Logics for Unranked Trees: An Overview

Leonid Libkin[1⋆]

University of Toronto
Email: libkin@cs.toronto.edu

Abstract. Labeled unranked trees are used as a model of XML documents, and logical languages for them have been studied actively over the past several years. Such logics have different purposes: some are better suited for extracting data, some for expressing navigational properties, and some make it easy to relate complex properties of trees to the existence of tree automata for those properties. Furthermore, logics differ significantly in their model-checking properties, their automata models, and their behavior on ordered and unordered trees. In this paper we present a survey of logics for unranked trees.

1 Introduction

Trees arise everywhere in computer science, and there are numerous formalisms in the literature for describing and manipulating trees. Some of these formalisms are declarative and based on logical specifications: for example, first-order logic, or monadic second-order logic, or various temporal or fixed-point logics over trees. Others are procedural formalisms such as various flavors of tree automata, or tree transducers, or tree grammars. All these formalisms have found numerous applications in verification, program analysis, logic programming, constraint programming, linguistics, and databases.

Until recently, most logical formalisms for trees dealt with *ranked* trees [18, 58]: in such trees, all nodes have the same fixed number of children (or, a bit more generally, the number of children of a node is determined by the label of that node). Over the past several years, however, the focus has shifted towards *unranked* trees, in which there are no restrictions on the number of children a node can have. Although unranked trees have been considered in the 60s and 70s, and are related to feature trees over an infinite set of features that have been investigated by computational linguists, their systematic study was initiated by the development of XML (eXtensible Markup Language). XML is a data format which has become the lingua franca for information exchange on the world wide web. XML data is typically modeled as labeled unranked trees [42].

This connection has led to a renewed interest in logical and procedural formalisms for unranked trees: one uses logical formalisms for expressing declarative queries, and procedural formalisms for evaluating them. Logics over unranked trees appeared in large numbers over the past 7–8 years, and they come in many flavors in shapes. Common to them is a close connection to automata models, and quite often to temporal and modal logics, especially when one describes properties of paths through a document.

Let us now review some of the parameters according to which logics for unranked trees can be classified.

⋆ Complete version of this survey can be found at www.cs.toronto.edu/~libkin/publ.html.

The yardstick logic Most formalisms are "inspired" by either *first-order logic* (FO), or *monadic second-order logic* (MSO) that extends FO by quantification over sets. Query languages and schema formalisms for XML tend to use MSO as the yardstick: for example, XML DTDs are (almost) equivalent to MSO sentences, and various language for extraction of data from XML documents have the power of MSO unary queries. On the other hand, navigational aspects of XML, in particular, logics capturing various fragments of XPath, are usually related to FO and its fragments.

Arity of queries Most commonly one considers Boolean or unary queries. Boolean queries evaluate to *true* or *false*. Checking if an XML document conforms to a schema specification is represented by a Boolean query. Unary queries correspond to formulae in one free variable, and thus produce a set of nodes. E.g., extracting sets of nodes, or evaluating XPath expressions relative to the root naturally give rise to unary queries.

Complexity of model-checking The model-checking problem asks whether a tree T satisfies a logical sentence φ. If φ is an MSO sentence φ, it can be evaluated in linear time in the size of T, by converting to a tree automaton. But there is a price to pay: in terms of the size of φ, the complexity becomes non-elementary. This type of trade-offs is one of the central issues in dealing with logics over trees.

Ordered vs. unordered trees In unranked XML trees, children of the same node are ordered by a *sibling ordering*. If such an order is present, we speak of ordered unranked trees. In many cases, however, this ordering is irrelevant, and some models, such as feature trees, do not impose any ordering on siblings. There is considerable difference between the expressiveness of logics and automata models depending on the availability of sibling ordering. The presence of ordering also affects the yardstick logic, since without order often counting is needed to match the power of automata models [19].

The paper is organized as follows. After we give basic definitions in Section 2, we review logics for ordered trees in Section 3. We start with MSO-related logics, including syntactic restrictions of MSO, a datalog-based logic, and the μ-calculus. We then turn to FO-related logics, present analogs of LTL and $\mathrm{CTL}^\star$ that have been studied for expressing navigational properties, and also look at conjunctive queries over trees. In Section 4 we turn to trees that lack the sibling ordering, and show that in many logics some form of counting needs to be added to compensate for the missing ordering. In Section 5 we look at the model-theoretic approach in the spirit of automatic structures.

2 Trees, logics, and automata

Tree domains, trees, and operations on trees Nodes in unranked trees are elements of $\mathbb{N}^*$ – that is, finite strings whose letters are natural numbers. A string $s = n_0 n_1 \dots$ defines a path from the root to a give node: one goes to the n_0th child of the root, then to the n_1th child, etc. We write $s_1 \cdot s_2$ for the concatenation of strings s_1 and s_2.

