Logical Definability and Query Languages over Unranked Trees

Leonid Libkin
University of Toronto
libkin@cs.toronto.edu

Frank Neven
University of Limburg
frank.neven@luc.ac.be

Abstract

Unranked trees, that is, trees with no restriction on the number of children of nodes, have recently attracted much attention, primarily as an abstraction of XML documents. In this paper, we study logical definability over unranked trees, as well as collections of unranked trees, that can be viewed as databases of XML documents. The traditional approach to definability is to view each tree as a structure of a fixed vocabulary, and study the expressive power of various logics on trees. A different approach, based on model theory, considers a structure whose universe is the set of all trees, and studies definable sets and relations; this approach extends smoothly to the setting of definability over collections of trees. We study the latter, model-theoretic approach. We find sets of operations on unranked trees that define regular tree languages, and show that some natural restrictions correspond to logics studied in the context of XML pattern languages. We then look at relational calculi over collections of unranked 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 (NC^1*) data complexity, that can express important XML properties like DTD validation and XPath evaluation.*

1. Introduction

In the literature, there are two different approaches to logical definability over strings and trees: in the older and by now classic way of providing logical descriptions of regularity [35], 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. In the other setting, one considers the family of all strings Σ^* 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 approach was studied in [3, 5, 10, 6, 20, 19].

The second approach led to the study of *automatic* structures, that is, structures in which every definable predicate can be represented by a finite automaton [19, 20]. It was shown in [6] 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 $\mathfrak{S}$. That structure $\mathfrak{S}$ also had several reducts defining regular and star-free languages, and having some nice properties that made them useful as the basis for relational calculi on databases over strings [3, 4]. Recently, automatic structures have been studied in the context of *ranked* trees [5]. In that case, the universe is the set of all trees, and the universal tree-automatic structure $\mathfrak{T}$ has as its definable relations precisely the relations recognized by tree automata [15].

In this paper, we study definability in automatic structures over *unranked trees*, and related database query languages (relational calculi) over collections of unranked trees. Unranked trees differ from ranked trees in that there is no restriction on the arity of nodes. Although unranked trees have been considered in the 60s and 70s [28, 34], and are related to feature trees over an infinite set of features [22, 23], it was the advent of XML that initiated their systematic study [8]. XML is a popular data format which is becoming the lingua franca for information exchange on the world wide web [37], and XML data is naturally modeled as unranked trees [25, 37]. This connection made recent advances in unranked tree language theory foundational for XML-related research in areas such as XML pattern languages [9, 24, 26, 27, 31] and XML schema languages [18, 29].

Most crucial XML concepts are closely related to unranked tree automata. For example, DTDs (Document Type Definitions, the most common form of typing XML) correspond to a subset of tree automata, and a proposed extension, called DTDs with specialization [29], has precisely the power of tree automata. A pattern language XPath allows one to navigate documents following paths of labels from a given regular language. In general, the connection between regularity and querying tree-structured data is well-recognized [17].

However, logics for unranked trees have only been considered for the setting where each tree is a model. By constructing automatic structures over unranked trees, we not only connect the two different definability approaches, but also make it easy to extend definability from a single tree to a collection of trees. Most often such collections appear as repositories of XML documents. If we have a structure $\mathfrak{M}$ over unranked trees, to construct relational calculus over $\mathfrak{M}$ we simply augment the vocabulary with symbols for sets/relations in the database. For example, if $\varphi_d(T)$ is a formula over $\mathfrak{M}$ saying that T conforms to DTD d, then $U(T) \wedge \varphi_d(T)$ is a relational calculus query asking for all trees T in U that conform to d.

In the first part of the paper, we consider definability over automatic structures of unranked trees. We construct a structure $\mathfrak{T}^{\mathfrak{u}}$ which turns out to be the universal one. Like the corresponding structure for ranked trees, $\mathfrak{T}^{\mathfrak{u}}$ is based on the extension relation $\prec$ among trees [5, 22, 23]; however, 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 also consider a weaker structure, $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$, that still defines all regular languages, but a smaller class of relations. We then look at restricted definability over $\mathfrak{T}^{\mathfrak{u}}$ and $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and connect it with definability in the traditional setting: it turns out that some natural sublogics of first-order over $\mathfrak{T}^{\mathfrak{u}}$ and $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ correspond to logics that have been studied in connection with XML pattern languages, and are closely connected to monadic path logic [36].

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 AC^0 to 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 adding relations of unranked trees to our structures and considering queries on them. 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 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, NC^1) while remaining quite expressive.

Many results in this paper are proved by a combination of two techniques. One is the encoding of SωS into S2S, due to Rabin (cf. [7]), which gives us a coding of unranked trees as ranked trees. To be able to use it, we need several results showing how to restrict quantification over various structures; those are proved by Ehrenfeucht-Fraïssé games.

Organization. Section 2 defines the main concepts and presents the main proof techniques. In Section 3, we present the basic definability results over $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathfrak{u}}$ and show that the latter defines precisely the regular relations. In Section 4, we look at restricted definability over $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$, $\mathfrak{T}^{\mathfrak{u}}$, and ranked tree models, and connect it with first-order, monadic path logic [36], and their unranked extension [26]. We also prove data-complexity results. In Section 5, we consider query languages for collections of unranked trees based on first-order logic over $\mathfrak{T}^{\mathfrak{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathfrak{u}}$. Section 6 presents conclusions. Due to space limitations, complete proofs are not included; a full version can be obtained from the authors.

