# Logical Definability and Query Languages over Ranked and Unranked Trees

MICHAEL BENEDIKT
Bell Laboratories
and
LEONID LIBKIN
University of Toronto
and
FRANK NEVEN
Hasselt University and Transnational University of Limburg

We study relations on trees defined by first-order constraints over a vocabulary that includes the *tree extension relation* $T \prec T'$, holding if and only if every branch of $T$ extends to a branch of $T'$, unary node-tests, and a binary relation checking if the domains of two trees are equal. We consider both ranked and unranked trees. These are trees with and without a restriction on the number of children of nodes. We adopt the model-theoretic approach to tree relations and study relations definable over the structure consisting of the set of all trees and the above predicates. We relate definability of sets and relations of trees to computability by tree automata. We show that some natural restrictions correspond to familiar logics in the more classical setting, where every tree is a structure over a fixed vocabulary, and to logics studied in the context of XML pattern languages. We then look at relational calculi over collections of trees, and obtain quantifier-restriction results that give us bounds on the expressive power and complexity. As unrestricted relational calculi can express problems complete for each level of the polynomial hierarchy, we look at their restrictions, corresponding to the restricted logics over the family of all unranked trees, and find several calculi with low ($\mathrm{NC}^1$) data complexity, while still expressing properties important for database and document applications. We also give normal forms for safe queries in the calculus.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation; F.1.3 [**Computation by Abstract Devices**]: Complexity Measures and Classes; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages; H.2.3 [**Database Management**]: Languages

General Terms: Languages, Theory

This paper is based on two conference papers that appeared in the Proceedings of the 17th and 18th Symposia on Logic in Computer Science: LICS 2002 by Benedikt and Libkin, and LICS 2003 by Libkin and Neven.
Author addresses. Michael Benedikt: Bell Laboratories, 263 Shuman Blvd, Naperville, IL 60566, USA. email: benedikt@research.bell-labs.com.
Leonid Libkin Department of Computer Science, 6 King's College Road, University of Toronto, Ontario M5S 3H5, Canada, email: libkin@cs.toronto.edu.
Frank Neven: Hasselt University and Transnational University of Limburg, School for Information Technology, Agoralaan, B-3590 Diepenbeek, Belgium, email: frank.neven@uhasselt.be

Additional Key Words and Phrases: Ranked Trees, Unranked Trees, Tree Automata, Model Theory, Query Languages

## 1. INTRODUCTION

Because much of computing practice involves the manipulation of tree structures, computer science abounds in formalisms for describing trees. Tree constraints and monadic-second order logic are two declarative approaches to specifying tree properties, while tree grammars, various flavors of tree automata and tree transducers are examples of more procedural formalisms. Naturally, an extensive literature exists comparing the expressive power of each of these formalisms (see [Comon et al. 1997; Thomas 1997]) and for translating between declarative formalisms and their procedural implementations. In particular, work on analyzing specifications of trees plays a significant role in program analysis [Aiken 1994; Müller et al. 2000], verification [Elgaard et al. 1998; Kupferman et al. 1996; Niwinski and Walukiewicz 1998], logic and constraint programming [Smolka 1995; Smolka and Treinen 1994] and linguistics [Dörre 1991; Kolb and Mönnich 1999].

In many applications, such as databases for tree-like data (as modeled, for example in XML) and program transformation, one does not wish to deal with properties of a single tree, but with relations between trees. For example, document databases store and manipulate large sets and relations of trees. A natural aim then is to use a tree constraint language to describe the properties of interest. Querying a tree-relation could then be seen as constraint solving – a model very much in line with the traditional declarative model for database processing. In the present paper, we will deal with constraint solving both on ranked trees (fixed number of children per node) and unranked trees.

In the literature, there are two different approaches to logical languages for defining strings and trees, depending on whether one characterizes a string or tree in terms of its internal structure or in terms of its relationship with other strings or trees. In the older and by now classic way of providing logical descriptions of regularity [Thomas 1997], individual strings and trees are represented as structures, and definability in a logic (e.g., first-order, monadic-second order) characterizes a class of strings/trees accepted by certain automata. This is the "finitary structure" or "internal" approach. In the other setting, one considers the family of all strings $\Sigma^*$ or the family of all trees, and defines some operations on them. This gives us a first-order structure $\mathfrak{M}$, and formulae in one free variable $\varphi(x)$ define sets of trees/strings $\{x \mid \mathfrak{M} \models \varphi(x)\}$. This "infinitary structure" approach was studied in [Benedikt et al. 2003; Bruyère et al. 1994; Blumensath and Gräel 2000; Khoussainov and Nerode 1995; Hodgson 1983] in the context of strings. Since the infinitary structure approach is attractive for dealing with relations, we will adopt it here in studying tree relations.

The infinitary approach led to the study of *automatic* structures, that is, structures in which every definable predicate can be represented by a finite automaton [Hodgson 1983; Khoussainov and Nerode 1995]. It was shown in [Blumensath and Gräel 2000] that there is a *universal* automatic structure over strings, that is, a structure $\mathfrak{S}$ such that every other automatic structure can be embedded into $\mathfrak{S}$. A finer study of string relations definable using the infinitary approach was conducted [Benedikt et al. 2003]; it takes the operations of the universal automatic structures as the primitive string relations, and studies string relations definable within it, and also within several of its reducts. [Benedikt et al. 2003] gives an overview of the expressiveness of several reducts of $\mathfrak{S}$, and uses these as the basis for relational calculi on string databases. In this paper we perform an analogous study for tree relations. We start with a "universal" structure for tree relations, consider definability within both this structure and its reducts, and then extend from formulae over tree tuples to relational algebra over tree relations.

The prior literature does consider properties of tree tuples and sets of trees, both in relation to logic programming and program analysis [Vorobyov and Voronkov 1998; Su et al. 2002] and with respect to database querying [Dantsin and Voronkov 2000]. Most of this work revolves around the use of equations and inequations among terms or trees. These works can be considered to take the infinitary approach to tree relations, studying collections of trees definable through formulae or constraints in *term* or *feature algebra*. Rephrased in the terminology of operations on labeled trees, term algebra corresponds to the set of trees with the operations of merging subtrees and extending branches by a single node. Term algebra, however, does not allow one to express the vertical ordering relationships among nodes that are important for many applications. For example, general regular tree patterns cannot be expressed using a term algebra; indeed, one cannot even express each query asking for the set of trees matching a regular expression pattern on one of its branches.

In this paper we deal with an infinitary structure different than term algebra. We investigate tree-tuple specifications given in a constraint formalism that includes *extension* relationships between trees: $T_1 \prec T_2$ iff $T_1$ is an initial subtree of $T_2$. This is the same as the standard subsumption ordering used for *feature trees* [Dörre 1991; Müller et al. 2001]: intuitively, it means that every branch of $T_1$ is also a branch of $T_2$. In the unranked case we split it into two relations: $\prec_{\rightarrow}$ (extend a tree by adding siblings), and $\prec_{\downarrow}$ (extend a tree by adding descendants). We denote the structure extended with the latter operation (and some others) by $\mathfrak{T}$ and $\mathfrak{T}^{\mathfrak{u}}$, for ranked and unranked trees, respectively.

In addition to dealing with a different structure on the set of trees, we deal with the *first-order theory* of our infinite structures, as opposed to just their equational theory (which is often the focus of investigation for tree algebras [Ésik 1998; Wilke 1996]). One of our key criteria is that the theory is decidable. This makes it impossible to combine the ordering $\prec$ with term algebra operations, since the resulting theory is known to be undecidable [Müller et al. 2001]. Instead, we introduce operations that allow us to extend trees at the leaves, rather than combine subtrees at the root. We shall call the set of trees with the extension relation (and a number of other operations to be introduced shortly) *tree algebra*, and the resulting formulae *tree formulae*. To get an idea of the combination of multi-tree constraints and single-tree formulae, we list below several properties that can be expressed in this algebra.

—*branch*$(T_1, T_2)$: $T_1$ is a single branch of $T_2$.

—*branch*$_i^a(T_1, T_2)$: $T_1$ and $T_2$ are single branches, and $T_2$ extends $T_1$ in direction $i$, labeling the leaf by $a$.

—$ab(T)$: Every node labeled $a$ in $T$ is followed by a node labeled $b$.

—$\neg\exists T_1\ (branch(T_1, T) \wedge ab(T_1))$: $T$ does not have a branch in which every node labeled $a$ is followed by a node labeled $b$.

We show that the formalism allows considerable expressive power, is closed under logical operations, and is decidable. After introducing the formalism, the first part of the paper is devoted to the synthesis of automata from formulae. We show that constraints given by tree formulae can be solved, with a multi-tree automaton that recognizes the defined collection of tree tuples. Since automata can be considered as a special kind of formula on finite structures, we are showing a correspondence between the infinite structure approach and the traditional finite-structure approach. We then consider, for the ranked case, restrictions of the tree algebra, possessing considerable expressive power (in fact, covering all the examples above) and yet having a simpler automaton construction. We present such a restriction, called *primal tree algebra* and denoted by $\mathfrak{T}_\mathfrak{p}$, and a corresponding class of automata, called *splitting automata*. We use these results to show separation between the two algebras.

We use these automata-theoretic tools to establish some properties of definable sets in the models. These properties will help to clarify the relationship of tree algebras to other formalisms. We start by investigating what sort of combinatorial objects can be defined within the model, and how the solution set of a formula $\varphi(\vec{x}, \vec{y})$ varies as the parameter $\vec{y}$ varies. Term algebras are *stable* in the model-theoretic sense (cf. [Hodges 1993]), implying that there is no definable linear order and the fibers of a formula cannot vary arbitrarily (i.e. the *VC-dimension* of definable families is bounded). We show that neither of these is true for the tree algebra – a linear order can be defined, even in the primal case, and the VC-dimension is unbounded. We also show that conversely there are properties of tree tuples expressible over term algebra that are not expressible via tree formulae.

Next, we look at restricted definability for both $\mathfrak{T}$, $\mathfrak{T}_\mathfrak{p}$, $\mathfrak{T}^\mathfrak{u}$, and $\mathfrak{T}^\mathfrak{u}_\mathfrak{p}$ and give some finer connections between these classes and those definable in the traditional (i.e. finitary model) setting in the following sense. W.r.t. the above mentioned restricted definability, it turns out that some natural sublogics of first-order over $\mathfrak{T}$ and $\mathfrak{T}^\mathfrak{u}$ correspond to logics that have been studied in connection with XML pattern languages, and are closely connected to monadic path logic [Thomas 1984].

If we have a formula $\varphi(T)$, its input, a tree $T$, can be viewed as a first-order structure, and hence we can consider the notion of *data complexity* of a formula. Reduction to tree automata and other techniques give us good bounds, from $\mathrm{AC}^0$ to $\mathrm{NC}^1$ to DLOGSPACE, on the data complexity of (restricted) logics over $\mathfrak{T}^\mathfrak{u}_\mathfrak{p}$ and $\mathfrak{T}^\mathfrak{u}$.

In the second part of the paper, we focus on database related aspects by moving from tree tuples to set or relations of tree-tuples. We do this by adding a finite relation on trees to our structures and considering queries against the resulting expanded structure. The notion of data complexity in this setting views the input as a database of trees. We show that the data-complexity of query evaluation is in the polynomial hierarchy, and find a class of queries for which it is $\mathrm{AC}^0$; this gives us some useful bounds on the expressive power. We then look at various relational calculi based on restricted logics from the first part of the paper, and find that they have low data complexity (e.g, $\mathrm{NC}^1$) while remaining quite expressive. We also address the question of characterizing *safe* queries, and give a range-restricted form that captures all safe queries with tree extension constraints.

*Organization.* In Section 2, we introduce the necessary background on strings, the logics $\mathcal{FO}$ and $\mathcal{MSO}$, and Ehrenfeucht-Fraïssé games. In Section 3 and 4, we discuss definability over ranked and unranked trees, respectively, and their relationship with automata and classical logics. In Section 4, we also discuss complexity. In Section 5, we consider relational calculi. We conclude in Section 6.

## 2. PRELIMINARIES

In this section, we give a brief overview of the two different approaches to logical definability over strings and we discuss Ehrenfeucht-Fraïssé games. To avoid confusion, when we deal with logics in the first setting, where strings and trees are represented as separate structures, we use calligraphic letters, e.g. $\mathcal{FO}$ (first-order), $\mathcal{MSO}$ (monadic second-order), etc. In the other setting, we normally consider first-order definability over some structure $\mathfrak{M}$, and then we write $\mathrm{FO}(\mathfrak{M})$. Throughout the paper, $\Sigma$ is a finite alphabet with at least two letters.

### 2.1 $\mathcal{FO}$ and $\mathcal{MSO}$.

We briefly discuss the syntax and semantics of $\mathcal{FO}$ and $\mathcal{MSO}$. We refer the unfamiliar reader to [Ebbinghaus and Flum 1999].

A vocabulary $\tau$ is a finite set of relation names and names of constants. Every relation name has an associated arity. A $\tau$-structure $\mathfrak{A}$ consists of a domain $A$, interprets every relation symbol in $\tau$ by a relation over $A$ of the right arity, and interprets every constant by an element in $A$.

An atomic $\mathcal{FO}$-formula is of the form $x = y$ or $R(\bar{x})$ where $x$ and $y$ are variables or constants, $\bar{x}$ is a sequence of variables and constants, and $R$ is a relation symbol. Every atomic $\mathcal{FO}$-formula is an $\mathcal{FO}$-formula. $\mathcal{FO}$-formulae are further closed under the boolean connectives ($\vee$, $\wedge$, $\neg$) and first-order quantification ($\exists x$). As usual, we denote by $\mathfrak{A} \models \varphi$ that $\mathfrak{A}$ is a model for $\varphi$.

$\mathcal{MSO}$ is $\mathcal{FO}$ extended with set quantification. More precisely, $\mathcal{MSO}$ is defined as follows. Every atomic $\mathcal{FO}$-formula is an atomic $\mathcal{MSO}$-formula. Additionally, let $X$ be a set variable and let $x$ be a first-order variable or a constant. Then, $X(x)$ is an atomic $\mathcal{MSO}$-formula expressing that $x$ belongs to the set $X$. In addition to $\mathcal{FO}$-formulae, $\mathcal{MSO}$-formulae are closed under set quantification. That is, $\exists X \varphi(X)$ is an $\mathcal{MSO}$-formulae where $X$ is a set variable and $\varphi(X)$ is an $\mathcal{MSO}$-formula. Over a structure $\mathfrak{A}$, such a formula expresses that there should be a set $X \subseteq A$ such that $\mathfrak{A} \models \varphi(A)$.

### 2.2 Strings.

As mentioned in the introduction, in the finitary approach to definability, a string $s = a_1 \ldots a_n$ over $\Sigma$ is represented as a structure $\langle \{1, \ldots, n\}, <, (O_a)_{a \in \Sigma} \rangle$, where $<$ is the usual ordering, and $O_a$ is interpreted as $\{i \mid a_i = a\}$. Classical results state that a set of strings is definable by an $\mathcal{MSO}$ ($\mathcal{FO}$) sentence iff it is regular (star-free, respectively), cf. [Thomas 1997].

The alternative approach to definability in the string setting is by using an infinite structure. In that case, we consider several operations on the set $\Sigma^*$ of all finite strings over $\Sigma$. One of them is the prefix relation $s_1 \leq s_2$ among strings. For each symbol $a \in \Sigma$ we have a function $l_a : \Sigma^* \to \Sigma^*$ that adds $a$ as the last symbol, that is, $l_a(s) = s \cdot a$. Finally, we have a relation $\mathrm{el}(s_1, s_2)$ which holds iff $|s_1| = |s_2|$; here $|s|$ is the length of the string $s$.

The structures most often considered in this setting are:

$$\begin{aligned} \mathfrak{S} &= \langle \Sigma^*, \leq, (l_a)_{a\in\Sigma}, \mathrm{el}\rangle, \quad \textit{and}, \\ \mathfrak{S}_{\mathfrak{p}} &= \langle \Sigma^*, \leq, (l_a)_{a\in\Sigma}\rangle. \end{aligned}$$

It is known that a subset of $\Sigma^*$ is $\mathrm{FO}(\mathfrak{S})$-definable iff it is regular [Blumensath and Gräel 2000; Bruyère et al. 1994], and it is $\mathrm{FO}(\mathfrak{S}_{\mathfrak{p}})$-definable iff it is star-free [Benedikt et al. 2003]. Furthermore, $\mathfrak{S}$ is the "universal" automatic structure, as any relation given by a finite automaton is FO-definable in $\mathfrak{S}$, and vice versa [Blumensath and Gräel 2000; Bruyère et al. 1994]. The index $\mathfrak{p}$ in $\mathfrak{S}_{\mathfrak{p}}$ stands for "primal".

To explain the notion of a *relation*, that is, a subset of $(\Sigma^*)^k, k > 1$, being definable by an automaton, let $\perp$ be a new symbol not in $\Sigma$, and $\Sigma_\perp = \Sigma \cup \{\perp\}$. Given a $k$-tuple of strings $\vec{s} = (s_1, \ldots, s_k)$, we define a string $[\vec{s}]$ over $\Sigma_\perp^k$, whose length is $\max_j |s_j|$, and whose $i$th symbol is $(s_1^i, \ldots, s_k^i)$, where

$$s_j^i \;=\; \begin{cases} \text{the } i\text{th symbol of } s_j, & \text{if } |s_j| \leq i \\ \perp, & \text{otherwise.} \end{cases}$$

In other words, we pad shorter strings with $\perp$ so that all strings are of the same length. We then say that a relation $R \subseteq (\Sigma^*)^k$ is regular if the language $\{[\vec{s}] \mid \vec{s} \in R\}$ is accepted by an automaton over $\Sigma_\perp^k$.

The structure $\mathfrak{S}$ is known as the *universal string-automatic structure*: any other structure $\mathfrak{M} = \langle \Sigma^*, \Omega\rangle$ which defines only regular relations is interpretable in $\mathfrak{S}$ [Blumensath and Gräel 2000].

### 2.3 Ehrenfeucht-Fraïssé games.

Most proofs in this paper make extensive use of Ehrenfeucht-Fraïssé (EF) games. The standard (FO) EF game is played on two structures, $\mathfrak{A}$ and $\mathfrak{B}$, of the same vocabulary, by two players, the spoiler and the duplicator. In round $i$, the spoiler selects a structure, say $\mathfrak{A}$, and an element $a_i$ of it; the duplicator responds by selecting an element $b_i$ of $\mathfrak{B}$. The duplicator wins in $k$-rounds if $\{(a_i, b_i) \mid i \leq k\}$ defines a partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$. We write $\mathfrak{A} \equiv_k \mathfrak{B}$ to denote this. A classical result states that $\mathfrak{A} \equiv_k \mathfrak{B}$ iff $\mathfrak{A}$ and $\mathfrak{B}$ agree on all FO sentences of quantifier rank up to $k$, cf. [Ebbinghaus and Flum 1999].

The game for $\mathcal{MSO}$ is similar, except that the players can play point moves, like in the FO game, and set moves, in which case the spoiler plays $A_i \subseteq \mathfrak{A}$ (or $B_i \subseteq \mathfrak{B}$), and the duplicator responds with $B_i \subseteq \mathfrak{B}$ (or $A_i \subseteq \mathfrak{A}$). The winning condition also requires that, in addition, the $\subseteq$ and $\in$ relations be preserved. Then we write $\mathfrak{A} \equiv_k^{\mathcal{MSO}} \mathfrak{B}$. Again, $\mathfrak{A} \equiv_k^{\mathcal{MSO}} \mathfrak{B}$ iff $\mathfrak{A}$ and $\mathfrak{B}$ agree on all $\mathcal{MSO}$ sentences of quantifier rank up to $k$, cf. [Ebbinghaus and Flum 1999].

We shall also make use of reduced EF games, which are helpful for logics with restricted quantification (for instance, to branches of trees). For that, let $\mathrm{FO}_V$

stand for FO with restricted quantification of the form $Qx \in V$, where $V$ is to be interpreted as a subset of the structure. If $V$ is interpreted as $V^{\mathfrak{A}}$ in $\mathfrak{A}$ and $V^{\mathfrak{B}}$ in $\mathfrak{B}$, then we write $(\mathfrak{A}, \vec{a}) \sim_k^V (\mathfrak{B}, \vec{b})$ if for every such restricted formula $\varphi(\vec{x})$ of quantifier rank $\leq k$, it is the case that $\mathfrak{A} \models \varphi(\vec{a})$ iff $\mathfrak{B} \models \varphi(\vec{b})$.

The $V$-restricted EF game is defined as the usual EF game except that moves can only come from $V^{\mathfrak{A}}$ and $V^{\mathfrak{B}}$. We write $(\mathfrak{A}, \vec{a}) \equiv_k^V (\mathfrak{B}, \vec{b})$ if the duplicator wins in $k$ rounds of the $V$-restricted game, starting from the position $(\vec{a}, \vec{b})$. Note that $\vec{a}$ and $\vec{b}$ do not have to come from $V^{\mathfrak{A}}$ and $V^{\mathfrak{B}}$. The proof of the following result mimics the usual proof for EF games, cf. [Ebbinghaus and Flum 1999].

LEMMA 2.1. $(\mathfrak{A}, \vec{a}) \equiv_k^V (\mathfrak{B}, \vec{b})$ *iff* $(\mathfrak{A}, \vec{a}) \sim_k^V (\mathfrak{B}, \vec{b})$.

In all the logics we consider, there will be finitely many formulae of quantifier rank $k$, up to logical equivalence. A *rank-$k$* type of a tuple $\vec{a}$ in $\mathfrak{A}$ is the set of all formulae $\varphi(\vec{x})$ of quantifier rank $\leq k$ such that $\mathfrak{A} \models \varphi(\vec{a})$. Given the above, there are only finitely many rank-$k$ types, and each of them is definable by a formula of quantifier rank $k$. For more information, see [Ebbinghaus and Flum 1999].

Types and games will be used heavily to prove expressibility of properties in a certain logic. The basic idea is as follows. Suppose we want to prove that quantification over $V$ is sufficient to express all FO-definable properties. For this, it will suffice to show that for every $k \geq 0$, there is an $m \geq 0$, such that $\mathfrak{A} \equiv_m^V \mathfrak{B}$ implies $\mathfrak{A} \equiv_k \mathfrak{B}$. Indeed, every property definable by an FO sentence of quantifier rank $k$ is a union of rank-$k$ types, and the above implication says that the $\equiv_m^V$ equivalence relation refines $\equiv_k$: hence, every rank-$k$ type is a union of $\equiv_m^V$ equivalence classes, and each one of those is definable by a formula with $V$-restricted quantification of quantifier rank $m$. We shall see this argument many times in the paper.

## 3. DEFINABILITY OVER RANKED TREES

### 3.1 Basic definitions

The trees we consider are based on two fixed alphabets: the alphabet for directions $\Delta$ of the form $\{1, \ldots, n\}$, and $\Sigma$ for node labeling. Unless explicitly stated otherwise, we assume $n > 1$. Most often we have $n = 2$ (binary trees). We write $s_1 \leq s_2$ if a string $s_1$ is a prefix of a string $s_2$. A *tree domain* is a prefix-closed finite subset $D$ of $\Delta^*$: $s_1 \in D$ and $s_2 \leq s_1$ imply $s_2 \in D$. A *tree* is a pair $T = (D, f)$ where $D \subset \Delta^*$ is a tree domain, and $f$ is a function from $D$ to $\Sigma$. We use $\text{dom}(T)$ to denote $D$. The set of all trees over $\Delta = \{1, \ldots, n\}$ and $\Sigma$ is denoted by $\text{TREE}_n(\Sigma)$. Note that $\text{TREE}_1(\Sigma)$ naturally corresponds to $\Sigma^*$; we shall say more about this correspondence later.

A node in a tree $T$ is a string $s \in D = \text{dom}(T)$, and $f(s)$ is its labeling. The root is the empty string $\epsilon$, and the leaves are those $s \in D$ such that $s$ is not a proper prefix of any other string in $D$. The set of leaves of $T$ is called the *frontier* of $T$ and is denoted by $\text{Fr}(T)$.

The *yield* of a tree $T$, $\text{yield}(T)$, is the string from $\Sigma^*$ read at $\text{Fr}(T)$. That is, if $\text{Fr}(T) = \{s_1, \ldots, s_k\}$ where $s_1, \ldots, s_k$ occur in lexicographic order, then $\text{yield}(T)$ is the string $f(s_1)f(s_2)\cdots f(s_k)$.

We now look at the operations (functions, predicates, and constants) on trees in the algebra we consider. The constants are $\epsilon_a, a \in \Sigma$, with domain $\{\epsilon\}$ labeled by

$a$. Unary term construction operators are as follows. Given $i \le n$ (direction) and $a \in \Sigma$, for $T = (D, f)$, $\mathrm{succ}_i^a(T) = (D', f')$ where $D' = D \cup \{s \cdot i \mid s \in \mathrm{Fr}(T)\}$, and $f'$ extends $f$ to $D'$ by $f'(s \cdot i) = a$ for each $s \in \mathrm{Fr}(T)$.

The basic binary relation – the one that gives the name to the algebra – is the extension order. Given two trees $T = (D, f)$ and $T' = (D', f')$, we write $T \preceq T'$ ($T'$ *extends* $T$) if $D \subseteq D'$ and $f$ is the restriction of $f'$ to $D$. Clearly it is a partial order. As usual $T \prec T'$ means $T \preceq T'$ and $T \neq T'$. We denote the greatest lower bound of $T$ and $T'$ by $T \sqcap T'$.

A tree $T$ is called a *branch* if $\mathrm{dom}(T)$ is linearly ordered by the prefix relation, that is, for any $s, s' \in \mathrm{dom}(T)$, either $s \le s'$ or $s' \le s$. Sometimes we use lowercase letters to denote branches. If $t$ is a branch and $t \preceq T$, we say that $t$ is a branch of $T$. If in addition $\mathrm{Fr}(t) \subseteq \mathrm{Fr}(T)$, then $t$ is called a maximal branch of $T$.

As we will see, first-order formulae over the above functions and predicates give us quite an expressive language. But to capture *all* properties of tree tuples that are implementable by tree automata, we will require an additional operation that allows us to compare trees based only on their domains, ignoring alphabet symbols. That is, for two trees $T, T'$, we write $T \approx_{\mathrm{dom}} T'$ iff $\mathrm{dom}(T) = \mathrm{dom}(T')$.

We now introduce the basic objects of our study. For each $n > 0$, we define the following:

The *Primal Tree Algebra* is the structure having the successor operations and the extension relation:

$$\mathfrak{T}_{\mathfrak{p}} \quad = \quad \langle \textsc{Tree}_n(\Sigma), \preceq, (\mathrm{succ}_i^a)_{i \le n, a \in \Sigma}, (\epsilon_a)_{a \in \Sigma} \rangle.$$

The *Tree Algebra* is the structure that in addition allows domain comparisons:

$$\mathfrak{T} \quad = \quad \langle \textsc{Tree}_n(\Sigma), \preceq, (\mathrm{succ}_i^a)_{i \le n, a \in \Sigma}, (\epsilon_a)_{a \in \Sigma}, \approx_{\mathrm{dom}} \rangle.$$

First-order formulae over $\mathfrak{T}$ are called *tree formulae.*

We can show that many of the basic tree operations and predicates are definable over $\mathfrak{T}_{\mathfrak{p}}$. There is a formula saying that $t$ is a branch: $\forall x, y\ (x \preceq t \wedge y \preceq t) \rightarrow (x \preceq y \vee y \preceq x)$. We denote this formula by $\eta(t)$. We also write $\eta(t, T)$ for $\eta(t) \wedge t \preceq T$ ($t$ is a branch of $T$) and $\eta_{\max}(t, T)$ for $\eta(t, T) \wedge \neg \exists t'\ (t \prec t' \wedge \eta(t', T))$ ($t$ is a maximal branch of $T$).

One can also see that $\mathrm{succ}_i^a$ could be defined in a number of different ways, for instance, as extending the leftmost branch, or the rightmost branch, or only extending branches. With each of those operations and $\preceq$ one would be able to define $\mathrm{succ}_i^a$. Furthermore, we can define $T \sqcap T'$ as the greatest lower bound of $T$ and $T'$ in $\preceq$ (which is a tree whose domain is the largest prefix-closed subset of $\mathrm{dom}(T) \cap \mathrm{dom}(T')$ on which $f$ and $f'$ coincide). We can also define a predicate $L_a$ on branches which tests if the leaf is labeled by $a$: $L_a(t) \equiv (t = \epsilon_a) \vee \exists t'(t' \prec t \wedge \bigvee_i t = \mathrm{succ}_i^a(t'))$.

*Complete vs incomplete domains.* In the literature (cf. [Comon et al. 1997; Thomas 1997]), it is common to define many concepts related to regular tree languages for trees over complete domains $D$: that is, for every $s \in D$, either all $s \cdot i, i \le n$ are in $D$, or none are.

We thus define a notion of completion of a tree: a *completion* of $T = (D, f)$ with respect to a symbol $a \in \Sigma$ as $T_a^c = (D', f')$ where $D'$ is the smallest complete domain that contains $D$, and $f'(s) = f(s)$ for $s \in D$, and $f'(s) = a$ for $s \in D' - D$. We often consider completions with some symbol $\perp \notin \Sigma$, and we shall write $\Sigma_\perp$ for $\Sigma \cup \{\perp\}$.

Completions $T_a^c$ are definable in $\mathfrak{T}_\mathfrak{p}$. Indeed, $T_a^c$ is the smallest, with respect to $\preceq$, tree $T' \succeq T$ such that for any nonmaximal branch $t$ of $T'$, and for each $i \leq n$, either $\mathrm{succ}_i^b(t) \preceq T$ for some $b \in \Sigma$, or $\mathrm{succ}_i^a(t) \preceq T'$. Clearly this is definable over $\mathfrak{T}_\mathfrak{p}$.

*$\mathcal{FO}$ and $\mathcal{MSO}$.* A tree $T = (D, f)$ in $\mathrm{Tree}_n(\Sigma)$ can be represented as a structure $\langle D, <, (Succ_i)_{1 \leq i \leq n}, (O_a)_{a \in \Sigma}\rangle$, where $<$ is the prefix relation on $D$, $Succ_i(s, s')$ iff $s' = s \cdot i$, and $O_a$ is interpreted as $\{s \mid f(s) = a\}$. We then consider $\mathcal{FO}$ and $\mathcal{MSO}$ defined over this vocabulary.

### 3.2 Tree algebra and tree automata

In this section, we show that sets definable by tree formulae are familiar objects: they are regular (recognizable) tree languages/relations. Furthermore, formulae over $\mathfrak{T}$ can be compiled into tree automata, and vice versa: this automata-theoretic characterization makes $\mathfrak{T}$ a natural model to work in.

Definition 3.1. Let $n \geq 1$. A *tree automaton* is a tuple $A = (Q, \Sigma_\perp, \delta_0, \delta_n, F)$ where $Q$ is a finite set of states; $F \subseteq Q$ is the set of final states; and, $\delta_0$ and $\delta_n$ are functions from $\Sigma_\perp$ and $Q^n \times \Sigma_\perp$ to $2^Q$, respectively.

A *run* of $A$ on a complete tree $T = (D, f)$ is a mapping $\lambda : D \to Q$ such that for every inner node $s \in D$ (with $n$ children), $\lambda(s) \in \delta_n(\lambda(s \cdot 1), \ldots, \lambda(s \cdot n), f(s))$ and for every leaf node $s$, $\lambda(s) \in \delta_0(f(s))$. A run is *accepting* if $\lambda(\epsilon) \in F$. The automaton *accepts* a tree when there is an accepting run.

A set of trees over complete domains is called regular if it is accepted by a tree automaton. Extending this to arbitrary domains, we say that a set $X \subseteq \mathrm{Tree}_n(\Sigma)$ is *regular* if the set $X_\perp^c = \{T_\perp^c \mid T \in X\}$ is accepted by a tree automaton.

The classical theorem of Doner [Doner 1970], and Thatcher and Wright [Thatcher and Wright 1968] relates definability and regularity.

Theorem 3.2. *A set of trees is $\mathcal{MSO}$-definable iff it is regular.*

We next define regular tree relations, that is, subsets of $\mathrm{Tree}_n(\Sigma) \times \ldots \times \mathrm{Tree}_n(\Sigma)$, following [Comon et al. 1997]. Let $\vec{T} = (T_1, \ldots, T_k)$ be a tuple of trees. We represent such a tuple as a tree $[\vec{T}]$ in $\mathrm{Tree}_n(\Sigma_\perp^k)$. Let $T_i = (D_i, f_i), i \leq k$. Then $[\vec{T}] = (D, F)$ where $D = D_1 \cup \ldots \cup D_k$ and for each $s \in D$, $F(s)$ is an element of $\Sigma_\perp^k$, that is, $F(s) = (a_1, \ldots, a_k)$ in which

$$a_i = \begin{cases} f_i(s) & \text{if } s \in D_i; \\ \perp & \text{otherwise.} \end{cases}$$

Over complete domains, the notion of recognizability says that the set of trees $[\vec{T}]$ is accepted by a tree automata over the alphabet $\Sigma_\perp^k$. To account for incomplete domains, we say that $X \subseteq \mathrm{Tree}_n(\Sigma)^k$ is *regular* iff the set $\{[\vec{T}]_{\boldsymbol{\perp}}^c \mid \vec{T} \in X\}$ is regular, that is, accepted by a tree automaton over the alphabet $\Sigma_\perp^k$. Here $\boldsymbol{\perp}$ stands for the $k$-tuple $(\perp, \ldots, \perp)$.

