still labels in t and t' that decrease the $\mathcal{R}$-class of v and we will decrease the third and fourth parameters to conclude by induction. The third case is the complement of the two first cases, we will show that if neither of the assumptions we state in those cases hold, we are able to conclude the proof using our identities.

5.1 First Case

In this case we suppose than there is more one $\sim$-class left in $H - X$, we will conclude by induction, adding one $\sim$-class to X. We take $G = \{g_1, ..., g_n\}$, a maximal $\sim$-class in $H - X$ relatively to $\sqsubseteq$, let $a_1, ..., a_n$ be leaf representatives for the $g_1, ..., g_n$ ($\alpha(a_i) = g_i$). We say a tree is a twig if its only node which is not a leaf is its root.

Definition 5. *From t and t', we construct new trees:*

1. *s and s' by replacing all twigs of type g_i by a_i, for all i.*
2. *r and r' by replacing all maximal subtrees of s and s' with type $g_i \in G$ (maximal in the sense that they are not subtrees of bigger subtrees of type in G) with the leaf a_i. Notice that r and r' are $X \cup G$ trimmed.*

Notice that by construction, we have $v\alpha(t) = v\alpha(s) = v\alpha(r)$ and $v\alpha(t') = v\alpha(s') = v\alpha(r')$. The following Lemma, which is the central point of this case, is a consequence of the assumption we made about $H - X$ containing more than one $\sim$-class and abour G being a maximal one.

Lemma 3. $r \cong_{\frac{k}{2}-4} r'$

Before we prove it, we show how it can be used to conclude this case. We have:

1. $r \cong_{\frac{k}{2}-4} r'$
2. r and r' are $X \cup G$ trimmed
3. $\frac{k}{2} - 4 \geq (4 \times 2^{2|H-X\cup G|})K$, since $k \geq (4 \times 2^{2|H-X|})K$

Therefore using the induction hypothesis $v\alpha(r) = v\alpha(r')$ and by construction, it follows that $v\alpha(t) = v\alpha(t')$.

We prove Lemma 3 in two steps, the first one is that we can detect subtrees of type in G under α in the game. The second one is that we can then detect precisely their type in G. The first step is mainly a consequence of the properties of *stab* described in Section 4. This step uses bisimulation in the unranked case, since we do not have bisimulation we use Equation (4) instead. Notice that this first step, because it relies on the results of Section 4, is also using Equations (1) and (3).

Claim. Let f be a subtree of s and f' a subtree of s' such that $\alpha(f) \in G$ and $\alpha(f') \notin G$. Spoiler wins the two rounds game on f and f'.

Proof. We claim that an X-Trimmed tree t has type outside G if and only if one the following conditions holds:

1. t is a leaf and it has type outside G
2. t has a twig subtree with type outside G
3. t has non-twig unary node with label b such that $\alpha(b) \notin stab_V(G)$
4. t has non-twig binary node with label c such that $\alpha(c) \notin stab_W(G)$
5. t has an inner node with a leaf of label a as brother such that $\alpha(a) \notin stab_H(G)$

As G is a maximal $\sim$-class in $H - X$, it follows from the properties of *stab* that the conditions are sufficient. We prove that they are necessary.

In the unranked case, closure under bisimulation entails that $G \subseteq stab_H(G)$, this no longer true in our case. However, (4) entails a weaker result that is sufficient in order to prove that the conditions are necessary.

Claim. If $stab_H(G) \neq \emptyset$, $G \subseteq stab_H(G)$

Proof. Because there exists an $h \in stab_H(G)$, $\exists w$ such that $w \diamond h \in stab_V(G)$. We have $(w \diamond h)^\omega \in stab_V(G)$ (recall that $stab_V(G)$ is a submonoid), therefore:

$$(w \diamond h)^\omega g = g' \in G$$

Using Equation (4) we get:

$$(w \diamond h)^\omega w(g', g') = g'$$

Therefore $g' \in stab_H(G)$, hence $w(g, g') \sim g$, which also means that $g \in stab_H(G)$. □

We say that (t_1, t_2) is a good pair if it belongs to $stab_H(G) \times G$ or $G \times stab_H(G)$ (notice that if $stab_H(G) = \emptyset$, there is no good pair). Let t be a tree of type outside G, if t is a leaf (1) holds, otherwise let t' be a minimal subtree of t of type outside G if it is a twig (2) holds, otherwise we are in one of the three following cases:

- $t' = b(t_1)$, t_1 has type in G so $\alpha(b) \notin stab_V(G)$, (3) holds
- $t' = c(t_1, t_2)$ and (t_1, t_2) is a good pair. Then, we have $\alpha(t_1) \in stab_H(G)$ and $\alpha(t_2) \in G$. Then, since t' has type outside G, $\alpha(c) \diamond \alpha(t_1) \notin stab_V(G)$, which implies that $\alpha(c) \notin stab_W(G)$ (recall that $stab_W(G) \diamond stab_H(G) \subseteq stab_V(G)$), (4) holds.
- $t' = c(t_1, t_2)$ and (t_1, t_2) is not a good pair, we consdier two cases. If $stab_H(G) = \emptyset$, we have by definition $stab_W(G) = \emptyset$, therefore (4) holds. Otherwise, we have $G \subseteq stab_H(G)$, since (t_1, t_2) is not a good pair, t_1 or t_2 is outside $stab_H(G)$ and it is necessarly a leaf since it cannot have type in G, (5) holds.

So f verifies none of the conditions and f' verifies at least one. It is clear that the four first conditions are detectable in two rounds, the fifth is because f is a subtree of s of type in G so by construction of s it has no leaf with type outside $stab_H(G)$ in twigs. □

The second step is proven using the induction hypothesis. We prove that if Duplicator can win the game on two X-trimmed trees of type in G, they have the same type. This is done by building a smaller algebra than (H, V, W) from *stab* which coincides with (H, V, W) on X-trimmed trees of type in G and using the induction hypothesis on that smaller algebra. Aside from technical details, this proof is similar to the one used in the unranked case. Using these two steps, we are able to derive a winning strategy for Duplicator on r and r' from her winning strategy on t and t', proving Lemma 3.

5.2 Second Case

In this case we suppose that there is a b in B such that $\alpha(b) \notin stab_V(v)$ or a $c \in C$ such that $\alpha(c) \notin stab_W(v)$. Let $B_\downarrow = \{b_1, ..., b_n\}$ be the set of all such

$b \in B$ and $C_\downarrow = \{c_1, ..., c_m\}$ be the set of all such $c \in C$. We reduce the size of the alphabet by suppressing all such b and c. For two trees s, s', we write:

$$s \equiv s' \text{ when } \forall u <_{\mathcal{R}} v \ u\alpha(s) = u\alpha(s')$$

Notice that $\equiv$ is an equivalence relation of finite index. Let $L_1, ..., L_\iota$ be the classes of this equivalence relation and $a_1, ..., a_\iota$ be leaf representatives of those classes. We write $a_{i,l}$ the leaves that have the same type as the trees $b_l(a_i)$ and $a_{i,j,l}$ leaves that have the same type as the trees $c_l(a_i, a_j)$. We modify t and t' in two steps:

1. First we consider each minimal subtree of t which has its root labeled by a $b_l \in B_\downarrow$ (minimal in the sense that it is not subtree of a tree of root in $B_\downarrow$ or $C_\downarrow$). We look at the subtree rooted in b_l, we compute its class L_i of $\equiv$, and we replace it by a_i. We do the same with the minimal subtrees that have their root labeled by $c_l \in C_\downarrow$. We look at the two subtrees that are rooted to the node and we compute their class L_i and L_j of $\equiv$, and we replace them by a_i and a_j. We obtain a new tree s. We modify t' the same way and we obtain s'.
2. Finally we replace the subtrees $b_l(a_i)$ by $a_{i,l}$ and $c_l(a_i, a_j)$ by $a_{i,j,l}$. We have now a pair of trees that we call r and r'.

We state two lemmas that are central to this case. The first one is a consequence of the definition of the relation $\equiv$ and of the sets $B_\downarrow$ and $C_\downarrow$.

Lemma 4. *$v\alpha(t) = v\alpha(s) = v\alpha(r)$ and $v\alpha(t') = v\alpha(s') = v\alpha(r')$*

Lemma 5. *$\bar{r} \cong_{\frac{k}{2}-1} \bar{r}'$*

Before proving those results, we use them to conclude this case. We have:

1. No node of label in $B_\downarrow$ or $C_\downarrow$ is present in r and r' so the size of the node alphabets have decreased in r and r'.
2. $\frac{k}{2} - 1 \geq 2^{2(|B|+|C|-1)}K$ because $k \geq 2^{2(|B|+|C|)}K$.