2. Preliminaries

In this section, we give a brief overview of the two different approaches to logical definability over strings and trees. 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 FO($\mathfrak{M}$). Throughout the paper, Σ is a finite alphabet with at least two letters.

2.1. Strings

A string $s = a_1 \ldots a_n$ over Σ can be 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. [35].

Another approach to definability is by using the standard model-theoretic setting. In that case, we consider several operations on the set Σ^* of all finite strings over Σ. One of them is the prefix relation $s_1 <_{\text{pre}} s_2$ 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 string s.

The structures most often considered in this setting are:

$$\begin{array}{rcl}\mathfrak{S} & = & \langle \Sigma^*, <_{\text{pre}}, (l_a)_{a\in\Sigma}, \mathrm{el}\rangle;\\ \mathfrak{S}_{\mathfrak{p}} & = & \langle \Sigma^*, <_{\text{pre}}, (l_a)_{a\in\Sigma}\rangle.\end{array}$$

It is known that a subset of Σ^* is FO-definable in $\mathfrak{S}$ iff it is regular [6, 10], and it is FO-definable in $\mathfrak{S}_{\mathfrak{p}}$ iff it is star-free [3]. 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 [6, 10]. The index $\mathfrak{p}$ in $\mathfrak{S}_{\mathfrak{p}}$ stands for "primal", following the notation introduced in [5].

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 Σ, 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 ith symbol is $(s_1^i, \ldots, s_k^i)$, where s_j^i is the ith symbol of s_j, if $|s_j| \leq i$, and $\perp$ otherwise. 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 $\Sigma_\perp^k$.

2.2. Ranked trees

When dealing with ranked trees, one usually fixes a natural number k and requires that all interior nodes of trees have at most k children. In this paper, without any loss of generality, we consider binary trees. The trees we consider are on two fixed alphabets: the alphabet for the domain of the tree consists of $\{0, 1\}$; we use the finite alphabet Σ for the node labels. A set D of strings is *prefix-closed* if $s \in D$ and $s' <_{\text{pre}} s$ imply $s' \in D$. A *ranked tree domain* is a finite prefix-closed subset of $\{0, 1\}^*$. A *ranked* Σ*-tree* is a pair $T = (D, f)$ where D is a ranked tree domain and $f : D \to \Sigma$ is a function. We refer to D as the *domain* of T, and to f as the *labeling function*. We use $\text{dom}(T)$ to denote D. By $\text{TREE}(\Sigma)$ we denote the set of all ranked Σ-trees.

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 ϵ, and the leaves are $s \in D$ such that s is not a prefix of any other string in D. Nodes are ordered lexicographically so we have the left-right relations between them.

A ranked tree $T = (D, f)$ is represented as a first-order structure $\langle D, <_{\text{pre}}, \text{succ}_0, \text{succ}_1, (O_a)_{a\in\Sigma}\rangle$, where $<_{\text{pre}}$ is the prefix relation, $O_a = \{s \in D \mid f(s) = a\}$, and $\text{succ}_0(s) = s \cdot 0$, $\text{succ}_1(s) = s \cdot 1$ for all $s, s \cdot 0, s \cdot 1 \in D$.

We denote by $\mathcal{FO}$ ($\mathcal{MSO}$) the set of first-order (monadic second-order) formulae over the above vocabulary. A set of trees $X \subseteq \text{TREE}(\Sigma)$ is $\mathcal{FO}$ ($\mathcal{MSO}$) definable if there is an $\mathcal{FO}$ ($\mathcal{MSO}$) sentence Φ such that $X = \{T \mid T \models \Phi\}$. It is known that a set of trees is $\mathcal{MSO}$-definable if it is regular, that is, accepted by a tree automaton.[1] We shall consider some restrictions of $\mathcal{MSO}$; among them is *path logic* $\mathcal{MSO}^{\text{path}}$ [36], which is $\mathcal{MSO}$ in which quantification is restricted to paths: sets linearly ordered by $<_{\text{pre}}$. It is known that $\mathcal{FO} \subsetneq \mathcal{MSO}^{\text{path}} \subsetneq \mathcal{MSO}$ [36].

It was shown recently [5] that regularity over trees can be captured by the model-theoretic notion of definability, similarly to the string case. For that, one finds natural tree analogs of the operations of $\mathfrak{S}$ and $\mathfrak{S}_\mathfrak{p}$. We introduce two new structures:

$$
\begin{array}{rcl}
\mathfrak{T} & = & \langle \text{TREE}(\Sigma), \prec, (\text{succ}_i^a)_{i=0,1;a\in\Sigma}, (\epsilon_a)_{a\in\Sigma}, \approx_{\text{dom}}\rangle \\
\mathfrak{T}_\mathfrak{p} & = & \langle \text{TREE}(\Sigma), \prec, (\text{succ}_i^a)_{i=0,1;a\in\Sigma}, (\epsilon_a)_{a\in\Sigma}\rangle
\end{array}
$$

Here $\prec$ is the subsumption [22, 23] (or extension [5], depending on how one looks at it) relation: $(D_1, f_1) \prec (D_2, f_2)$ if $D_1 \subset D_2$ and f_1 and f_2 agree on D_1. We write $T_1 \preceq T_2$ if $T_1 \prec T_2$ or $T_1 = T_2$.