THEOREM 3.3. *(1) For any $k, n \geq 1$, a subset of* $\textsc{Tree}_n(\Sigma)^k$ *is FO($\mathfrak{T}$)-definable iff it is regular.*

*(2) For any $n \geq 1$, a subset of* $\textsc{Tree}_n(\Sigma)$ *is FO($\mathfrak{T}_\mathfrak{p}$)-definable iff it is regular.*

*Furthermore, for both (1) and (2), the translations between formulae and automata are effective.*

*Proof.* (1) It is easy to see that the relations

$$\{(T, T') \mid T \prec T'\},$$

$$\{(T, T') \mid T' = \mathrm{succ}_i^a(T)\}, \qquad a \in \Sigma, i \leq n,$$

$$\{(T, T') \mid T \approx_{\mathrm{dom}} T'\},$$

as well as the sets $\{\epsilon_a\}$, $a \in \Sigma$, are regular (by constructing tree automata for them). Furthermore, it is known that regular subsets of $\textsc{Tree}_n(\Sigma)^k$ are (effectively) closed under the Boolean operations and projection [Comon et al. 1997], which implies that every relation definable in $\mathfrak{T}$ is regular. The statement about the projection operations requires some care, since such closure is known for trees on complete domains [Comon et al. 1997]. Dealing with arbitrary domains is not a problem however. Suppose we have a formula $\varphi(\vec{T}, V)$, and let $R_\varphi = \{[\vec{T}, V]_\perp^c \mid \varphi(\vec{T}, V) \text{ holds}\}$. Then

$$\begin{array}{ll}
\{[\vec{T}]_\perp^c \mid \exists V \varphi(\vec{T}, V) \text{ holds}\} & = \\
\{[\vec{T}]_\perp^c \mid \exists V\ [\vec{T}, V]_\perp^c \in R_\varphi\} & = \\
\{[\vec{T}]_\perp^c \mid \exists V\ [[\vec{T}]_\perp^c, V_\perp^c] \in R_\varphi\} & = \\
\{[\vec{T}]_\perp^c \mid \exists V_\perp^c\ [[\vec{T}]_\perp^c, V_\perp^c] \in R_\varphi\} &
\end{array}$$

and now the closure under projection for trees over complete domains implies that $\exists V \varphi(\vec{T}, V)$ defines a regular relation.

To show that every regular relation is definable over $\mathfrak{T}$, we use equivalence of being regular and being definable in $\mathcal{MSO}$ (Theorem 3.2). We make use of a different vocabulary which enables us to get rid of first-order variables in formulae. In particular, $\mathcal{MSO}$ is defined over the structure whose universe is a (complete) tree domain, and the vocabulary is $\subseteq$, sng, $\textsc{Succ}_i$, $P_a, a \in \Sigma$, cf. [Thomas 1997]. We may assume that all variables are second-order and they range over the subsets of the domain of a tree. The meaning of the predicates is as follows: $\mathrm{sng}(X)$ means that $X$ is a singleton; $\textsc{Succ}_i(X, Y)$ means that $X, Y$ are singletons, and $Y$ is the $i$th successor of $X$ (that is, if $X = \{s\}$, then $Y = \{s \cdot i\}$); $P_a(X)$ means that every node in $X$ is labeled $a$. Furthermore, when $n > 2$, one can assume that for trees over complete domains, sets $X$ only range over antichains, that is, sets of strings in $\{1, \ldots, n\}^*$ such that none of them is a proper prefix of another [Potthoff and Thomas 1993; Thomas 1997] (this fragment of $\mathcal{MSO}$ is called *antichain* logic). As for $n = 1$ the theorem reduces to the string case, we can safely assume that $n > 1$.

Given a tuple of trees $\vec{T} = (T_1, \ldots, T_k)$, we have to show how to code antichain logic on $\mathrm{dom}([\vec{T}]_\perp^c)$ in FO over $\mathfrak{T}$. First, an antichain $X$ in $\mathrm{dom}([\vec{T}]_\perp^c)$ is coded by a tree $V$ whose leaves are precisely $X$ (the labeling could be arbitrary in this case; we assume that all nodes are labeled by a symbol $a$). One can test in $\mathfrak{T}$ if a tree $V$ codes an antichain in $\mathrm{dom}([\vec{T}]_\perp^c)$: this happens iff for each branch $v$ of $V$, there is a branch $t_i$ of $(T_i)^c_\perp$, $i \leq k$, such that $\mathrm{dom}(v) \subseteq \mathrm{dom}(t_i)$. Note that this is the same as saying that there is a branch $t_i$ of $(T_i)^c_a$, $i \leq k$, such that $\mathrm{dom}(v) \subseteq \mathrm{dom}(t_i)$. Since completions are definable over $\mathfrak{T}_\mathfrak{p}$, we just need to express $\mathrm{dom}(T) \subseteq \mathrm{dom}(T')$ for arbitrary $T, T'$. But this is simply $\exists T''\ (T'' \preceq T') \wedge (T \approx_{\mathrm{dom}} T')$.

Since antichains of $\mathrm{dom}([\vec{T}]^c_\perp)$ are coded by trees, it remains to show how to code the basic predicates of $\mathcal{MSO}$. For sng, this is simply the formula $\eta(\cdot)$ defining a branch. The successor relation $\textsc{Succ}(X,Y)$ is $\eta(V_X) \wedge \eta(V_Y) \wedge \bigvee_i (V_Y = \mathrm{succ}^a_i(V_X))$, where $V_X$ and $V_Y$ are trees coding $X$ and $Y$. The subset relation is translated into a formula saying that every maximal branch of $V_X$ is a maximal branch of $V_Y$. Finally, for each letter $\vec{a} \in \Sigma^k_\perp$, we must define $P_{\vec{a}}(X)$. If $V_X$ codes $X$, this amounts to saying that for every $i \leq k$, every leaf of $V_X$ is labeled by $a_i$ in $T_i$, for $\vec{a} = (a_1, \ldots, a_k)$. Note that each $a_i$ is either an element of $\Sigma$ or $\perp$. It suffices to define this for $V_X$ being a branch (then one could say that the leaves of all maximal branches are labeled by $\vec{a}$). Then the formula $\varphi_i(t)$ which is true for $V_X$ iff its leaf is labeled $a_i$ in $T_i$, is the following: for $a_i \in \Sigma$,

$$\varphi_i(t) \equiv \exists t' \ \Big(\eta(t', T_i) \wedge (t \approx_{\mathrm{dom}} t') \wedge \big((t' = \epsilon_{a_i}) \vee \bigvee_{j=1}^{n} (\exists t''\ (t' = \mathrm{succ}^{a_i}_j(t'')))\big)\Big),$$

and for $a_i = \perp$,

$$\varphi_i(t) \equiv \exists t' \exists t'' \ \big(\eta_{\max}(t', T_i) \wedge \eta(t'') \wedge t' \prec t'' \wedge t'' \approx_{\mathrm{dom}} t\big).$$

For (2), one direction has already been proved (that $\mathfrak{T}_\mathfrak{p}$ only defines regular relations). For the other direction, we use the same proof as for $\mathfrak{T}$, coding antichain logic. When $\vec{T}$ consists of a single tree $T$, $\mathrm{dom}([\vec{T}]^c_\perp) = \mathrm{dom}(T^c_\perp)$, that is, the completion of the domain of $T$, and an antichain $X$ is coded by a tree $V_X \preceq T^c_a$ whose leaves are exactly the nodes in $X$. This eliminates the need for the $\approx_{\mathrm{dom}}$ predicate, and one can easily modify the previous proof to code $\mathcal{MSO}$. $\Box$

Thus, definability of sets of trees in $\textsc{Tree}_n(\Sigma)$ is the same in $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$. For relations, however, definability is different, as we shall see in the next section.

Furthermore, $\mathfrak{T}$ plays the role of $\mathfrak{S}$ for trees: that is, it is a universal tree-automatic structures, as any other structure that only defines regular tree relations is interpretable in it.

*Consequences of the automata-theoretic representation.* First, we can show that in any formula $\varphi(\vec{T})$, quantifiers only need to range over a finite set. Given a tuple $\vec{T} \in \textsc{Tree}_n(\Sigma)^k$, let $\textsc{Tree}_n(\Sigma)|_{\mathrm{dom}(\vec{T})}$ be the set of all trees whose domain is a subset of $\bigcup_{T \in \vec{T}} \mathrm{dom}(T)$. By encoding the run of a tree automaton over $\mathfrak{T}$, one can see the following.

Corollary 3.4. *The finite set $\textsc{Tree}_n(\Sigma)|_{\mathrm{dom}(\vec{T})}$ is definable from $\vec{T}$ over $\mathfrak{T}$. Furthermore, every formula $\varphi(\vec{T})$ over $\mathfrak{T}$ is equivalent to a formula in which quantifiers range over $\textsc{Tree}_n(\Sigma)|_{\mathrm{dom}(\vec{T})}$.* $\Box$

The tree automata representation also gives us decidability and lower complexity bounds.

Corollary 3.5. *The theory of $\mathfrak{T}$ (and thus of $\mathfrak{T}_\mathfrak{p}$) is decidable. Decision procedures for both $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ have non-elementary complexity.*

*Proof.* Since translation from formulae to tree automata given in the proof of Theorem 3.3 is effective, and since emptiness of tree automata is decidable, we conclude that the theory of $\mathfrak{T}$ is decidable.

For the non-elementary lower bound, we code WS1S, the weak monadic theory of the successor relation on $\mathbb{N}$. We fix the alphabet $\Sigma$ to contain a single letter $a$. Each number $n$ is represented by a tree (in fact, a branch) $t_n$ where $t_0 = \epsilon_a$, and $\mathrm{dom}(t_n) = \{1^i \mid i \leq n\}$. A finite set $X = \{n_1, \ldots, n_k\}$, where $n_1 < \ldots < n_k$, is coded by a tree $T_X$ with $\mathrm{dom}(T_X) = \{1^i \mid i \leq n_k\} \cup \{1^{n_i}2 \mid i \leq k\}$. Over $\mathfrak{T}_\mathfrak{p}$, we can express that $t$ is a branch coding $n \in \mathbb{N}$ (by saying that every proper subbranch can only be extended by $\mathrm{succ}_1^a$), and that $T$ is a tree coding a finite set $X \subset \mathbb{N}$ (by saying that all its branches are either of the form $t_i$, or of the form $\mathrm{succ}_2^a(t_i)$). Since $\prec$ over $t_n$ expresses the usual ordering on $\mathbb{N}$, we see that the successor relation is definable. Finally, if $T$ is of the form $T_X$ and $t$ is of the form $t_n$, then $\mathrm{succ}_2^a(t) \prec T$ holds iff $n \in X$. This shows that any weak WS1S sentence $\Phi$ over $\langle \mathbb{N}, \mathit{succ} \rangle$ can be transformed, in polynomial time, into a sentence $\Phi_T$ over $\mathfrak{T}_\mathfrak{p}$ such that $\Phi$ is valid iff $\Phi_T$ is. Hence, the decision problem for $\mathfrak{T}_\mathfrak{p}$ is non-elementary. $\square$

### 3.3 Primal tree algebra and automata

The goal of this section is to compare the power of $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$. Since the previous results show that all regular sets of trees can be defined in $\mathfrak{T}_\mathfrak{p}$ (assuming $n > 1$: we discuss the special case $n = 1$ later), one might ask whether the domain-comparison operator $\approx_{\mathrm{dom}}$ is in fact already definable in $\mathfrak{T}_\mathfrak{p}$. We show here that $\approx_{\mathrm{dom}}$ is not expressible in the primal tree algebra, and thus $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ are different. We make the difference between the two models more concrete by presenting a restricted model of tree automata that exactly captures definability in $\mathfrak{T}_\mathfrak{p}$.

Let $\vec{T} = (T_1, \ldots, T_k)$ be a tuple of trees. We say that $t$ is a branch of $\vec{T}$ if $t$ is a branch of one of $T_i$s. In this case we also write $t \in \vec{T}$ for $\bigvee_i \eta(t, T_i)$. The automaton model is called a *splitting automaton*; such a device accepts or rejects a tuple $\vec{T}$ by defining a run over the set of all branches of $\vec{T}$ (as opposed to products of branches as for general tree-tuple automata). Intuitively, a splitting automaton has parallel threads moving up distinct branches of $\vec{T}$, with these threads merging at the point where the branches meet.

A *splitting-vector* is a function $V$ that assigns to each $(i, a) \in \Delta \times \Sigma$ a finite set of integers (which will be indices in the tuple of trees on which the automaton is running) in such a way that for any fixed $i$, the sets $V(i, a), a \in \Sigma$, are disjoint. The *range* of a splitting vector $V$ is $\mathit{range}(V) = \bigcup_{(i,a)\in\Delta\times\Sigma} V(i, a)$.

For a finite set $S$, an *$S$-splitting vector* is a finite set $V$ of tuples $(i, a, J, s) \in \Delta \times \Sigma \times \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \times S$, such that the projection on the first three components, denoted by $\mathit{Subset}(V) = \{(i, a, J) \mid \exists s\ (i, a, J, s) \in V\}$, is a splitting vector, and such that for every $(i, a)$ there is exactly one $(i, a, J, s) \in V$. We let $\mathit{State}(V)(i, a)$ be the unique $s \in S$ such that for some $J$, $(i, a, J, s) \in V$. For an $S$-splitting vector, we define the range of $V$ to be the range of the ordinary splitting-vector $\mathit{Subset}(V)$.

An *$S$-splitting rule* is a rule of the form

$$(I, s) \Leftarrow V,$$

where $I$ is a finite set of integers, $s \in S$, and $V$ is an $S$-splitting vector with

$range(V) \subseteq I$. Intuitively, a splitting vector describes for each successor $\mathrm{succ}_i^a(t)$ of a branch $t$ which components of $\vec{T}$ have that successor. An $S$-splitting vector describes the state of the machine on each of these successors of a branch, while a rule describes a bottom-up transition to a new state and new set of trees in $\vec{T}$.

An *acceptance partition* $F$ is a function assigning to each $a \in \Sigma$ a set $J \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N})$, while an *$S$-acceptance partition* $F$ is a function assigning to each $a \in \Sigma$ a pair $(J, s) \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N}) \times S$. For such a function $F$, we let $Subset(F)$ and $State(F)$ be the two projection functions: $Subset(F)$ is the function mapping $a \in \Sigma$ to the $J$ such that $(J, s) \in F(a)$, and $State(F)$ is the function that maps $a \in \Sigma$ to the $s$ such that $(J, s) \in F(a)$.

For a branch $t$, let $supp(t, \vec{T})$ be $\{i \mid t \in T_i\}$. Given $\vec{T}$ and a branch $t$ of $\vec{T}$, $v(t, \vec{T})$ is the splitting vector assigning to $(i, a)$ the set $\{j \mid \mathrm{succ}_i^a(t) \in T_j\}$. We let $v(\emptyset, \vec{T})$ be the acceptance partition assigning to each $a \in \Sigma$ the set $\{i \mid \epsilon_a \in T_i\}$.

A *$k$-dimensional (bottom-up) splitting automaton* $A$ is a tuple $(S, \delta, IR, \mathcal{F})$ where:

—$S$ a finite set (the *states* of $A$).

—$\delta$, the *transition relation*, is a finite set of $S$-splitting rules $(I, s) \Leftarrow V$ with $I \subseteq \{1, \ldots, k\}$, with every $S$-splitting vector $V$ contained in at least one rule.

—$IR$, the set of *initialization rules* is a set of rules of the form $(I, s) \Leftarrow$ , where $I \subseteq \{1, \ldots, k\}$, and each subset $I$ is in at least one rule of $IR$.

—A collection of $S$-acceptance partitions $\mathcal{F}$, the *accepting partitions of $A$*.

A bottom-up splitting automaton is *deterministic* if there is at most one initialization rule $(I, s) \Leftarrow$, for each $I \subseteq \{1, \ldots, k\}$, and in $\delta$ there is at most one rule with a given right-hand side.

A run $r$ of a $k$-dimensional bottom-up splitting automaton $A$ on $\vec{T}$ of size $k$ is a function from the branches of $\vec{T}$ to the states $S$ of $A$ such that:

—For every frontier branch $t$ (i.e. a branch such that no extension of $t$ is a branch of $\vec{T}$) with $supp(t, \vec{T}) = I$, $r(t)$ is a state $s$ such that $(I, s) \Leftarrow$ is in $IR$.

—For every non-frontier branch $t$, $r(t)$ is a state $s$ such that $(supp(t, \vec{T}), s) \Leftarrow V$ is in $\delta$, where $Subset(V) = v(t, \vec{T})$ and $State(V)(i, a) = r(\mathrm{succ}_i^a(t))$, whenever $\mathrm{succ}_i^a(t)$ is in $\vec{T}$.

A run is *accepting* if there is an $S$-acceptance partition $F \in \mathcal{F}$ with $Subset(F) = v(\emptyset, \vec{T})$ and $State(F)(a) = r(\epsilon_a)$ for each $a$ such that $\epsilon_a$ is in $\vec{T}$.

The following is an example of a 2-splitting automaton $A$ over alphabet $\Sigma = \{a, b\}$ $\Delta = \{1, 2\}$:

$$
\begin{array}{lcl}
\mathbf{IR} & = & \{(\{1\}, s_0) \Leftarrow \} \\
\delta & = & \{(\{1\}, s_0) \Leftarrow \{(1, b, \{1\}, s_0), (2, b, \{1\}, s_0)\} \\
 & & (\{1,2\}, s_1) \Leftarrow \{(1, b, \{1\}, s_0)\} \\
 & & (\{1,2\}, s_1) \Leftarrow \{(1, a, \{1,2\}, s_1)\} \\
\mathcal{F} & = & \{(a, \{1,2\}, s_1)\}
\end{array}
$$

The initial rule says what $A$ does on nodes that have no successors: these nodes must only be in the first tree, and on each such node we start in state $s_0$. The first rule in $\delta$ says that if a node $n_0$ has both of its successors $n_1$ and $n_2$ in the first tree with label $b$ and $A$ is in state $s_0$, then $n_0$ is only in the first tree, and $A$ remains in state $s_0$ on $n_0$. The second rule says that on a node $n_0$ with only a 1-successor

$n_1$, if $n_1$ is labeled with $b$ and is only in the first tree, and $A$ is in $s_0$ on $n_1$, then $n_0$ is in both trees, and $A$ is in state $s_1$ on $n_0$. The final rule says that if $n_0$ has only 1-successor $n_1$, $n_1$ is in both trees, and $n_1$ is labeled with $a$, then $n_0$ is in both trees, and the state of $A$ is unchanged in moving to $n_0$. The acceptance partition describes the requirement that the root node be common to both trees, labeled with $a$, and in state $s_1$. This automaton accepts a pair $(T_1, T_2)$ iff $T_1$ consists of a binary tree labeled with $b$ placed below a linear stem labeled with $a$, $T_2 \prec T_1$, and $T_2$ is exactly the linear stem.

In the special case of a 1-dimensional automaton, a splitting vector $V$ is just a collection of pairs $(i, a) \in \Delta \times \Sigma$, such that for each $i$ there is at most one $a$ with $(i, a) \in V$. A splitting vector can thus be identified with a function from $\Delta$ into $\Sigma_\perp$. An $S$-splitting vector likewise corresponds to a function from $\Delta$ into $\Sigma_\perp \times S$, and the set of rules of the automaton can be identified with a partial function mapping $f$ in $(\Sigma_\perp \times S)^\Delta$ to $s \in S$. Under this identification, a run of a splitting automaton over a single tree $T$ corresponds to a run of a standard bottom-up tree automaton over $T_\perp^c$, and hence the set of trees accepted by a 1-dimensional splitting automaton is regular.

Theorem 3.6. *A subset of* $\mathrm{TREE}_n(\Sigma)^k$ *is definable by a formula of* $\mathfrak{T}_\mathfrak{p}$ *iff it is accepted by a k-dimensional splitting automaton. Furthermore, the translations from formulae to automata and vice versa are effective.*

*Proof.* We first note the following (which can easily be proved via the usual subset construction):

Lemma 3.7. *For every bottom-up splitting automaton there is a deterministic bottom-up splitting automaton accepting the same sets of tuples of trees.* $\square$

Lemma 3.8. *The set of families of tree-sequences definable from bottom-up splitting automata is closed under complement, union, and projection.*

Proof of Lemma 3.8. Union is straightforward for deterministic automata. For complement, if we are given a deterministic bottom-up splitting automaton $A$, let $A'$ be formed by simply complementing the set of acceptance partitions of A. Then for any vector $\vec{T}$ it is clear that the unique run of $A$ on $\vec{T}$ accepts iff the unique run of $A'$ on $\vec{T}$ rejects.

We now turn to projection. Suppose we have a deterministic automaton $A = (S, \delta, IR, \mathcal{F})$ accepting sequences $T_1 \dots T_{k+1}$. We will create a new nondeterministic automaton $A' = (S', \delta', IR', \mathcal{F}')$ as follows.

Let $\delta_1$ be the set of rules such that every head is of the form $(\{k+1\}, s)$, $IR_1$ be all initialization rules of the form $(\{k+1\}, s) \Leftarrow$, and $S_1$ be all the states reachable from $IR_1$ by applying the rules in $\delta_1$. Then $A_1 = (S_1, \delta_1, IR_1, \emptyset)$ can be considered as a standard bottom-up tree automaton (that is, an automaton over a single tree).

The set of states $S'$ of $A'$ will be $S \times \{0, 1\}$. An $S'$-splitting vector $V'$ is *consistent* if for every $i \in \Delta$ there are no distinct $a_1, a_2 \in \Sigma$ and $s_1, s_2 \in S$ such that $State(V'(i, a_1)) = (s_1, 1)$ and $State(V'(i, a_2)) = (s_2, 1)$.

Similarly, an $S'$-acceptance partition $F'$ is consistent if there are no distinct $a_1, a_2 \in \Sigma$ and $s_1, s_2 \in S$ such that $State(F'(i, a_1)) = (s_1, 1)$ and $State(F'(i, a_2)) = (s_2, 1)$.

We construct $\delta'$ and $IR'$ iteratively. First, initialize both to $\emptyset$. Then, for all $R =$

$(I_h, s_h) \Leftarrow V$ in $\delta$, use the following algorithm. If $V$ contains $\{(i_0, a_0, \{k+1\}, s_0))$ with $s_0$ not in $S_1$, or if $I_h = \{k+1\}$, then skip to the next $R$. Otherwise, transform $R$ to the rule $R'$ obtained by:

—dropping every tuple of the form $(i, a, \{k+1\}, s)$ from $V$,

—replacing every element of $V$ of the form $(i, a, I \cup \{k+1\}, s)$ with $(i, a, I, (s, 1))$,

—replacing every element of $V$ of the form $(i, a, I, s)$ with $I \subset k$ with $(i, a, I, (s, 0))$,

—if $I_h = I \cup \{k+1\}$ for some $I \subset \{1 \dots k\}$, then replacing the head of $R$ with $(I_h, (s, 1))$, otherwise replacing the head of $R$ with $(I_h, (s, 0))$.

If the resulting rule $R'$ has empty righthand side, then we add $R'$ to $IR'$. Otherwise, if the right-hand side is consistent, we add it to $\delta'$.

Now for all initialization rules $R = (I_h, s_h) \Leftarrow$ in $IR$ use the following algorithm. If $I_h = \{k+1\}$, then skip to the next rule in $IR$. Otherwise transform $R$ to the rule $R'$ obtained by:

$$R' = \begin{cases} (I, (s, 1)) \Leftarrow , & \text{if } I_h = I \cup \{k+1\} \\ (I_h, (s, 0)) \Leftarrow , & \text{otherwise} \end{cases}$$

For every $S$-acceptance partition $f \in \mathcal{F}$, let $f'$ be the $S'$-acceptance partition such that:

—$f'(a) = (J - \{k+1\}, (s, 1))$ if $f(a) = (J, s)$ and $k+1 \in J$,

—$f'(a) = (J, (s, 0))$ if $f(a) = (J, s)$ and $k+1$ not in $J$.

We let $\mathcal{F}' = \{f' \mid f \in \mathcal{F}$ and $f'$ is consistent$\}$.

The above fulfills the definition of a splitting-automaton, except possibly for the "completeness" requirement that every splitting vector be included in the righthand side of some rule. We can complete the automaton with trivial rules that transition every "missing" vector to a sink state.

Claim 3.9. *$A' = (S', \delta', IR', \mathcal{F}')$ is an automaton accepting*

$$\{(T_1 \dots T_k) \mid \exists T_{k+1}\ (T_1 \dots T_{k+1}) \text{ is accepted by } A\}.$$

*Proof of claim 3.9.* In one direction, suppose $(T_1, \dots, T_{k+1})$ is accepted by $A$, with run being an accepting run. Take the function $r'$ mapping branches of $(T_1, \dots, T_k)$ to $S'$ as follows: $r'$ assigns $(s, 1)$ to branch $t$ iff $r$ assigns $s$ to $t$ and $t \in T_{k+1}$, while $r'$ assigns $(s, 0)$ to branch $t$ iff $r$ assigns $s$ to $t$ and $t \notin T_{k+1}$. Then one can check that $r'$ is a run of $A'$ accepting $(T_1, \dots, T_k)$.

Conversely, suppose that $r'$ is a run of $A'$ accepting $(T_1, \dots, T_k)$. Let $B_1$ be the set of branches $t$ such that $r'(t) = (s, 1)$ for some $s$. The consistency conditions can be used to verify that $B_1$ contains no two distinct branches with the same domain (if there were two such branches, either their meet is nonempty and we get a violation of the consistency conditions on transitions, or their meet is empty and we end up with a violation of the consistency condition on acceptance partitions).

Let $B_2$ be the set of branches $t$ in $B_1$ that are frontier branches of $\vec{T}$. Since the initial rules of $A'$ are constructed from either transition rules of $A$ or initialization rules of $A$, we see that if $r'(t) = (s, 1)$ for $t \in B_2$, we must have either $(s, J \cup \{k+1\}) \in IR$ for some $J$, or $(I, s) \Leftarrow V$ for some $s \in S$, $I \subset \{1 \dots k+1\}$, and $S$-splitting vector $V$ with $range(V) = \{k+1\}$ and $State(V) \subset S_1$. Let $B_3$ be the subset of $B_2$ for which the latter alternative holds, and for each $t$ in $B_3$ let $V_t$ be the $S$-splitting vector above. Let $Ext(t) = \{(i,a) \in \Delta \times \Sigma \mid \exists s_{t,i,a} \in S_1\ (i,a,\{k+1\},s_{t,i,a}) \in V_t\}$, and for each $(i,a) \in Ext(t)$ choose a witness $s_{t,i,a}$ and a tree $T_{t,i,a}$ such that the automaton $A_1$ run on $T_{t,i,a}$ ends in state $s_{t,i,a}$ (one exists from the definition of $S_1$ and $A_1$). Now let $T_{k+1} = B_1 \cup \bigcup_{t \in B_3} \bigcup_{(i,a) \in Ext(t)} T_{t,i,a}$. We can check that $(T_1, \ldots, T_{k+1})$ is accepted by $A$. This completes the proof of the claim.

Since the atomic formulae of $\mathfrak{T}_\mathfrak{p}$ are clearly definable by splitting automata, we have shown that every definable set of $\mathfrak{T}_\mathfrak{p}$ is definable by a bottom-up splitting automaton.

We now show the converse, that is, that every set accepted by a splitting automaton is definable in $\mathfrak{T}_\mathfrak{p}$.

Let $A = (S, \delta, IR, \mathcal{F})$ be a deterministic $k$-dimensional splitting automaton. We assume that for every state $s \in S$ there is exactly one $I \subset \{1 \ldots k\}$ such that $(I,s)$ occurs in a rule of $\delta$ or $IR$ . One can transform $A$ into an automaton with this property by creating one copy of each state for every subset of $\{1 \ldots k\}$ and then replacing every appearance of $(I,s)$ in a rule with $(I,s_I)$, where $s_I$ is the copy of $s$ for $I$. With the assumptions above, for $s \in S$, we let $I(A,s)$ denote the subset $I$ such that $(I,s)$ occurs in a rule of $A$.

Fix a non-empty subset of $\{1 \ldots k\}$, denoted by $I_0$. Given a sequence of trees $\vec{T}$, we let $\vec{T}^{=}(I_0) = \{t \mid t \text{ a branch of } T_i \leftrightarrow i \in I_0\}$. Note that every branch of $\vec{T}^{=}(I_0)$ is a branch of $\sqcap_{i \in I_0} T_i$.

An $I_0$-*component* is a maximally connected subset of $\vec{T}^{=}(I_0)$. (A collection $S$ of branches of a single tree is connected if the set $\{\mathrm{Fr}(t) \mid t \in S\}$ is a connected subset of the infinite tree $\Sigma^*$.) An $I_0$-root is a minimal element of an $I_0$ component. Given a branch $t \in \vec{T}$, we let $Comp(t, \vec{T})$ be the component of $t$ in $\vec{T}$.

For a state $s \in S$, we let $A(s, \vec{T}) = \{x \mid x \text{ is an } I_0\text{-root and the unique run of } A$ on $\vec{T}$ assigns $x$ to state $s\}$, where $I_0 = I(A,s)$.

Lemma 3.10. *For every state $s \in S$, there is a $\mathfrak{T}_\mathfrak{p}$ formula $\varphi(v_1 \ldots v_k, w)$ such that $\mathfrak{T}_\mathfrak{p} \models \varphi(\vec{T}, t)$ iff $t$ is a branch of $\vec{T}$ and $t$ is in $A(s, \vec{T})$.*

This lemma suffices, since the set of $\vec{T}$ accepted by the automaton is a boolean combination of the sets $\{\vec{T} \mid \epsilon_a \text{ is in } A(s, \vec{T})\}$ for $s \in S$ and $a \in \Sigma$.

Before we prove the Lemma, we need some preliminaries. For a tree $T$, let $Bd(T)$ be the set of branches $t$ of $T$ such that for some $i$, $t$ has no extension in direction $i$. The property $t \in Bd(T)$ is definable in $\mathfrak{T}_\mathfrak{p}$.

We note that the argument in Theorem 3.3 can be used to show the following.

Claim 3.11. *Let $\varphi(E_1, \ldots, E_j, d)$ be a formula of $\mathcal{MSO}$ (in the vocabulary given in the proof of Theorem 3.3), with $E_i$'s second-order variables and $d$ a first-order variable. Let $F(\vec{v}, w, y)$ be a $\mathfrak{T}_\mathfrak{p}$ definable predicate such that for every tree-sequence $\vec{T}$ and branch $t \in \vec{T}$, the set $F(\vec{T}, t) = \{t' \mid \mathfrak{T}_\mathfrak{p} \models F(\vec{T}, t, t')\}$ is a connected subset of some $T_i$ with $t \in F(\vec{T}, t)$. For $i \leq j$, let $G_i(\vec{v}, w, y)$ be definable predicates of $\mathfrak{T}_\mathfrak{p}$ such that $G_i(\vec{T}, t, t')$ implies $t' \in Bd(F(\vec{T}, t))$. For a tree-sequence $\vec{T}$ and branch $t$, we let $G_i(\vec{T}, t)$ be the set $\{\mathrm{Fr}(t') \mid \mathfrak{T}_\mathfrak{p} \models G_i(\vec{T}, t, t')\}$.*

*Let $\boldsymbol{F}(\vec{T}, t)$ be the tree whose branches are precisely those of $F(\vec{T}, t)$, considered* *as a structure of the $\mathcal{MSO}$ vocabulary for trees. Then the set*

$$\{(\vec{T},t) \ \mid \ \boldsymbol{F}(\vec{T},t) \models \varphi((F(\vec{T},t), G_1(\vec{T},t), \ldots G_j(\vec{T},t), \mathrm{Fr}(t)))\}$$

*is definable in $\mathfrak{T}_{\mathtt{p}}$.*

The claim is proved exactly as in Theorem 3.3, part b). Instead of coding quantification over subsets of a variable $v$, we have to code quantification over subsets of the connected set defined by definable predicate $F(\vec{v}, w, z)$. However, the elements $z$ of this set are all contained in some fixed $v_i$, and so this is the same as quantifying over subsets $S$ of $v_i$ such that all elements of $S$ happen to satisfy the predicate $F(\vec{v}, w, z)$. Once again, we can work in the completion $\boldsymbol{F}(\vec{T},t)^c_\perp$, and there it suffices to code quantification over antichains of $\boldsymbol{F}(\vec{v}, w, z)^c_\perp$, which we can do via subtrees of $v_i$. First-order quantification is mimicked by quantification over branches, and tests for membership in the additional predicates $E_i = G_i(\vec{T},t)$ can be performed by the assumption that $G_i$ is definable in $\mathfrak{T}_{\mathtt{p}}$.

We now prove Lemma 3.10 by induction on $\mid I_0 \mid = I(A, s)$. For $I_0 = \{i\}$, any run of $A$ that witnesses $t \in A(s, \vec{T})$ uses transitions and initial states involving only $\{i\}$. Hence this run is simply the run of a standard bottom-up automaton over the completion of $T_i$ with respect to $\perp$. There is thus an MSO statement $\varphi(E, d)$ (with $E$ a free second-order variable) such that for every branch $t$,

$$(Comp(t, \vec{T}), \mathrm{Fr}(t)) \models \varphi \ \text{ iff } \ \mathrm{Fr}(t) \in A(\vec{T}, s)$$

The $\varphi$ above simply asserts the existence of states witnessing the run. By the claim above this implies that the set $\{\vec{T}, t \mid t \prec \vec{T}(I_0) \text{ and } (\vec{T}(t), \mathrm{Fr}(t)) \models \varphi\}$ is definable in $\mathfrak{T}_{\mathtt{p}}$, and the base case now follows.