Therefore using Lemma 5 by induction we get $v\alpha(r) = v\alpha(r')$, using Lemma 4 we conclude $v\alpha(t) = v\alpha(t')$.

Lemma 5 is proven in two steps. The first step is that Spoiler can detect in t and t' the subtrees that are to be deleted in order to build r and r'. The second step is that he is able to detect for each of those subtrees the label of the leaf that will be used in order to replace it in r and r'. The first step is a consequence of the fact that those subtrees are rooted with nodes whose label do not appear above. The second steps is a consequence of the fact that if Duplicator wins the EF game on two trees it means that they have the same type under all contexts that are strictly smaller than v relatively to $\leq_{\mathcal{R}}$ (induction hypothesis), which means that by definition they are equivalent under $\equiv$. Given these two steps, we are able to derive a winning strategy for Duplicator on r and r' from her strategy on t and t'. While technically more difficult those proofs are similar in spirit to the unranked case.

5.3 Third case

We suppose that we are not in one of the two previous cases, meaning that for all $b \in B$, $\alpha(b) \in stab_V(v)$, for all $c \in C$, $\alpha(c) \in stab_W(v)$ and $H - X = H_{min}$. If t is a leaf, so is t' and vice versa. Therefore we can suppose that t and t' are not leaves. In this case, t and t' have their types in H_{min} (they are X-trimmed). We prove that for all $h, g \in H_{min}$ $vg = vh$, in particular $v\alpha(t) = v\alpha(t')$ which concludes this case.

Let u_h and u_g be as defined in the Lemma 1. We assume that:

$$\begin{aligned} u_h &= \alpha(c_1 \diamond a_1)...(c_n \diamond a_n) \\ u_g &= \alpha(c_{n+1} \diamond a_{n+1})...(c_m \diamond a_m) \end{aligned}$$

We supposed that there were no unary symbols in order to simplify the expressions, this does not affect the proof. We also supposed that all subtrees off the main path were leaves, this is not restrictive since we supposed that all types were reachable with a leaf. By hypothesis, $\forall i$ $c_i \in stab_W(v)$, so we have an a'_i such that $\alpha(c_i \diamond a'_i) \in stab_V(v)$. We write:

$$\begin{aligned} w_h &= \alpha(c_1 \diamond a'_1)...(c_n \diamond a'_n) \\ w_g &= \alpha(c_{n+1} \diamond a'_{n+1})...(c_m \diamond a'_m) \end{aligned}$$

By construction of w_g and w_h:

$$\begin{aligned} vw_hw_hw_gw_g &\mathrel{\mathcal{R}} v \\ vw_hw_hw_gw_gv' &= v \text{ for some v'} \\ v(w_hw_hw_gw_gv')^\omega &= v \end{aligned}$$

Hence by definition of u_h:

$$v(w_hw_hw_gw_gv')^\omega(u_hw_hu_gw_gv')^\omega g = vh$$

Notice that we have $w_gw_g \dashv u_gw_g$ and $w_hw_h \dashv u_hw_h$, which allows us to use Equation (5):

$$\begin{aligned} vh &= v(w_hw_hw_gw_gv')^\omega u_gw_gv'(u_hw_hu_gw_gv')^\omega g \\ vh &= v(w_hw_hw_gw_gv')^\omega g \qquad \text{(by definition } \forall g' \ u_gg' = g\text{)} \\ vh &= vg \end{aligned}$$

This completes the proof.

6 Characterization of Δ_2

In this section we give a decidable characterization of Δ_2. Like the characterization for unranked trees in [3], our characterization use the piece relation. However, our definition of piece is more restrictive than the one used in [3], a piece of a tree is obtained by supressing some nodes and attaching the remaning nodes such that the ancestor relation is preserved. With this definition, every piece of an unranked tree is an unranked tree, but a piece of a binary tree need not be a binary tree. Therefore, our piece relation only considers binary pieces.

Definition 6. *Over an alphabet (A, B, C), we say that a tree t is a piece of a tree s if and only if there is an injective morphism of the nodes of t to the nodes of s that preserves labels and the ancestor relation, we write $s \preceq t$. This relation is naturaly extended on contexts (recall that a context is a tree with a special node $*$).*

Given $u, v \in V$ we say that u is a piece of v and write $u \preceq v$ if and only if there is some morphism $\alpha : (A, B, C)^{\Delta} \to (H, V, W)$ with $\alpha(p) = u$ and $\alpha(q) = v$ such that $p \preceq q$.