The successor functions work as follows: if $T = (D, f)$, then $\text{succ}_i^a(T)$ is the tree (D', f') where $D' = D \cup \{s \cdot i \mid s \text{ a leaf of } T\}$, and f' extends f by labeling each node in $D' - D$ by a. The one-node tree labeled a is denoted by ϵ_a (that is, $\text{dom}(\epsilon_a)$ is ϵ, the empty string.) Finally, $(D_1, f_1) \approx_{\text{dom}} (D_2, f_2)$ iff $D_1 = D_2$.

We consider first-order logic (FO) over these structures. To emphasize the structure at hand, we write FO($\mathfrak{T}_\mathfrak{p}$) and FO($\mathfrak{T}$) to denote FO over $\mathfrak{T}_\mathfrak{p}$ and $\mathfrak{T}$, respectively.

Note that these structures are natural extensions of $\mathfrak{S}$ and $\mathfrak{S}_\mathfrak{p}$; that is, if we associate strings with unary trees ($D \subseteq 0^*$), then the definable relations of $\mathfrak{T}$ are precisely those of $\mathfrak{S}$, and the definable relations of $\mathfrak{T}_\mathfrak{p}$ are precisely those of $\mathfrak{S}_\mathfrak{p}$.

As with strings, definability by automata can be extended to tuples of trees. Let $\vec{T} = (T_1, \ldots, T_k)$ be a tuple of trees. We represent it as a tree $[\vec{T}]$ in $\text{TREE}(\Sigma_\perp^k)$. 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, $f(s) = (a_1, \ldots, a_k)$ where

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

We say that a subset R of $\text{TREE}(\Sigma)^k$ is regular if $\{[\vec{T}] \mid \vec{T} \in R\}$ is accepted by a tree automaton (again, with adding nodes to ensure that the automaton runs on a tree without unary branching).

We shall use the following results from [5].

Fact 1 *a) If $X \subseteq \text{TREE}(\Sigma)$, then*

X is regular $\Leftrightarrow$ X is definable in $\mathfrak{T}_\mathfrak{p}$ $\Leftrightarrow$ X is definable in $\mathfrak{T}$.

b) If $k > 1$ and $R \subseteq \text{TREE}(\Sigma)^k$, then X is definable in $\mathfrak{T}$ iff it is regular.
c) The relation $\approx_{\text{dom}}$ is not definable in $\mathfrak{T}_\mathfrak{p}$.

We will also look at some restrictions of FO($\mathfrak{T}_\mathfrak{p}$) and FO($\mathfrak{T}$). Thereto, we introduce the following notion: a *branch* is a tree T such that the set of trees $\{T' \mid T' \preceq T\}$ is linearly ordered by $\prec$. This is definable in $\mathfrak{T}_\mathfrak{p}$ by the following formula $\eta(T)$:

$$
\forall T', T'' \ \big(T' \preceq T \wedge T'' \preceq T\big) \to \big(T' \preceq T'' \vee T'' \preceq T'\big).
$$

We shall consider restrictions of FO over $\mathfrak{T}$ and $\mathfrak{T}_\mathfrak{p}$ in which only quantification over branches is allowed. These

[1] Note that tree automata are normally defined for trees in which every non-leaf node has two children. Since we do not impose this requirement, we can define *regular* as being accepted by a tree automaton over the completion of a tree, in which a successor node labeled by a symbol not in Σ is added to each node with exactly one successor.

restrictions will be denoted by $\mathrm{FO}_\eta(\mathfrak{T}_\mathfrak{p})$ and $\mathrm{FO}_\eta(\mathfrak{T})$. 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 $\mathrm{FO}(\mathfrak{T})$.

2.3. Unranked trees

For unranked trees there is no bound on the number of children. We use consecutive positive integers to enumerate children of a node. 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* Σ*-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 Σ is denoted by $\mathrm{UTREE}(\Sigma)$.

An unranked tree $T = (D, f)$ is represented as a first-order structure $\langle D, <_{\text{pre}}, <_{\text{sib}}, (O_a)_{a\in\Sigma}\rangle$, where $<_{\text{pre}}$ and O_a are as before, and $<_{\text{sib}}$ is the order relation on siblings ($s \cdot i <_{\text{sib}} s \cdot j$ for all $s \cdot i, s \cdot j \in D$, $i, j \in \mathbb{N}$, and $i < j$). Denote the above vocabulary by ν_Σ. Apart from $\mathcal{FO}$ and $\mathcal{MSO}$ over ν_Σ, we also consider the extension of the monadic path 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.

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:

1. 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 on the right);

2. $T_1 \prec_{\downarrow} T_2$ if $T_1 \prec T_2$ and for every $s \in D_2 - D_1$, $s' <_{\text{pre}} s$ for some leaf s' of T_1 (extension down).

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, $T_1 \approx_{\text{dom}} T_2$ holds iff $D_1 = D_2$.

With these operations, we now define two structures:

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

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 $\mathrm{FO}_\eta(\mathfrak{T}^\mathfrak{u})$.

Next, we extend the notion of tree automata to the unranked case, following [8].

Definition 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, δ is a function from $Q \times \Sigma$ to 2^{Q^*} assigning a regular string language over Q to every pair (q, σ).

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 $R \subseteq \mathrm{UTREE}(\Sigma)^k$ is regular if so is the set $\{[\vec{T}] \mid \vec{T} \in R\}$ over $\mathrm{UTREE}(\Sigma^k_\perp)$.

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 [27].