For the inductive step, fix $I_0$ and $s_0 \in S$ with $I(A, s_0) = I_0$. By induction, we know that for each $s$ with $|I(A, s)| < |I_0|$, there is a predicate defining $A(\vec{T}, s)$. Let $F(\vec{v}, w, y)$ hold iff $supp(w, \vec{v}) = supp(y, \vec{v}) = I_0$, $w$ is an $I_0$-root of $\vec{v}$, and $y$ is in the $\vec{w}$ component of $y$. That is, $F(\vec{T}, t)$ defines the $I_0$ component whose root is $t$. For any $S$-splitting vector $V$ such that $|Subset(V(i, a))| < |I_0|$ for each $i$ and $a$, we have a formula $G_V(\vec{v}, w, y)$ that holds of $\vec{T}, t, t'$ iff $t$ and $t'$ are branches in $F(\vec{T}, t)$, $t$ is in $Bd(F(\vec{T}, t))$ and $\mathrm{succ}^a_i(t)$ is assigned by the run of $A$ to $State(V(i, a))$ whenever $|supp(\mathrm{succ}^a_i(t), \vec{T})| < |I_0|$. Let $V_1, \ldots, V_l$ enumerate all these splitting vectors, with $G_{V_i}$ being the corresponding predicates. Let $G_e(\vec{v}, w, y)$ be an additional formula saying of $(\vec{T}, t, t')$ that $t'$ is a maximal branch of some $T_i$ that is in $Bd(F(\vec{T}, t))$.

We now apply the claim above, with $F(\vec{v}, w, y)$ and the predicates $G_{V_i}(\vec{v}, w, y)$ and $G_e$ above. The formula $\varphi(E_{V_1} \ldots E_{V_l}, E_e, d)$ will state of $T, d$ that $d$ is the root of $T$ and there is a partition of $Bd(T)$ into sets $\{B_s \mid s \in S, I(A, s) = I_0\}$ such that $d \in B_{s_0}$ and for all $q$ in $Bd(T)$, if $q \in B_s$ then either $(I_0, s) \in IR$ and $q \in E_e$ or there is $(I_0, s) \Leftarrow V$ in $\delta$ and an $S$-splitting vector $V_m$ with $|Subset(V_m(i, a))| < |I_0|$ for each $i$ and $a$, such that

—for every $(i, a, I', s') \in V$ with $|I'| < |I_0|$, $q \in E_{V_m}$ iff $V_m(i, a) = s'$, and
—for every $(i, a, I_0, s') \in V$, $\mathrm{succ}^a_i(q) \in B_{s'}$.

The formula $\varphi$ applied to the predicates $G_{V_i} : i \leq l$ and $F$ states exactly that we have a run of $A$ starting at the leaves of the boundary that assigns $d$ to $s_0$. Definability of $A(s_0, \vec{T})$ in $\mathfrak{T}_{\mathtt{p}}$ now follows from the claim.

This completes the proof of Lemma 3.10 and Theorem 3.6. □

*Consequences of the automaton representation for $\mathfrak{T}_\mathfrak{p}$.* The main consequence is that $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ are different: $\mathfrak{T}$ defines more subsets of $\mathrm{TREE}_n(\Sigma)^k$ for any $k \geq 2$.

COROLLARY 3.12. *If* $|\Sigma| > 1$, *then the predicate* $\approx_{\mathrm{dom}}$ *is not expressible in* $\mathfrak{T}_\mathfrak{p}$.

*Proof.* For each $n$, let $H_n$ and $K_n$ be distinct integers such that for every string automaton $SA$ with $n$ or fewer states, $SA$'s run on the string $b^{H_n}$ terminates in the same state as its run on the string $b^{K_n}$. Let $1_{a,H_n}$ consist of the tree with domain $1^{H_n}$ and which maps everything to $a \in \Sigma$ and similarly for $1_{a,K_n}$. Let $1_{b,H_n}$ consist of the tree with domain $1^{H_n}$ and which maps everything to $b \in \Sigma$, and similarly for $1_{b,K_n}$. Note that $1_{a,H_n} \approx_{\mathrm{dom}} 1_{b,H_n}$ but $1_{a,H_n} \approx_{\mathrm{dom}} 1_{b,K_n}$ does not hold. We will show that the pair $\mathrm{Same}_n = (1_{a,H_n}, 1_{b,H_n})$ is indistinguishable from the pair $\mathrm{Diff}_n = (1_{a,H_n}, 1_{b,K_n})$ on splitting automata with at most $n$ states. By Theorem 3.6 this implies that there can be no $\mathfrak{T}_\mathfrak{p}$ formula that distinguishes each $\mathrm{Diff}_n$ from $\mathrm{Same}_n$. Since there is such a formula in $\mathfrak{T}$, this separates $\mathfrak{T}$ from $\mathfrak{T}_\mathfrak{p}$.

Let $A$ be any deterministic 2-dimensional splitting automaton with $n$ or fewer states accepting $\mathrm{Same}_n$. Clearly, the run of $A$ on $\mathrm{Same}_n$ or $\mathrm{Diff}_n$ must begin with initial rules of the form $(\{1\}, s) \Leftarrow$ and $(\{2\}, s) \Leftarrow$, make transitions $\delta$ only of the form $(\{1\}, s) \Leftarrow \{(1, a, \{1\}, s')\}$ and $(\{\{2\}, s) \Leftarrow \{(1, b, \{2\}, s')\}$ (for brevity, we omit all tuples of the form $(i, a, \emptyset, s) \mid i \in \Delta, a \in \Sigma$ in the righthand side of rules). Furthermore, if the run assigns $\epsilon_a$ to state $s_a$ and $\epsilon_b$ to $s_b$ on some run, then there must be an acceptance partition in $\mathcal{F}$ of the form $\{(a, \{1\}, s_a), (b, \{2\}, s_b)\}$.

Let $s_a$ and $s_b$ be the states assigned to $\epsilon_a$ and $\epsilon_b$ by the accepting run of $A$ on $\mathrm{Same}_n$, and let $s'_a$ and $s'_b$ be the two states reached in $A$ on the run on $\mathrm{Diff}_n$. Clearly $s'_a = s_a$, since the components extending $\epsilon_a$ in each pair are identical.

We can form a string automaton $SA_b$ whose states are the states of $A$, initial states $\{s \mid (\{2\}, s) \Leftarrow \quad \in IR\}$. and whose transitions $(s, a, s')$ are exactly those such that $(\{2\}, s) \Leftarrow \{(1, b, \{2\}, s')\} \in \delta$. That is, a run of $SA_b$ corresponds exactly to a run of $A$ on the second component.

Based on this correspondence, we see that the run of $SA_b$ on $b^{H_n}$ reaches the state $s_b$. But then by construction, the run of $SA_b$ on the string $b^{K_n}$ also reaches state $s_b$. But using the correspondence above in reverse, we see that the run of $SA$ on $\mathrm{Diff}_n$ assigns $s_b$ to $\epsilon_b$. Hence we see that $SA$ accepts $\mathrm{Diff}_n$ also, and therefore $SA$ cannot distinguish $\mathrm{Diff}_n$ from $\mathrm{Same}_n$, which completes the proof. □

Another consequence is that in $\mathfrak{T}_\mathfrak{p}$, quantification can be restricted to a finite set. For a symbol $a \in \Sigma$, let $\mathrm{TREE}_n(\Sigma)|_{\vec{T}^c_a}$ be the set of all trees $T_0$ such that every branch of $T_0$ is a branch of a tree $T^c_a, T \in \vec{T}$. Note that this set is definable from $\vec{T}$ over $\mathfrak{T}_\mathfrak{p}$. By encoding the run of a splitting automaton, we can show:

COROLLARY 3.13. *For any* $a \in \Sigma$, *and every* $\mathfrak{T}_\mathfrak{p}$ *formula* $\varphi(\vec{T})$, *there is an equivalent formula in which the quantifiers range over* $\mathrm{TREE}_n(\Sigma)|_{\vec{T}^c_a}$. □

### 3.4 Expressibility and model theory

We now study model theoretic properties and the expressive power of the tree algebras. We start with two results that show a sharp contrast between $\mathfrak{T}$ and $\mathfrak{T}_\mathfrak{p}$ on the one hand, and term algebra on the other hand. As mentioned in the introduction, term algebras have a particularly well behaved model theory: they are stable (which implies that no linear order is definable), have finite VC dimension, and admit quantifier-elimination. In contrast to this, we can show:

PROPOSITION 3.14. *There is a linear ordering on* $\text{TREE}_n(\Sigma)$ *definable in* $\mathfrak{T}_\mathfrak{p}$.

*Proof.* We show how to define a linear order $<_b$ on branches. Once this is done, by a standard construction for lifting a linear order to finite subsets of a set, the formula $\varphi_<(T_1, T_2)$ given by

$$\forall t_1 \Big( \big(\eta_{\max}(t_1, T_1) \wedge \neg\eta_{\max}(t_1, T_2)\big) \;\rightarrow\; \exists t_2 \big(\eta_{\max}(t_2, T_2) \wedge \neg\eta_{\max}(t_2, T_1) \wedge t_1 <_b t_2\big)\Big)$$

defines a linear ordering on $\text{TREE}_n(\Sigma)$. The ordering $t <_b t'$ is defined as follows. First we fix an ordering $<$ on $\Sigma$. Assume that $t \neq t'$. If $t \prec t'$, then $t <_b t'$. Otherwise, let $t_0 = t \sqcap t'$ and let $s = \text{Fr}(t_0)$. Assume that $s \cdot i \in \text{dom}(t) \cap \text{dom}(t')$ for some $i \leq n$. Then $t <_b t'$ iff $f(s \cdot i) < f'(s \cdot i)$. Otherwise, we have $i \neq j$ such that $s \cdot i \in \text{dom}(t)$ and $s \cdot j \in \text{dom}(t')$. Then $t <_b t'$ iff $i < j$.

Clearly for $t \neq t'$ either $t <_b t'$ or $t' <_b t$. It remains to prove transitivity. For this, 12 cases arise, and it is routine to check all of them. □

Since there is no linear order definable in a term algebra, the operations of $\mathfrak{T}_\mathfrak{p}$ clearly cannot be first-order definable in term algebra.

Let $join_a(T_1, T_2)$ be the binary tree whose root is labeled $a$, and whose left and right subtrees are $T_1$ and $T_2$ respectively. The structure $\langle \text{TREE}_n(\Sigma), \prec, (join_a)_{a \in \Sigma}, (\epsilon_a)_{a \in \Sigma}\rangle$ corresponds to the first-order theory of $FT_\leq$ constraints over feature trees studied in [Müller and Niehren 2000; Müller et al. 2001]. Since that theory is known to be undecidable [Müller et al. 2001], we obtain:

PROPOSITION 3.15. $join_a$ *is not definable in* $\mathfrak{T}$. □

One can also give a direct proof, showing that with $join_a$ one can define predicates not recognizable by tree automata.

Thus, first-order logic over $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ is incomparable with first-order logic over term and feature algebras [Müller et al. 2001; Dantsin and Voronkov 2000].

*VC dimension.* We now show that the behavior of definable families in $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ formula is not as tame as in a term algebra. A standard notion of tameness for definable families is given by the concept of *VC dimension* (cf. [Anthony and Biggs 1992]) (also known as not having the independence property [Laskowski 1992]). Given an infinite set $X$ and a family $\mathcal{C}$ of its subsets, $X$ *shatters* a finite set $F \subset X$ if $\{F \cap C \mid C \in \mathcal{C}\}$ is the powerset of $F$. The VC dimension of $\mathcal{C}$ is the maximum size of a finite set it shatters (or $\infty$ if arbitrarily large finite sets are shattered). Given a structure $\mathfrak{M}$ over a set $U$ and a formula $\varphi(x_1, \ldots, x_m, y_l, \ldots, y_k)$ in the language of $\mathfrak{M}$, the definable family given by $\varphi$ is $\{\ \{\vec{a} \in U^m \mid \mathfrak{M} \models \varphi(\vec{a}, \vec{b})\} \ \mid\ \vec{b} \in U^k\}$. We say that $\mathfrak{M}$ has *finite VC dimension* if every definable family has finite VC dimension. From the point of view of learning theory, this means that every definable family is PAC-learnable [Anthony and Biggs 1992]. Finite VC dimension also implies strong bounds on the expressiveness of relational query languages [Benedikt and Libkin 2000; Benedikt et al. 2003]. It turns out that the presence of the extension predicate $\prec$, prevents $\mathfrak{M}$ from having finite VC dimension.

PROPOSITION 3.16. *For any nonempty* $\Sigma$*, the structure* $\langle \text{TREE}_2(\Sigma), \preceq\rangle$ *does not have finite VC dimension. Hence* $\mathfrak{T}_\mathfrak{p}$ *and* $\mathfrak{T}$ *do not have finite VC dimension.* □

| Model | $\mathfrak{S}_{\mathtt{p}}$ | $\mathfrak{T}_{\mathtt{p}}$ | $\mathfrak{S}$ | $\mathfrak{T}$ |
|---|---|---|---|---|
| one-dimension definable sets | *-free | regular | regular | regular |
| arbitrary definable sets | counter-free prefix automata | splitting automata | regular | regular |
| VC dimension | finite | infinite | infinite | infinite |
| FO theory | decidable | decidable | decidable | decidable |
| (weak) MSO theory | decidable | undecidable | undecidable | undecidable |

Fig. 1. Comparison of string and tree models

*Proof.* The formula $\eta_{\max}(t, T)$ saying that $t$ is maximal branch of $T$ is definable in $\langle \text{TREE}_2(\Sigma), \preceq\rangle$, and it is very easy to see that it has infinite VC dimension. □

*Model theory of strings vs. model theory of trees.* We remarked before that if the alphabet of directions has a unique element, then trees over such alphabet are naturally associated with strings: that is, trees in $\text{TREE}_1(\Sigma)$ are in 1-1 correspondence with $\Sigma^*$. The analogs of $\mathfrak{T}$ and $\mathfrak{T}_{\mathtt{p}}$ under this correspondence are $\mathfrak{S}$ and $\mathfrak{S}_{\mathtt{p}}$, respectively. Figure 1 summarizes results on $\mathfrak{T}, \mathfrak{T}_{\mathtt{p}}$, and their string analogs $\mathfrak{S}$ [Bruyère et al. 1994; Blumensath and Gräel 2000] and $\mathfrak{S}_{\mathtt{p}}$ [Benedikt et al. 2003]. It turns out that model-theoretically $\mathfrak{S}$ and $\mathfrak{T}$ are rather close, but $\mathfrak{S}_{\mathtt{p}}$ and $\mathfrak{T}_{\mathtt{p}}$ are very different. The first line of the table of Figure 1 talks about one-dimensional definable sets, that is, subsets of $\Sigma^*$ or $\text{TREE}_n(\Sigma)$. The second line is about arbitrary definable sets. The automaton construction for $\mathfrak{S}_{\mathtt{p}}$ is a counter-free restriction of regular prefix automata of [Angluin and Hoover 1984]. No similar restriction (either to first-order definable or star-free tree languages [Thomas 1984]) is possible over $\mathfrak{T}_{\mathtt{p}}$, since even in the one-dimensional case, arbitrary regular tree languages are definable.

The third line compares VC dimension of definable families. The fourth and the fifth line compare first-order and weak monadic second-order theories; the undecidability of the latter for $\mathfrak{T}_{\mathtt{p}}$ is stated below; the routine proof is omitted.

PROPOSITION 3.17. *One can code arithmetic $(+, \times)$ in weak MSO over $\mathfrak{T}_{\mathtt{p}}$. Consequently, the weak MSO theory of $\mathfrak{T}_{\mathtt{p}}$ (and even the weak EMSO theory) is undecidable.* □

### 3.5 Branch quantification

In this section we present two results on restrictions of FO that capture some familiar subclasses of regular tree languages: those definable in $\mathcal{FO}$ and the *monadic chain logic* [Thomas 1987]. Let $T = (D, f)$ be a tree. Then a set $V \subseteq D$ is a chain iff for all $s, s' \in V$, $s \leq s'$ or $s' \leq s$. Monadic chain logic, denoted $\mathcal{MSO}^{\text{chain}}$, is the restriction of $\mathcal{MSO}$ where set quantification is restricted to sets that are chains. We write $\exists^{\text{chain}} X$ to emphasize that $X$ can only be interpreted by a chain.

The restrictions on $\text{FO}(\mathfrak{T}_{\mathtt{p}})$ and $\text{FO}(\mathfrak{T})$ are based on quantification over branches. That is, quantification is not over arbitrary trees, but only over branches. These restrictions will be denoted by $\text{FO}_\eta(\mathfrak{T}_{\mathtt{p}})$ and $\text{FO}_\eta(\mathfrak{T})$. We also use the notation $\exists^\eta$ and $\forall^\eta$ to emphasize that quantification is over branches. Clearly, these can be defined in $\mathrm{FO}(\mathfrak{T}_\mathfrak{p})$ and $\mathrm{FO}(\mathfrak{T})$.

We state the results for definable subsets of $\mathrm{T}\textsc{ree}_n(\Sigma)$. For $\mathfrak{T}$, the results straightforwardly extend to definability of relations. A $k$-ary relation $R$ on trees over $\Sigma$ is definable in a logic like $\mathcal{FO}, \mathcal{MSO}$, etc, if the set $\{[\vec{T}] \mid \vec{T} \in R\}$ of trees over $\Sigma_\perp^k$ is definable in the logic.

THEOREM 3.18. *a) A set of* $\mathrm{T}\textsc{ree}_n(\Sigma)$*-trees is* $\mathcal{FO}$*-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$*-definable.*

*b) A set of* $\mathrm{T}\textsc{ree}_n(\Sigma)$*-trees is* $\mathcal{MSO}^{\mathrm{chain}}$*-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T})$*-definable.*

*Proof.* We recall that $\eta(t)$ and $\eta(t, T)$ express that $t$ is a branch and that $t$ is a branch of $T$, respectively.

We start by proving part a).

We first show that every $\mathcal{FO}$-definable set of trees is also $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$-definable. To this end, we associate with every node in $T$ the branch ending in that node. So, we prove by induction on the structure of an $\mathcal{FO}$-formula $\varphi(x_1, \dots, x_k)$ that there is a $\mathrm{FO}(\mathfrak{T}_\mathfrak{p})$-formula $\varphi'(T, T_1, \dots, T_k)$ such that

$$T \models \varphi(v_1, \dots, v_k) \text{ iff } \mathfrak{T}_\mathfrak{p} \models \varphi'(T, T_1, \dots, T_k),$$

where quantification in $\varphi'$ is restricted to branches and $T_i$ is the branch of $T$ ending in $v_i$. The proof proceeds by induction on the structure of $\mathcal{FO}$-formulae.

If $\varphi$ is of the form $\mathrm{succ}_i(x_1, x_2)$, then $\varphi(T, T_1, T_2)$ is

$$\eta(T_1, T) \wedge \eta(T_2, T) \wedge \bigvee_{a \in \Sigma} \mathrm{succ}_i^a(T_1) = T_2.$$

If $\varphi$ is of the form $O_a(x_1)$ then $\varphi'(T, T_1)$ is

$$\eta(T_1, T) \wedge L_a(T_1),$$

where $L_a(T_1)$ is the formula $\epsilon_a = T_1 \vee \exists^\eta T_2 \bigvee_{i \le n} \mathrm{succ}_i^a(T_2) = T_1$ expressing that the last vertex of $T_1$ is labeled with $a$.

If $\varphi$ is of the form $x_1 \le x_2$ then $\varphi'(T, T_1, T_2)$ is

$$\eta(T_1, T) \wedge \eta(T_2, T) \wedge T_1 \preceq T_2.$$

Closure under boolean connectives is immediate. Quantification is restricted to branches of $T$.

We now prove that every $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$-definable set of trees is also $\mathcal{FO}$-definable. First, we recall and introduce some new notation. We shall write $T_1 \equiv_k T_2$ if the duplicator can win the $k$-round game on $T_1$ and $T_2$ considered as first-order structures, $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_k^\eta (\mathfrak{T}_\mathfrak{p}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$, and $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$ in which in addition all moves are branches of $T_1$ and $T_2$ respectively. Here, $\eta(T_1, T_2)$ denotes the set of branches occurring in $T_1$ or $T_2$. We will show that for every $k > 0$, there exists $m > 0$, such that:

(1) $T_1 \equiv_k T_2$ implies $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$, and

(2) $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_m^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$ implies $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_k^\eta (\mathfrak{T}_\mathfrak{p}, T_2)$.

Combining these, we shall conclude that each $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$-definable set of ranked trees is a union of $\mathcal{FO}$ types and thus is $\mathcal{FO}$-definable.

For 1), to produce the winning strategy for $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$, every time a branch, say $t_1 \in T_1$ is played, the duplicator pretends that the spoiler plays the leaf of that branch in the game on $T_1$ and $T_2$, finds the response (which is a node in $T_2$), and plays the branch that ends in that node. It is easy to see that this guarantees a win.

For 2), we let $m = k + 3$. The general idea, here and for many other proofs, is as follows. For each move by the spoiler in the branch-restricted game on $(\mathfrak{T}_\mathfrak{p}, T_1)$ and $(\mathfrak{T}_\mathfrak{p}, T_2)$, the duplicator constructs a certain branch in $\eta(T_1, T_2)$ and pretends that the spoiler makes a move with that branch in the $\eta(T_1, T_2)$-restricted game on $(\mathfrak{T}_\mathfrak{p}, T_1)$ and $(\mathfrak{T}_\mathfrak{p}, T_2)$, where, by the assumption, he has a winning strategy. He then uses this winning strategy to find his response, and uses this response to construct the response in the branch-restricted game on $(\mathfrak{T}_\mathfrak{p}, T_1)$ and $(\mathfrak{T}_\mathfrak{p}, T_2)$. The chosen bound $m$ ensures that this strategy guarantees him a win.

We now describe the strategy. Let $T' \cap T''$ be the tree whose domain is the largest prefix-closed subset of domains of $T'$ and $T''$ on which labellings from $T'$ and $T''$ coincide (and the labeling of $T' \cap T''$ is inherited from $T'$ and $T''$).

We denote moves in the game on $(\mathfrak{T}_\mathfrak{p}, T_1)$ and $(\mathfrak{T}_\mathfrak{p}, T_2)$ by $t_{1i}$ and $t_{2i}$, respectively, for round $i \leq k$. Furthermore, after each round $i$, we also have a configuration $((t'_{11}, \ldots, t'_{1i}), (t'_{21}, \ldots, t'_{2i}))$ corresponding to the winning strategy in $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_m^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$.

The game proceed as follows. Suppose in round $i$ ($1 \leq i \leq k$), the spoiler plays $t_{1i}$ in $(\mathfrak{T}_\mathfrak{p}, T_1)$ (the case of a move in $(\mathfrak{T}_\mathfrak{p}, T_2)$ is symmetric). The duplicator computes $t'_{1i}$ as $t_{1i} \cap T_1$ and augments the configuration $((t'_{11}, \ldots, t'_{1\,i-1}), (t'_{21}, \ldots, t'_{2\,i-1}))$ to $((t'_{11}, \ldots, t'_{1i}), (t'_{21}, \ldots, t'_{2i}))$ where $t'_{2i}$ is the response to $t'_{1i}$ according to the winning strategy $(\mathfrak{T}_\mathfrak{p}, T_1) \equiv_m^{\eta(T_1,T_2)} (\mathfrak{T}_\mathfrak{p}, T_2)$.

Let $s$ be the leaf node of $t'_{1i}$, and let $s \cdot j_1$, $s \cdot j_1 \cdot j_2$, …, $s \cdot j_1 \cdot j_2 \cdots j_p$ be the remaining nodes in $t_{1i}$. Let $s'$ be the leaf of $t'_{2i}$; we construct $t_{2i}$ by adding the nodes $s' \cdot j_1$, …, $s' \cdot j_1 \cdot j_2 \cdots j_p$ to $t'_{2i}$, where for each $q \leq p$, $s' \cdot j_1 \cdot j_2 \cdots j_q$ has the same label as $s \cdot j_1 \cdot j_2 \cdots j_q$ in $t_{1i}$.

It remains to show that this strategy guarantees a win for the duplicator. First, we argue that

$$t'_{1i} = t_{1i} \cap T_1 \quad \text{and} \quad t'_{2i} = t_{2i} \cap T_2. \tag{1}$$

For $t'_{1i}$ the claim holds by construction. For $t'_{2i}$, suppose the leaf node of $t'_{1i}$ is $s$, its child in $t_{1i}$ is $s \cdot j$ and labeled by $a$, and $s'$ is the leaf of $t'_{2i}$. Then the node $s' \cdot j$ is either not in $T_2$ or is not labeled $a$. Indeed, for each $a$ and $j$ there is a quantifier-rank 3 formula $\alpha_j^a(t, T)$ that holds iff $t$ is a branch of $T$ whose leafs $j$th successor is either not in the domain of $T$, or is not labeled $a$. As $m$ was chosen to be $k + 3$, $\alpha_j^a(t'_{1i}, T_1)$ holds iff $\alpha_j^a(t'_{2i}, T_2)$ does. Hence, $t'_{2i} = t_{2i} \cap T_2$.

We show that $\{(t_{1j}, t_{2j}) \mid 1 \leq j \leq i)\}$ defines a partial isomorphism from $(\mathfrak{T}_\mathfrak{p}, T_1)$ to $(\mathfrak{T}_\mathfrak{p}, T_2)$ under the assumption that both $\{(t_{1j}, t_{2j}) \mid 1 \leq j \leq i - 1)\}$ and $\{(t'_{1j}, t'_{2j}) \mid 1 \leq j \leq i)\}$ define a partial isomorphism from $(\mathfrak{T}_\mathfrak{p}, T_1)$ to $(\mathfrak{T}_\mathfrak{p}, T_2)$. By (1), we get that $t_{1i}$ is a branch of $T_1$ iff $t_{2i}$ is a branch of $T_2$. By construction, $t_{1i} = \epsilon_a$ iff $t_{2i} = \epsilon_a$ for all $a \in \Sigma$. Suppose $t_{1i} \preceq t_{1j}$, for some $j$. We distinguish two cases: (a) $t_{1i}$ is a branch of $T_1$. Then $t_{2i}$ is a branch of $T_2$. Hence, $t'_{1i} \preceq t'_{1j} \preceq t_{1j}$

and $t_{2i} = t'_{2i} \preceq t'_{2j} \preceq t_{2j}$. (b) $t_{1i}$ is not a branch of $T_1$. From (1) and $t_{1i} \preceq t_{1j}$ it follows that $t'_{1i} = t'_{1j}$. So, $t'_{2i} = t'_{2j}$ and $t_{2i} \preceq t_{2j}$.

Finally, let $succ^a_i(t_{1i}, t_{1j})$. We distinguish two cases. (a) $t_{1i}$ is a branch of $T_1$. Clearly, if $t_{1j}$ is also a branch of $T_1$, then $succ^a_k(t_{2i}, t_{2j})$. Otherwise, as before, $t_{1i} = t'_{1j}$ and therefore $t_{2i} = t'_{2j}$. Moreover, if $s_1$ and $s_2$ are the leaf node of $t'_{1j}$ and $t'_{2j}$, respectively, then $t_{1j}$ and $t_{2j}$ are obtained by adding the nodes $s_1 \cdot k$ and $s_2 \cdot k$ with label $a$ to $t'_{1j}$ and $t'_{2j}$, respectively. So, $succ^a_k(t_{2i}, t_{2j})$. (b) $t_{1i}$ is not a branch of $T_1$. Then, $t'_{1i} = t'_{1j}$ and $t'_{2i} = t'_{2j}$. Let $s_1$ and $s_2$ be the leaf node of $t'_{1j}$ and $t'_{2j}$, respectively. Let $s_1 \cdot j_1$, $s_1 \cdot j_1 \cdot j_2$, $\ldots$, $s_1 \cdot j_1 \cdot j_2 \cdots j_p$ be the remaining nodes in $t_{1i}$. Then, $s_1 \cdot j_1$, $s_1 \cdot j_1 \cdot j_2$, $\ldots$, $s_1 \cdot j_1 \cdot j_2 \cdots j_p$, $s_2 \cdot j_1 \cdot j_2 \cdots j_p \cdot k$ are the remaining nodes in $t_{1j}$. However, with $s_1$ replaced with $s_2$, these are also the remaining nodes of $t_{2i}$ and $t_{2j}$. Hence, $succ^a_k(t_{2i}, t_{2j})$.

The remaining dual cases are straightforward adaptations.

We next prove part b).

We start by showing that every $\mathcal{MSO}^{\text{chain}}$-definable set of trees is $\text{FO}_\eta(\mathfrak{T})$-definable. We only discuss the cases not addressed in the previous proof: chain quantification $\exists^{\text{chain}} X \ldots$ and $X(x)$. With every set variable $X$ we associate a variable $T_X$ which will be interpreted by a branch where positions are labeled with zeros and one's and whose domain is a subset of $T$. The idea is that 1-labeled nodes are in the set $X$ while 0-labeled nodes are not. Note that we tacitly assume that $0, 1 \in \Sigma$. More precisely, the formula $X(x_1)$ is equivalent to

$$\eta(T_1, T) \wedge \exists^\eta T'(T' \approx_{\text{dom}} T_1 \wedge T' \preceq T_X \wedge L_1(T_1)).$$

The formula $\exists^{\text{chain}} X \ldots$ is equivalent to

$$\exists^\eta T_X \exists T'(\eta(T', T) \wedge T' \approx_{\text{dom}} T_X \wedge \neg\exists^\eta T''(T'' \preceq T_X \wedge \neg L_0(T'') \wedge \neg L_1(T'')) \ldots).$$

Here, $L_a$ with $a = 0, 1$ is the formula defined in (1).

The proof that shows that every $\text{FO}_\eta(\mathfrak{T})$-definable set of trees is $\mathcal{MSO}^{\text{chain}}$-definable proceeds along the same lines as the proof for $\mathfrak{T}_\mathfrak{p}$. We write $T_1 \equiv_k^{\mathcal{MSO}^{\text{chain}}} T_2$ if the duplicator can win the $k$-round $\mathcal{MSO}^{\text{chain}}$ game on $T_1$ and $T_2$; $(\mathfrak{T}, T_1) \equiv_k^\eta (\mathfrak{T}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$; and, $(\mathfrak{T}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$ in which in addition all moves are branches whose domains are contained in $\text{dom}(T_1)$ or $\text{dom}(T_2)$, respectively. We will show that for every $k > 0$, there exists $m, l > 0$, such that:

(1) $T_1 \equiv_l^{\mathcal{MSO}^{\text{chain}}} T_2$ implies $(\mathfrak{T}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}, T_2)$, and

(2) $(\mathfrak{T}, T_1) \equiv_m^{\eta(T_1,T_2)} (\mathfrak{T}, T_2)$ implies $(\mathfrak{T}, T_1) \equiv_k^\eta (\mathfrak{T}, T_2)$.

The result will follow from these.

Assume, without loss of generality, that $\Sigma = \{a, b\}$. For the first item, we choose $l$ to be $2k+3$. Each move in the game $(\mathfrak{T}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}, T_2)$, say $t_1$ with $\text{dom}(t_1) \subseteq \text{dom}(T_1)$, is mimicked by two moves in the game $T_1 \equiv_l^{\mathcal{MSO}^{\text{chain}}} T_2$: in the first move, a chain $C_a$ corresponding to $t_1$'s nodes labeled $a$ is played, and in the second move, it is the chain $C_b$ corresponding to $t_1$'s nodes labeled $b$. Assume that the duplicator responses according to the winning strategy are $C'_a$ and $C'_b$. The fact that the game can continue for an extra 3 moves guarantees that $C'_a \cap C'_b = \emptyset$, and that $C'_a \cup C'_b$ encodes a branch in $T_2$. So, there is a branch $t_2$ with $\mathrm{dom}(t_2) \subseteq \mathrm{dom}(T_2)$ whose nodes labeled $a$ (respectively, $b$) are precisely those in $C'_a$ (respectively, $C'_b$). This branch is the duplicator's response to $t_1$. It is routine to show that the duplicator wins the game.

The proof for the second item mimics that for $\mathfrak{T}_\mathfrak{p}$, except that $t'_{1i}$ is chosen to be the maximum subbranch of $t_{1i}$ such that $\mathrm{dom}(t'_{1i}) \subseteq \mathrm{dom}(T_1)$. The rest of the proof is almost identical to the one for $\mathfrak{T}_\mathfrak{p}$. □

## 4. DEFINABILITY OVER UNRANKED TREES

### 4.1 Basic definitions

In dealing with unranked trees we have no a priori bound on the number of children. Hence we use consecutive positive integers to enumerate children of a node (that is, there cannot be a second child without a first). That is, we define an *unranked tree domain* as a prefix-closed finite subset $D$ of $\mathbb{N}_+^*$ (finite strings of positive integers) such that $s \cdot i \in D$ implies $s \cdot j \in D$ for all $j \leq i$. An *unranked* $\Sigma$*-tree* is a pair $T = (D, f)$ where $D$ is an unranked tree domain and $f : D \to \Sigma$. The set of all unranked trees over $\Sigma$ is denoted by $\textsc{Utree}(\Sigma)$. Note that in contrast with ranked trees, an unranked one cannot have an $i$-th child without having an $(i-1)$-th one.