Theorem 2. *A binary tree language L is definable in Δ_2 if and only if it belongs to TDA and verifies the following identity:*

$$u^{\omega} = u^{\omega} v u^{\omega} \quad \forall u, v \in V \text{ such that } v \preceq u \tag{6}$$

Notice that it is sufficient to consider only identities (2) and (6), since identities (1) and (3) of TDA are direct consequences of (6).

Those identities are almost identical to the ones stated in the unranked characterization given in [3], the only difference being the restriction we made by considering only binary pieces in our piece relation $\preceq$. The impact of this restriction on the proof remains minimal in this case. However, technical differences related to the ranked tree structures do appear in the proof. Of the three logics we consider in this paper, Δ_2 is the one whose characterization has the closest proof to its unranked variant. The main differences reside in the common part with the $EF + F^{-1}$ logic in which we defined the *stab* sets and proved their properties. The ranked tree structure actually makes the rest of the proof technically simpler in this case.

Since the identities are almost identical to the ones of the unranked case, the proof of the easy direction of the Theorem is exactly the same as in the unranked case.

7 Characterization of Boolean Combinations of Σ_1

In this section we give a decidable characterization of boolean combinations of Σ_1. We use the notion of piece we defined when we stated the characterization of Δ_2.

Theorem 3. *A binary tree language L is definable in Σ_1 if and only if its syntactic binary algebra verifies the following properties:*

$$u^{\omega} v = u^{\omega} = v u^{\omega} \quad v \preceq u \tag{7}$$

for all $u, v \in V$.

Notice that once again the identity is very close to the one used in the characterization over unranked trees. Like the characterization of Δ_2, the only difference in the statement, is the restriction to binary pieces. However, while this restriction was anecdotic in Δ_2, it leads to many problems for this logic. An example is that for unranked trees, if you take a piece of a tree and split the tree into a context and another tree, the piece is also naturaly splited into a piece of the context and a piece of the new tree. This property which is extensively used in the proof of the unranked characterization is not true on ranked trees, which forces us to be careful when we split trees.

Since, the identity is similar to the one stated in [7] for unranked trees, the easy direction of Theorem 3 is identical, if a language is definable by a boolean combination of Σ_1 formulas, its syntactic algebra verifies (7).

8 Discussion

The statements of our characterizations for $EF + F^{-1}$, Δ_2 and Boolean Combinations of Σ_1 were limited to trees of rank two. However, these results could easily be extended to trees of rank k for a fixed k by extending the algebraic framework. For example, trees of rank three would be characterized by adding an other set representing tri-contexts, (contexts with three holes which are all siblings).

A relevant question would be to consider extensions of these logics. The only relation we considered is the order $<$, but what about other relations? We could add an order between siblings or a vertical successor relation, which is not definable from $<$ with the expressive power of the logics we considered.

References

1. M. Benedikt and L. Segoufin. Regular tree languages definable in FO. In *22nd Annual Symposium on Theoretical Aspects of Computer Science*, volume 3404 of *Lecture Notes in Computer Science*, pages 327–339, 2005.
2. M. Bojańczyk. Two-way unary temporal logic over trees. In *22nd IEEE Symposium on Logic in Computer Science*, pages 121–130, 2007.
3. M. Bojańczyk and L. Segoufin. Tree languages defined in first-order logic with one quantifier alternation. 2008.
4. M. Bojańczyk and I. Walukiewicz. Characterizing EF and EX tree logics. *Theoritical Compututer Science*, 358(2-3):255–272, 2006.
5. M. Bojańczyk and I. Walukiewicz. Forest algebras. In *Automata and Logic: History and Perspectives*, pages 107–132. Amsterdam University Press, 2007.
6. M. Y. Vardi K. Etessami and T. Wilke. First-order logic with two variables and unary temporal logic. In *12th IEEE Symposium on Logic in Computer Science*, pages 228–235, 1997.
7. L. Segoufin M. Bojańczyk and H. Straubing. Piecewise testable tree languages. 2008.
8. R. McNaughton and S. Papert. *Counter-Free Automata.* MIT Press, 1971.
9. M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.
10. D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *30th ACM Symposium on Theory of Computing*, pages 234–240, 1998.
11. T. Wilke. An algebraic characterization of frontier testable tree languages. *Theoretical Computer Science*, 154(1):85–106, 1996.