2.4. XML: DTDs and XPath

Document Type Definitions (DTDs) is the most commonly used XML schema definition language. It can be abstracted by extended context-free grammars (with regular expressions as right-hand sides of productions). Formally, a DTD over Σ is a pair (s, d) where $s \in \Sigma$ is the start symbol and $d : \Sigma \to 2^{\Sigma^*}$ maps every Σ-symbol to a regular language over Σ. A tree $T = (D, f)$ conforms to d iff $f(s \cdot 1) \cdots f(s \cdot n) \in d(f(s))$ for every $s \in D$ with n children.

XPath [14] is an XML pattern language employed by several XML transformation languages like XSLT [13] and XQuery [12]. 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. Let $\mathcal{FO}(\exists^*)$ be the fragment of $\mathcal{FO}$ over the vocabulary ν_Σ consisting of formulae $\varphi(x, y)$ in the prenex normal form and all quantifiers existential. Additionally, formulas can make use of the unary predicates root(x), leaf(x), first(x), and last(x) (denoting that x is the root, a leaf, the first and the last child, respectively) and the binary predicate succ(x, y) (denoting that y is the right sibling of x, respectively). Note that these predicates are $\mathcal{FO}$-definable but not $\mathcal{FO}(\exists^*)$-definable. 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 [2].

2.5. The toolbox

Many results in this paper are proved by a combination of two techniques: translating unranked trees into ranked

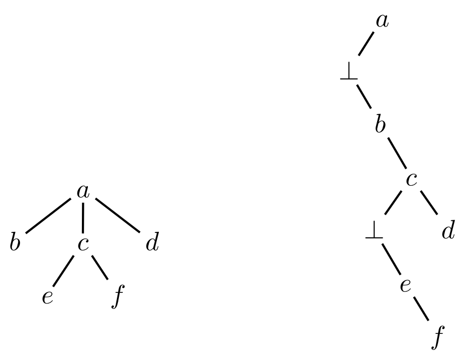

Figure 1. A tree T and $\mathcal{R}(T)$.

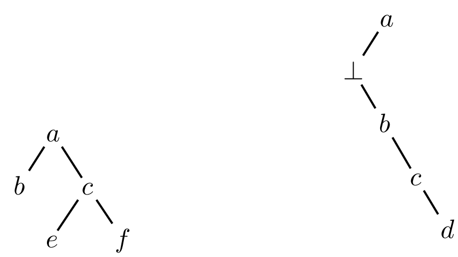

Figure 2. A branch of T with endpoint labeled f and a branch of $\mathcal{R}(T)$ with endpoint labeled d.

ones, and Ehrenfeucht-Fraïssé games. Below we briefly review them.

2.5.1 Encoding unranked trees

This encoding is basically the same as Rabin's encoding of SωS into S2S, cf. [7]. 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 $T = (D, f) \in \textsc{Utree}(\Sigma)$, we define $\mathcal{R}(T) = (D', f') \in \textsc{Tree}(\Sigma_\perp)$ 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 1.

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 versa, and since the image of $\mathcal{R}(\cdot)$ is $\mathcal{MSO}$-definable, we have the following folklore result (for an explicit proof, see [33]):

Proposition 1 *For any finite alphabet Σ and $X \subseteq \textsc{Utree}(\Sigma)$, X is regular iff $\mathcal{R}(X)$ is regular.*

We conclude with a note on branches. 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 given in Section 2.3. We give an example in Figure 2.

2.5.2 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. [16].

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. [16].

We shall also make use of reduced EF games, which are helpful for logics with restricted quantification (for instance, to branches). For that, let 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 quantification 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. [16].

Lemma 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 $\varphi(\vec{x})$ 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.

3. Basic definability results over unranked trees

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

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

Lemma 2 *Let $\mathfrak{M}^{\mathrm{u}}$ be either $\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$ or $\mathfrak{T}^{\mathrm{u}}$, over* $\textsc{UTree}(\Sigma)$, *and let $\mathfrak{M}$ be the corresponding ranked tree model, over* $\textsc{Tree}(\Sigma_\perp)$. *Then for every FO($\mathfrak{M}^{\mathrm{u}}$)-formula φ there exists an* FO($\mathfrak{M}$)*-formula φ' such that:*

$$\mathfrak{M}^{\mathrm{u}} \models \varphi(\vec{T}) \quad \Leftrightarrow \quad \mathfrak{M} \models \varphi'(\mathcal{R}(\vec{T})).$$

Moreover, if φ is an $\mathrm{FO}_\eta(\mathfrak{M}^{\mathrm{u}})$*-formula, then φ' can be chosen to be an* $\mathrm{FO}_\eta(\mathfrak{M})$*-formula.*

Using this Lemma, together with the encoding $\mathcal{R}(\cdot)$, we show that Fact 1 extends from ranked to unranked trees.

Theorem 1 *Let X be a subset of* $\textsc{UTree}(\Sigma)$. *Then the following are equivalent:*

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

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

So, $\mathfrak{T}^{\mathrm{u}}$ 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 [5, 6], this implies decidability:

Corollary 1 *First-order theories of $\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathrm{u}}$ are decidable.*

The decision procedure is hyperexponential even for ranked trees, as was shown in [5] by reduction from WS1S.

As a corollary of Fact 1, and Lemma 2, we obtain separation of $\mathfrak{T}^{\mathrm{u}}$ and $\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$:

Corollary 2 *The relation $\approx_{\mathrm{dom}}$ is not* FO($\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$)*-definable.*

The proof of Theorem 1 would have been a trivial corollary of Fact 1 and Proposition 1, had the graph of $\mathcal{R}(\cdot)$ been definable in $\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$ or $\mathfrak{T}^{\mathrm{u}}$. 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.