An unranked tree $T = (D, f)$ is represented as a first-order structure $\langle D, <_{\mathrm{pre}}, <_{\mathrm{sib}}, (O_a)_{a\in\Sigma}\rangle$, where $O_a$ is as before, $<_{\mathrm{pre}}$ is the prefix relation (we added 'pre' for clarity) and $<_{\mathrm{sib}}$ is the order relation on siblings ($s \cdot i <_{\mathrm{sib}} s \cdot j$ for all $s \cdot i, s \cdot j \in D$, $i, j \in \mathbb{N}$, and $i < j$). Denote the above vocabulary by $\nu_\Sigma$. Apart from $\mathcal{FO}$ and $\mathcal{MSO}$ over $\nu_\Sigma$, we also consider the extension of the monadic chain logic to unranked trees $\mathcal{MSO}_{\rightarrow}^{\downarrow}$ and a logic $\mathcal{FOREG}$, which is the extension of $\mathcal{FO}$ with horizontal and vertical regular path expressions. These will be defined in Section 4.3.

We now look at the operations on unranked trees. The relation $\prec$ is defined as before. However, using just this relation would force us to introduce infinitely many successor relations in the vocabulary. To keep the vocabulary finite, we split $\prec$ into two relations:

*Extension on the right* $\prec_\rightarrow$. That is, only younger siblings can be added. Formally, for $T_1 = (D_1, f_1)$ and $T_2 = (D_2, f_2)$, $T_1 \prec_\rightarrow T_2$ if $T_1 \prec T_2$ and for every $s \cdot i \in D_2 - D_1$, there is $j < i$ such that $s \cdot j \in D_1$.

*Extension down* $\prec_\downarrow$. That is, descendant of leaves can be added. Formally, $T_1 \prec_\downarrow T_2$ if $T_1 \prec T_2$ and for every $s \in D_2 - D_1$, $s' <_{\mathrm{pre}} s$ for some leaf $s'$ of $T_1$.

Likewise we define $\preceq_\rightarrow$ and $\preceq_\downarrow$. Clearly, $\preceq = \preceq_\rightarrow \circ \preceq_\downarrow$.

Let $L_a(T)$ hold, for $a \in \Sigma$, iff the rightmost node in $T$ (that is, the largest one in the lexicographic ordering) is labeled $a$. As before, $T_1 \approx_{\mathrm{dom}} T_2$ holds iff $D_1 = D_2$.

With these operations, we now define two structures:

$$\begin{aligned} \mathfrak{T}^\mathfrak{u} &= \langle \textsc{Utree}(\Sigma), \prec_\rightarrow, \prec_\downarrow, (L_a)_{a\in\Sigma}, \approx_{\mathrm{dom}}\rangle \\ \mathfrak{T}^\mathfrak{u}_\mathfrak{p} &= \langle \textsc{Utree}(\Sigma), \prec_\rightarrow, \prec_\downarrow, (L_a)_{a\in\Sigma}\rangle. \end{aligned}$$

We define unranked branches as trees satisfying $\eta(T)$. Similarly to the ranked case, we define the logics $\mathrm{FO}(\mathfrak{T}^\mathfrak{u}_\mathfrak{p})$, $\mathrm{FO}(\mathfrak{T}^\mathfrak{u})$, $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u}_\mathfrak{p})$ and $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u})$.

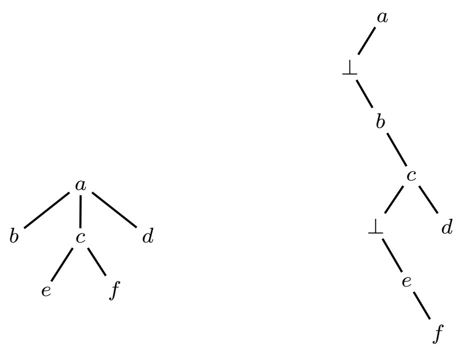


Fig. 2. A tree $T$ and $\mathcal{R}(T)$.

Note that although the predicates and operations for ranked and unranked trees differ slightly, it is not difficult to see that when restricted to $\text{Tree}_n(\Sigma)$, the logics over $\mathfrak{T}^{\mathfrak{u}}$ and $\mathfrak{T}$, and $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and $\mathfrak{T}_{\mathfrak{p}}$, are equivalent.

Next, we extend the notion of tree automata to the unranked case, following [Brüggemann-Klein et al. 2001].

Definition 4.1. An *unranked tree automaton* is a tuple $A = (Q, \Sigma, \delta, F)$ where $Q$ is a finite set of states; $F \subseteq Q$ is the set of final states; and, $\delta$ is a function from $Q \times \Sigma$ to $2^{Q^*}$ assigning a regular language of strings over $Q$ to every pair $(q, \sigma)$.

A *run* of $A$ on a tree $T = (D, f)$ is a mapping $\lambda : D \to Q$ such that for every node $s \in D$ with $n$ children, $\lambda(s \cdot 1) \cdots \lambda(s \cdot n) \in \delta(\lambda(s), f(s))$. Note that for leaf nodes $s$ this implies that $\varepsilon \in \delta(\lambda(s), f(s))$. A run is *accepting* if $\lambda(\epsilon) \in F$. The automaton *accepts* a tree when there is an accepting run.

A set of unranked trees is *regular* if there is an unranked tree automaton accepting it. A relation $R \subseteq \text{UTREE}(\Sigma)^k$ is regular if so is the set $\{[\vec{T}] \mid \vec{T} \in R\}$ over $\text{UTREE}(\Sigma^k_\perp)$.

Similarly to the ranked case, $\mathcal{MSO}$ over the vocabulary $(<_{\text{sib}}, <_{\text{pre}}, (O_a)_{a \in \Sigma})$ defines precisely the regular unranked tree languages [Neven and Schwentick 2002].

*Encoding unranked trees.* This encoding is basically the same as Rabin's encoding of S$\omega$S into S2S, cf. [Rabin 1969; Börger et al. 1997]. Given a string $n_1 \cdots n_k$ of positive integers,

$$\mathcal{R}(n_1 \cdots n_k) \;=\; 01^{n_1}01^{n_2}0 \cdots 01^{n_k} \;\in\; \{0,1\}^*.$$

Also, $\mathcal{R}(\epsilon) = \epsilon$.

Given a tree $T = (D, f) \in \text{UTREE}(\Sigma)$, we define $\mathcal{R}(T) = (D', f') \in \text{TREE}(\Sigma_\perp)$ as follows:

—$D'$ is the prefix-closure of $\mathcal{R}(D) = \{\mathcal{R}(s) \mid s \in D\}$;
—If $s \in D$, then $f'(\mathcal{R}(s)) = f(s)$; if $s' \in D' - \mathcal{R}(D)$, then $f'(s') = \perp$.

We give an example in Figure 2.

It turns out that unranked regular and ranked regular are in fact similar notions. For a set of unranked trees $X$, let $\mathcal{R}(X) = \{\mathcal{R}(T) \mid T \in X\}$. Since $\mathcal{MSO}$ over unranked trees can rather easily be encoded in $\mathcal{MSO}$ over ranked trees and vice

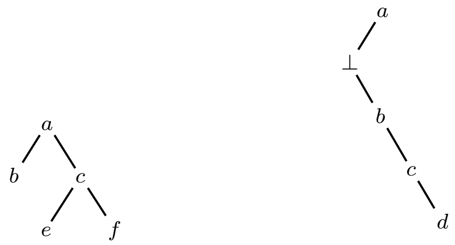


Fig. 3. A branch of $T$ with endpoint labeled $f$ and a branch of $\mathcal{R}(T)$ with endpoint labeled $d$.

versa, and since the image of $\mathcal{R}(\cdot)$ is $\mathcal{MSO}$-definable, we have the following folklore result (for an explicit proof, see [Suciu 2002]):

PROPOSITION 4.2. *For any finite alphabet $\Sigma$ and $X \subseteq$ UTREE$(\Sigma)$, $X$ is regular iff $\mathcal{R}(X)$ is regular.*

We conclude with a note on branches. Recall that a branch $t$ is defined by the formula $\eta(t) = \forall x, y\ (x \preceq t \wedge y \preceq t) \rightarrow (x \preceq y \vee y \preceq x)$. Branches are a crucial notion in many of the restricted logical formalisms. It is therefore worthwhile to point out that this notion differs for ranked and unranked trees. Indeed, a branch in an unranked tree includes the left siblings of every node in the branch, while a branch in a ranked tree does not. The latter follows immediately from the definition of branches. We give an example in Figure 3.

### 4.2 Basic definability results

In this section, we link definability over unranked trees in the structures $\mathfrak{T}^{\mathsf{u}}_{\mathsf{p}}$ and $\mathfrak{T}^{\mathsf{u}}$ to regular languages.

Here, and throughout the paper, we make use of the following lemma. The proof is by induction on FO$(\mathfrak{T}^{\mathsf{u}}_{\mathsf{p}})$ and FO$(\mathfrak{T}^{\mathsf{u}})$-formulae.

LEMMA 4.3. *Let $\mathfrak{M}^{\mathsf{u}}$ be either $\mathfrak{T}^{\mathsf{u}}_{\mathsf{p}}$ or $\mathfrak{T}^{\mathsf{u}}$, over* UTREE$(\Sigma)$*, and let $\mathfrak{M}$ be the corresponding ranked tree model, over* TREE$(\Sigma_\perp)$*. Then*

—*for every FO$(\mathfrak{M}^{\mathsf{u}})$-formula $\varphi$ there exists an* FO$(\mathfrak{M})$*-formula $\varphi'$ such that $\mathfrak{M}^{\mathsf{u}} \models \varphi(\vec{T}) \Leftrightarrow \mathfrak{M} \models \varphi'(\mathcal{R}(\vec{T}))$. Moreover, if $\varphi$ is an* FO$_\eta(\mathfrak{M}^{\mathsf{u}})$*-formula, then $\varphi'$ can be chosen to be an* FO$_\eta(\mathfrak{M})$*-formula.*

—*for every* FO$(\mathfrak{M})$*-formula $\varphi$ there exists an FO$(\mathfrak{M}^{\mathsf{u}})$-formula $\varphi'$ such that $\mathfrak{M} \models \varphi(\mathcal{R}(\vec{T})) \Leftrightarrow \mathfrak{M}^{\mathsf{u}} \models \varphi'(\vec{T})$. Moreover, if $\varphi$ is an* FO$_\eta(\mathfrak{M})$*-formula, then $\varphi'$ can be chosen to be an* FO$_\eta(\mathfrak{M}^{\mathsf{u}})$*-formula.*

Using this Lemma, together with the encoding $\mathcal{R}(\cdot)$, we show that Theorem 3.3 extends from ranked to unranked trees.

THEOREM 4.4. *Let $X$ be a subset of* UTREE$(\Sigma)$*. Then the following are equivalent:*

(1) *$X$ is definable in $\mathfrak{T}^{\mathsf{u}}_{\mathsf{p}}$;*
(2) *$X$ is definable in $\mathfrak{T}^{\mathsf{u}}$;*
(3) *$X$ is regular.*

*Furthermore, for $k > 1$ and $R \subseteq \mathrm{UTREE}(\Sigma)^k$, $R$ is $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}})$-definable iff it is regular. Moreover, the conversions between formulae and automata are effective.*

*Proof.* ($1 \Rightarrow 2$) immediate.

($2 \Rightarrow 3$) Let $\varphi(T)$ be an $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}})$-formula. By Lemma 4.3, there is a formula $\varphi'(T)$ such that $\mathfrak{T}^{\mathfrak{u}} \models \varphi(T) \Leftrightarrow \mathfrak{T} \models \varphi'(\mathcal{R}(T))$. By Theorem 3.3, the set $\{\mathcal{R}(T) \mid \mathfrak{T} \models \varphi'(\mathcal{R}(T))\}$ is ranked regular. By Proposition 4.2, $\{T \mid \mathfrak{T} \models \varphi'(\mathcal{R}(T))\}$ is unranked regular.

($3 \Rightarrow 1$) If $X$ is unranked regular, then, by Proposition 4.2, $\mathcal{R}(X)$ is ranked regular. By Theorem 3.3, there is an $\mathrm{FO}(\mathfrak{T}_{\mathfrak{p}})$-formula $\varphi'(T')$ defining $\mathcal{R}(X)$. Hence, by Lemma 4.3, there is an $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$-formula $\varphi(T)$ defining $X$.

The proof that the definable relations over $\mathfrak{T}^{\mathfrak{u}}$ are regular, and vice versa, follows from Theorem 3.3, Lemma 4.3, and Proposition 4.2 in exactly the same way as the one-dimensional case. The effectiveness follows from the effectiveness of the translation of Lemma 4.3, the effectiveness of Theorem 3.3, and Proposition 4.2 (see [Börger et al. 1997; Suciu 2002]). □

So, $\mathfrak{T}^{\mathfrak{u}}$ is the universal automatic structure for unranked trees, meaning that any other structure that only defines regular relations can be interpreted in it. As over strings and ranked trees, this implies decidability:

COROLLARY 4.5. *The first-order theories of $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathfrak{u}}$ are decidable.*

As we have seen, the decision procedure is non-elementary even for ranked trees.

As a corollary of Theorem 3.3, and Lemma 4.3, we obtain separation of $\mathfrak{T}^{\mathfrak{u}}$ and $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$:

COROLLARY 4.6. *The relation $\approx_{\mathrm{dom}}$ is not $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$-definable.*

The proof of Theorem 4.4 would have been a trivial corollary of Theorem 3.3 and Proposition 4.2, had the graph of $\mathcal{R}(\cdot)$ been definable in $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ or $\mathfrak{T}^{\mathfrak{u}}$. That is, if by considering a binary tree $T'$ as an unranked one we could have tested by a formula $\varphi(T, T')$ if $T' = \mathcal{R}(T)$. The following result shows that such a formula does not exist. It can be proved by a simple pumping argument.

PROPOSITION 4.7. *The graph of $\mathcal{R}$ is not $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}})$-definable.*

We explain how the notion of *data complexity* for logics can be applied to our infinite structures. For a complexity class $\mathcal{C}$, we say that the data complexity of such a logic over $\mathfrak{M}$ (e.g., $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$ or $\mathrm{FO}_{\eta}(\mathfrak{T}^{\mathfrak{u}})$) is $\mathcal{C}$, if for every formula $\varphi(x)$ in the logic, the set $\{T \mid \mathfrak{M} \models \varphi(T)\}$ is in $\mathcal{C}$. Here $T$ is encoded as the appropriate first-order structure (depending on whether $T$ is ranked or unranked).

As an immediate corollary of Theorem 4.4, we see that the data complexity of both $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$ and $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}})$ is polynomial, since formulae can be converted into unranked tree automata. Moreover, [Gottlob et al. 2005] places the complexity of unranked regular tree languages in DLOGSPACE (for ranked trees, the bound is $\mathrm{NC}^1$ [Lohrey 2001]). We shall see several low data complexity bounds in the next section. (Notice, however, that the combined complexity, that is, the complexity of $\{(\varphi, T) \mid \varphi(T) \text{ holds}\}$ is hyper-exponential for both $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathfrak{u}}$).

## 4.3 Restricted logics over trees

In this section we consider branch quantification for unranked trees. Recall that for ranked trees, with branch quantification over $\mathfrak{T}_{\mathtt{p}}$ and $\mathfrak{T}$ we obtained $\mathcal{FO}$ and $\mathcal{MSO}^{\text{chain}}$-definable tree languages.

The first of these results extends easily to unranked trees.

PROPOSITION 4.8. *A set of unranked trees is $\mathcal{FO}$-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}})$-*definable.*

*Proof.* That $\mathcal{FO}$ can be coded in $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}})$ can be shown by a straightforward extension of the proof for the ranked case (Theorem 3.18). For the converse, it suffices to show that for every $k \geq 0$, there is $m \geq 0$ such that $T_1 \equiv_m T_2$ implies $(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_1) \equiv^\eta_k (\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_2)$.

Fix $k \geq 0$. Let $\varphi(\cdot)$ be a formula defining a rank-$k$ type, with respect to branch quantification, of a single tree in $\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}$. Then $\varphi$ is an $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}})$-formula, and by Lemma 4.3, there exists a formula $\varphi'$ in $\mathrm{FO}_\eta(\mathfrak{T}_{\mathtt{p}})$ such that $\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}} \models \varphi(T)$ iff $\mathfrak{T}_{\mathtt{p}} \models \varphi'(\mathcal{R}(T))$. Let $l$ be the maximum quantifier rank of formulae $\varphi'$ for all such formulae $\varphi$. Then $(\mathfrak{T}_{\mathtt{p}}, \mathcal{R}(T_1)) \equiv^\eta_l (\mathfrak{T}_{\mathtt{p}}, \mathcal{R}(T_1))$ implies $(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_1) \equiv^\eta_k (\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_2)$.

From the proof of Theorem 3.18 we know that there exists $m_1 > 0$ such that $\mathcal{R}(T_1) \equiv_{m_1} \mathcal{R}(T_2)$ implies $(\mathfrak{T}_{\mathtt{p}}, \mathcal{R}(T_1)) \equiv^\eta_l (\mathfrak{T}_{\mathtt{p}}, \mathcal{R}(T_1))$. It is straightforward to show that for some $m$, $T_1 \equiv_m T_2$ implies $\mathcal{R}(T_1) \equiv_{m_1} \mathcal{R}(T_2)$, and hence $(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_1) \equiv^\eta_k (\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, T_2)$. This completes the proof. $\square$

We now show that $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}})$ can be described by a natural extension of the chain logic to unranked trees. Over ranked trees, we allow quantification over chains with respect to the $<_{\text{pre}}$ partial order. For unranked trees, we use the vocabulary $(<_{\text{pre}}, <_{\text{sib}}, (O_a))$, and hence the extension, called $\mathcal{MSO}^{\downarrow}_{\rightarrow}$, allows quantification over both *vertical* chains (with respect to $<_{\text{pre}}$) and *horizontal* chains (those with respect to $<_{\text{sib}}$). In other words, in $\mathcal{MSO}^{\downarrow}_{\rightarrow}$, quantification is over sets $X$ such that $X$ is either linearly ordered by $<_{\text{pre}}$, or linearly ordered by $<_{\text{sib}}$. Note that in the latter case, $X$ must be the set of children of some node.

THEOREM 4.9. *Let $|\Sigma| > 1$. Then a subset of* $\mathrm{UTREE}(\Sigma)^k$, *$k \geq 1$, is definable in* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}})$ *iff it is definable in $\mathcal{MSO}^{\downarrow}_{\rightarrow}$.*

*Proof.* We shall prove the result for $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}})$ formulae with one free variable. A generalization to multiple free variables and $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ over $[\vec{T}]$ is straightforward.

For the $\mathcal{MSO}^{\downarrow}_{\rightarrow} \subseteq \mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}})$ direction, we only have to show how to code chains and sets of children of the same node by branches. Recall that we assume the presence of at least two symbols, say $a$ and $b$, in the alphabet $\Sigma$. Now, given a chain $X \subseteq \mathrm{dom}(T)$, we construct a tree $t_X$ such that $\mathrm{dom}(t_X)$ is the smallest set of nodes that contains $X$ and is a domain of an unranked tree. That is, if $s \in X$ and $s' \leq_{\text{pre}} s$, then $s' \in \mathrm{dom}(t_X)$, and if $s'' \cdot i \leq_{\text{pre}} s$, then $s'' \cdot j \in \mathrm{dom}(t_X)$ for all $j \leq i$. Furthermore, if $s \in X$, then it is labeled $a$ in $t_X$, otherwise it is labeled $b$. For a set $Y$ of siblings of a single node, we create a tree $t^Y$ as follows. Let $y_0$ be the rightmost node in $Y$. Then $\mathrm{dom}(t^Y) = \mathrm{dom}(t_{\{y_0\}})$, and nodes in $Y$ are labeled $a$, and the remaining ones are labeled $b$. It is easy to see that both $t_X$ and $t^Y$ are branches, and it is completely routine to verify that with this coding, every $\mathcal{MSO}^{\downarrow}_{\rightarrow}$

formula can be expressed in $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$.

In the rest of the proof we show the converse. We define, as before, the relation $(\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_m^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2)$ indicating that the duplicator wins the $m$-round game in which moves on $(\mathfrak{T}^{\mathrm{u}}, T_i)$ are restricted to branches with $\mathrm{dom}(t) \subseteq \mathrm{dom}(T_i)$, $i = 1, 2$. We first show that for every $l \geq 0$, there exists $k \geq 0$ such that

Lemma 4.10. *For every $l \geq 0$, there exists $k \geq 0$ such that*

$$(\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2) \quad \textit{implies} \quad (\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_l^{\eta} (\mathfrak{T}^{\mathrm{u}}, T_2).$$

The proof of Lemma 4.10 is very similar to proof of the analogous result for the ranked trees, used in Theorem 3.18. Let $k = l + 3$. Suppose a branch $t_{1i}$ in $T_1$ is played in the $i$th round of the game on $(\mathfrak{T}^{\mathrm{u}}, T_1)$ and $(\mathfrak{T}^{\mathrm{u}}, T_2)$. Let $t'_{1i}$ be the restriction of $t_{1i}$ to $\mathrm{dom}(T_1)$. The duplicator assumes that the spoiler plays $t'_{1i}$ in the game $(\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2)$, and finds a response $t'_{2i}$, which is a branch of $T_2$. To define $t_{2i}$, the response to $t_{1i}$, the duplicator simply "attaches" the remaining part of $t_{1i}$ to $t'_{2i}$. That is, if $s_p = s'_p \cdot j_p$ is the rightmost node of $t'_{pi}$, $p = 1, 2$, then for any node $s'_1 \cdot j \cdot v$ in $\mathrm{dom}(t_{1i})$, where $j \geq j_1$, we add a node $s'_2 \cdot (j_2 + j - j_1) \cdot v$ to $t_{2i}$ and label it the same. The extra 3 moves in the game $(\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_{l+3}^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2)$ ensure that always $\mathrm{dom}(t'_2) = \mathrm{dom}(t_2) \cap \mathrm{dom}(T_2)$. The latter then guarantees the winning strategy of the duplicator. This proves Lemma 4.10.

We now need some notation. We write $T_1 \equiv_m^{\mathcal{MSO}^{\downarrow}_{\rightarrow}} T_2$ to mean that the duplicator wins the $m$-round $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ game. Such a game is a restricted $\mathcal{MSO}$ game in which all moves are either chains of nodes (those will be called *vertical* moves), or sets of nodes which are children of a single node (those moves will be called *horizontal*). A straightforward extension of the standard $\mathcal{MSO}$ game argument shows that if $T_1 \equiv_m^{\mathcal{MSO}^{\downarrow}_{\rightarrow}} T_2$, then $T_1$ and $T_2$ agree on all $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ sentences of quantifier rank up to $m$.

The main part of the proof is the following lemma.

Lemma 4.11. *For every $k \geq 0$, there exists $m \geq 0$ such that*

$$T_1 \equiv_m^{\mathcal{MSO}^{\downarrow}_{\rightarrow}} T_2 \quad \textit{implies} \quad (\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2).$$

Note that proving Lemma 4.11 is sufficient: together with Lemma 4.10, it shows that every rank-$l$ type of a tree with respect to branch quantification is a union of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ rank-$m$ types, and thus every $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ formula of quantifier rank $l$ is equivalent to a Boolean combination of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ rank-$m$ types, and thus to an $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ formula of quantifier rank $m$.

We now prove Lemma 4.11. Without loss of generality, we assume $\Sigma = \{a, b\}$ (the proof straightforwardly extends to larger alphabets). Let $\nu_k$ be the vocabulary $(<, U_a, U_b, U_1, \ldots, U_{2k})$, where where all $U_i$s are unary, and $M$ be the number of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ rank-$(2k+1)$ types over $\nu_k$. We take $m$ to be $kM + 2k + 6$.

For a branch $t$, let $e(t)$ be the rightmost leaf of $t$ (the ending node of $t$), and let $\mathrm{rbd}(t)$ (the right boundary) be the set of all nodes $s \leq_{\mathrm{pre}} e(t)$. For any non-root node $s$ in a tree $T$ (not necessarily a branch), $\mathrm{sibl}(s)$ denotes the set of all siblings of $s$ (that is, children of $s$'s parent). We normally view $\mathrm{sibl}(s)$ as a structure in the vocabulary $<$, where $<$ is interpreted as $<_{\mathrm{sib}}$, the left-to-right order on the siblings of a node.

We shall refer to the game $(\mathfrak{T}^{\mathrm{u}}, T_1) \equiv_k^{\eta(T_1,T_2)} (\mathfrak{T}^{\mathrm{u}}, T_2)$ as the *branch game*, and to the game $T_1 \equiv_m^{\mathcal{MSO}^{\downarrow}_{\rightarrow}} T_2$ as the *chain game*, and shall denote the moves in the branch game by $(t_i^1, t_i^2)$.

After $i$ moves of the branch game, $((t_1^1, t_1^2), \ldots, (t_i^1, t_i^2))$, we have a $\nu_i$ structure $S_{i,s}^{T_j}$ associated with each node $s$ of $T_j$, $j = 1, 2$. If $s \in \mathrm{dom}(T_j)$, then the universe of $S_{i,s}^{T_j}$ is $\mathrm{sibl}(s)$, $<$ is interpreted as $<_{\mathrm{sib}}$, $U_a$ ($U_b$) is interpreted as nodes labeled $a$ ($b$) in $T_i$, and $U_{2q-1}, U_{2q}$, $q \leq i$, are interpreted as nodes labeled $a$ and $b$ respectively in $t_q^i$.

We let $\tau^j(i,s)$ denote the rank-$2(k-i)+1$ MSO-type of $S_{i,s}^{T_j}$, $j = 1, 2$. We show that the branch game can be played such that the following holds. After each round $i$, there is a number $M_i \leq M$ and a partition of $\mathrm{rbd}(t_i^1)$ (respectively, $\mathrm{rbd}(t_i^2)$) into sets $\mathrm{rbd}_q(t_i^1)$ (respectively, $\mathrm{rbd}_q(t_i^2)$), $q \leq M_i$, such that

—from the position

$$\left\{ (\mathrm{rbd}_q(t_j^1), \mathrm{rbd}_q(t_j^2)) \ \mid \ j \leq i, \ q \leq M_j \ \right\} \tag{2}$$

the duplicator has a winning strategy for $(k-i)M+2k+6$ moves in the chain game; more precisely, the above position consists of $\Sigma_{j=1}^{i}(j \times M_j)$ predicates, every predicate being interpreted by the sets $\mathrm{rbd}_q(t_j^1)$ and $\mathrm{rbd}_q(t_j^2)$ in the corresponding structures;

—for every $q$, $s_1 \in \mathrm{rbd}_q(t_i^1)$, and $s_2 \in \mathrm{rbd}_q(t_i^2)$:

$$\tau^1(i, s_1) = \tau^2(i, s_2). \tag{3}$$

Claim 4.12. *Conditions (2) and (3) guarantee that the duplicator has a winning strategy in the branch game.*

For the proof of Claim 4.12, we show that $\prec_{\downarrow}$ and $\approx_{\mathrm{dom}}$ are preserved; dealing with other predicates is analogous.

Suppose $t_{j_1}^1 \prec_{\downarrow} t_{j_2}^1$, and $j_1 < j_2$. Then $\mathrm{rbd}(t_{j_1}^1) \subsetneqq \mathrm{rbd}(t_{j_2}^1)$. This means that for every $s \in \mathrm{rbd}(t_{j_1}^1)$ in the structure $S_{k,s}^{T_1}$ the following holds: $U_{2j_1-1} = U_{2j_2-1}$ and $U_{2j_1} = U_{2j_2}$. Moreover, for every $s \in \mathrm{rbd}(t_{j_2}^1) - \mathrm{rbd}(t_{j_1}^1)$, in $S_{k,s}^{T_1}$ we have $U_{2j_1-1} = U_{2j_1} = \emptyset$ and $U_{2_1-1}, U_{2j_2} \neq \emptyset$. As these properties can be expressed as $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ formulae of quantifier-rank 1 they are consistent with $\tau^1(k,s)$ for every $s \in \mathrm{rbd}(t_{j_2}^1)$. Given condition (2), the duplicator can use the remaining moves in the game to show that the partition $\bigcup_{q=1}^{M_{j_2}} \mathrm{rbd}_q(t_{j_2}^2)$ follows the same pattern as that of $\bigcup_{q=1}^{M_{j_2}} \mathrm{rbd}_q(t_{j_2}^1)$. That is, the nodes in the $\mathrm{rbd}_q(t_{j_2}^2)$'s for which the $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ type of $S_{j_2,s}^{T_1}$ is consistent with $(U_{2j_1-1} = U_{2j_2-1}) \wedge (U_{2j_1} = U_{2j_2})$ are followed in the $<_{\mathrm{pre}}$-order by the nodes in the $\mathrm{rbd}_q(t_{j_2}^2)$'s for which this type is consistent with $U_{2j_1-1} = U_{2j_1} = \emptyset$ and $U_{2j_2-1}, U_{2j_2} \neq \emptyset$, and every node in $\mathrm{rbd}(t_{j_1})$ is contained in $\mathrm{rbd}(t_{j_2})$. Using condition (3), we then conclude that $t_{j_1}^2 \prec_{\downarrow} t_{j_2}^2$.

Suppose $t^1_{j_1} \approx_{\text{dom}} t^1_{j_2}$. Then

$$\bigcup_{q \le M_{j_1}} \text{rbd}_q(t^1_{j_1}) = \bigcup_{q \le M_{j_2}} \text{rbd}_q(t^1_{j_2}).$$

Note that this condition can be stated as a quantifier-rank 1 sentence of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ in terms of the predicates for sets $\text{rbd}_q(t^1_{j_1})$ and $\text{rbd}_q(t^1_{j_2})$. Hence, if $\bigcup_{q \in M_{j_1}} \text{rbd}_q(t^2_{j_1}) \neq \bigcup_{q \in M_{j_2}} \text{rbd}_q(t^2_{j_2})$, then the spoiler wins in 1 move in the chain game from the position $\{(\text{rbd}_q(t^1_j), \text{rbd}_q(t^2_j)) \mid j \le i,\ q \le M_j\}$, which contradicts (2). Hence,

$$\bigcup_{q \le M_{j_1}} \text{rbd}_q(t^2_{j_1}) = \bigcup_{q \le M_{j_2}} \text{rbd}_q(t^2_{j_2}).$$

and thus $\text{rbd}(t^2_{j_1}) = \text{rbd}(t^2_{j_2})$; since $t^2_{j_1}$ and $t^2_{j_2}$ are branches, this implies $t^2_{j_1} \approx_{\text{dom}} t^2_{j_2}$. This proves Claim 4.12.

It remains to show that the duplicator can play preserving conditions (2) and (3). Consider the $i$th move by the spoiler; we assume, without loss of generality, that the spoiler plays $t^1_i$ in $T_1$. For each $s \in \text{rbd}(t^1_i)$, compute $\tau^1(i-1, s)$, and form the set $X^1_q = \text{rbd}_q(t^1_i)$ of all $s \in \text{rbd}(t^1_i)$ for which $\tau^1(i, s)$ is the $q$th type in the enumeration $1, \ldots, M_i$. Then in the chain game, the duplicator assumes that the spoiler plays $M_i \le M$ moves $X^1_q$, and finds the responses $X^2_q$ over $T_2$. Clearly such a play is possible given $m$. To show (2) and (3), we must prove that there exists a branch $t^2_i$ in $T_2$ such that the set of nodes $s \in \text{rbd}_q(t^2_i)$ for which $\tau^2(i, s)$ is the $q$th type in the enumeration is precisely $X^2_q$.

First, notice that there is a branch $t'$ in $T_2$ for which $\text{rbd}(t') = \bigcup_q X^2_q$. Since $X^1_q$'s satisfy the condition

$$\begin{aligned} &\forall s, s' \ \Big( (\textstyle\bigvee_q X^1_q(s) \wedge s' <_{\text{pre}} s) \rightarrow \bigvee_q X^1_q(s') \Big) \\ \wedge\ &\forall s, s' \ \Big( (\textstyle\bigvee_q X^1_q(s) \wedge \bigvee_q X^1_q(s')) \rightarrow (s \le_{\text{pre}} s' \vee s' \le_{\text{pre}} s) \Big), \end{aligned}$$

the same must be true for $X^2_q$'s (if it were not, the spoiler would have won in 2 moves, which is impossible given our bound on $m$). This implies that $\bigcup_q X^2_q$ is of the form $\text{rbd}(t')$, where $t'$ is a branch. We now take $t^2_i$ so that $\text{rbd}(t^2_i) = \text{rbd}(t') = \bigcup_q X^2_q$.

Since the moves $X^1_q$'s and $X^2_q$'s are played according to the winning strategy in the chain game, condition (2) is preserved. Condition (3) is guaranteed by the following. We show that for every $q$, every $s_1 \in \text{rbd}_q(t^1_i)$ and $s_2 \in X^2_q$, it is the case that $\tau^1(i-1, s_1) = \tau^2(i-1, s_2)$; in other words,[1]

$$S^{T_1}_{i-1, s_1} \equiv^{\mathcal{MSO}}_{2(k-(i-1))+1} S^{T_2}_{i-1, s_2}. \tag{4}$$

Then we have two extra moves to play the sets $U_{2i-1}$ and $U_{2i}$ corresponding to positions labeled $a$ and $b$ among siblings of $s_1$ in $t^1_i$. The responses give us labellings of siblings of $s_2$, and (4) implies that the MSO game on $S^{T_1}_{i,s_1}$ and $S^{T_2}_{i,s_2}$ can continue for $\big(2(k-(i-1))+1\big) - 2 = 2(k-1)+1$ rounds, thus giving us (3).