We need some basic binary relations on $\mathbb{N}^*$ – the *child* and *next-sibling* relations:

$$s \prec_{\mathrm{ch}} s' \;\Leftrightarrow\; s' = s \cdot i \text{ for some } i \in \mathbb{N};$$
$$s \prec_{\mathrm{ns}} s' \;\Leftrightarrow\; s = s_0 \cdot i \text{ and } s' = s_0 \cdot (i+1) \text{ for some } s_0 \in \mathbb{N}^* \text{ and } i \in \mathbb{N}.$$

We also use the *first child relation*: $s \prec_{\mathrm{fc}} s \cdot 0$. We shall use * to denote the reflexive-transitive closure of a relation. Thus, $\prec^*_{\mathrm{ch}}$ is the *descendant* relation (including self), and $\prec^*_{\mathrm{ns}}$ is a linear ordering on siblings.

Definition 1 (Tree domain). *A* tree domain D *is a finite prefix-closed subset of* $\mathbb{N}^*$ *(i.e., if* $s \in D$ *and* s' *is a prefix of* s*, then* $s' \in D$*) such that* $s \cdot i \in D$ *implies* $s \cdot j \in D$ *for all* $j < i$*.*

Let Σ be a finite alphabet.

Definition 2 (Σ-trees). *An* ordered unranked Σ-labeled tree T *is a structure*

$$T = \langle D, \prec_{\mathrm{ch}}^*, \prec_{\mathrm{ns}}^*, (P_a)_{a\in\Sigma}\rangle,$$

where D *is a tree domain,* $\prec_{\mathrm{ch}}^*$ *and* $\prec_{\mathrm{ns}}^*$ *are the descendant relation and the sibling ordering, and* P_a*'s are interpreted as disjoint sets whose union is the entire domain* D*.*

An unordered *unranked tree is defined as a structure* $\langle D, \prec_{\mathrm{ch}}^*, (P_a)_{a\in\Sigma}\rangle$*.*

Thus, a tree consists of a tree domain together with a labeling on its nodes: if $s \in P_a$, then the label of s is a. In this case we write $\lambda_T(s) = a$.

First-order and monadic second-order logic We only consider relational vocabularies: finite lists $(R_1, \ldots, R_m)$ of relation symbols, each R_i with an associated arity n_i. Over trees, relation symbols are binary (e.g., $\prec_{\mathrm{ch}}, \prec_{\mathrm{ns}}, \prec_{\mathrm{ch}}^*$) or unary ($P_a$'s for $a \in \Sigma$).

Formulae of *first-order* logic (FO) are built from atomic formulae $x = x'$, and $R(\bar{x})$, where x, x' are variables, and $\bar{x}$ is a tuple of variables, using the Boolean connectives $\vee, \wedge, \neg$ and quantifiers $\exists$ and $\forall$. If a formula φ has free variables $\bar{x}$, we shall write $\varphi(\bar{x})$.

Formulae of *monadic second-order* logic (MSO) in addition allow quantification over sets. We shall normally denote sets of nodes by upper case letters. Thus, MSO formulae have the usual first-order quantifiers $\exists x\varphi$ and $\forall x\varphi$ as well as second-order quantifiers $\exists X\varphi$ and $\forall X\varphi$, and new atomic formulae $X(x)$, where X is a second-order variable and x is a first-order variable. An MSO formula may have both free first-order and second-order variables. If it only has free first-order variables, then it defines a relation on the universe of the structure.

Note that relations $\prec_{\mathrm{ch}}$ and $\prec_{\mathrm{ns}}$ are definable, in FO, from $\prec_{\mathrm{ch}}^*$ and $\prec_{\mathrm{ns}}^*$. In MSO one can define $\prec_{\mathrm{ch}}^*$ from $\prec_{\mathrm{ch}}$; however, it is well-known that in FO this is *not* possible. This is why we chose $\prec_{\mathrm{ch}}^*$ and $\prec_{\mathrm{ns}}^*$, rather than $\prec_{\mathrm{ch}}$ and $\prec_{\mathrm{ns}}$, as our basic relations.