Proposition 2 *The graph of $\mathcal{R}$ is not* FO($\mathfrak{T}^{\mathrm{u}}$)*-definable.*

We introduce the notion of *data complexity* for logics over infinite structures as follows. For a complexity class $\mathcal{C}$, we say that the data complexity of such a logic over $\mathfrak{M}$ (e.g., FO($\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$) or $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$) 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 1, we see that the data complexity of both FO($\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$) and FO($\mathfrak{T}^{\mathrm{u}}$) is polynomial, since formulae can be converted into unranked tree automata. Moreover, a recent result [30] places the complexity of unranked regular tree languages in DLOGSPACE (for ranked trees, the bound is NC^1 [21]). We shall see a number 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 hyperexponential for both $\mathfrak{T}^{\mathrm{u}}_{\mathfrak{p}}$ and $\mathfrak{T}^{\mathrm{u}}$).

4. Restricted logics over trees

In this section we present a number of restrictions of FO over both ranked and unranked tree models, that capture some familiar subclasses of regular tree languages, including those definable in $\mathcal{FO}$, monadic path logic [36] (over ranked trees), and several other logics recently proposed in connection with XML research. We shall also establish data complexity results for those logics. All restrictions are based on quantification over branches.

We normally state the results for definable subsets of $\textsc{Tree}(\Sigma)$ or $\textsc{UTree}(\Sigma)$. For structures with the $\approx_{\mathrm{dom}}$ predicate, the results straightforwardly extend to definability of relations. A k-ary relation R on trees over Σ 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.

We start by looking at *ranked* trees, and show that $\mathrm{FO}_\eta(\mathfrak{T})$ and $\mathrm{FO}_\eta(\mathfrak{T}_{\mathfrak{p}})$ capture familiar classes of regular tree languages.

Theorem 2 1. *A set of ranked trees is $\mathcal{FO}$-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T}_{\mathfrak{p}})$*-definable.*

2. *A set of ranked trees is $\mathcal{MSO}^{\mathrm{path}}$-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T})$*-definable.*

Proof sketch. Coding $\mathcal{FO}$ in $\mathrm{FO}_\eta(\mathfrak{T}_{\mathfrak{p}})$ and $\mathcal{MSO}^{\mathrm{path}}$ in $\mathrm{FO}_\eta(\mathfrak{T})$ is routine. For the other direction, we show that for every $k \geq 0$, there is $m > 0$ such that $T_1 \equiv_m T_2$ implies that the duplicator wins in k rounds in the branch-restricted EF game on $(\mathfrak{T}_{\mathfrak{p}}, T_1)$ and $(\mathfrak{T}_{\mathfrak{p}}, T_2)$ (for the first item). For the second item, we show that likewise one can find $m > 0$ such that $T_1 \equiv_m^{\mathcal{MSO}^{\mathrm{path}}} T_2$ implies the win for the duplicator in the branch-restricted game on $(\mathfrak{T}, T_1)$ and $(\mathfrak{T}, T_2)$. The result easily follows from these claims. $\square$

Theorem 2, 1), extends to the unranked case:

Proposition 3 *A set of unranked trees is $\mathcal{FO}$-definable iff it is* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$*-definable.*

We now turn our attention to $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$, first-order logic with branch quantification over $\mathfrak{T}^{\mathrm{u}}$. We show that this logic is quite powerful, that it corresponds to an unranked analog of $\mathcal{MSO}^{\mathrm{path}}$, and yet has low data complexity.

The first result, which will follow immediately from Theorem 3 (to be proved shortly), shows that with branch quantification over $\mathfrak{T}^{\mathrm{u}}$, one can express standard XML computations such as DTD validation and XPath pattern-matching.

Proposition 4
- *For any DTD d, there exists an* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ *formula φ_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 now show that $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ can be described by a natural extension of the path logic to unranked trees. Over ranked trees, we allow quantification over chains with respect to the $<_{\mathrm{pre}}$ partial order. For unranked trees, we use the vocabulary $(<_{\mathrm{pre}}, <_{\mathrm{sib}}, (O_a))$, and hence the extension, called $\mathcal{MSO}^{\downarrow}_{\rightarrow}$, allows quantification over both *vertical* chains (with respect to $<_{\mathrm{pre}}$) and *horizontal* chains (those with respect to $<_{\mathrm{sib}}$). In other words, in $\mathcal{MSO}^{\downarrow}_{\rightarrow}$, quantification is over sets X such that X is either linearly ordered by $<_{\mathrm{pre}}$, or linearly ordered by $<_{\mathrm{sib}}$. Note that in the latter case, X must be a set of children of the same node.

Theorem 3 *A subset of* $\mathrm{UTREE}(\Sigma)^k$*, $k \geq 1$, is definable in* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ *iff it is definable in $\mathcal{MSO}^{\downarrow}_{\rightarrow}$.*

Proof sketch. For the $\mathcal{MSO}^{\downarrow}_{\rightarrow} \subseteq \mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$ direction we show how to code vertical and horizontal chains with branches. For the other direction, we again use EF games and show that for each k, there is an m such that $T_1 \equiv_m^{\mathcal{MSO}^{\downarrow}_{\rightarrow}} T_2$ implies the win for the duplicator in the k-round branch-restricted game on $(\mathfrak{T}^{\mathrm{u}}, T_1)$ and $(\mathfrak{T}^{\mathrm{u}}, T_2)$. This is done in two stages: in the first stage the game is further restricted to branches of T_1 and T_2, and in the second stage it mimics the $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ game by calculating $\mathcal{MSO}$ types of the string associated with each level of an unranked branch, and playing $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ moves corresponding to the chains of nodes with the same type. $\square$

As one corollary of Theorem 3, we obtain the separation $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}) \subsetneq \mathrm{FO}(\mathfrak{T}^{\mathrm{u}})$ (since $\mathcal{MSO}^{\downarrow}_{\rightarrow} \subsetneq \mathcal{MSO}$, which follows from the fact that over ranked trees, path logic is properly contained in $\mathcal{MSO}$ [36]).