[1]Note that the horizontal moves on $\text{sibl}(s_1)$ and $\text{sibl}(s_2)$ in the chain game are precisely the $\mathcal{MSO}$ moves on string structures $S^{T_1}_{i-1,s_1}$ and $S^{T_2}_{i-1,s_2}$.

We now prove (4). Consider $s_1 \in X_q^1$ and $s_2 \in X_q^2$. We may assume, without loss of generality, that $\{s_1\}$ is the response, in the chain game, to $\{s_2\}$. Indeed, any response to $\{s_2\}$ is a singleton set in $X_q^1$, and all elements in $X_q^1$ have the same type $\tau^1(i-1,s)$, so we can assume that the response is actually $s_1$.

We now consider structures $\mathrm{sibl}(s_1)$ and $\mathrm{sibl}(s_2)$ ordered by $<_{\mathrm{sib}}$, and consider horizontal moves in the chain game on them (which can be associated with $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ moves). In particular, we show that if the spoiler plays $U_a$, $U_b$, or $U_j$, $j \leq 2i$, over $\mathrm{sibl}(s_1)$, then the duplicator must respond by the corresponding set over $\mathrm{sibl}(s_2)$ (and symmetrically, changing the roles of $s_1$ and $s_2$). This takes $2i+2$ moves; from the bound on $m$, we know that the duplicator can continue to play for another $2k+6-(2i+2) = 2k-2i+4$ moves. Furthermore, the duplicator cannot move away from $\mathrm{sibl}(s_1)$ and $\mathrm{sibl}(s_2)$: if he does, the spoiler can catch him in one move. So we may assume that the duplicator can continue to play for $2k-2i+3 = 2(k-(i-1))+1$ moves on $\mathrm{sibl}(s_1)$ and $\mathrm{sibl}(s_2)$, from the position in which $U_a, U_b$ and $U_j$, $j \leq i$, are already played. That is, starting from $S^{T_1}_{i-1,s_1}$ and $S^{T_2}_{i-1,s_2}$ the duplicator can still respond to $2(k-(i-1))+1$ moves by the spoiler over $\mathrm{sibl}(s_1)$ and $\mathrm{sibl}(s_2)$, which are precisely $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ game moves. Hence, this will give us (4).

To complete the proof now, assume that the spoiler plays $U_a$ over $\mathrm{sibl}(s_1)$ as a horizontal move in the chain game. The duplicator must respond by $U_a$ over $\mathrm{sibl}(s_2)$: otherwise the spoiler would win in 3 moves. Likewise, the duplicator must respect $U_b$. Suppose the spoiler plays $U_{2j-1}$ over $\mathrm{sibl}(s_1)$ as a horizontal move, for $j < i$, that is, the set of nodes among $\mathrm{sibl}(s_1)$ that belong to $\mathrm{dom}(t_j^1)$ and are labeled $a$. We must show that the duplicator responds by playing $U_{2j-1}$ over $\mathrm{sibl}(s_2)$.

Consider two cases. In the first case, $\mathrm{dom}(t_j^1) \cap \mathrm{sibl}(s_1) = \emptyset$. Then $\mathrm{dom}(t_j^2) \cap \mathrm{sibl}(s_2) = \emptyset$ (otherwise the spoiler wins in 3 moves), and hence the duplicator is forced to play the correct move $\emptyset$. In the second case, $\mathrm{dom}(t_j^1) \cap \mathrm{sibl}(s_1) \neq \emptyset$ (and hence, by symmetry, $\mathrm{dom}(t_j^2) \cap \mathrm{sibl}(s_2) \neq \emptyset$). Let $r_l = \mathrm{rbd}(t_j^l) \cap \mathrm{sibl}(s_l), l = 1,2$. Then $U_{2j}$ is the intersection of $\{s \mid s \leq_{\mathrm{sib}} r_1\} \cap U_a$, and hence the duplicator is forced to play the intersection of $U_a$ over $\mathrm{sibl}(s_2)$ and $\{s \mid s \leq_{\mathrm{sib}} r_2\}$ – otherwise the spoiler needs only 3 moves to win the game, and the bound on $m$ makes this impossible.

This completes the proof of the theorem. $\square$

As one corollary of Theorem 4.9, we obtain the separation $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}) \subsetneqq \mathrm{FO}(\mathfrak{T}^{\mathrm{u}})$ (since $\mathcal{MSO}^{\downarrow}_{\rightarrow} \subsetneqq \mathcal{MSO}$, which follows from the fact that over ranked trees, chain logic is properly contained in $\mathcal{MSO}$ [Thomas 1984]).

Next, we show that $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ has low data complexity. Recall that $\mathrm{NC}^1$ is the class of languages accepted by bounded fan-in logarithmic-depth polysize circuits; it is contained in DLOGSPACE. By using a different representation of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ and a logical characterization of $\mathrm{NC}^1$ [Vollmer 1999], we show:

Proposition 4.13. *The data-complexity of* $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ *is* $\mathrm{NC}^1$.

*Proof.* It is shown in [Neven and Schwentick 2000] that every formula in $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ can be transformed to a modal logic called $\mathcal{FOREG}^*$. Hence our proof will proceed by showing that the data-complexity of $\mathcal{FOREG}^*$ formulae is $\mathrm{NC}^1$.

We start with the definition of $\mathcal{FOREG}^*$. We use two kinds of first-order variables. One kind is called *quantifier variables* and used for quantification of vertices and is denoted by symbols like $x, y$. The second kind (*expression variables*, denoted by $r, s$) is only used in vertical or horizontal path expressions. We use the following two kinds of path formulae.

—If $P$ is a regular expression (in the usual sense) over formulae then $\varphi = [P]^{\downarrow}_{r,s}(x, y)$ is a *vertical path formula*. The free variables of $\varphi$ are $\{x, y\} \cup (\text{free}(P) - \{r, s\})$, where free($P$) denotes the free variables that occur in at least one of the formulae that are used to build $P$.

—If $P$ is a regular expression over formulae then $[P]^{\rightarrow}_{r}(x)$ is a *horizontal path formula*. The free variables of $\varphi$ are $\{x\} \cup (\text{free}(P) - \{r\})$

We refer to path formulae also by the term *path expressions*. A simple example of a formula which uses a horizontal path expression is $[(O_a(r))^* O_b(r)]^{\rightarrow}_{r}(x)$.

The semantics of such formulae is defined as follows. Let $\varphi = [P]^{\downarrow}_{r,s}(x, y)$ be a vertical path formula, $t$ a tree and $v, w$ vertices of $t$. We assume interpretations for the free variables occurring in formulae in $P$. Then, $t \models \varphi[v, w]$, iff $v$ is an ancestor of $w$ and there is a labeling of the edges on the path from $v$ to $w$ with formulae, such that

(1) each edge $(u, u')$ is labeled with a formula $\theta(r, s)$ such that $t \models \theta[u, u']$; and
(2) the sequence of labels along the path from $v$ to $w$ matches $P$.

For $\psi = [P]^{\rightarrow}_{r}(x)$, $t \models \psi[v]$, iff there is a labeling of the children of $v$ with formulae, such that

(1) each child $w$ of $v$ is labeled with a formula $\theta(r)$ such that $t \models \theta[w]$; and
(2) the sequence of labels matches $P$.

E.g., the above example formula says that the rightmost child of $x$ is labeled with $b$ and all other children are labeled with an $a$.

The logic $\mathcal{FOREG}^*$ is obtained from $\mathcal{FO}$ by allowing vertical and horizontal path formulae. More formally,

(1) Every FO formula is an $\mathcal{FOREG}^*$ formula.
(2) If $P$ is a regular expression over $\mathcal{FOREG}^*$ formulae with free variables $r, s$ then $\varphi = [P]^{\downarrow}_{r,s}(x, y)$ is an $\mathcal{FOREG}^*$ formula.
(3) If $P$ is a regular expression over $\mathcal{FOREG}^*$ formulae with free variable $r$ then $[P]^{\rightarrow}_{r}(x)$ is an $\mathcal{FOREG}^*$ formula.
(4) $\mathcal{FOREG}^*$ formulae are closed under first-order quantification, disjunction, and negation, as usual.

To show that the data complexity of $\mathcal{FOREG}^*$ is $\text{NC}^1$, we use a logical characterization of $\text{NC}^1$ [Vollmer 1999]. The logic capturing $\text{NC}^1$ is of the form $\mathcal{FO}[\mathcal{Q}]$, where $\mathcal{Q}$ is a generalized quantifier (whose precise definition is not needed for the proof). We shall need the fact that every regular language is in $\text{NC}^1$ [Vollmer 1999] (and therefore expressible in $\mathcal{FO}[\mathcal{Q}]$).

To show that $\mathcal{FOREG}^* \subseteq \mathcal{FO}[\mathcal{Q}]$, one proceeds by induction on $\mathcal{FOREG}^*$ expressions. The only nontrivial case is that of path expressions of the form $[P]^{\downarrow}_{r,s}(u, v)$ and $[P]^{\rightarrow}_{r}(u)$. Consider, for example, $[P]^{\downarrow}_{r,s}(u, v)$, and let $\psi_1, \ldots, \psi_m$ be all the formulae used in $P$. Let $\Psi_P$ be an $\mathcal{FO}[\mathcal{Q}]$ sentence in the string vocabulary $(<, U_1, \ldots, U_m)$ expressing this regular language. Let $\psi'_i(r,s)$ be the $\mathcal{FO}[\mathcal{Q}]$ formula equivalent to $\psi_i$ (which exists by the induction hypothesis). Then the $\mathcal{FO}[\mathcal{Q}]$ formula equivalent to $[P]^{\downarrow}_{r,s}(u,v)$ is obtained from $\Psi_P$ by replacing each quantifier $\exists z$ with $\exists z_1, z_2\ (u \leq_{\text{pre}} z_1 <_{\text{pre}} z_2 \leq_{\text{pre}} v \wedge \neg\exists z_3(z_1 <_{\text{pre}} z_3 <_{\text{pre}} z_2))$, and then replacing each $U_i(z)$ with $\psi'_i(z_1, z_2)$ and $z < y$ with $z_1 < y_1$. The case of $[P]^{\rightarrow}_r(u)$ is similar. This completes the proof. □

We conclude this section by connecting an extension of $\text{FO}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$ with another logic studied in connection with XML, as an abstraction of XML pattern languages: $\mathcal{FOREG}$ [Neven and Schwentick 2000]. $\mathcal{FOREG}$ is the extension of $\mathcal{FO}$ with predicates $r^{\downarrow}(x)$ and $r^{\rightarrow}(x)$ for every regular expression $r$. For a tree $T$ and a node $s$, $T \models r^{\downarrow}(s)$ iff the labels on the path from the root to $s$ satisfy $r$; $T \models r^{\rightarrow}(s)$ iff the string formed by concatenating the labels of the left siblings of $s$ satisfies $r$.

Now, we introduce the logic $\text{FO}^{\text{reg}}_\eta$. This is $\text{FO}_\eta$ extended with the following predicates: for every regular expression $r$ we have $r^{\downarrow}(T)$ and $r^{\rightarrow}(T)$ with the following meaning:

—$r^{\downarrow}(T)$ iff $T = (D, f)$ is a branch, $\{s_1, \ldots, s_p\}$ is the set of right-most siblings with $s_1 <_{\text{pre}} \cdots <_{\text{pre}} s_p$ and $f(s_1)\cdots f(s_p) \in L(r)$;

—$r^{\rightarrow}(T)$ iff $T = (D, f)$ is a branch, $\{s_1, \ldots, s_p\} = \{s \mid s \leq_{\text{sib}} e(T)\}$ with $s_1 <_{\text{sib}} \cdots <_{\text{sib}} s_p$, $e(T)$ the endpoint of $T$, and $f(s_1)\cdots f(s_p) \in L(r)$.

Clearly, $\text{FO}^{\text{reg}}_\eta(\mathfrak{T}^{\mathfrak{u}}) = \text{FO}_\eta(\mathfrak{T}^{\mathfrak{u}})$. Therefore, we only consider $\text{FO}^{\text{reg}}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$.

THEOREM 4.14. *A set $X \subseteq \text{UTREE}(\Sigma)$ is definable in $\text{FO}^{reg}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$ iff $X$ is definable in $\mathcal{FOREG}$.*

*Proof.* The proof is similar to the proof of Theorem 3.18(1). To show that $\mathcal{FOREG}$ can be expressed in $\text{FO}^{\text{reg}}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$, we only need to note that the regular expressions can easily be expressed in $\text{FO}^{\text{reg}}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$.

We now prove the other direction. The proof is an extension of the proof of Theorem 3.18(1). Let $R$ be a finite set of regular expressions. We shall write $T_1 \equiv^{\mathcal{FOREG},R}_k T_2$ if the duplicator can win the $k$-round game on $T_1$ and $T_2$ considered as first-order structures where only the atomic predicates $r^{\downarrow}$ and $r^{\rightarrow}$ for $r \in R$ should be satisfied. We write, $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv^{\eta,R}_k (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$ where only the atomic predicates $r^{\downarrow}$ and $r^{\rightarrow}$ for $r \in R$ should be satisfied. Finally, we write $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv^{\eta(T_1,T_2),R}_k (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$ if the duplicator wins a $k$-round branch-restricted game that starts with $(T_1, T_2)$ in which in addition all moves are branches of $T_1$ and $T_2$ respectively, and where only the atomic predicates $r^{\downarrow}$ and $r^{\rightarrow}$ for $r \in R$ should be satisfied.

We will show that for every $k > 0$ and every finite set of regular expressions $R$, there exists $m > 0$ and a finite set of regular expressions $R \subseteq R'$ such that:

(1) $T_1 \equiv^{\mathcal{FOREG},R}_k T_2$ implies $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv^{\eta(T_1,T_2),R}_k (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$, and

(2) $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv^{\eta(T_1,T_2),R'}_m (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$ implies $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv^{\eta,R}_k (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$.

Combining these, we shall conclude that each $\text{FO}^{\text{reg}}_\eta(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$-definable set of ranked trees is $\mathcal{FOREG}$-definable.

For 1), to produce the winning strategy for $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv_k^{\eta(T_1,T_2),R} (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$, every time a branch, say $t_1 \in T_1$ is played, the duplicator pretends that the spoiler plays the leaf of that branch in the game on $T_1$ and $T_2$, finds the response (which is a node in $T_2$), and plays the branch that ends in that node. It is easy to see that this guarantees a win.

For 2), we let $m = k + 3$. For every regular expression $r \in R$, let $A_r = (Q_r, \Sigma, q_0^r, \delta_r, F_r)$ be the equivalent DFA accepting $L(r)$. The following property holds for every such $A_r$:

for every $w_1, w_2 \in \Sigma^*$, $\delta_r^*(w_1) = \delta_r^*(w_2)$ implies $\delta_r^*(w_1 v) \in F_r$ iff $\delta_r^*(w_2 v) \in F_r$ for all $v \in \Sigma^*$. (*)

Here, $\delta_r^*$ is the transition function $\delta_r$ extended to strings. Let $A_r^q$ be the automaton $A_r$ where $F = \{q\}$ and let $s_r^q$ be the regular expression equivalent to $A_r^q$. Then $R' = R \cup \{s_r^q \mid r \in R, q \in Q_r\}$.

We denote moves in the game on $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1)$ and $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$ by $t_{1i}$ and $t_{2i}$, respectively, for round $i \leq k$. Furthermore, after each round $i$, we also have a configuration $((t'_{11}, \ldots, t'_{1i}), (t'_{21}, \ldots, t'_{2i}))$ corresponding to the winning strategy in $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv_m^{\eta(T_1,T_2),R'} (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$.

The game proceed as follows. Suppose in round $i$ $(1 \leq i \leq k)$, the spoiler plays $t_{1i}$ in $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1)$ (the case of a move in $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$ is symmetric). The duplicator computes $t'_{1i}$ as $t_{1i} \cap T_1$ and augments the configuration $((t'_{11}, \ldots, t'_{1\,i-1}), (t'_{21}, \ldots, t'_{2\,i-1}))$ to $((t'_{11}, \ldots, t'_{1i}), (t'_{21}, \ldots, t'_{2i}))$ where $t'_{2i}$ is the response to $t'_{1i}$ according to the winning strategy $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv_m^{\eta(T_1,T_2),R'} (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$. Here, $t_{1i} \cap T$ is the tree $(D, f)$ where $D$ is the maximal set which is prefix-closed, contained in $\mathrm{dom}(t_{1i})$, and on which the labellings of $t_{1i}$ and $T$ coincide; $f$ is the labeling of $t_{1i}$ restricted to $D$.

To define $t_{2i}$, the response to $t_{1i}$, the duplicator simply "attaches" the remaining part of $t_{1i}$ to $t'_{2i}$. That is, if $s_p = s'_p \cdot j_p$ is the rightmost node of $t'_{pi}$, $p = 1, 2$, then for any node $s'_1 \cdot j \cdot v$ in $\mathrm{dom}(t_{1i})$, where $j \geq j_1$, we add a node $s'_2 \cdot (j_2 + j - j_1) \cdot v$ to $t_{2i}$ and label it the same.

It remains to show that this strategy guarantees a win for the duplicator. First, note that $t'_{2i} = t_{2i} \cap T_2$ for every $i$. The latter follows from the fact that we have three extra moves in the game $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_1) \equiv_m^{\eta(T_1,T_2),R'} (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, T_2)$. Further, $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models r^{\downarrow}(t_{1i})$ iff $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models r^{\downarrow}(t_{2i})$ for all $r$. Indeed, suppose $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models r^{\downarrow}(t_{1i})$ but $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \not\models r^{\downarrow}(t_{2i})$. Then, from (*) it follows that there is a $q \in Q_r$ such that $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models s_r^{q,\downarrow}(t'_{1i})$ and $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \not\models s_r^{q,\downarrow}(t'_{2i})$. So, $t'_{2i}$ cannot be an answer to $t'_{1i}$ in the small game. If $t_{1i}$ extends $t'_{1i}$ downwards, then clearly, $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models r^{\rightarrow}(t_{1i})$ iff $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}} \models r^{\rightarrow}(t_{2i})$. Otherwise, if $t_{1i}$ only extends $t'_{1i}$ to the right, then an argument similar to the one above applies. □

Note that $\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}})$ can express the properties of Proposition 5.10. Since $\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}) \subseteq \mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathfrak{u}}) = \mathrm{FO}_\eta(\mathfrak{T}^{\mathfrak{u}})$, we conclude that its data complexity is $\mathrm{NC}^1$. The bound cannot be lowered, due to the completeness of regular languages for $\mathrm{NC}^1$ [Vollmer 1999].

## 5. QUERY LANGUAGES

One of the motivations for tree algebras is to get tree constraints relevant in database (and in particular, XML) applications. In such applications, one writes queries, typically first-order, not only over trees but also over collections of trees.

Using database terminology, we deal with *relational calculus* with tree extension constraints. From the logical point of view, we consider definability over $\mathfrak{T}$, $\mathfrak{T}_\mathfrak{p}$, $\mathfrak{T}^\mathfrak{u}$, and $\mathfrak{T}^\mathfrak{u}_\mathfrak{p}$ parameterized by sets or relations on trees.

In this section, after giving the basic definition for relational calculi with constraints, we obtain normal forms for queries that will allow us to classify the expressive power and complexity of query evaluation for relational calculi with tree extension constraints. We then obtain normal forms for queries that are known to produce finite output on any input.

### 5.1 Relational calculi over databases of trees

*Definitions.* A database *schema* $\sigma$ is a finite collection of relation symbols. Given an underlying structure $\mathfrak{M}$, relational calculus over $\mathfrak{M}$ and $\sigma$, $\mathrm{RC}(\mathfrak{M}, \sigma)$, is the class of first-order queries (formulae) in the language of $\mathfrak{M}$ supplemented with the symbols from $\sigma$. If $\sigma$ is understood, or irrelevant, we write $\mathrm{RC}(\mathfrak{M})$. For example, if $\sigma$ has a single binary relation $E$, then the $\mathrm{RC}(\mathfrak{T}_\mathfrak{p}, \sigma)$ query

$$\forall x \forall y \; E(x, y) \rightarrow \big(\eta(x) \wedge \eta(y) \wedge x \prec y\big)$$

tests whether $E$ is a subgraph of $\prec$ whose nodes are branches.

We shall always interpret $\sigma$ relations as finite relations over the universe of $\mathfrak{M}$ (in our case, $\textsc{Tree}_n(\Sigma)$). The *active domain* of a $\sigma$-structure $\mathcal{A}$ is the set $adom(\mathcal{A})$ of all the elements of $\textsc{Tree}_n(\Sigma)$ that occur in $\mathcal{A}$. Active-domain quantifiers are quantifiers of the form $\exists x \in adom$ and $\forall x \in adom$; $\exists x \in adom \; \varphi(x)$ is interpreted as the existence of an element $a \in adom(\mathcal{A})$ that satisfies $\varphi(a)$.

*Normal forms.* The main tools for analyzing the expressive power of $\mathrm{RC}(\mathfrak{M}, \sigma)$ come in the form of normal forms results. The most commonly used one, *restricted-quantifier normal form* [Benedikt and Libkin 2000; Flum and Ziegler 1999; Benedikt et al. 2003], states that every $\mathrm{RC}(\mathfrak{M}, \sigma)$ formula is equivalent to a formula in which no $\sigma$ symbol appears in the scope of a quantifier $\forall x$ or $\exists x$ (that is, they appear only in the scope of quantifiers $\forall x \in adom$ and $\exists x \in adom$). The ability to put queries in restricted-quantifier normal form implies very strong expressivity bounds on relational calculi with constraints. In particular, it gives a strong bound on *generic* queries expressible in such calculi. Recall that a query is generic if it commutes with permutations of the domain of $\mathfrak{M}$. For example, *parity*, testing if the number of elements of $adom(\mathcal{A})$ is generic, as is *graph connectivity*. In fact, any query definable in a standard relational query language (relational calculus, datalog, etc.) without constraints is generic. If all queries can be put in restricted-quantifier normal form, then every generic query in $\mathrm{RC}(\mathfrak{M}, \sigma)$ is definable in first-order logic over finite $\sigma$ structures and a linear ordering on their domain. This in turn implies that queries such as parity and connectivity are not definable. However, it was shown in [Benedikt et al. 2003] that no structure with infinite VC dimension admits restricted-quantifier normal form. Hence,

COROLLARY 5.1. *$\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$ do not admit restricted-quantifier normal form, even if $|\Sigma| = 1$.*

We thus need to find a different way of getting bounds on the expressive power and complexity of $\mathrm{RC}(\mathfrak{T})$ and $\mathrm{RC}(\mathfrak{T}_\mathfrak{p})$. The main tool is a different and weaker normal form result that shows how to restrict quantification to a finite extension of the active domain. From that result, we derive both complexity and expressibility bounds.

### 5.2 Relational calculi and ranked trees

We first show that a certain weaker restricted quantifier normal form holds for relational calculus over $\mathfrak{T}$. Recall that for a set $X$ of trees, $\text{TREE}_n(\Sigma)|_{\text{dom}(X)}$ is the set of all trees $T$ such that $\text{dom}(T) \subseteq \bigcup_{T_0 \in X} \text{dom}(T_0)$. Note that if $X$ is definable by a formula of $\mathfrak{T}$, then so is $\text{TREE}_n(\Sigma)|_{\text{dom}(X)}$.

The main result of this section is that for every $\text{RC}(\mathfrak{T}, \sigma)$-query, quantifiers can be restricted to range over a finite and definable set of trees. In particular, these trees can only contain nodes present in the domains of trees in the tuple of free variables and in the trees in the active domain of the finite $\sigma$-structure.

THEOREM 5.2. *For* $|\Sigma| > 1$*, any relational calculus formula* $\varphi(\vec{T})$ *in* $\text{RC}(\mathfrak{T}, \sigma)$ *is equivalent to a formula in which quantifiers range over* $\text{TREE}_n(\Sigma)|_{\text{dom}(X)}$ *where* $X = adom(\mathcal{A}) \cup \vec{T}$.

*Proof.* For $n = 1$, the result follows from [Benedikt et al. 2003], so we only consider the case of $n > 1$. Without loss of generality, we assume $n = 2$ and $\Sigma = \{a, b\}$ (this will keep the notation more manageable; the extension to larger $n$ and larger alphabets does not pose any problems). We shall use a different (relational) vocabulary for $\mathfrak{T}$. Namely, instead of $\prec$ and succ, we have the following predicates: $\eta(t, T)$ indicating that $t$ is a branch of $T$, $\text{lsucc}(t, t')$ which is true if $t' = \text{succ}_1^a(t)$ or $t' = \text{succ}_1^b(t)$ and both $t, t'$ are branches ($t'$ extends $t$ on the left), similarly $\text{rsucc}(t, t')$ is true if $t' = \text{succ}_2^a(t)$ or $t' = \text{succ}_2^b(t)$ and $t, t'$ are branches ($t'$ extends $t$ on the right), and $L_a(t)$ ($L_b(t)$) saying that $t$ is a branch and the leaf is labeled $a$ ($b$, resp.). Clearly these can be expressed over $\mathfrak{T}_\mathfrak{p}$, and both $\prec$ and succ functions can be expressed in the new signature. Indeed, $T \preceq T'$ iff $\forall t\ (\eta(t, T) \rightarrow \eta(t, T'))$. Furthermore, $\eta_{\max}(t, T)$ can be expressed as $\eta(t, T) \wedge \neg\exists t'(t' \neq t \wedge \eta(t', T) \wedge \eta(t, t'))$. Then $T' = \text{succ}_1^a(T)$ iff the following holds:

$$\begin{aligned} &\forall t\ \big(\eta_{\max}(t, T) \rightarrow \exists t'\ (\eta_{\max}(t', T') \wedge \text{lsucc}(t, t') \wedge L_a(t'))\big) \\ \wedge\ &\forall t'\ \big(\eta_{\max}(t', T') \rightarrow L_a(t') \wedge \exists t\ (\eta_{\max}(t, T) \wedge \text{lsucc}(t, t'))\big) \end{aligned}$$

and other succ functions are defined similarly. Therefore, we shall view $\mathfrak{T}$ as a structure of the vocabulary $(\eta, \text{lsucc}, \text{rsucc}, L_a, L_b, \epsilon_a, \epsilon_b, \approx_{\text{dom}})$.

It suffices to prove the theorem for sentences, as a tuple of free variables can always be encoded as an extra relation with one tuple. Given a prefix-closed subset $X$ of $\{1, 2\}^*$, its *frontier* $\text{Fr}(X)$ is the set of strings from $X$ which are not prefixes of other strings from $X$. In other words, for any tree $T$ with $\text{dom}(T) = X$, $\text{Fr}(X) = \text{Fr}(T)$.

For a $\sigma$-structure $\mathcal{A}$, we write $\text{dom}(\mathcal{A})$ for $\bigcup_{T \in adom(\mathcal{A})} \text{dom}(T)$, and $\text{dom}^c(\mathcal{A})$ for the completion of $\text{dom}(\mathcal{A})$, that is, the set that contains the following:

(1) $\text{dom}(\mathcal{A})$;

(2) for every $s \cdot 1 \in \text{dom}(\mathcal{A})$ and $s \cdot 2 \notin \text{dom}(\mathcal{A})$, the string $s \cdot 2$, and

(3) for every $s \cdot 2 \in \text{dom}(\mathcal{A})$ and $s \cdot 1 \notin \text{dom}(\mathcal{A})$, the string $s \cdot 1$.

Note the frontier of $\text{dom}^c(\mathcal{A})$ is $\text{Fr}(\text{dom}(\mathcal{A}))$ plus the strings added by the rules 2 and 3.

Let $\mathfrak{T}[\mathcal{A}]$ be the structure in the (new) vocabulary for $\mathfrak{T}$ plus $\sigma$ whose universe is the set of all trees $T$ with $\mathrm{dom}(T) \subseteq \mathrm{dom}(\mathcal{A})$, and let $\mathfrak{T}'[\mathcal{A}]$ be the structure in the same vocabulary whose universe is the set of all trees $T$ with $\mathrm{dom}(T) \subseteq \mathrm{dom}^c(\mathcal{A})$. The interpretation of $\mathfrak{T}$ predicates is inherited from $\mathfrak{T}$, and the interpretation of $\sigma$ predicates from $\mathcal{A}$. The proof follows from two lemmas:

LEMMA 5.3. *For any $k \geq 0$, there exists $m \geq 0$ such that $\mathfrak{T}^c[\mathcal{A}] \equiv_m \mathfrak{T}^c[\mathcal{B}]$ implies $(\mathfrak{T}, \mathcal{A}) \equiv_k (\mathfrak{T}, \mathcal{B})$.*

LEMMA 5.4. *For any $k \geq 0$, there exists $m \geq 0$ such that $\mathfrak{T}[\mathcal{A}] \equiv_m \mathfrak{T}[\mathcal{B}]$ implies $\mathfrak{T}^c[\mathcal{A}] \equiv_k \mathfrak{T}^c[\mathcal{B}]$.*

These lemmas suffice, since they imply that every rank-$k$ type over $\mathfrak{T}$ and $\sigma$ (that is, every $\mathrm{RC}(\mathfrak{T}, \sigma)$ query of quantifier rank $k$) is a union of rank-$m$ types over the restriction of $\mathfrak{T}$ to trees with $\mathrm{dom}(T) \subseteq \mathrm{dom}(\mathcal{A})$, for some $m > 0$ that depends on $k$ only. Each such type can be expressed by a query of quantifier rank $m$ with restricted quantification, since the universe of $\mathfrak{T}[\mathcal{A}]$ (or $\mathfrak{T}[\mathcal{B}]$) is definable in $\mathrm{RC}(\mathfrak{T}, \sigma)$.

*Proof of Lemma 5.3.* A new notation: for a tree $T$ and a node $s$, $T(s)$ is the subtree rooted at $s$; that is, $\mathrm{dom}(T(s)) = \{s' \mid s \cdot s' \in \mathrm{dom}(T)\}$, and the labeling is inherited from $T$. Note that $T(s)$ could be empty. Observe that any tree $T$ is uniquely determined by its restriction to $\mathrm{dom}^c(\mathcal{A})$, denoted by $T'$, and all the trees $T(s)$, $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$.

Let $d_i$ be the (finite) number of rank $k - i$ types of $i$-tuples of trees over $\mathfrak{T}$. We choose $m$ to be

$$\sum_{i=1}^{k}(d_i + 1) \ + \ 3.$$

To play the $k$-round game on $(\mathfrak{T}, \mathcal{A})$ and $(\mathfrak{T}, \mathcal{B})$ (called the *big* game), the duplicator is also playing an $m$-round game on $\mathfrak{T}^c[\mathcal{A}]$ and $\mathfrak{T}^c[\mathcal{B}]$ (called the *small* game), and is using his winning strategy there to guarantee the win in the big game.

We shall refer to moves in the big game as $(M_i, N_i), i \leq k$, with $M_i$ played in $(\mathfrak{T}, \mathcal{A})$. Associated with that in the small game we have moves $(M_i', N_i')$ which are the restrictions of $M_i$ and $N_i$ to $\mathrm{dom}^c(\mathcal{A})$ and $\mathrm{dom}^c(\mathcal{B})$. The idea of duplicator's strategy is as follows. Suppose in the $i$th round the spoiler plays $M_i$ in the big game. The duplicator pretends that that spoiler plays $M_i'$ in the small game and find the response $N_i'$. Next, the duplicator looks at what happens with $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$. For each $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$, there are $i$ trees rooted at $s$ which are subtrees of $M_j$, $j \leq i$ (some of them could be empty). There are $d_i$ rank $k - i$ types of such $i$ tuples. For each type $\tau$, the duplicator constructs (in the small game) a tree $U_\tau$ whose leaves are exactly the leaves $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$ at which type $\tau$ is realized. Assuming the spoiler plays this tree, the duplicator finds his response $V_\tau$. For any $s \in \mathrm{Fr}(V_\tau)$ in $\mathfrak{T}^c[\mathcal{B}]$, the duplicator then chooses a tree $N_s^i$ so that the rank $k - i$ type of $(N_1(s), \ldots, N_{i-1}(s), N_s^i)$ is precisely $\tau$. Finally, in the big game the duplicator responds with $N_i$ which is $N_i'$ with $N_s^i$ attached as subtrees rooted at $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{B}))$.

We now present the winning strategy for the duplicator in greater detail. Let $\tau_1^i, \ldots, \tau_{d_i}^i$ enumerate rank $k - i$ types of $i$-tuples over $\mathfrak{T}$. After $i$ rounds have been played in the big game, in the small game we have the following moves:

—$M'_j, N'_j$ which are restrictions of $M_j, N_j$ to $\mathrm{dom}^c(\mathcal{A})$ and $\mathrm{dom}^c(\mathcal{B})$ respectively,

—$U_l^j, j \leq i, l \leq d_j$ over $\mathrm{dom}^c(\mathcal{A})$, and $V_l^j, j \leq i, l \leq d_j$ over $\mathrm{dom}^c(\mathcal{B})$. The labeling function on these trees is irrelevant (e.g., it could be assumed to be constantly $a$).

These trees will have the following properties:

(1) $\mathrm{Fr}(U_l^j), l \leq d_j$, partition $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$, and likewise $\mathrm{Fr}(V_l^j), l \leq d_j$, partition $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{B}))$. Some of $U_l^j$s and $V_l^j$s are allowed to be empty; since emptiness is a quantifier-free formula, from the assumption we see that $U_l^j = \emptyset$ iff $V_l^j = \emptyset$.