Definition 3 (Definability in logic). *Given a logic* $\mathcal{L}$*, we say that a set of trees* $\mathcal{T}$ *is definable in* $\mathcal{L}$ *if there is a sentence* φ *of* $\mathcal{L}$ *such that* $T \in \mathcal{T}$ *iff* $T \models \varphi$*. We say that a unary query* $\mathcal{Q}$ *(that selects nodes from trees) is definable in* $\mathcal{L}$ *if there is a formula* $\psi(x)$ *of* $\mathcal{L}$ *such that* $s \in \mathcal{Q}(T)$ *iff* $T \models \psi(s)$*, for every tree* T *and a node* s *in* T*.*

Unranked tree automata An *nondeterministic unranked tree automaton, NUTA* [56, 9], over Σ-labeled trees is a triple $\mathcal{A} = (Q, F, \delta)$ where Q is a finite set of states, $F \subseteq Q$ is the set of final states, and δ is a mapping $Q \times \Sigma \to 2^{Q^*}$ such that $\delta(q, a)$ is a regular language over Q (normally represented by a regular expression over Q). A *run* of $\mathcal{A}$ on a tree T with domain D is a function $\rho_{\mathcal{A}} : D \to Q$ such that, if s is a node with n children, and it is labeled a, then the string $\rho_{\mathcal{A}}(s\cdot 0)\cdots\rho_{\mathcal{A}}(s\cdot(n-1))$ is in $\delta(\rho_{\mathcal{A}}(s), a)$.

In particular, if s is a leaf labeled a, then $\rho_{\mathcal{A}}(s) = q$ implies that $\varepsilon \in \delta(q, a)$. A run is *accepting* if $\rho_{\mathcal{A}}(\varepsilon) \in F$, that is, the root is in an accepting state. A tree T is *accepted* by $\mathcal{A}$ if there exists an accepting run. We let $L(\mathcal{A})$ denote the set of all trees accepted by $\mathcal{A}$. Such sets of trees will be called *regular*.

Binary trees and translations A *binary tree domain* is a prefix-closed subset D of $\{0,1\}^*$ such that if $s \cdot i \in D$, then $s \cdot (1-i) \in D$ (that is, a node is either a leaf, or both its children are in D). A (binary) *nondeterministic tree automaton, NTA,* is a quadruple $\mathcal{A}_b = (Q, q_0, F, \delta)$ where Q and F are as before, q_0 is the initial state, and δ is a function $Q \times Q \times \Sigma \to 2^Q$. A run $\rho_{\mathcal{A}_b}$ on a binary tree T with domain D is a function from D to Q such that if s is a leaf labeled a, then $\rho_{\mathcal{A}_b}(s) \in \delta(q_0, q_0, a)$, and if $s \cdot 0, s \cdot 1$ belong to D, and s is labeled a, then $\rho_{\mathcal{A}_b}(s) \in \delta(\rho_{\mathcal{A}_b}(s \cdot 0), \rho_{\mathcal{A}_b}(s \cdot 1), a)$. A run is accepting if $\rho_{\mathcal{A}_b}(\varepsilon) \in F$, and $L(\mathcal{A}_b)$ is the set of all binary trees for which there exists an accepting run. Such sets are called regular.

There is a well-known regularity-preserving translation between ranked and unranked trees. It was used in [49] to show decidability of SωS (but here we shall apply it only to finite tree domains). The idea of the translation is that the first successor in the binary tree corresponds to the first child, and the second successor to the next sibling. More precisely, we define a mapping $\mathcal{R} : \mathbb{N}^* \to \{0,1\}^*$ such that $\mathcal{R}(\varepsilon) = \varepsilon$, and if $\mathcal{R}(s) = s'$, where $s = s_0 \cdot i$, then $\mathcal{R}(s \cdot 0) = s' \cdot 0$ and $\mathcal{R}(s_0 \cdot (i+1)) = s' \cdot 1$. If D is an unranked tree domain, we let $\mathcal{R}(D)$ be $\{\mathcal{R}(s) \mid s \in D\}$ together with $\mathcal{R}(s) \cdot 1$ if s is a non-leaf last child, and $\mathcal{R}(s) \cdot 0$ if s a leaf, other than the last sibling (these additions ensure that $\mathcal{R}(D)$ is a binary tree domain). We define $\mathcal{R}(T)$ to be a tree with domain $\mathcal{R}(D)$, where $\mathcal{R}(s)$ has the same label as s, and the added nodes are labeled by a symbol $\perp \notin \Sigma$. The following is a folklore result.

Lemma 1. *For every NUTA $\mathcal{A}$, there is an NTA $\mathcal{A}_b$ such that $L(\mathcal{A}_b) = \{\mathcal{R}(T) \mid T \in L(\mathcal{A})\}$, and for every NTA $\mathcal{A}_b$ there is an NUTA $\mathcal{A}$ such that the above holds.*

3 Ordered trees

In this section we only deal with ordered unranked trees. We first survey MSO-based logics, and then move to FO-based ones.

3.1 MSO and its relatives

As we mentioned already, MSO is often used as a yardstick logic for trees, because of its close connection to regular languages. The following result belonged to folklore, and was explicitly stated in [41].

Theorem 1. *A set of unranked trees is regular iff it is definable in* MSO.

When one considers binary trees, this result says that regular sets of binary trees are precisely those MSO-definable, and if we look at strings, which may be viewed as trees without branching, we obtain that regular languages are precisely those MSO-definable. Of course these are well-known results by Büchi [10], and Thatcher, Wright [57].