Next, we show that it has low data complexity. Recall that 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 NC^1 [38], we show:

Proposition 5 *The data-complexity of $\mathcal{MSO}^{\downarrow}_{\rightarrow}$ is* NC^1.

We conclude this section by connecting an extension of $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ with another logic studied in connection with XML, as an abstraction of XML pattern languages: $\mathcal{FOREG}$ [26]. $\mathcal{FOREG}$ 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 $\mathrm{FO}^{\mathrm{reg}}_\eta$. This is FO_η 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 $s_1 <_{\mathrm{pre}} \cdots <_{\mathrm{pre}} s_p$ and $f(s_1) \cdots f(s_p) \in L(r)$;
- $r^{\rightarrow}(T)$ iff $T = (D, f)$ is a branch, $\{s_1, \ldots, s_p\} = \{s \mid s \leq_{\mathrm{sib}} e(T)\}$ with $s_1 <_{\mathrm{sib}} \cdots <_{\mathrm{sib}} s_p$, $e(T)$ the endpoint of T, and $f(s_1) \cdots f(s_p) \in L(r)$.

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

Theorem 4 *A set $X \subseteq$* $\mathrm{UTREE}(\Sigma)$ *is definable in* $\mathrm{FO}^{reg}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ *iff X is definable in $\mathcal{FOREG}$.*

Note that $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ can express the properties of Proposition 4. Since $\mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}) \subseteq \mathrm{FO}^{\mathrm{reg}}_\eta(\mathfrak{T}^{\mathrm{u}}) = \mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$, we conclude that its data complexity is NC^1. The bound cannot be lowered, due to the completeness of regular languages for NC^1 [38].

5. Query languages

The setting of first-order definability over structures whose elements are trees extends smoothly to definability in relational calculus, that is, in the presence of a finite relational structure whose elements are trees. In the

XML setting, the most likely scenario is that of a unary database vocabulary, corresponding to several repositories of trees. In general, however, there is no need to impose such a restriction, and we may consider arbitrary relations over $\mathrm{UTREE}(\Sigma)$.

Formally, in this setting we have a relational vocabulary σ; each m-ary relation S in σ is interpreted as a *finite* subset of $\mathrm{UTREE}(\Sigma)^m$. The logics we consider are $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$ and $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ (we shall also see some restrictions later). These extend $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ and $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}})$ by allowing atomic formulae of the form $S(T_1, \ldots, T_m)$, for S in σ.

We shall also use *active-domain* quantification $\exists T \in \mathit{adom}$ and $\forall T \in \mathit{adom}$. The active domain of a σ-structure $\mathcal{A}$ over $\mathrm{UTREE}(\Sigma)$ is the set $\mathit{adom}(\mathcal{A})$ of all $T \in \mathrm{UTREE}(\Sigma)$ such that T occurs in one of the tuples of a relation S in $\mathcal{A}$. We write $(\mathfrak{M}, \mathcal{A}) \models \exists T \in \mathit{adom}\ \varphi(T, \cdot)$ to mean that for some $T_0 \in \mathit{adom}(\mathcal{A})$, $(\mathfrak{M}, \mathcal{A}) \models \varphi(T_0, \cdot)$.

5.1 Relational calculi over $\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}$ and $\mathfrak{T}^{\mathrm{u}}$: expressiveness and complexity

We start by studying the expressiveness and complexity of $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$ and $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$. The general 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.

The upper bounds theorem not only provides us with a PH bound (which is not very useful for proving expressivity bounds), but it also gives us a bound on the complexity of *generic* queries. A generic (Boolean) query is a collection Q of isomorphism types of σ-structures. It is expressible in $\mathrm{FO}(\mathfrak{M}, \sigma)$ if there is a sentence Φ such that for any σ-structure $\mathcal{A}$ over $\mathfrak{M}$, $(\mathfrak{M}, \mathcal{A}) \models \Phi$ if the isomorphism type of $\mathcal{A}$ is in Q.

Define a generic encoding of a structure $\mathcal{A}$, $\mathit{enc}_{\mathrm{gen}}(\mathcal{A})$, with an n-element active domain $\{T_1, \ldots, T_n\}$ as an encoding in which T_i is encoded as i in binary. We say that *generic data complexity* of $\mathrm{FO}(\mathfrak{M}, \sigma)$ is in complexity class $\mathcal{C}$ if for any σ and any generic query Q expressible in $\mathrm{FO}(\mathfrak{M}, \sigma)$, the language $\{\mathit{enc}_{\mathrm{gen}}(\mathcal{A}) \mid \mathcal{A} \in Q\}$ is in $\mathcal{C}$. Note that a generic encoding is determined by an ordering on the active domain and hence is not unique; however, for a generic Q, the choice of a particular generic encoding does not affect the definition of generic data complexity.