(2) Let $I_\emptyset^j \subseteq \{1, \dots, d_j\}$ be the set of all indices $l$ such that $\tau_l^j$ (viewed as a formula) is consistent with the formula saying that the last element of the $j$-tuple being $\emptyset$. Then $\mathrm{Fr}(U_l^j), l \in I_\emptyset^j$, partition $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{A})) - \mathrm{Fr}(M'_j)$, and likewise $\mathrm{Fr}(V_l^j), l \in I_\emptyset^j$, partition $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{B})) - \mathrm{Fr}(N'_j)$.

(3) For any $j < i$ and $l \leq d_j$, let $I$ be the set of indices $p$ of all $\tau_p^{j+1}$ consistent with $\tau_l^j$. Then $\mathrm{Fr}(U_p^{j+1}), p \in I$, form a partition of $\mathrm{Fr}(U_l^j)$, and likewise for the $V$ moves.

(4) For any $l \leq d_j$, $l \notin I_\emptyset^j$, there is a symbol $a \in \Sigma$ such that every leaf of $M'_j$ which is in $\mathrm{dom}(U_l^j)$ is labeled $a$, and likewise every leaf of $N'_j$ which is in $\mathrm{dom}(V_l^j)$ is labeled $a$.

(5) For any $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$, let $U_l^i$ be the unique tree among the $U_q^i$s such that $s \in U_l^i$. Then the rank $k - i$ type of $(M_1(s), \dots, M_i(s))$ is $\tau_l^i$.

Claim 5.5. *The duplicator can play in a way that maintains the above conditions (1)–(5).*

This is shown by induction on $i$. Assume, w.l.o.g., that the $(i+1)$st move of the spoiler is $M_{i+1}$. The duplicator finds $M'_{i+1}$, the restriction to $\mathrm{dom}^c(\mathcal{A})$, and adds a round $(M'_{i+1}, N'_{i+1})$ to the small game. Next, for each type $\tau_l^{i+1}, l \leq d_{i+1}$, the duplicator collects all the nodes $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$ such that the rank $k - (i+1)$ type of $i+1$ tuple $(M_1(s), \dots, M_{i+1}(s))$ is $\tau_l^{i+1}$. Let $X(\tau_l^{i+1})$ be the set of all such nodes. The duplicator forms a tree $U_l^{i+1}$ whose leaves are precisely $X(\tau_l^{i+1})$. Then, assuming the spoiler plays $U_l^{i+1}, l \leq d_{i+1}$ in the small game, the duplicator finds responses $V_l^{i+1}, l \leq d_{i+1}$ over $\mathrm{dom}^c(\mathcal{B})$. Note that if conditions 1)–4) above are violated, the spoiler would be able to win the small game in three extra moves; thus, by our assumptions, conditions 1)–4) are maintained in the game, since we allowed for three extra moves in calculating $m$. (A note on condition 4: this holds since exactly one formula $\epsilon_a \preceq T$, $a \in \Sigma$, is in the atomic type of $T$; hence all leafs of $M'_j$ that are in $U_l^j$ will have the same label. Furthermore, the condition in 4 can be stated as a quantifier rank 3 sentence, and hence it will hold for $V_l^j$ and $N'_j$ as well.)

The duplicator now looks at every node $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{B}))$. If $s \notin \mathrm{Fr}(N'_{i+1})$, then there is no subtree of $N_{i+1}$ rooted at $s$. If $s \in \mathrm{Fr}(N'_{i+1})$, then there exists a unique $l \leq d_{i+1}$ such that $s \in \mathrm{Fr}(V_l^{i+1})$. Then there exists $l' \leq d_i$ such that $U_l^{i+1}$ is a subtree of $U_{l'}^i$ and $V_l^{i+1}$ is a subtree of $V_{l'}^i$. Let $\varphi_l^{i+1}(T_1, \dots, T_i, T_{i+1})$ be the formula of quantifier rank $k - (i+1)$ defining $\tau_l^{i+1}$. We know that for any $s' \in \mathrm{Fr}(U_l^{i+1})$,

$$\exists U \; \varphi_l^{i+1}(M_1(s'), \dots, M_i(s'), U)$$

holds (the witness is $M_{i+1}(s')$). Since $s' \in \mathrm{Fr}(U^i_{l'})$ and $s \in \mathrm{Fr}(V^i_{l'})$, the rank $k-i$ types of $(M_1(s'), \ldots, M_i(s'))$ and $(N_1(s), \ldots, N_i(s))$ are the same, and hence $\exists U\ \varphi^{i+1}_l(N_1(s), \ldots, N_i(s), U)$ holds. We pick a witness $U$, and make it $N_{i+1}(s)$, that is, the subtree of $N_{i+1}$ rooted at $s$. Since the root of $M_{i+1}(s')$ has the same label as $s'$ in $U^{i+1}_l$, and the label of the root is determined by the atomic type, we see from condition 4) that the label of $s$ in $N'_{i+1}$ is the same as the label of the root of $U$; hence, $U$ can be attached at $s$. This completes the construction of $N_{i+1}$ which is the duplicator's response in the big game. The usual elementary properties of types ensure that all the conditions (1)–(5) continue to hold. This proves Claim 5.5.

Claim 5.6. *A strategy that maintains conditions (1)–(5) is a winning strategy for the duplicator.*

To prove this claim, we need to show that $(M_i, N_i), i \leq k$ is a partial isomorphism, that is, the duplicator wins the big game. If for some $k$-ary $R$ from $\sigma$, $R(M_{i_1}, \ldots, M_{i_k})$ holds, then $M_{i_j}$s are all trees over $\mathrm{dom}^c(\mathcal{A})$ and hence $M'_{i_j} = M_{i_j}$, $N'_{i_j} = N_{i_j}$, and $R(N_{i_1}, \ldots, N_{i_k})$ since the duplicator has a winning strategy in the small game.

Next, suppose that $\eta(M_i, M_j)$ is true. If $M'_i = M_i$, by the construction and the winning strategy in the small game we get $\eta(N_i, N_j)$. If not, let $s$ be the intersection of $\mathrm{dom}(M_i)$ and $\mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$ (since $M_i$ is a branch, the intersection is a single node); that is, $s$ is the leaf of $M'_i$. Since $\eta(M_i, M_j)$ holds, $s$ is a node in $\mathrm{Fr}(M'_j)$, and $\eta(M'_i, M'_j)$ holds. Thus, $\eta(N'_i, N'_j)$ holds, and by the winning strategy, the atomic types of $(M_i(s), M_j(s))$, and $(N_i(s'), N_j(s'))$ are the same, where $s'$ is the leaf of $N'_i$. Hence, $\eta(N_i(s'), N_j(s'))$ holds, and thus $\eta(N_i, N_j)$ is true. The converse – that $\eta(N_i, N_j)$ implies $\eta(M_i, M_j)$ – is identical.

Next, assume $\mathrm{lsucc}(M_i, M_j)$ holds. Then either $M'_j = M_j, M_i = M'_i$, in which case $N_i = N'_i, N_j = N'_j$ and $\mathrm{lsucc}(N_i, N_j)$ by the winning strategy in the small game, or the leaf of $M'_i$ is on the frontier of $\mathrm{dom}^c(\mathcal{A})$. In this case we conclude that for the leaf $s'$ of $N'_i$, $(M_i(s), M_j(s))$ and $(N_i(s'), N_j(s'))$ have the same atomic type, which implies $\mathrm{lsucc}(N_i, N_j)$. The converse, and the case of the rsucc predicate, are analogous.

We omit the very easy cases of $L_a, L_b, \epsilon_a, \epsilon_b$, and consider $M_i \approx_{\mathrm{dom}} M_j$. Clearly $M'_i \approx_{\mathrm{dom}} M'_j$, and thus $N'_i \approx_{\mathrm{dom}} N'_j$. Furthermore, for any $s' \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{B}))$, the atomic type of $(N_1(s'), \ldots, N_k(s'))$ must be equal to the atomic type of $(M_1(s), \ldots, M_k(s))$ for some $s \in \mathrm{Fr}(\mathrm{dom}^c(\mathcal{A}))$. In fact, both have type $\tau^k_l$ where $l$ is the index such that $s \in \mathrm{Fr}(U^k_l)$ and $s' \in \mathrm{Fr}(V^k_l)$. Since the atomic type of $(M_1(s), \ldots, M_k(s))$ includes $M_i(s) \approx_{\mathrm{dom}} M_j(s)$, we conclude $N_i(s') \approx_{\mathrm{dom}} N_j(s')$, from which $N_i \approx_{\mathrm{dom}} N_j$ follows. The converse is the same. This shows that duplicator wins the big game in $k$ rounds, and thus completes the proof of Lemma 5.3.

*Proof of Lemma 5.4.* We claim that $m$ can be taken to be $5k+2$ (more generally, $(n\cdot|\Sigma|+1)\cdot k+2$, where $n$ is the number of directions). Again we shall refer to the games as the big game and the small game: the big game is played over $\mathrm{dom}^c(\mathcal{A})$ and $\mathrm{dom}^c(\mathcal{B})$, and the small one over $\mathrm{dom}(\mathcal{A})$ and $\mathrm{dom}(\mathcal{B})$. The moves in the big game will be denoted by $(M_i, N_i), i \leq k$.

For each move in the big game, there will be five moves in the small game. These moves, denoted by $M_i', U_{1a}^i, U_{2a}^i, U_{1b}^i, U_{2b}^i$ over $\text{dom}(\mathcal{A})$ and $N_i', V_{1a}^i, V_{2a}^i, V_{1b}^i, V_{2b}^i$ over $\text{dom}(\mathcal{B})$ will satisfy the following properties:

(1) $M_i'$ is the restriction of $M_i$ to $\text{dom}(\mathcal{A})$ and $N_i'$ is the restriction of $N_i$ to $\text{dom}(\mathcal{B})$.
(2) $U_j^i \approx_{\text{dom}} M_i'$, and $V_j^i \approx_{\text{dom}} N_i'$, $j \in \{1a, 2a, 1b, 2b\}$.
(3) The labeling of a node $s$ in $U_{1a}^i$ ($V_{1a}^i$) is $a$ iff $s \cdot 1 \in \text{dom}^c(\mathcal{A}) - \text{dom}(\mathcal{A})$, $s \in \text{dom}(M_i)$ and the labeling of $s$ in $M_i$ is $a$ (respectively, $s \cdot 1 \in \text{dom}^c(\mathcal{B}) - \text{dom}(\mathcal{B})$, $s \in \text{dom}(N_i)$ and the labeling of $s$ in $N_i$ is $a$).
(4) The rules for $U_j^i, V_j^i, j \in \{2a, 1b, 2b\}$ are similar: for $j = 2a$, $s \cdot 2$ is in the completion, and the labeling of the node is $a$; for $j = 1b$, $s \cdot 1$ is in the completion, and the labeling is $b$; finally, for $j = 2b$, $s \cdot 2$ is in the completion, and the labeling is $b$.
(5) A node $s \in \text{dom}(M_i')$ can be labeled $a$ in at most one of $U_j^i$s, and likewise a node $s \in \text{dom}(N_i')$ can be labeled $a$ in at most one of $U_j^i$s.

The game proceeds as follows. Suppose $i$ rounds have been played in the big game (and thus $5i$ rounds in the small game), and the spoiler makes his $i+1$st move in the big game, say, $M_{i+1}$ with $\text{dom}(M_{i+1}) \subseteq \text{dom}^c(\mathcal{A})$ (the case when the spoiler plays $N_{i+1}$ with $\text{dom}(N_{i+1}) \subseteq \text{dom}^c(\mathcal{B})$ is symmetric). Let $M_{i+1}'$ be the restriction of $M_{i+1}$ to $\text{dom}(\mathcal{A})$. Let $U_j^{i+1}$, $j \in \{1a, 2a, 1b, 2b\}$, be four trees with the same domain as $M_{i+1}'$, where all nodes $s$ are labeled $b$ except in the following four cases.

—If $s \cdot 1 \notin \text{dom}(M_{i+1}')$ and it is labeled $a$ in $M_{i+1}$, then $s$ is labeled $a$ in $U_{1a}^{i+1}$.
—If $s \cdot 2 \notin \text{dom}(M_{i+1}')$ and it is labeled $a$ in $M_{i+1}$, then $s$ is labeled $a$ in $U_{2a}^{i+1}$.
—If $s \cdot 1 \notin \text{dom}(M_{i+1}')$ and it is labeled $b$ in $M_{i+1}$, then $s$ is labeled $a$ in $U_{1b}^{i+1}$.
—If $s \cdot 2 \notin \text{dom}(M_{i+1}')$ and it is labeled $b$ in $M_{i+1}$, then $s$ is labeled $a$ in $U_{2b}^{i+1}$.

The duplicator then assumes that the spoiler played $M_{i+1}', U_j^{i+1}, j \in \{1a, 2a, 1b, 2b\}$ in the small game, and finds the responses $N_{i+1}', V_j^{i+1}, j \in \{1a, 2a, 1b, 2b\}$, according to his winning strategy in the small game. We may assume that these satisfy conditions 2) an 5), since those conditions can be tested by a sentence of quantifier rank 2, and hence, if they are violated, the spoiler would win the small game in the next 2 moves, which we know is not the case. We finally construct $N_{i+1}$, duplicator's response to $M_{i+1}$, by using $V_j^{i+1}$s. That is, if a certain node $s$ is labeled $a$ in $V_{1a}^{i+1}$, we know that $s \cdot 1 \in \text{dom}^c(\mathcal{B}) - \text{dom}(\mathcal{B})$, so we add $s \cdot 1$ labeled $a$ to $N_{i+1}$. For $V_{2a}^{i+1}$, we add $s \cdot 2$ labeled $a$, for $V_{1b}^{i+1}$, we add $s \cdot 1$ labeled $b$, and for $V_{2b}^{i+1}$, we add $s \cdot 2$ labeled $b$. This completes the construction of duplicator's response $N_{i+1}$; clearly, all conditions 1)–5) are satisfied.

It remains to show that $(M_i, N_i), i \leq k$ defines a partial isomorphism. Clearly all $\sigma$-relations are preserved, as they apply only to trees whose domains are contained in $\text{dom}(\mathcal{A})$ and $\text{dom}(\mathcal{B})$, and thus the assumption of the lemma applies to them. Hence, it remains to check that $\mathfrak{T}$ predicates are preserved.

Suppose $\eta(M_i, M_j)$ holds. If $\text{dom}(M_i) \subseteq \text{dom}(\mathcal{A})$, then $M_i = M_i'$ and $\eta(N_i, N_j)$ is immediate. Suppose the leaf of $M_i$ is a node in $\text{dom}^c(\mathcal{A}) - \text{dom}(\mathcal{A})$, say $s \cdot 1$ for $s \in \text{dom}(\mathcal{A})$, and assume it is labeled $a$ (the other three cases are identical). Then $s$ is labeled $a$ in both $U_{1a}^i$ and $U_{1a}^j$. Let $s'$ be the leaf of $N_i'$. By the winning strategy of the small game, $s'$ is labeled $a$ in $V^i_{1a}$. Furthermore, $U^i_{1a}$ and $U^j_{1a}$ satisfy the condition that the leaf of $U^i_{1a}$ is also labeled $a$ in $U^j_{1a}$. Since this is expressed as a sentence of quantifier rank 2, this condition must be true of $V^i_{1a}$ and $V^j_{1a}$. Hence, $s'$ is labeled by $a$ in $V^j_{1a}$, meaning that in $N_j$, the branch leading to $s$ is extended to $s \cdot 1$ and $s \cdot 1$ is labeled $a$. This implies $\eta(N_i, N_j)$.

The proofs for $\mathrm{lsucc}, \mathrm{rsucc}, L_a, L_b$ are similar, and the proofs for $\epsilon_a, \epsilon_b$ are easy. It thus remains to show that $\approx_{\mathrm{dom}}$ is preserved. Assume again $M_i \approx_{\mathrm{dom}} M_j$ (the other case is symmetric). Then $M'_i \approx_{\mathrm{dom}} M'_j$ and therefore $N'_i \approx_{\mathrm{dom}} N'_j$. Next, consider a tree $W^i_\ell$ whose domain is $\mathrm{dom}(M'_i)$, and whose labeling combines $U^i_{1a}$ and $U^i_{1b}$; that is, a node $s$ is labeled $a$ iff it is labeled $a$ in precisely one of $U^i_{1a}$ and $U^i_{1b}$. This indicates that a branch goes to the left at $s$ in $M_i$. Similarly define $W^j_\ell$, and also $W^i_r$ by combining $U^i_{2a}$ and $U^i_{2b}$, and likewise $W^j_r$. Over $\mathrm{dom}(\mathcal{B})$, we define trees $Z^i_\ell, Z^j_\ell, Z^i_r, Z^j_r$, whose domain is $\mathrm{dom}(N'_i)$, and the labeling is defined as for $W$s, except that $V^i_p, V^j_p$ trees are used. From $M_i \approx_{\mathrm{dom}} M_j$, we infer $W^i_\ell = W^j_\ell$ and $W^i_r = W^j_r$. There is a formula $\alpha(T_1, T_2, T_3, T_4)$ of quantifier rank 2 such that $\alpha(U^i_{1a}, U^i_{1b}, U^j_{1a}, U^j_{1b})$ holds iff $W^i_\ell = W^j_\ell$. Hence, by the assumptions on the small game, $\alpha(U^i_{1a}, U^i_{1b}, U^j_{1a}, U^j_{1b})$ implies $\alpha(V^i_{1a}, V^i_{1b}, V^j_{1a}, V^j_{1b})$ and hence $Z^i_\ell = Z^j_\ell$. This means that the set of nodes $s$ in $\mathrm{dom}(N'_i)$ from which a branch extends to $s \cdot 1 \in \mathrm{dom}^c(\mathcal{B}) - \mathrm{dom}(\mathcal{B})$ is the same as the set of nodes $s$ with the same property in $\mathrm{dom}(N'_j)$. The same argument shows that $Z^i_r = Z^j_r$; that is, the set of nodes $s \in \mathrm{dom}(\mathcal{B})$ from which a branch extends to $s \cdot 2 \in \mathrm{dom}^c(\mathcal{B}) - \mathrm{dom}(\mathcal{B})$ is the same for $N_i$ and $N_j$. Combining this with $N'_i \approx_{\mathrm{dom}} N'_j$, we conclude $N_i \approx_{\mathrm{dom}} N_j$.

This concludes the proof that $(M_i, N_i), i \leq k$ is a partial isomorphism. Hence, the duplicator wins the big game, which proves Lemma 5.4, and thus the theorem. □

5.2.1 *Data complexity of relational calculi over $\mathfrak{T}$ and $\mathfrak{T}_\mathfrak{p}$.* Using Theorem 5.2, we obtain bounds on query evaluation over $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$. For relational calculi, we are interested in *data complexity* [Abiteboul et al. 1995]: the complexity of evaluating a fixed query as databases vary. The result below says that data complexity is essentially PH (polynomial hierarchy): PH is an upper bound, and for every level of PH, there is a complete problem that can be encoded. Since the encoding can be done in $\mathrm{RC}(\mathfrak{T}_\mathfrak{p})$, this gives us matching bounds for the complexity over $\mathfrak{T}$ and the simpler algebra $\mathfrak{T}_\mathfrak{p}$.

THEOREM 5.7. *The data complexity of* $\mathrm{RC}(\mathfrak{T})$ *(and thus* $\mathrm{RC}(\mathfrak{T}_\mathfrak{p})$*) is in PH. Furthermore, there is an infinite set $S \subseteq \mathrm{TREE}_n(\Sigma)$ definable in $\mathfrak{T}_\mathfrak{p}$ such that for every $n$, there are problems complete for $\Sigma^{\mathrm{p}}_n$ and $\Pi^{\mathrm{p}}_n$ which can be expressed in* $\mathrm{RC}(\mathfrak{T}_\mathfrak{p})$ *(and thus* $\mathrm{RC}(\mathfrak{T})$*) over databases whose active domain lies in $S$.*

*Proof.* We start by proving the PH bound for $\mathrm{RC}(\mathfrak{T}, \sigma)$. Recall that PH is the set of all problems that can be solved on an alternating TM in polynomial time with a constant number of alternations. As before, it suffices to prove the result for sentences. Assume that $\varphi$ is a sentence of $\mathrm{RC}(\mathfrak{T}, \sigma)$ and let $\mathcal{A}$ be a $\sigma$-structure with active domain $\{T_1, \ldots, T_n\}$. Let $D = \bigcup_{i \leq n} \mathrm{dom}(T_i)$. By Theorem 5.2, we may assume that quantifiers in $\varphi$ range over trees over $D$. Therefore, for every existential quantifier $\exists x$ the machine simply guesses a tree $T_x$ whose domain is in

$D$. Since the size of $D$ is polynomial in the input size, this guessing takes polynomial time. For a universal quantifier, the machine does the same, except that guessing is done in a universal state. At the end, the machine just has to verify a Boolean combination of atomic statements where every variable $x$ is substituted by a tree $T_x$. The latter is clearly possible in deterministic polynomial time. As the total number of alternations is bounded by the number of quantifier alternations of the formula the overall complexity is in PH.

We now turn to the lower bound. We note that since $\mathrm{RC}(\mathfrak{T}, \sigma)$ subsumes the corresponding calculus over the universal string structure $\mathfrak{S}$, the lower bound for $\mathrm{RC}(\mathfrak{T}, \sigma)$ follows from the hardness result in [Benedikt et al. 2003]. So we focus on $\mathrm{RC}(\mathfrak{T}_\mathfrak{p}, \sigma)$. We let $S$ be the set of all trees whose domain is a subset of $1^*$, and whose labeling is identically $a$ for a fixed $a \in \Sigma$. Assume, without loss of generality, that $\sigma$ contains one binary relation $E$. We show that for any MSO sentence $\Phi$ over $\sigma$ there exists a $\mathrm{RC}(\mathfrak{T}_\mathfrak{p}, \sigma)$ sentence $\varphi$ such that for any $\mathcal{A}$ with $adom(\mathcal{A}) \subseteq S$, $\mathcal{A} \models \Phi$ iff $\mathcal{A} \models \varphi$. Since MSO can express problems complete for any level of PH over graphs, this suffices to prove the lower bound.

To see that MSO can be modeled in $\mathrm{RC}(\mathfrak{T}_\mathfrak{p}, \sigma)$, notice that a set $X$ of trees with domains $\{\ \{1^j \mid j \leq i\}\ \mid\ i \in \{m_1, \ldots, m_k\}\}$, $m_1 < \ldots < m_k$, and labeling $a$ is uniquely identified by the tree $T(X)$ whose domain consists of $\{1^i \mid i \leq m_k\}$ and $\{1^{m_j} \cdot 2 \mid j \leq k\}$. That is, one has a formula $\alpha(T_1, T_2)$ over $\mathfrak{T}_\mathfrak{p}$ such that $\alpha(T, T(X))$ iff $T \in X$. With this, MSO over $\sigma$ is straightforwardly encoded in $\mathrm{RC}(\mathfrak{T}_\mathfrak{p}, \sigma)$. This completes the proof. □

5.2.2 *Generic data complexity and expressivity bounds.* The PH bounds on query expressivity might lead one to imagine that arbitrary NP-complete calculations on tree sets can be performed by tree extension queries. We show, however, that the complexity of *generic* queries is in fact quite low. A generic (Boolean) query is just an isomorphism type $Q$ of $\sigma$-structures. We say that $Q$ is expressible in $\mathrm{RC}(\mathfrak{M}, \sigma)$ if there is a sentence $\Phi$ of $\mathrm{RC}(\mathfrak{M}, \sigma)$ such that for any $\sigma$-structure $\mathcal{A}$ over $\mathfrak{M}$, $(\mathfrak{M}, \mathcal{A}) \models \Phi$ if $\mathcal{A}$ is of isomorphism type $Q$.

Normally, data complexity of a Boolean query $Q$ is defined as the complexity of the language that consists of encodings of structures $\mathcal{A} \in Q$. If $\mathcal{A}$ is a relation over $\mathrm{T{\scriptsize REE}}_n(\Sigma)$, such an encoding must also encode all the trees in $adom(\mathcal{A})$. Since in generic queries it is irrelevant which trees belong to $adom(\mathcal{A})$, we can use a different encoding, where elements of the active domain, of size $k$, are encoded as $1, 2, \ldots, k$ in binary, just as in the case of relational calculus without any additional constraints [Abiteboul et al. 1995]. We denote such an encoding by $enc_{\mathrm{gen}}(\mathcal{A})$, and say that *generic data complexity* of $\mathrm{RC}(\mathfrak{M})$ is in a complexity class $K$ if for any $\sigma$ and any generic query $Q$ expressible in $\mathrm{RC}(\mathfrak{M}, \sigma)$, the language $\{enc_{\mathrm{gen}}(\mathcal{A}) \mid \mathcal{A} \in Q\}$ is in $K$.

THEOREM 5.8. *Generic data complexity of* $\mathrm{RC}(\mathfrak{T})$ *is in (uniform)* $\mathrm{AC}^0$.

*Proof.* Recall the structure $\mathfrak{S} = \langle \Sigma^*, <, L_a, \mathrm{el} \rangle$ [Blumensath and Gräel 2000; Benedikt et al. 2003]. We augment its vocabulary with definable functions $g_a, a \in \Sigma$, that append the symbol $a$ at the end of the string. We shall show the following. Suppose $\Phi$ in $\mathrm{RC}(\mathfrak{T}, \sigma)$ defines a generic query. Then there is a sentence $\Psi$ in $\mathrm{RC}(\mathfrak{S}, \sigma)$ that defines a generic query, and, for every $\mathcal{A}$ over $\mathfrak{T}$, there exists a $\sigma$-structure $\mathcal{B}$ over $\mathfrak{S}$ such that $\mathcal{A}$ and $\mathcal{B}$ are isomorphic as $\sigma$-structures, and $\mathcal{A} \models \Phi$ iff

$\mathcal{B} \models \Psi$. This implies that generic data complexity of $\Phi$ is the same as generic data complexity of $\Psi$, because $enc_{\text{gen}}(\mathcal{A}) = enc_{\text{gen}}(\mathcal{B})$, and the generic data complexity of $\text{RC}(\mathfrak{S}, \sigma)$ is known to be in uniform $\text{AC}^0$, see [Benedikt et al. 2003].

By Theorem 5.2, we can assume that quantification in $\Phi$ is restricted to trees whose domains lie in $\text{dom}(\mathcal{A})$. Consider $\Phi$ restricted to structures $\mathcal{A}$ such that for every $T \in adom(\mathcal{A})$, $\text{dom}(T) \subset 1^*$. Every tree $T$ with $\text{dom}(T) \subset 1^*$ is uniquely identified by a string $s_T$ whose $i$th position contains the label of $1^{i-1}$. In $\Phi$ restricted to such trees, quantification is also restricted to trees with domain contained in $1^*$, that is, to strings. Furthermore, for such trees $T_1 \approx_{\text{dom}} T_2$ iff $\text{el}(s_{T_1}, s_{T_2})$, $T_1 \prec T_2$ iff $s_{T_1} < s_{T_2}$, $T_1 = \text{rsucc}(T_2)$ never holds, $T_1 = \text{lsucc}(T_2)$ iff $\bigvee_{a \in \Sigma} g_a(s_{T_2}) = s_{T_1}$ holds, and $L_a(T)$ iff $L_a(s_T)$. We thus obtain $\Psi$ from $\Phi$ by replacing all the $\mathfrak{T}$ predicates by $\mathfrak{S}$ predicates as above.

From the construction and Theorem 5.2, we see that whenever $\mathcal{A}$ is a structure over $\mathfrak{T}$ with $\text{dom}(\mathcal{A}) \subset 1^*$, and $\mathcal{A}_{str}$ is obtained from $\mathcal{A}$ by replacing each $T$ with $s_T$, then $\mathcal{A} \models \Phi$ iff $\mathcal{A}_{str} \models \Psi$. Furthermore, $\mathcal{A}$ and $\mathcal{A}_{str}$ are isomorphic as $\sigma$-structures. This also implies that $\Psi$ is generic. Indeed, for two structures $\mathcal{B}, \mathcal{B}'$ over $\mathfrak{S}$ which are isomorphic as $\sigma$-structures, let $\mathcal{A}$ and $\mathcal{A}'$ be structures over $\mathfrak{T}$ such that $\mathcal{A}_{str} = \mathcal{B}$ and $\mathcal{A}'_{str} = \mathcal{B}'$. Then $\mathcal{B} \models \Psi$ iff $\mathcal{A}_{str} \models \Phi$ iff (by genericity of $\Phi$) $\mathcal{A}'_{str} \models \Phi$ iff $\mathcal{B}' \models \Psi$. Thus, $\Psi$ satisfies all the conditions listed above, which completes the proof of $\text{AC}^0$ generic data complexity. $\square$

From $\text{AC}^0$ lower bounds (cf. [Immerman 1998]), we obtain:

COROLLARY 5.9. *Parity test and connectivity test are not definable in* $\text{RC}(\mathfrak{T})$. $\square$

### 5.3 Relational calculi and unranked trees

We start by giving a few XML-motivated examples of queries one may want to express over unranked trees. First, we briefly review DTDs [Bray et al. 2000] and XPath [Clark and DeRose 1999]. Document Type Definitions (DTDs) is the most commonly used schema language for XML. It can be abstracted by extended context-free grammars (with regular expressions as right-hand sides of productions). Formally, a DTD over $\Sigma$ is a pair $(s, d)$ where $s \in \Sigma$ is the start symbol and $d : \Sigma \to 2^{\Sigma^*}$ maps every $\Sigma$-symbol to a regular language over $\Sigma$. A tree $T = (D, f)$ conforms to $d$ iff $f(v \cdot 1) \cdots f(v \cdot n) \in d(f(v))$ for every $v \in D$ with $n$ children and the root of $T$ is labeled with $s$.

XPath [Clark and DeRose 1999] is an XML pattern language employed by several XML transformation languages like XSLT [Clark 1999] and XQuery [Chamberlin et al. 2002]. We have the core XPath fragment in mind, which is normally defined by the following grammar:

$$p := p_1 | p_1 \;\mid\; /p \;\mid\; p_1/p_2 \;\mid\; p_1//p_2 \;\mid\; p_1[p_2] \;\mid\; \sigma \;\mid\; *$$

We refrain from giving a direct formal semantics, but, instead consider a logic containing this fragment. Recall that $\nu_\Sigma$ is the vocabulary for unranked trees as defined in Section 4.1. Let $\mathcal{FO}(\exists^*)$ be the fragment of $\mathcal{FO}$ over $\nu_\Sigma$ consisting of formulae $\varphi(x, y)$ in prenex normal form and all quantifiers existential. Additionally, formulae can make use of the unary predicates $\text{root}(x)$, $\text{leaf}(x)$, $\text{first}(x)$, and $\text{last}(x)$ (denoting that $x$ is the root, a leaf, the first and the last child, respectively) and the binary predicate $\text{succ}(x, y)$ (denoting that $y$ is the right sibling of $x$). Note that these predicates are $\mathcal{FO}$-definable but not by existential formulae. A pattern $\varphi(x,y)$ is always evaluated against some context node $u$; we write $\varphi(u,T)$ for the set $\{v \mid T \models \varphi(u,v)\}$. For logical characterizations of fragments of XPath, we refer to [Marx 2004].

Using connections between FO over unranked tree structures and $\mathcal{MSO}_{\rightarrow}^{\downarrow}$, one can easily verify the following.

PROPOSITION 5.10. *—For any DTD d, there exists an* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ *formula* $\varphi_d$ *with one free variable such that* $\mathfrak{T}^{\mathrm{u}} \models \varphi_d(T)$ *iff* $T$ *conforms to d.*

*—For every XPath expression* $e = \psi(x,y)$*, there exists an* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ *formula* $\varphi_e(T,t,t')$ *such that* $\mathfrak{T}^{\mathrm{u}} \models \varphi_e(T,t,t')$ *iff* $t$*,* $t'$ *are branches of* $T$ *with leaves* $u,u'$*, and* $\psi(u,u')$ *holds in* $T$*.*

We study the expressiveness and complexity of $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}},\sigma)$ and $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}},\sigma)$, and show that the situation is reminiscent of that for ranked trees: one can prove a quantifier-restriction result for those calculi that gives us a PH (polynomial hierarchy) upper bound on query evaluation. The proof, however, is not an immediate consequence of its ranked counterpart. The reason is that all known quantifier-restriction results for ranked trees, when applied to trees of the form $\mathcal{R}(T)$, involve quantification over trees that are *not* encodings of unranked trees. This, as before, is remedied by combining the encoding techniques with some (restricted) EF games.

THEOREM 5.11. *(1) The data complexity of both* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}},\sigma)$ *and* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}},\sigma)$ *is PH;*

*(2) The generic data complexity of both* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}},\sigma)$ *and* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}},\sigma)$ *is* $\mathrm{AC}^0$*.*

*Proof.* Both 1) and 2) are based on the following quantifier-restriction result. By $\mathrm{FO}_{\mathrm{dom}}(\mathfrak{M},\sigma)$, where $\mathfrak{M}$ is one of the ranked or unranked tree models considered here, we denote the set of $\mathrm{FO}(\mathfrak{M},\sigma)$ sentences in which quantification is restricted to $\mathrm{dom}(\mathcal{A})$. Recall that $\mathrm{dom}(\mathcal{A}) = \bigcup_{T\in adom(\mathcal{A})} \mathrm{dom}(T)$, where $\mathcal{A}$ is a $\sigma$-structure. That is, $(\mathfrak{M},\mathcal{A}) \models \exists T\ \psi(T,\cdot)$ means that for some $T_0$ with $\mathrm{dom}(T_0) \subseteq \mathrm{dom}(\mathcal{A})$, $(\mathfrak{M},\mathcal{A}) \models \psi(T_0,\cdot)$.

LEMMA 5.12. *Every* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}},\sigma)$ *sentence is equivalent to an* $\mathrm{FO}_{\mathrm{dom}}(\mathfrak{T}^{\mathrm{u}},\sigma)$ *sentence.*

The proof of the lemma is by reduction to a similar result for the ranked trees ($\mathrm{FO}(\mathfrak{T},\sigma) = \mathrm{FO}_{\mathrm{dom}}(\mathfrak{T},\sigma)$), but combining it with the translation $\mathcal{R}(\cdot)$ requires some work. Let $\mathcal{A}$ be a $\sigma$-structure over $\mathrm{UTREE}(\Sigma)$. Then $\mathcal{R}(\mathcal{A})$ is a $\sigma$-structure over $\mathrm{TREE}(\Sigma_\perp)$ whose active domain is $\{\mathcal{R}(T) \mid T \in adom(\mathcal{A})\}$, and which contains, for every tuple $(T_1,\ldots,T_n)$ in an $n$-ary relation $S$ in $\mathcal{A}$, a tuple $(\mathcal{R}(T_1),\ldots,\mathcal{R}(T_n))$. Recall that $\mathfrak{T}^{\mathrm{u}}[\mathcal{A}]$ stands for the structure $(\mathfrak{T}^{\mathrm{u}},\mathcal{A})$ in which the universe is the set of all trees $T$ with $\mathrm{dom}(T) \subseteq \mathrm{dom}(\mathcal{A})$; we define $\mathfrak{T}[\mathcal{R}(\mathcal{A})]$ likewise.

The proof of Lemma 5.12 will be based on the following claim.

CLAIM 5.13. *There exist two functions* $f,g : \mathbb{N} \to \mathbb{N}$ *such that, for any two* $\sigma$*-structures* $\mathcal{A},\mathcal{B}$ *over* $\mathrm{UTREE}(\Sigma)$ *and every* $k \geq 0$*:*

*(1)* $\mathfrak{T}^{\mathrm{u}}[\mathcal{A}] \equiv_{f(k)} \mathfrak{T}^{\mathrm{u}}[\mathcal{B}]$ *implies* $\mathfrak{T}[\mathcal{R}(\mathcal{A})] \equiv_k \mathfrak{T}[\mathcal{R}(\mathcal{B})]$*;*

*(2)* $(\mathfrak{T},\mathcal{R}(\mathcal{A})) \equiv_{g(k)} (\mathfrak{T},\mathcal{R}(\mathcal{B}))$ *implies* $(\mathfrak{T}^{\mathrm{u}},\mathcal{A}) \equiv_k (\mathfrak{T}^{\mathrm{u}},\mathcal{B})$*.*

First, we see how Lemma 5.12 follows from this. From Lemma 5.3, we know that there is a function $h : \mathbb{N} \to \mathbb{N}$ such that $\mathfrak{T}[\mathcal{R}(\mathcal{A})] \equiv_{h(k)} \mathfrak{T}[\mathcal{R}(\mathcal{A})]$ implies $(\mathfrak{T}, \mathcal{R}(\mathcal{A})) \equiv_k (\mathfrak{T}, \mathcal{R}(\mathcal{A}))$. Hence, for every $k \geq 0$,

$$\begin{array}{ccccl} \mathfrak{T}^{\mathrm{u}}[\mathcal{A}] & \equiv_{f \circ h \circ g(k)} & \mathfrak{T}^{\mathrm{u}}[\mathcal{B}] & & \\ & \Downarrow & & & \text{by Claim 5.13, 1)} \\ \mathfrak{T}[\mathcal{R}(\mathcal{A})] & \equiv_{h \circ g(k)} & \mathfrak{T}[\mathcal{R}(\mathcal{B})] & & \\ & \Downarrow & & & \text{by Lemma 5.3} \\ (\mathfrak{T}, \mathcal{R}(\mathcal{A})) & \equiv_{g(k)} & (\mathfrak{T}, \mathcal{R}(\mathcal{B})) & & \\ & \Downarrow & & & \text{by Claim 5.13, 2)} \\ (\mathfrak{T}^{\mathrm{u}}, \mathcal{A}) & \equiv_k & (\mathfrak{T}^{\mathrm{u}}, \mathcal{B}) & & \end{array}$$

Thus, every quantifier-rank $k$ sentence of $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ is equivalent to a quantifier-rank $f(h(g(k)))$ sentence of $\mathrm{FO}_{\mathrm{dom}}(\mathfrak{T}^{\mathrm{u}}, \sigma)$, which proves Lemma 5.12.

We now prove Claim 5.13, part 1). For that, we need a way of coding a ranked tree $T$ with $\mathrm{dom}(T) \subseteq \mathrm{dom}(\mathcal{R}(X))$, where $X$ is a set of unranked trees. Such trees are either single nodes, or every node in their domains is of the form $01s$ where $s$ does not contain an occurrence of 00.

We fix the alphabet of unranked trees to be $\Sigma = \{a, b\}$ (and hence the alphabet of ranked translations will be $\Sigma_\perp = \{a, b, \perp\}$). Extension to larger alphabets is straightforward.

Not all ranked trees are translations of unranked trees, but each ranked tree can be encoded by a tuple of unranked trees in a way that allows the game to proceed. For this, we define special 8-tuples of unranked trees. We say that a tuple $\vec{V} = (V_1, \ldots, V_8)$ of unranked trees over $\Sigma$ is *nice* if $\mathrm{dom}(V_1) = \ldots = \mathrm{dom}(V_8) = D$, and

—nodes labeled $a$ in $V_1, V_2, V_3$ form a partition of $D$;

—only non-leaf nodes can be labeled $a$ in $V_4, V_5$, and no node is labeled $a$ in both $V_4$ and $V_5$;

—only leaf nodes can be labeled $a$ in $V_6, V_7, V_8$, and a node can be labeled $a$ in at most one of these trees.

Given a nice tuple $\vec{V}$, we associate a tree $\mathcal{T}(\vec{V})$ in $\mathrm{T{\scriptsize REE}}(\Sigma_\perp)$ with it as follows. Its domain contains $D'$, the prefix-closure of $\mathcal{R}(D)$. For the root node or a node $s$ of the form $s' \cdot 1$ in $D'$, let $d = \mathcal{R}^{-1}(s)$. Then

$$\text{label of } s \text{ in } \mathcal{T}(\vec{V}) \quad = \quad \begin{cases} a & \text{if } d \text{ is labeled } a \text{ in } V_1 \\ b & \text{if } d \text{ is labeled } a \text{ in } V_2 \\ \perp & \text{if } d \text{ is labeled } a \text{ in } V_3 \end{cases}$$

Other nodes in $D'$ are of the form $s = s' \cdot 0$. Let $d' = \mathcal{R}^{-1}(s')$. Note that $d'$ is not a leaf node. Then

$$\text{label of } s \text{ in } \mathcal{T}(\vec{V}) \quad = \quad \begin{cases} a & \text{if } d' \text{ is labeled } a \text{ in } V_4 \\ b & \text{if } d \text{ is labeled } a \text{ in } V_5 \\ \perp & \text{if } d \text{ is labeled } b \text{ in both } V_4 \text{ and } V_5 \end{cases}$$

Finally, consider a leaf node $d$ in $D$. Let $s = \mathcal{R}(d)$. If $d$ is labeled $a$ in one of

$V_6, V_7, V_8$, we add the node $s \cdot 0$ to the domain of $\mathcal{T}(\vec{V})$ and label it as follows:

$$\text{label of } s \cdot 0 \text{ in } \mathcal{T}(\vec{V}) \quad = \quad \begin{cases} a & \text{if } d \text{ is labeled } a \text{ in } V_6, \\ b & \text{if } d \text{ is labeled } a \text{ in } V_7, \text{ and} \\ \bot & \text{if } d \text{ is labeled } a \text{ in } V_8. \end{cases}$$

This completes the description of $\mathcal{T}(\vec{V})$.

Next, suppose we have a ranked tree $T$ which is either a single node, or whose domain, $D_T$, contains strings of the form $01 \cdot s$ where $s$ does not contain 00. We now show how to code it by a tuple $\vec{V}$ so that $\mathcal{T}(\vec{V}) = T$. First, let $D_T^0$ be the set of leaf nodes of the form $s \cdot 0 \in D_T$. Then $D_T^1 = D_T - D_T^0$ has the property that for some unranked tree domain $D$, $D_T^1$ is the prefix closure of $\mathcal{R}(D)$.

We now define $V_1, \ldots, V_8$ over $D$ in the natural way: that is, in $V_i$, $i = 1, 2, 3$, we label $a$ the nodes whose image is labeled $a$, $b$, $\bot$, respectively, in $T$; in $V_4$ and $V_5$ we label by $a$ the nodes $d$ such that $\mathcal{R}(d) \cdot 0$ is labeled $a$ (respectively $b$) in $T$, and in $V_i$, $i = 6, 7, 8$, we label by $a$ the leaves $d$ such that $\mathcal{R}(d) \cdot 0$ is in $D_T^0$, and is labeled $a$, $b$, or $\bot$, respectively. It is routine to show that $\mathcal{T}(\vec{V}) = T$.

Note, furthermore, that if for some unranked tree $T_u$, we have $T = \mathcal{R}(T_u)$, then $T_u = V_1$, and furthermore, there is a formula $\beta(\vec{V})$ over $\mathfrak{T}^{\mathrm{u}}$ that holds iff $\vec{V}$ codes an image, under $\mathcal{R}(\cdot)$, of an unranked tree. Moreover, it is easy to see that that there are $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}})$ formulae $\alpha_0(\vec{V})$, $\alpha_{P_1}(\vec{V})$, and $\alpha_{P_2}(\vec{V}, \vec{W})$, where $P_1, P_2$ range over unary and binary predicates in the vocabulary of $\mathfrak{T}$, such that

—$\alpha_0(\vec{V})$ holds iff $\vec{V}$ is nice;

—$\alpha_{P_1}(\vec{V})$ holds iff $P_1(\mathcal{T}(\vec{V}))$ holds; and

—$\alpha_{P_2}(\vec{V}, \vec{W})$ holds iff $P_2(\mathcal{T}(\vec{V}), \mathcal{T}(\vec{W}))$ holds.

We now take $c$ to be the maximum quantifier rank of $\alpha_0, \alpha_{P_1}, \alpha_{P_2}$, and $\beta$, and define $f(k) = 8k + c$.

We now show how to play the game $(\mathfrak{T}, \mathcal{R}(\mathcal{A})) \equiv_k^{\mathrm{dom}} (\mathfrak{T}, \mathcal{R}(\mathcal{B}))$. The moves in the game will be denoted by $T_{\mathcal{A}}^i$ and $T_{\mathcal{B}}^i$. Corresponding to those moves, we will have moves $V_j^i, W_j^i$, $j = 1, \ldots, 8$, played in the game $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A}) \equiv_{8k+c}^{\mathrm{dom}} (\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$.

Suppose the spoiler plays $T_{\mathcal{A}}^i$ over $(\mathfrak{T}, \mathcal{R}(\mathcal{A}))$ in round $i$ (the case when the spoiler plays over $(\mathfrak{T}, \mathcal{R}(\mathcal{B}))$ is symmetric). Let $\vec{V}^i$ be such that $\mathcal{T}(\vec{V}^i) = T_{\mathcal{A}}^i$. The duplicator assumes that the spoiler plays 8 moves $\vec{V}^i$ in the game on $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A})$ and $(\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$ from the position $\big((V_j^l \mid l < i, j = 1, \ldots, 8),\ (W_j^l \mid l < i, j = 1, \ldots, 8)\big)$ and finds the response $\vec{W}^i$ according to his winning strategy. Since $\alpha_0(\vec{V}^i)$ holds, by the assumption we have $\alpha_0(\vec{W}^i)$ and hence there is a ranked tree $T'$ such that $\mathcal{T}(\vec{W}^i) = T'$. The duplicator then responds by playing $T_{\mathcal{B}}^i := T'$.

It remains to show that the strategy works. Suppose for some binary predicate $P_2$ in the vocabulary of $\mathfrak{T}$, we have $P_2(T_{\mathcal{A}}^{i_1}, T_{\mathcal{A}}^{i_2})$. Then $\alpha_{P_2}(\vec{V}^{i_1}, \vec{V}^{i_2})$ holds, and hence $\alpha_{P_2}(\vec{W}^{i_1}, \vec{W}^{i_2})$ holds, which implies $P_2(T_{\mathcal{B}}^{i_1}, T_{\mathcal{B}}^{i_2})$. The proof of preservation of unary predicates is the same.

Finally, assume that for some $m$-ary predicate $S$ from $\sigma$, $S(T_{\mathcal{A}}^{i_1}, \ldots, T_{\mathcal{A}}^{i_m})$ holds in $\mathcal{R}(\mathcal{A})$. Then, for each $l \leq m$, there exists an unranked tree $U_{\mathcal{A}}^{i_l}$ in $adom(\mathcal{A})$ such that $T_{\mathcal{A}}^{i_l} = \mathcal{R}(U_{\mathcal{A}}^{i_l})$, and $S(U_{\mathcal{A}}^{i_1}, \ldots, U_{\mathcal{A}}^{i_m})$ holds in $\mathcal{A}$. Furthermore, this means that

$\beta(\vec{V}^{i_l})$ holds, and $V_1^{i_l} = U_{\mathcal{A}}^{i_l}$, $l \le m$. By the definition of $f(k)$, we conclude that $\beta(\vec{W}^{i_l})$ for each $l \le m$, and thus $\mathcal{R}(W_1^{i_l}) = T_{\mathcal{B}}^{i_l}$. Since $S(V_1^{i_1}, \dots, V_1^{i_m})$ holds in $\mathcal{A}$, by the assumption on the game on $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A})$ and $(\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$ we have $S(W_1^{i_1}, \dots, W_1^{i_m})$ in $\mathcal{B}$, and hence $S(T_{\mathcal{B}}^{i_1}, \dots, T_{\mathcal{B}}^{i_m})$ in $\mathcal{R}(\mathcal{B})$.

This shows that the duplicator has the $k$-round winning strategy, and completes the proof of part 1) of the Claim.

Next we prove Claim 5.13, part 2). Let $c$ be the constant which is the maximum quantifier rank of formulae $\varphi_P$ such that $\mathfrak{T}^{\mathrm{u}} \models P(T_1, T_2)$ iff $\mathfrak{T} \models \varphi_P(\mathcal{R}(T_1), \mathcal{R}(T_2))$, where $P$ is an atomic formula, and the formula over $\mathfrak{T}$ defining the range of $\mathcal{R}(\cdot)$. Once can see from the proof of Lemma 4.3 that $c$ is a constant and does not depend on $k$. We then take $g(k) = k + c + 1$.

The moves in the $i$th round of the game $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A}) \equiv_k (\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$ will be denoted by $T_{\mathcal{A}}^i$ and $T_{\mathcal{B}}^i$, respectively. With each such pair of moves, we will have associated moves $V_{\mathcal{A}}^i$ and $V_{\mathcal{B}}^i$ for the game $(\mathfrak{T}, \mathcal{R}(\mathcal{A})) \equiv_{g(k)} (\mathfrak{T}, \mathcal{R}(\mathcal{B}))$. The game on $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A})$ and $(\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$ is played as follows. Suppose that in round $i$, the spoiler plays $T_{\mathcal{A}}^i$ in $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A})$ (the case when the spoiler plays in $(\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$ is similar). The duplicator then sets $V_{\mathcal{A}}^i = \mathcal{R}(T_{\mathcal{A}}^i)$, assumes that the spoiler plays $V_{\mathcal{A}}^i$ in the position $\big((V_{\mathcal{A}}^j, j < i),\ (V_{\mathcal{B}}^j, j < i)\big)$ in the game on $(\mathfrak{T}, \mathcal{R}(\mathcal{A}))$ and $(\mathfrak{T}, \mathcal{R}(\mathcal{B}))$, and finds the response $V_{\mathcal{B}}^i$.

Let $\alpha(T)$ be the formula over $\mathfrak{T}$ stating that $T$ is in the image of $\mathcal{R}(\cdot)$. Since its quantifier rank is at most $c$ and $\alpha(V_{\mathcal{A}}^i)$ holds, then $\alpha(V_{\mathcal{B}}^i)$ holds, and therefore there exists an unranked tree $T'$ which is mapped to $V_{\mathcal{B}}^i$ by $\mathcal{R}(\cdot)$. We then set $T_{\mathcal{B}}^i{:=}T'$.

It remains to show that this provides a winning strategy for the duplicator. Suppose for some $m$-ary relation symbol $S$ in $\sigma$, we have $(T_{\mathcal{A}}^{i_1}, \dots, T_{\mathcal{A}}^{i_m}) \in S^{\mathcal{A}}$. Then

$$(V_{\mathcal{A}}^{i_1}, \dots, V_{\mathcal{A}}^{i_m}) = (\mathcal{R}(T_{\mathcal{A}}^{i_1}), \dots, \mathcal{R}(T_{\mathcal{A}}^{i_m})) \in S^{\mathcal{R}(\mathcal{A})},$$

and by the assumption, $(V_{\mathcal{B}}^{i_1}, \dots, V_{\mathcal{B}}^{i_m}) \in S^{\mathcal{R}(\mathcal{B})}$. Hence, $(T_{\mathcal{B}}^{i_1}, \dots, T_{\mathcal{B}}^{i_m}) \in S^{\mathcal{B}}$. A symmetric argument shows that $(T_{\mathcal{B}}^{i_1}, \dots, T_{\mathcal{B}}^{i_m}) \in S^{\mathcal{B}}$ implies $(T_{\mathcal{A}}^{i_1}, \dots, T_{\mathcal{A}}^{i_m}) \in S^{\mathcal{A}}$.

Next, suppose $P(T_{\mathcal{A}}^i, T_{\mathcal{A}}^j)$ holds, where $P$ is an atomic formula over $\mathfrak{T}^{\mathrm{u}}$. Then by Lemma 4.3 $\varphi_P(V_{\mathcal{A}}^i, V_{\mathcal{A}}^j)$ holds and, since $g(k) = k + c$ and the quantifier rank of $\varphi_P$ is at most $c$, $\varphi_P(V_{\mathcal{B}}^i, V_{\mathcal{B}}^j)$ holds, which in turn implies that $P(T_{\mathcal{B}}^i, T_{\mathcal{B}}^j)$ holds. This, and the symmetric argument, show that $\{(T_{\mathcal{A}}^i, T_{\mathcal{B}}^i) \mid i \le k\}$ is a $\mathfrak{T}^{\mathrm{u}}$-partial isomorphism, and hence, by the previous paragraph, a partial isomorphism between $(\mathfrak{T}^{\mathrm{u}}, \mathcal{A})$ and $(\mathfrak{T}^{\mathrm{u}}, \mathcal{B})$. This completes the proof of Claim 1, part 2).

*Proof of Theorem 5.11.* Lemma 5.12 gives the proof of the first part of the theorem, since with restriction of quantification to $\mathrm{dom}(\mathcal{A})$, one can straightforwardly check the validity of a sentence by using an alternating polynomial time algorithm.

The second part is by using Lemma 5.12 and a reduction to the ranked case. Suppose $Q$ is a generic query expressible in $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ by a sentence $\Phi$. From Lemma 5.12, we may assume that quantification in $\Phi$ is already restricted. Suppose $\mathcal{A}$ is such that $adom(\mathcal{A}) \subset \textsc{Tree}(\Sigma)$. Then all quantification in $\Phi$ is also restricted to $\textsc{Tree}(\Sigma)$. Since the restrictions of predicates $\prec_{\rightarrow}$ and $\prec_{\downarrow}$ to $\textsc{Tree}(\Sigma)$ can be expressed in $\mathrm{FO}(\mathfrak{T})$, we see, by genericity, that there is a sentence $\Phi'$ in $\mathrm{FO}(\mathfrak{T}, \sigma)$ that expresses $Q$. Hence, the generic data complexity of $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ is at most the generic data complexity of $\mathrm{FO}(\mathfrak{T}, \sigma)$, which was shown to be $\mathrm{AC}^0$. This completes the proof of the theorem. □

Corollary 5.14. *Parity and transitive closure cannot be expressed in* $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}, \sigma)$.

The upper bound of Theorem 5.11 cannot be lowered, since the relational calculi we introduced can express problems complete for each level of PH. Moreover, this can be done in a rather simple setting; for example, over $\mathfrak{T}$ and $\mathfrak{T}^{\mathfrak{u}}$, all one needs is one unary relation and quantification over branches.

Proposition 5.15. *Let* $\sigma_1$ *contain one unary relation* $U$*, and* $\sigma_2$ *one binary relation* $E$*. Then for every* $i$*, both* $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma_2)$ *and* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathfrak{u}}, \sigma_1)$ *(in fact, even* $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$*) can define* $\Sigma^p_i$*- and* $\Pi^p_i$*-hard problems.*

*Proof.* We have seen in Theorem 5.7 that for every $i$, $\mathrm{FO}(\mathfrak{T}_{\mathfrak{p}}, \sigma_2)$ can define $\Sigma^p_i$- and $\Pi^p_i$-hard problems. As $\mathrm{FO}(\mathfrak{T}_{\mathfrak{p}}, \sigma_2)$ is easily definable in $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma_2)$, the first result follows. The case of $\mathrm{FO}_\eta(\mathfrak{T}^{\mathfrak{u}}, \sigma_1)$ in the presence of a relation of arity at least two follows from the corresponding results for the string model [Benedikt et al. 2003].

We now turn to the proof for $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$. This is a reduction from QSAT. For each quantifier-prefix $\pi$, we construct an $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$ formula $\xi$ such that, if a unary relation $S$ models a CNF formula $\varphi$, then $(\mathfrak{T}, S) \models \xi$ iff $\varphi$ preceded by the prefix $\pi$ is satisfiable.

Assume a formula of the form $\psi := \theta_1 Y_1 \theta_2 Y_2 \cdots \theta_k Y_k \varphi$ where $\varphi$ is the conjunction of $\ell$ disjuncts $C_i$, each $Y_i$ is a sequence of variables, and each $\theta_i$ is $\exists$ or $\forall$. Let $x_1, \ldots, x_n$ be the variables in $\varphi$. We can assume that no disjunct contains both $x_i$ and $\neg x_i$.

The unary database relation, which we will name $S$, contains $\ell$ trees $t_1, \ldots, t_\ell$. Every tree is a unary branch of depth $n-1$ (so, containing $n$ nodes), every node has at most one child and is labeled with a label from $(\{\mathrm{pos}, \mathrm{neg}\} \times \{1, \ldots, k\}) \cup \{0\}$. Intuitively, the tree $t_i$ encodes the disjunct $C_i$. The node at depth $j$ is labeled $(\mathrm{pos}, r), (\mathrm{neg}, r)$, or 0 when $x_{j+1}$ occurs positively, negatively or does not occur in $C_i$, respectively, and $x_{j+1}$ is in $Y_r$.

Next, we construct the $\mathrm{FO}(\mathfrak{T})$-formula $\xi$ which is obtained from $\psi$ as follows. For clarity, we write $\exists t \in S$ rather than $\exists t \in \mathit{adom}$. We precede $\psi$ by $(\exists t \in S)$. We replace every quantifier $\exists Y_i \ldots$ by $\exists^\eta s_i (t \approx_{\mathrm{dom}} s_i \wedge \alpha_{01}(s_i) \wedge \ldots)$ and every quantifier $\forall Y_i \ldots$ by $\forall^\eta s_i (t \approx_{\mathrm{dom}} s_i \wedge \alpha_{01}(s_i) \to \ldots)$. Here, $\alpha_{01}(s_i)$ is the formula expressing that every node in $s_i$ is labeled with 0 or 1. Intuitively, every $s_i$ represents a truth assignment for all variables. Of course, we will only use $s_i$ to determine the truth value of the variables specified in $Y_i$.

The formula $\varphi$ is replaced by

$$\begin{aligned}&(\forall t' \in S)(\exists^\eta t'')(\exists^\eta s'')(t'' \approx_{\mathrm{dom}} s'' \wedge t'' \preceq t' \wedge \\ &\qquad\qquad \bigvee_{r \in \{1,\ldots,k\}} (O_{(\mathrm{pos},r)}(t'') \wedge s'' \preceq s_r \wedge O_1(s'')) \\ &\qquad\qquad\qquad \vee \bigvee_{r \in \{1,\ldots,k\}} (O_{(\mathrm{neg},r)}(t'') \wedge s'' \preceq s_r \wedge O_0(s''))\end{aligned}$$

Intuitively, the formula expresses that for every disjunct $t'$ in $S$ there must be at least one literal $t''$ that becomes true under the interpretation induced by $s_r$. Here,

$O_\sigma(t)$ denotes the formula expressing that the endpoint of $t$ is labeled with $\sigma$.

Note that $\xi$ only depends on the quantifier prefix of $\psi$, not on $\varphi$ or the sets $Y_i$.

Clearly, $(\mathfrak{T}, S) \models \xi$ iff $\psi \in$ QSAT. $\square$

### 5.4 Restricted query languages

Given the high bounds on the complexity of query languages, we propose some restricted relational calculi with lower data complexity, but still sufficiently expressive so that they can do, for example, DTD validation and XPath pattern-matching. The source of high complexity in a query language is the possibility of quantifying over the entire set of unranked trees. We therefore impose restrictions on such quantification.

Let $\mathrm{FO}^{\mathrm{act}}(\mathfrak{M}, \sigma)$ be the logic that is build from atomic formulae over $\sigma$ and *arbitrary* formulae of $\mathrm{FO}(\mathfrak{M})$ by using the Boolean connectives and quantification $\exists T \in adom$ and $\forall T \in adom$. If only $\mathrm{FO}_\eta(\mathfrak{M})$ formulae are used, we refer to $\mathrm{FO}^{\mathrm{act}}_\eta(\mathfrak{M}, \sigma)$.

Combining the $\mathrm{AC}^0$ data complexity of the relational calculus [Abiteboul et al. 1995] with the results of the previous section we get:

Corollary 5.16. *The data complexity of both* $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$ *and* $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^{\mathtt{u}}, \sigma)$ *is DLOGSPACE, and the data complexity of* $\mathrm{FO}^{\mathrm{act}}_\eta(\mathfrak{T}^{\mathtt{u}}, \sigma)$ *is* $\mathrm{NC}^1$. $\square$

Indeed, since the data complexity of the relational calculus is $\mathrm{AC}^0$, the data complexity of the calculi is the same as the data complexity of the corresponding formulae over $\mathfrak{T}^{\mathtt{u}}$ or $\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}$, which is then DLOGSPACE by [Gottlob et al. 2005]. If quantification is over branches only, the data complexity becomes $\mathrm{NC}^1$.

Notice that all these languages can do both DTD and XPath checking.

As another restriction, we use the logic $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}})$ of Section 4.3, which extends $\mathrm{FO}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}})$ with predicates $r^\downarrow(T)$ and $r^\rightarrow(T)$ testing if the labeling of the right boundary or the siblings of the rightmost node of a branch is in the language denoted by the regular expression $r$. By adding atomic formulae of the form $S(\vec{T})$ where $S \in \sigma$, and restricted quantification $\exists T \in adom$ and $\forall T \in adom$, we obtain a logic $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$. Note that this logic is closer to $\mathrm{FO}(\mathfrak{M}, \sigma)$ than to $\mathrm{FO}^{\mathrm{act}}(\mathfrak{M}, \sigma)$, since it can mix quantification over (a subset of) $\mathfrak{M}$ and quantification over the active-domain in an arbitrary way. Still, it has low data complexity.

Theorem 5.17. *The data-complexity of* $\mathrm{FO}^{reg}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$ *is* $\mathrm{NC}^1$.

*Proof.* By $\mathrm{FO}^{\mathrm{reg}}_{\eta,\mathrm{dom}}(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$, we denote the set of $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$ sentences in which quantification is restricted to branches of trees in $adom(\mathcal{A})$. That is, $(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \mathcal{A}) \models \exists T\ \psi(T, \cdot)$ means that for some branch $T_0$ of a tree $T_1 \in adom(\mathcal{A})$, $(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \mathcal{A}) \models \psi(T_0, \cdot)$.

We need the following lemma.

Lemma 5.18. *Every* $\mathrm{FO}^{reg}_\eta(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$ *sentence is equivalent to an* $FO^{reg}_{\eta,dom}(\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}, \sigma)$ *sentence.*

*Proof.* Denote by $\eta(\mathcal{A})$ the set of branches occurring in trees in the active domain of $\mathcal{A}$. Let $\mathfrak{T}^{\mathtt{u},R}_{\mathtt{p}}$ be the structure $\mathfrak{T}^{\mathtt{u}}_{\mathtt{p}}$ extended with the predicates $r^\rightarrow$ and $r^\downarrow$ for $r \in R$.

Let $(\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{A}) \equiv_k^{\eta(\mathcal{A},\mathcal{B})} (\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{B})$ denote the $k$-move game consisting of two kind of moves: (1) a move and the response on it is a branch in $\eta(\mathcal{A})$ and $\eta(\mathcal{B})$ or vice versa; or, (2) a move and the response on it is a tree in $adom(\mathcal{A})$ and $adom(\mathcal{B})$ or vice versa. By $(\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{A}) \equiv_k^{\eta} (\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{B})$, we denote the game where (1)-moves are not restricted to $\eta(\mathcal{A})$ and $\eta(\mathcal{B})$ but can be arbitrary branches.

We show that for any $k$ and any finite set of regular expressions $R$ there is an $\ell$ and a finite set of regular expressions $R'$ such that

$$(\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R'}, \mathcal{A}) \equiv_\ell^{\eta(\mathcal{A},\mathcal{B})} (\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R'}, \mathcal{B}) \text{ implies } (\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{A}) \equiv_k^{\eta} (\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R}, \mathcal{B}).$$

We refer to the left game as the *small* game and to the right game as the *big* game.

We set $\ell = k+3$. For every regular expression $r \in R$, let $A_r = (Q_r, \Sigma, q_0^r, \delta_r, F_r)$ be the equivalent DFA accepting $L(r)$. For every DFA, the following property holds:

for every $w_1, w_2 \in \Sigma^*$, $\delta_r^*(w_1) = \delta_r^*(w_2)$ implies $\delta_r^*(w_1 v) \in F_r$ iff $\delta_r^*(w_2 v) \in F_r$ for all $v \in \Sigma^*$. (*)

Here, $\delta_r^*$ is the transition function $\delta_r$ extended to strings. Let $A_r^q$ be the automaton $A$ where $F = \{q\}$ and let $s_r^q$ be the regular expression equivalent to $A_r^q$. So, if $w, v \in \Sigma^*$ satisfy $s_r^q$, then for all $z \in \Sigma^*$, $wz$ satisfies $r$ iff $vz$ satisfies $r$. Set $R' = R \cup \{s_r^q \mid r \in R, q \in Q_r\}$.

The winning strategy is similar to the one in the proof of Theorem 4.14. We only describe the strategy when the spoiler plays in $\mathcal{A}$ the converse situation is similar. If the spoiler makes a (2)-move $t_{1i} \in adom(\mathcal{A})$ in the big game then the duplicator answers with $t_{2i}$ where $t_{2i}$ is the answer to $t_{1i}$ in the small game. Suppose the spoiler makes a (1)-move $t_{1i} \in \mathcal{A}$ in the big game. Let $t'_{1i}$ be the restriction to $\eta(\mathcal{A})$. Let $t'_{2i}$ be the answer to $t'_{1i}$ in the small game. Then define $t_{2i}$ as the tree $t'_{2i}$ where the "remaining" part of $t_{1i}$ is attached. That is, if $s_p = s'_p \cdot j_p$ is the rightmost node of $t'_{pi}$, $p = 1, 2$, then for any node $s'_1 \cdot j \cdot v$ in $\mathrm{dom}(t_{1i})$, where $j \geq j_1$, we add a node $s'_2 \cdot (j_2 + j - j_1) \cdot v$ to $t_{2i}$ and label it the same.

We need to show that this strategy is indeed a winning strategy. Suppose $\big((t_{11}, \ldots t_{1i}), (t_{21}, \ldots, t_{2i})\big)$ have been chosen in the big game. There are corresponding moves $\big((t'_{11}, \ldots t'_{1i}), (t'_{21}, \ldots, t'_{2i})\big)$ in the small game. Note that when $t_{pj}$, $p \in \{1, 2\}$, is a branch, then $t'_{1j} = t_{1j} \cap \eta(\mathcal{A})$ and $t'_{2j} = t_{2j} \cap \eta(\mathcal{B})$. Indeed, one of them is chosen as such, the reason the other one has to be of the specified form, follows from the fact that we have three extra moves in the small game.

We introduce the following notation. When $t_{pj}$ is a branch, then define $e_{pj} := t_{pj} - t'_{pj}$ as the difference between $t_{pj}$ and $t'_{pj}$. That is, the tree with domain $(m - j_p) \cdot v$ such that $s_p \cdot m \cdot v$ is a node in $\mathrm{dom}(t_{pj})$. The labeling is inherited from $t_{pj}$. By definition, $e_{1j} = e_{2j}$, so we write $e_j$.

We show that the mapping $\vec{t_1} : t_{11}, \ldots, t_{2i} \mapsto \vec{t_2} : t_{21}, \ldots, t_{2i}$ is indeed a partial isomorphism. We only show that for every atomic formula $\alpha$, $\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R} \models \alpha(\vec{t_1})$ implies $\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R} \models \alpha(\vec{t_2})$. The converse direction is similar.

Suppose $\mathfrak{T}_{\mathtt{p}}^{\mathtt{u},R} \models t_{1j} \preceq_\downarrow t_{1r}$.