The upper bounds theorem is:

Theorem 5 *1. The data complexity of both* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$ *and* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ *is PH;*

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

Proof sketch. Both 1) and 2) are based on the following quantifier-restriction result. By $\mathrm{FO}_{\mathrm{dom}}(\mathfrak{M}, \sigma)$ we denote the set of $\mathrm{FO}(\mathfrak{M}, \sigma)$ sentences in which quantification is restricted to $\mathrm{dom}(\mathcal{A}) = \bigcup_{T \in \mathit{adom}(\mathcal{A})} \mathrm{dom}(T)$. That is, $(\mathfrak{M}, \mathcal{A}) \models \exists T\ \psi(T, \cdot)$ means that for some T_0 with $\mathrm{dom}(T_0) \subseteq \mathrm{dom}(\mathcal{A})$, $(\mathfrak{M}, \mathcal{A}) \models \psi(T_0, \cdot)$. We show that, by combining two game arguments and a similar result for ranked trees, that every $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ sentence is equivalent to an $\mathrm{FO}_{\mathrm{dom}}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ sentence. Both complexity bounds can easily be derived from this. □

Corollary 3 *Parity and transitive closure cannot be expressed in* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$.

The upper bound of Theorem 5 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 $\mathfrak{T}^{\mathrm{u}}$, all one needs is one unary relation and quantification over branches.

Proposition 6 *Let* σ_1 *contain one unary relation* U*, and* σ_2 *one binary relation* E*. Then for every* i*, both* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma_2)$ *and* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}, \sigma_1)$ *(in fact, even* $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$*) can define* Σ^p_i*- and* Π^p_i*-hard problems.*

Proof sketch. The proof for $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma_2)$ is by reduction to a similar result for ranked trees [5]. The proof for $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$ is a reduction from QSAT. For each quantifier-prefix π, we construct an $\mathrm{FO}_\eta(\mathfrak{T}, \sigma_1)$ formula ξ_π such that, if a unary relation S codes a CNF formula φ, then $(\mathfrak{T}, S) \models \xi_\pi$ iff φ preceded by the prefix π is satisfiable. □

5.2. Restricted query languages

Given the high bounds of the previous section, 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, and – unlike in some string models such as $\mathfrak{S}_{\mathrm{p}}$ – over trees one cannot in general get rid of this unrestricted quantification [5]. We therefore impose restrictions on such quantification, following the definition of restricted quantifier normal form (cf. [3, 5]).

Let $\mathrm{FO}^{\mathrm{act}}(\mathfrak{M}, \sigma)$ be the logic that is build from atomic formulae over σ 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 $\mathrm{FO}_\eta^{\mathrm{act}}(\mathfrak{M}, \sigma)$.

Combining the AC^0 data complexity of the relational calculus [1] with the results of the previous section we get:

Corollary 4 *The data complexity of both* $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$ *and* $\mathrm{FO}^{\mathrm{act}}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ *is DLOGSPACE, and the data complexity of* $\mathrm{FO}_\eta^{\mathrm{act}}(\mathfrak{T}^{\mathrm{u}}, \sigma)$ *is* NC^1.

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

As another restriction, we use the logic $\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ of Section 4, which extends $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ 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 $\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$. 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 6 *The data-complexity of* $\mathrm{FO}_\eta^{reg}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$ *is* NC^1.

Proof sketch. This is shown by proving a quantifier-restriction result: it suffices to use active-domain quantification together with quantification restricted to branches of trees in $adom(\mathcal{A})$. This is proved by EF games. □

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 φ_i's. The existence of effective syntax for pure relational calculus is a standard result of relational database theory [1], but [32] 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 [3, 5]. We now extend this result to the calculi studied here for unranked trees.

Proposition 7 *Safe queries in all of* $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$, $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$, $\mathrm{FO}_\eta^{reg}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}, \sigma)$, $\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}, \sigma)$, *and* $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}, \sigma)$ *have effective syntax.*

A slightly more complicated version of range-restriction guarantees safety for calculi with active-domain quantification; we leave the details for the full version.

Logic	Class of Languages	Data Complexity
$\mathrm{FO}(\mathfrak{T}^{\mathrm{u}})$	$\mathcal{MSO}$ = regular	DLOGSPACE
$\mathrm{FO}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$	$\mathcal{MSO}$ = regular	DLOGSPACE
$\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}})$	$\mathcal{MSO}^{\downarrow}_{\rightarrow}$	NC^1
$\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$	$\mathcal{FO}$	AC^0
$\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathrm{u}})$	$\mathcal{MSO}^{\downarrow}_{\rightarrow}$	NC^1
$\mathrm{FO}_\eta^{\mathrm{reg}}(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$	$\mathcal{FOREG}$	NC^1

Figure 3. Definability over $\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}$ and $\mathfrak{T}^{\mathrm{u}}$.

Logic	Class of Languages
$\mathrm{FO}(\mathfrak{T})$	$\mathcal{MSO}$ = regular [5]
$\mathrm{FO}(\mathfrak{T}_{\mathrm{p}})$	$\mathcal{MSO}$ = regular [5]
$\mathrm{FO}_\eta(\mathfrak{T})$	$\mathcal{MSO}^{\mathrm{path}}$
$\mathrm{FO}_\eta(\mathfrak{T}_{\mathrm{p}})$	$\mathcal{FO}$

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

6. Conclusion

Motivated by applications to XML, we examined definability and query languages for unranked trees. The obtained results are summarized in Figure 3–5. The definability results over unranked trees provide a rather complete picture concerning definability in the classical setting versus definability in the model-theory setting. In the second part of the paper, we focused on relational calculi. Depending on the allowed operators, the complexity of these range from AC^0 to PH. All of the calculi in Figure 5, except $\mathrm{FO}_\eta(\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}})$ and the generic restrictions, are suitable for basic XML querying as they are able to express DTDs and XPath.

In the study of definability and querying over automatic structures, we found many commonalities between strings, ranked trees, and unranked trees: for example, the complexity of the relational calculus is in PH, while for generic queries it drops to AC^0; the safe queries always have effective syntax, etc. However, the *proofs* of these results in [3], [5], and here are quite different. We would like to find a better explanation of how the definability by automata affects such basic properties of query languages. We also would like to understand the precise expressiveness of generic queries in relational calculi over automatic structures: we know that AC^0 is an upper bound, but we suspect that these queries may capture an interesting subclass of AC^0 that properly extends first-order. Finally, we would like to find precise descriptions of fragments of various XML query languages (e.g., [12, 11]) that correspond to the logics studied here.

Acknowledgments Part of this work was done when Frank

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

Figure 5. Relational calculi over $\mathfrak{T}^{\mathrm{u}}_{\mathrm{p}}$ and $\mathfrak{T}^{\mathrm{u}}$.

Neven visited the University of Toronto and Leonid Libkin visited the University of Limburg. Libkin was partly supported by NSERC, BUL, and PREA grants. We thank referees for their comments.

References

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

[2] M. Benedikt, W. Fan, G. M. Kuper. Structural Properties of XPath Fragments. In *ICDT'03*.

[3] M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. A model-theoretic approach to regular string relations. In *LICS'01*, pages 431–440.

[4] M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. String operations in query languages. In *PODS'01*, pages 183–194.

[5] M. Benedikt, L. Libkin. Tree extension algebras: logics, automata, and query languages. In *LICS'02*, pages 203–212.

[6] A. Blumensath and E. Grädel. Automatic structures. In *LICS'00*, pages 51–62.

[7] E. Börger, E. Grädel, Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[8] A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets: Version 1, 2001. Tech. Report HKUST-TCSC-2001-0.

[9] A. Brüggemann-Klein and D. Wood. Caterpillars: A context specification technique. *Markup Languages*, 2(1):81–106, 2000.

[10] V. Bruyère, G. Hansel, C. Michaux, R. Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.* 1 (1994), 191–238.

[11] L. Cardelli, G. Ghelli. A query language based on the ambient logic. In *ESOP 2001*, pages 1–22.

[12] D. Chamberlin, et al. XQuery 1.0: An XML query language. `http://www.w3.org/TR/xquery/`, 2002.

[13] J. Clark. XSL transformations version 1.0. http://www.w3.org/TR/WD-xslt, august 1999.

[14] J. Clark. XML Path Language (XPath). `http://www.w3.org/TR/xpath`.

[15] H. Comon et al. *Tree Automata: Techniques and Applications*. Available at `www.grappa.univ-lille3.fr/tata`.

[16] H.-D. Ebbinghaus, J. Flum. *Finite Model Theory*. Springer Verlag, Berlin, 1995.

[17] G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *LICS'02*, pages 189–202.

[18] H. Hosoya and B. C. Pierce. Regular expression pattern matching for XML. In *POPL'01*, pages 67–80.

[19] B. Hodgson. Décadabilité par automate fini. *Ann. Sc. Math. Québec*, 7 (1983), 39-57.

[20] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC'94*, pages 367–392.

[21] M. Lohrey. On the parallel complexity of tree automata. In *RTA'01*, pages 201–215.

[22] M. Müller, J. Niehren, R. Treinen. The first-order theory of ordering constraints over feature trees. In *LICS'98*, pages 432–443.

[23] M. Müller, J. Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. In *RTA'98*, pages 196–210.

[24] M. Murata. Extended path expressions for XML. In *PODS'01*, pages 126–137.

[25] F. Neven. Automata, logic, and XML. *CSL'02*, pp. 2–26.

[26] F. Neven, T. Schwentick. Expressive and efficient pattern languages for tree-structured data. *PODS'00*, pp. 145–156.

[27] F. Neven and T. Schwentick. Query automata on finite trees. *TCS*, 275:633–674, 2002.

[28] C. Pair and A. Quere. Définition et etude des bilangages réguliers. *Information and Control*, 13(6):565–593, 1968.

[29] Y. Papakonstantinou and V. Vianu. DTD inference for views of XML data. In *PODS'01*, pages 35–46.

[30] L. Segoufin. Typing and querying XML documents: some complexity bounds. In *PODS'03*, to appear.

[31] T. Schwentick. On diving in trees. *MFCS'01*, pp. 660–669.

[32] A. Stolboushkin, M. Taitslin. Finite queries do not have effective syntax. *Information and Computation*, 153(1) (1999), 99–116.

[33] D. Suciu. Typechecking for semistructured data. In *DBPL'01*, pages 1–20.

[34] M. Takahashi. Generalizations of regular sets and their application to a study of context-free languages. *Information and Control*, 27(1):1–36, 1975.

[35] W. Thomas. Languages, automata, and logic. *Handbook of Formal Languages, Vol. 3*, Springer, 1997.

[36] W. Thomas. Logical aspects in the study of tree languages. *CAAP'84*, pages 31–50.

[37] V. Vianu. A Web Odyssey: From Codd to XML. In *PODS'01*, pages 1–15.

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