(1) both $t_{1j}$ and $t_{1r}$ are branches, and $t_{1j} \notin \eta(\mathcal{A})$. As $t_{1m} = t'_{1m} \cap \eta(\mathcal{A})$ $(m \leq i)$, there is no $t \in \eta(\mathcal{A})$ such that $t'_{1m} \prec t \preceq t_{1m}$. So, $t_{1j} \preceq_\downarrow t_{1r}$ implies that $t'_{1j} = t'_{1r}$ and $e_j \preceq_\downarrow e_r$. Therefore, $t'_{2j} = t'_{2r}$. As $e_j \preceq e_i$, we have that $t_{2j} \preceq_\downarrow t_{2r}$.

(2) both $t_{1j}$ and $t_{1r}$ are branches, $t_{1j} \in \eta(\mathcal{A})$ and $t_{1r} \notin \eta(\mathcal{A})$. Then, $t_{1j} \preceq_\downarrow t'_{1r}$. From the small game $t_{2j} \preceq_\downarrow t'_{2r}$ and, therefore, $t_{2j} \preceq_\downarrow t_{2r}$.
(3) $t_{1j}, t_{1r} \in \eta(\mathcal{A})$. Follows from the small game.
(4) $t_{1j}, t_{1r} \in \mathit{adom}(\mathcal{A})$. Then $t_{2j}, t_{2r} \in \mathit{adom}(\mathcal{B})$ and $\mathfrak{T}^{\mathfrak{u},R}_{\mathfrak{p}} \models t_{2j} \preceq_\downarrow t_{2r}$ follows from the small game.
(5) $t_{1j} \in \mathit{adom}(\mathcal{A})$ and $t_{1r} \in \eta(\mathcal{A})$, or $t_{1r} \in \mathit{adom}(\mathcal{A})$ and $t_{1j} \in \eta(\mathcal{A})$. The former case reduces to case 3. The latter follows from the small game.

Suppose $r^\downarrow(t_{1j})$. Then, $t_{1j}$ is a branch. From the small game, $s^{q,\downarrow}_r(t'_{1j})$ iff $s^{q,\downarrow}_r(t'_{2j})$ for all regular expressions $s^q_r$. As the same part is added to both $t'_{1j}$ and $t'_{2j}$, we have $r^\downarrow(t_{2j})$. Showing that $t_{1j} \preceq_\downarrow t_{1r}$ implies $t_{2j} \preceq_\downarrow t_{2r}$, $r^\downarrow(t_{1j})$ implies $r^\downarrow(t_{2j})$, and $L_a(t_{1j})$ implies $L_a(t_{2j})$ is similar. Finally, the case of the database relations follows from the small game. ☐

We show that the data-complexity is in $\mathrm{NC}^1$. We translate every $\mathrm{FO}^{\mathrm{reg}}_{\eta,\mathrm{dom}}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$ formula to an $\mathcal{FOREG}$ formula over an adapted vocabulary $\mu_\Sigma(\sigma)$. The translation of the $\sigma$-database to the $\mu_\Sigma(\sigma)$-structure can be done in $\mathrm{NC}^1$. From the proof of Theorem 4.13, it then follows that the data-complexity is $\mathrm{NC}^1$.

Suppose $\mathit{adom}(\mathcal{A})$ contains $K$ trees. Consider a fixed enumeration of these $K$ trees where every tree $T$ has an associated enumeration number $\mathrm{enum}(T)$. Then, the $\mu_\Sigma(\sigma)$-structure $\mathcal{A}'$ uses the domain $\{1, \ldots, K\} \cup \{s \in \mathrm{dom}(T) \mid T \in \mathit{adom}(\mathcal{A})\}$. We associate node $s$ of the $j$-th tree with the pair $(j, s)$; similarly, we associate the $j$-th tree itself with its root $(j, \epsilon)$.

We define $\mathcal{A}'$ as follows:

—every tree will be coded in the relations $<_{\mathrm{pre}}, <_{\mathrm{sib}}, O_a$ of arity 3, 3, and, 2, respectively, where the first element of every pair is the enumeration number of the tree. Formally, for every tree $T = (D, f) \in \mathit{adom}(\mathcal{A})$, $<_{\mathrm{sib}}$ contains $(\mathrm{enum}(T), s \cdot i, s \cdot j)$ for all $s \cdot i, s \cdot j \in D$, $i, j \in \mathbb{N}$ and $i < j$; $<_{\mathrm{pre}}$ contains all $(\mathrm{enum}(T), s, s')$ where $s$ is a prefix of $s'$; and, every $O_a$ contains all pairs $(\mathrm{enum}(T), s)$ with $f(s) = a$.
—For every relation $R \in \sigma$ of arity $k$, $R'$ is a $2k$-ary relation; we have $R^{\mathcal{A}}(T_1, \ldots, T_k)$ iff $R^{\mathcal{A}'}(i_1, \epsilon, \ldots, i_k, \epsilon)$ where $\mathrm{enum}(T_j) = i_j$.

Clearly, the transformation of $\mathcal{A}$ to $\mathcal{A}'$ is in $\mathrm{NC}^1$.

The translation is by induction on the structure of $\mathrm{FO}^{\mathrm{reg}}_{\eta,\mathrm{dom}}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$ formulae. To be precise, for every $\mathrm{FO}^{\mathrm{reg}}_{\eta,\mathrm{dom}}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$ formula $\varphi(T_1, \ldots, T_n)$, we construct an $\mathcal{FOREG}$ formula $\varphi'(x_1, y_1, \ldots, x_n, y_n)$ such that

$$(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}) \models \varphi(T_1, \ldots, T_n) \Leftrightarrow \mathcal{A}' \models \varphi'(a_1, b_1, \ldots, a_n, b_n),$$

where $a_i = \mathrm{enum}(T_i)$ and $b_i = \varepsilon$ if $T_i \in \mathit{adom}(\mathcal{A})$, and $a_i = \mathrm{enum}(T_i)$ and $b_i = s$ if $T_i$ is a branch of a tree $T \in \mathit{adom}(\mathcal{A})$ ending in node $s$. So, with every tree $T$ we associate two variables $x$ and $y$. Note that this translation only depends on $\sigma$ and not on $\mathcal{A}$.

To facilitate the translation, we make a distinction between branch variables and domain variables. The former are interpreted by branches, while the latter are interpreted by trees in the active domain. If we do not care then we use variables $U_1, U_2, \ldots$.

Let $\mathrm{dom}(x,y)$ be the formula defining that $y$ is a node of the tree $T$ with $\mathrm{enum}(T) = x$. Let $\mathrm{root}(x,y)$ denote the formula expressing that $y$ is the root of $x$. Let $\mathrm{last}(x,y)$ denote the formula expressing that $y$ is the last node of the tree $T$ with $\mathrm{enum}(T) = x$. Let $s(U_i)$ be the formula $\mathrm{root}(x_i, y_i)$ if $U_i$ is a domain variable; otherwise, it is the formula $\mathrm{dom}(x_i, y_i)$. Finally, let $\mathrm{leaf}(x,y)$ be the formula expressing that $y$ is a leaf of $x$.

—if $\varphi := U_1 = U_2$ then $\varphi'$ is $s(U_1) \wedge s(U_2) \wedge x_1 = x_2 \wedge y_1 = y_2$;
—Let $\gamma$ be the formula

$$\forall y, y' \bigwedge_{R\in\{<_{\mathrm{pre}},<_{\mathrm{sib}}\}} (R(x_1, y, y') \to R(x_2, y, y')) \wedge \forall y, y'(O_a(x_1, y) \to O_a(x_2, y))$$

expressing the prefix relation.
Suppose $\varphi := U_1 \preceq_{\downarrow} U_2$. Then $\varphi'$ is the formula

$$s(U_1)\wedge s(U_2)\wedge\gamma\wedge\forall y(\mathrm{dom}(x_2, y)\wedge\neg\mathrm{dom}(x_1, y) \to \exists y'(<_{\mathrm{pre}} (x_1, y', y)\wedge\mathrm{leaf}(x_1, y'))).$$

Suppose $\varphi := U_1 \preceq_{\to} U_2$. Then $\varphi'$ is the formula

$$s(U_1)\wedge s(U_2)\wedge\gamma\wedge\forall y(\mathrm{dom}(x_2, y)\wedge\neg\mathrm{dom}(x_1, y) \to \exists y'(<_{\mathrm{sib}} (x_1, y', y)\wedge\mathrm{dom}(x_1, y'))).$$

—If $\varphi := L_a(U_1)$, then $\varphi'$ is the formula $s(U_1) \wedge \exists y(\mathrm{last}(x_1, y) \wedge O_a(y))$.
—if $\varphi := R(U_1, \ldots, U_n)$ then $\varphi'$ is $\bigwedge_{i=1}^{n} s(U_i) \wedge R'(x_1, y_1, \ldots, x_n, y_n)$;
—the boolean connectives are immediate;
—If $\varphi$ is of the form $\exists U_i \psi(U_i)$ then $\varphi'$ is of the form $\exists x_i \exists y_i(s(U_i) \wedge \psi'(x_i, y_i))$. $\square$

### 5.5 Safe query languages.

We conclude by a remark on *safety* in relational calculi. An $\mathrm{FO}(\mathfrak{M}, \sigma)$ query $\varphi(\vec{T})$ is safe if for every $\mathcal{A}$, the number of tuples $\vec{T_0}$ such that $(\mathfrak{M}, \mathcal{A}) \models \varphi(\vec{T_0})$ is finite. This property is undecidable even for the pure relational calculus (that is, when the vocabulary of $\mathfrak{M}$ is empty), but the class of safe queries often has *effective syntax*: that is, an r.e. collection of safe queries $\varphi_i, i \in \mathbb{N}$, such that every safe query in $\mathrm{FO}(\mathfrak{M}, \sigma)$ is equivalent to one of $\varphi_i$'s. The existence of effective syntax for pure relational calculus is a standard result of relational database theory [Abiteboul et al. 1995], but [Stolboushkin and Taitslin 1999] showed that it may not extend even to some structures with quantifier-elimination and decidable first-order theory. Nevertheless, for previously studied automatic structures, safe queries were shown to have effective syntax [Benedikt et al. 2003]. We now extend this result to the calculi studied here for trees.

THEOREM 5.19. *Safe queries in all of* $\mathrm{FO}(\mathfrak{T}_{\mathfrak{p}}, \sigma)$, $\mathrm{FO}(\mathfrak{T}, \sigma)$, $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$, $\mathrm{FO}_{\eta}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$, $\mathrm{FO}^{reg}_{\eta}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$, $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}, \sigma)$, *and* $\mathrm{FO}_{\eta}(\mathfrak{T}^{\mathfrak{u}}, \sigma)$ *have effective syntax.*

*Proof.* We start with ranked trees. We prove the result for $\mathfrak{T}$ and sketch a very similar proof for $\mathfrak{T}_{\mathfrak{p}}$. For each query $\varphi(\vec{T})$, define $\varphi^b(t) \equiv \exists T_1, \ldots, T_n\ \varphi(T_1, \ldots, T_n) \wedge (\bigvee_i \eta(t, T_i))$. That is, $\varphi^b(t)$ defines all the branches of all the trees in the active domain of the output of $\varphi$. Clearly, $\varphi$ is safe iff $\varphi^b$ is.

Let $\ell(t)$ be the length of a branch $t$, and let $\ell(\mathcal{A})$ be the maximum length of a branch in a tree in $adom(\mathcal{A})$. We need the following lemma.

Lemma 5.20. *For every $k \geq 0$, there is and can be effectively found a number $m > 0$ with the following property. Let $\psi(t)$ be a* $\mathrm{RC}(\mathfrak{T}, \sigma)$ *query such that if $\psi(t)$ is true, then $t$ is a branch. Assume that $\psi$ is of quantifier rank $k$. Suppose $\mathcal{A} \models \psi(t_0)$ where $\ell(t_0) - \ell(\mathcal{A}) > m$. Then $\psi(\mathcal{A})$ is infinite.*

*Proof of the lemma.* Consider all (finitely many) rank $k$ types of a single branch. Let $m$ be the smallest number with the following property: if for a rank $k$ type $\tau$ there are only finitely many branches that realize it, then each such branch has length at most $m$. Clearly this number can be computed from $k$, since all the types can be effectively listed, and the theory of $\mathfrak{T}$ is decidable.

Now suppose that $\mathcal{A} \models \psi(t_0)$ and $\ell(t_0) - \ell(\mathcal{A}) > m$. Let $s$ be the largest (with respect to the prefix relation) node of $t_0$ that also belongs to $\mathrm{dom}(\mathcal{A})$. Define $t$ to be the subbranch of $t_0$ rooted at $s$. Let $t_0[s] \prec t_0$ be the restriction of $t_0$ to the nodes which are proper prefixes of $s$. Note that $\ell(t) > m$. Let $t'$ be a branch of the same rank $k$ type as $t$, and let $t'_0$ be the extension of $t_0[s]$ with $t'$ attached as the subbranch rooted at $s$. We now claim that $(\mathfrak{T}, \mathcal{A}, t_0) \equiv_k (\mathfrak{T}, \mathcal{A}, t'_0)$. Clearly this suffices, since this implies $\mathcal{A} \models \psi(t'_0)$, and there are infinitely many branches $t'$ of the same rank $k$ type as $t$ (and hence infinitely many branches $t'_0$).

That $(\mathfrak{T}, \mathcal{A}, t_0) \equiv_k (\mathfrak{T}, \mathcal{A}, t'_0)$ follows by a straightforward composition argument. Suppose in round $i$ the spoiler plays $U_i$ in $(\mathfrak{T}, \mathcal{A}, t_0)$. To construct the response $V_i$ in $(\mathfrak{T}, \mathcal{A}, t'_0)$, the duplicator splits $U_i$ into $U'_i$ and $U''_i$, where $U'_i$ is the restriction of $U_i$ to nodes which are not suffixes of $s$, and $U'_i$ is the subtree of $U_i$ rooted at $s$. The duplicator's response is a tree $V_i$ whose restrictions to nodes which are not suffixes of $s$ is $V'_i$, and whose subtree rooted at $s$ is $V''_i$. To define such a tree, the duplicator chooses $V'_i = U'_i$, and finds $V''_i$ such that $(\mathfrak{T}, t, U''_1, \ldots, U''_i) \equiv_{k-i} (\mathfrak{T}, t', V''_1, \ldots, V''_i)$. The latter is possible by following a $k$-round winning strategy for $(\mathfrak{T}, t)$ and $(\mathfrak{T}, t')$. It is easy to check the the duplicator wins the $k$-round game on $(\mathfrak{T}, \mathcal{A}, t_0)$ and $(\mathfrak{T}, \mathcal{A}, t'_0)$. This completes the proof of the lemma.

To finish the proof of the theorem for $\mathfrak{T}$, note that by the lemma, for each query $\varphi$, there is (and can be effectively calculated) a number $m$ such that if $\varphi$ is safe on $\mathcal{A}$, then every branch in $\varphi^b(\mathcal{A})$ has length at most $\ell(\mathcal{A}) + m$ (by letting $k$ be the quantifier rank of $\varphi^b$ and applying the lemma). Note that for each fixed $m$, the predicate $\ell(t') \leq \ell(t'') + m$ is expressible over $\mathfrak{T}$. Thus, it suffices to take $\varphi_{\mathrm{rr}}(T_1, \ldots, T_n)$ to be

$$\bigwedge_{i=1}^{m} \forall t \ \Big(\eta(t, T_i) \rightarrow \exists T \in adom\ \exists t' \ \big(\eta(t', T) \wedge (\ell(t) \leq \ell(t') + m)\big)\Big)$$

For $\mathfrak{T}_{\mathfrak{p}}$, the proof proceeds similarly, except that in the analogous lemma the condition $\ell(t_0) - \ell(\mathcal{A}) \leq m$ is replaced by the condition that there is $t' \preceq t$ such that $\ell(t) \leq \ell(t') + m$ and $t'$ is a branch of a tree in $adom(\mathcal{A})$. With this modification (which is proved by a similar game argument), the rest of the proof is identical, as the predicate $t' \preceq t \wedge (\ell(t) \leq \ell(t') + m)$ is expressible over $\mathfrak{T}_{\mathfrak{p}}$ for every fixed $m$.

We next move to unranked trees. We first prove the result for the calculi based on $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$. Suppose $t$ is a branch and $\mathcal{A}$ a $\sigma$-structure. By $t_{\mathcal{A}}$ we denote the maximal subbranch of $t$ that is also a branch of some $T \in adom(\mathcal{A})$. By $d(t, \mathcal{A})$ we denote the number of nodes in $\mathrm{dom}(t) - \mathrm{dom}(t_{\mathcal{A}})$. Our goal is to show that following claim.

CLAIM 5.21. *For every $k \geq 0$, there exists $p > 0$ such that for every $\mathcal{A}$ and every branch $t$ with $d(t, \mathcal{A}) > p$, there are infinitely many branches $t'$ such that*

$$(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, t) \ \equiv_k \ (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, t').$$

We now show how the result follows from the claim. Suppose $\varphi(T_1, \ldots, T_m)$ is a query in one of the languages; we form a query

$$\varphi_\eta(t) \quad = \quad \exists T_1, \ldots, T_m \ (\bigvee_{i=1}^{m} \eta(t, T_i) \wedge \varphi(T_1, \ldots, T_m))$$

in $\mathrm{FO}(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \sigma)$. Let $k$ be the quantifier rank of this query. Then, if there $t_0$ such that $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}) \models \varphi_\eta(t_0)$ and $d(t_0, \mathcal{A}) > p$, then, by Claim 5.21, infinitely many branches satisfy $\varphi_\eta(\cdot)$ and hence $\varphi$ is not safe.

We now consider the formula

$$\gamma_p(T) \ = \ \forall^\eta t \preceq T \ \exists T' \in adom \ \exists^\eta t' \preceq T' \ \Big(\neg \exists t_1, \ldots, t_{p+1} \ \big(t' \prec t_1 \prec \ldots \prec t_{p+1} \prec t\big)\Big)$$

Thus, if $\varphi$ is safe, then $\varphi$ is equivalent to

$$\varphi(T_1, \ldots, T_m) \ \wedge \ \gamma_p(T_1) \wedge \ \ldots \ \wedge \ \gamma_p(T_m).$$

Since every query of the above form is safe, and adding the conjunction $\bigwedge_i \gamma_p(T_i)$ does not take the query out of the original syntactic class (among those listed in the Proposition), we get the result.

It remains to prove the claim. We know that for every $k$, we can find $l > 0$ such that

$$(\mathfrak{T}_{\mathfrak{p}}, \mathcal{R}(\mathcal{A}), \mathcal{R}(T)) \ \equiv_l \ (\mathfrak{T}_{\mathfrak{p}}, \mathcal{R}(\mathcal{A}), \mathcal{R}(T')) \quad \text{implies} \quad (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, T) \ \equiv_k \ (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, T'). \tag{5}$$

This is an easy consequence of Lemma 4.3; the game is played in the obvious way, that is, each move $V$ is modeled by $\mathcal{R}(V)$, and the bounds on $l$ ensure that the response to that in the game over $\mathfrak{T}_{\mathfrak{p}}$ is of the form $\mathcal{R}(W)$; then $W$ is selected as the response.

We have seen that for every $\sigma$-structure $\mathcal{B}$ over the structure of ranked trees, for every ranked branch $t_0$, and every $s > 0$, there exists $p > 0$ such that $d(t_0, \mathcal{B}) > p$ implies the existence of infinitely many branches $t_1$ with $(\mathfrak{T}_{\mathfrak{p}}, \mathcal{B}, t_0) \equiv_s (\mathfrak{T}_{\mathfrak{p}}, \mathcal{B}, t_1)$. We now apply this to $\mathcal{B} = \mathcal{R}(\mathcal{A})$ and $s = l + 3$, and find the corresponding $p > 0$.

Suppose now we have an unranked branch $t$ with $d(t, \mathcal{A}) > p$. Let $t_0 = \mathcal{R}(t)$; then $d(t_0, \mathcal{R}(\mathcal{A})) > p$. We thus find infinitely many ranked trees $t'_i, i > 0$ such that $(\mathfrak{T}_{\mathfrak{p}}, \mathcal{R}(\mathcal{A}), t_0) \ \equiv_{l+3} \ (\mathfrak{T}_{\mathfrak{p}}, \mathcal{R}(\mathcal{A}), t'_i)$. Since there is a formula of quantifier-rank 3 stating that a ranked tree $V$ is of the form $\mathcal{R}(W)$, where $W$ is an unranked branch, we conclude that $t'_i$ is of the form $\mathcal{R}(t_i)$, where $t_i$ is an unranked branch. Now (5) gives us $(\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, t) \equiv_k (\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}, \mathcal{A}, t_i)$ for infinitely many branches $t_i$, which proves the claim.

The proof of the result for the calculi based on $\mathfrak{T}^{\mathfrak{u}}$ is similar, except that one uses a different notion of distance: $d'(t, \mathcal{A})$ is the number of nodes in $\mathrm{dom}(t)$ that do not belong to $\mathrm{dom}(\mathcal{A}) = \bigcup_{T \in adom(\mathcal{A})} \mathrm{dom}(T)$. □

| Logic | Class of Languages |
|---|---|
| $\mathrm{FO}(\mathfrak{T})$ | $\mathcal{MSO}$ = regular |
| $\mathrm{FO}(\mathfrak{T}_\mathfrak{p})$ | $\mathcal{MSO}$ = regular |
| $\mathrm{FO}_\eta(\mathfrak{T})$ | $\mathcal{MSO}^{\mathrm{chain}}$ |
| $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$ | $\mathcal{FO}$ |

Fig. 4. Definability over $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$.

| Logic | Class of Languages |
|---|---|
| $\mathrm{FO}(\mathfrak{T}^\mathfrak{u})$ | $\mathcal{MSO}$ = regular |
| $\mathrm{FO}(\mathfrak{T}^\mathfrak{u}_\mathfrak{p})$ | $\mathcal{MSO}$ = regular |
| $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u})$ | $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ |
| $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u}_\mathfrak{p})$ | $\mathcal{FO}$ |
| $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^\mathfrak{u})$ | $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ |
| $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^\mathfrak{u}_\mathfrak{p})$ | $\mathcal{FOREG}$ |

Fig. 5. Definability over $\mathfrak{T}^\mathfrak{u}_\mathfrak{p}$ and $\mathfrak{T}^\mathfrak{u}$.

| Relational Calculi | Data Complexity |
|---|---|
| $\mathrm{FO}(\mathfrak{T}^\mathfrak{u}, \sigma)$ | PH |
| $\mathrm{FO}(\mathfrak{T}^\mathfrak{u}_\mathfrak{p}, \sigma)$ | PH |
| generic $\mathrm{FO}(\mathfrak{T}^\mathfrak{u}, \sigma)$ | $\mathrm{AC}^0$ |
| $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^\mathfrak{u}, \sigma)$ | DLOGSPACE |
| $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^\mathfrak{u}_\mathfrak{p}, \sigma)$ | DLOGSPACE |
| $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u}, \sigma)$ | PH |
| $\mathrm{FO}^{\mathrm{act}}_\eta(\mathfrak{T}^\mathfrak{u}, \sigma)$ | $\mathrm{NC}^1$ |
| $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^\mathfrak{u}_\mathfrak{p}, \sigma)$ | $\mathrm{NC}^1$ |

Fig. 6. Relational calculi over $\mathfrak{T}^\mathfrak{u}_\mathfrak{p}$ and $\mathfrak{T}^\mathfrak{u}$.

## 6. CONCLUSION

We examined definability and complexity for both logical constraints over tuples of trees and query languages over relations on trees. The results obtained for logics are summarized in Figure 4 and Figure 5. In each case, we show the class of languages one obtains from using formulae in the logic with no relational variables. This gives a rather complete picture concerning definability in the classical setting versus definability in the model-theory setting.

The second half of the paper deals with query languages. Figure 6 summarizes the expressiveness and data complexity of the corresponding query languages. We would like to understand the precise expressiveness of generic queries in relational calculi over automatic structures: we know that $\mathrm{AC}^0$ is an upper bound, but we suspect that these queries may capture an interesting subclass of $\mathrm{AC}^0$ that properly extends the class of first-order definable properties. In addition, we would like to find precise descriptions of fragments of various XML query languages (e.g., [Chamberlin et al. 2002; Cardelli and Ghelli 2004]) that correspond to the logics studied here.

Although the bounds presented for primal tree formulae are the same as for general formulae in the worst case, it remains to check to what extent primal formulae can be evaluated with greater parallelism. It is clear that splitting automata can be implemented more efficiently than general tree-tuple automata on trees with small overlap, but we currently have no formal results that capture this advantage.

The results here apply to tuples of finite trees. In future work, we will examine what occurs for tuples of *infinite* trees. The natural motivation for this comes from verification: given a set of state machines $S_1 \dots S_n$, one is often interested in synthesizing a state machine $S$ such that the product of $S$ with $S_i$ satisfies some property. If one passes from the machine $S_i$ to the behavior tree $T_i$ obtained by unwinding it, this corresponds to generating a regular infinite tree $T$ that satisfies a formula $\varphi(T, T_1 \dots T_n)$.

## ACKNOWLEDGMENTS

We are very grateful to anonymous referees for their comments. Part of this work was done when Michael Benedikt and Frank Neven visited the University of Toronto, supported by a PREA grant, and when Leonid Libkin visited the University of Limburg. Libkin was partly supported by grants from NSERC, BUL, and CITO.

## REFERENCES

Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases.* Addison-Wesley.

Aiken, A. 1994. Set constraints: Results, applications, and future directions. In *PPCP '94: Proceedings of the Second International Workshop on Principles and Practice of Constraint Programming.* Lecture Notes in Computer Science, vol. 874. Springer-Verlag, 326–335.

Angluin, D. and Hoover, D. N. 1984. Regular prefix relations. *Mathematical Systems Theory 17,* 3, 167–191.

Anthony, M. and Biggs, N. 1992. *Computational Learning Theory.* Cambridge University Press.

Benedikt, M. and Libkin, L. 2000. Relational queries over interpreted structures. *J. ACM 47,* 4, 644–680.

Benedikt, M., Libkin, L., Schwentick, T., and Segoufin, L. 2003. Definable relations and first-order query languages over strings. *J. ACM 50,* 5, 694–751.

Blumensath, A. and Gräel, E. 2000. Automatic structures. In *LICS '00: Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science.* IEEE Computer Society, 51.

Börger, E., Grädel, E., and Gurevich, Y. 1997. *The classical decision problem.* Springer.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., and Maler, E. 2000. Extensible Markup Language (XML) 1.0 (second edition) W3C recommendation. http://www.w3.org/XML.

Brüggemann-Klein, A., Murata, M., and Wood, D. 2001. Regular tree and regular hedge languages over unranked alphabets: Version 1, april 3, 2001. Tech. Rep. HKUST-TCSC-2001-0, The Hongkong University of Science and Technology.

Bruyère, V., Hansel, G., Michaux, C., and Villemaire, R. 1994. Logic and $p$-recognizable sets of integers. *Bulletin of the Belgian Mathematical Society 1*, 191–238.

Cardelli, L. and Ghelli, G. 2004. Tql: a query language for semistructured data based on the ambient logic. *Mathematical Structures in Computer Science 14,* 3, 285–327.

Chamberlin, D., Clark, J., Florescu, D., Robie, J., Simeon, J., and Stefanascu, M. 2002. XQuery 1.0: An XML query language. `http://www.w3.org/TR/xquery/`.

Clark, J. 1999. XSL transformations version 1.0. `http://www.w3.org/TR/WD-xslt`.

Clark, J. and DeRose, S. 1999. XML Path Language (XPath). `http://www.w3.org/TR/xpath`.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. 1997. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. release October, 1rst 2002.

Dantsin, E. and Voronkov, A. 2000. Expressive power and data complexity of nonrecursive query languages for lists and trees (extended abstract). In *PODS '00: Proceedings of the* *nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM Press, 157–165.

Doner, J. 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences 4*, 406–451.

Dörre, J. 1991. Feature logic with weak subsumption constraints. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 256–263.

Ebbinghaus, H.-D. and Flum, J. 1999. *Finite Model Theory*, 2 ed. Springer.

Elgaard, J., Klarlund, N., and Møller, A. 1998. Mona 1.x: New techniques for ws1s and ws2s. In *CAV '98: Proceedings of the 10th International Conference on Computer Aided Verification*, A. J. Hu and M. Y. Vardi, Eds. Lecture Notes in Computer Science, vol. 1427. Springer-Verlag, 516–520.

Ésik, Z. 1998. Axiomatizing the equational theory of regular tree languages (extended anstract). In *STACS '98: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science*, M. Morvan, C. Meinel, and D. Krob, Eds. Lecture Notes in Computer Science, vol. 1373. Springer-Verlag, 455–465.

Flum, J. and Ziegler, M. 1999. Pseudo-finite homogeneity and saturation. *Journal of Symbolic Logic 64*, 1689–1699.

Gottlob, G., Koch, C., Pichler, R., and Segoufin, L. 2005. The complexity of xpath query evaluation and xml typing. *J. ACM 52,* 2, 284–335.

Hodges, W. 1993. *Model Theory.* Cambridge.

Hodgson, B. 1983. Décadabilité par automate fini. *Ann. Sc. Math. Québec 7*, 39–57.

Immerman, N. 1998. *Descriptive Complexity.* Springer.

Khoussainov, B. and Nerode, A. 1995. Automatic presentations of structures. In *LCC: Logical and Computational Complexity*, D. Leivant, Ed. Lecture Notes in Computer Science, vol. 960. Springer, 367–392.

Kolb, H.-P. and Mönnich, U. 1999. *The Mathematics of Syntactic Structure: Trees and Their Logics.* Walter De Gruyter.

Kupferman, O., Safra, S., and Vardi, M. Y. 1996. Relating word and tree automata. In *LICS '96: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science.* IEEE Computer Society, 322.

Laskowski, M. C. 1992. Vapnik-chervonenkis classes of definable sets. *Journal of the London Mathematical Society 45*, 377–384.

Lohrey, M. 2001. On the parallel complexity of tree automata. In *RTA '01: Proceedings of the 12th International Conference on Rewriting Techniques and Applications*, A. Middeldorp, Ed. Lecture Notes in Computer Science, vol. 2051. Springer-Verlag, 201–215.

Marx, M. 2004. Conditional xpath, the first order complete xpath dialect. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM Press, 13–22.

Müller, M. and Niehren, J. 2000. Ordering constraints over feature trees expressed in second-order monadic logic. *Information and Computation 159,* 1–2, 22–58.

Müller, M., Niehren, J., and Podelski, A. 2000. Ordering constraints over feature trees. *Constraints 5,* 1–2, 7–41.

Müller, M., Niehren, J., and Treinen, R. 2001. The first-order theory of ordering constraints over feature trees. *Discrete Mathematics & Theoretical Computer Science 4,* 2, 193–234.

Neven, F. and Schwentick, T. 2000. Expressive and efficient pattern languages for tree-structured data (extended abstract). In *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* ACM Press, 145–156.

Neven, F. and Schwentick, T. 2002. Query automata on finite trees. *Theoretical Computer Science 275*, 633–674.

Niwinski, D. and Walukiewicz, I. 1998. Relating hierarchies of word and tree automata. In *STACS '98: Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer*

*Science*, M. Morvan, C. Meinel, and D. Krob, Eds. Lecture Notes in Computer Science, vol. 1373. Springer-Verlag, 320–331.

Potthoff, A. and Thomas, W. 1993. Regular tree languages without unary symbols are star-free. In *FCT '93: Proceedings of the 9th International Symposium on Fundamentals of Computation Theory*, Z. Ésik, Ed. Lecture Notes in Computer Science, vol. 710. Springer-Verlag, 396–405.

Rabin, M. 1969. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society 141*, 1–35.

Smolka, G. 1995. The oz programming model (extended abstract). In *Euro-Par '95: Proceedings of the First International Euro-Par Conference on Parallel Processing*, S. Haridi and P. Magnusson, Eds. Lecture Notes in Computer Science, vol. 966. Springer-Verlag, 5–8.

Smolka, G. and Treinen, R. 1994. Records for logic programming. *Journal of Logic Programming 18*, 229–258.

Stolboushkin, A. and Taitslin, M. 1999. Finite queries do not have effective syntax. *Information and Computation 153*, 99–116.

Su, Z., Aiken, A., Niehren, J., Priesnitz, T., and Treinen, R. 2002. The first-order theory of subtyping constraints. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages.* ACM Press, 203–216.

Suciu, D. 2002. Typechecking for semistructured data. In *DBPL '01: Revised Papers from the 8th International Workshop on Database Programming Languages*, G. Ghelli and G. Grahne, Eds. Lecture Notes in Computer Science, vol. 2397. Springer-Verlag, 1–20.

Thatcher, J. and Wright, J. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory 2,* 1, 57–81.

Thomas, W. 1984. Logical aspects in the study of tree languages. In *Proceedings of the 9th Int ernational Colloquium Trees in Algebra and Programming*, B. Courcelle, Ed. Cambridge University Press, 31–50.

Thomas, W. 1987. On chain logic, path logic, and first-order logic over infinite trees. In *Symposium on Logic in Computer Science.* IEEE Press, 245–256.

Thomas, W. 1997. Languages, automata, and logic. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Vol. 3. Springer, Chapter 7, 389–456.

Vollmer, H. 1999. *Introduction to Circuit Complexity.* Springer Verlag.

Vorobyov, S. and Voronkov, A. 1998. Complexity of nonrecursive logic programs with complex values. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems.* ACM Press, 244–253.

Wilke, T. 1996. An algebraic characterization of frontier testable tree languages. *Theoretical Computer Science 154,* 1, 85–106.

Received ???; revised ???; accepted ???