There is also a close connection between automata, MSO, and a formalism for describing XML schemas, called DTDs (which are essentially extended context-free grammars). A DTD d over an alphabet Σ is a collection of rules $a \to e_a$, where $a \in \Sigma$ and e_a is a regular expression over Σ. We shall assume there is at most one such rule for each $a \in \Sigma$. A Σ-labeled tree T satisfies d, if for each node s of T with n children, and $\lambda_T(s) = a$, the string $\lambda_T(s \cdot 0) \cdots \lambda_T(s \cdot (n-1))$ is in the language denoted by e_a.

Each DTD is easily definable by an unranked tree automaton: in fact its states just correspond to labels of nodes. This, however, is too restrictive to capture full definability in MSO, but a slight extension of DTDs does precisely that. An *extended DTD* over Σ is a triple (Σ', d', g) where $\Sigma' \supseteq \Sigma$, with g being a mapping $g : \Sigma' \mapsto \Sigma$, and d' is a DTD over Σ'. A Σ-labeled tree T satisfies (Σ', d', g) if there is a Σ'-labeled tree T' that satisfies d' such that $T = g(T')$. The following was established in [56].

Proposition 1. *A set of unranked trees is* MSO *definable iff it is the set of all trees satisfying some extended DTD* (Σ', d', g).

Theorem 1 talks about MSO sentences, but it can be extended to unary MSO queries using the concept of *query automata* [44]. A (nondeterministic) *query automaton* over unranked Σ-labeled trees is a quadruple $\mathcal{QA} = (Q, F, \delta, S)$ where $\mathcal{A} = (Q, F, \delta)$ is an UNTA, and S is a subset of $Q \times \Sigma$. Such a query automaton defines a unary query $\mathcal{Q}_{\mathcal{QA}}$ that selects nodes s in T such that $(\rho_{\mathcal{A}}(s), \lambda_T(s)) \in S$ for some accepting run $\rho_{\mathcal{A}}$.

Theorem 2. (see [44, 41, 24]) *A unary query $\mathcal{Q}$ on unranked trees is* MSO*-definable iff it is of the form $\mathcal{Q}_{\mathcal{QA}}$ for some query automaton.*

One can also define the semantics universally ($(\rho_{\mathcal{A}}(s), \lambda_T(s)) \in S$ for all accepting runs) and the result still holds. Query automata have a deterministic counterpart; however, in the deterministic version, two passes over the tree are required; see [44].

Theorems 1 and 2 are constructive. In particular, every MSO sentence φ can be effectively transformed into an automaton $\mathcal{A}_\varphi$ that accepts a tree T iff $T \models \varphi$. Since tree automata can be determinized, this gives us a $O(\|T\|)$ algorithm to check whether $T \models \varphi$, if φ is fixed[1]. However, it is well-known that the size of $\mathcal{A}_\varphi$ (even for string automata) cannot be bounded by an elementary function in $\|\varphi\|$ [55]. An even stronger result of [23] says that there could be no algorithm for checking whether $T \models \varphi$ that runs in time $O(f(\|\varphi\|) \cdot \|T\|)$, where f is an elementary function, unless PTIME=NP.

Nonetheless, these results do not rule out the existence of a logic $\mathcal{L}$ that has the same power as MSO and yet permits faster model-checking algorithms. Even looking at a simpler case of FO on strings, where results of [23] also rule out $O(f(\|\varphi\|) \cdot |s|)$ algorithms for checking if a string s satisfies φ, with f being an elementary function, the logic LTL (linear-time temporal logic) has the same expressiveness as FO [33] and admits model-checking algorithm with running time $2^{O(\|\varphi\|)} \cdot |s|$.

Logic ETL The first logic for unranked trees that has the power of MSO and model-checking complexity matching that of LTL appeared in [43] and was called ETL (*efficient tree logic*). It was obtained by putting syntactic restrictions on MSO formulae, and at the same time adding new constructors for formulae, which are not present in MSO, but are MSO-definable.

The atomic formulae of ETL are the same as for MSO, except that we are allowed to use both $\prec_{\mathrm{ch}}$ and $\prec^*_{\mathrm{ch}}$ and are *not* allowed to use the next-sibling relation $\prec^*_{\mathrm{ns}}$. ETL is closed under Boolean combinations (which are required to be in DNF), *guarded quantification*, and *path formulae*. The rules for guarded quantification are:

[1] We use the notation $\|T\|$, $\|\varphi\|$ to denote the sizes of natural encodings of trees and formulae.

– if $\varphi(x, y, X)$ is an ETL formula, then $\exists y\ (x \prec_{\mathrm{ch}} y \wedge \varphi)$ and $\exists y\ (x \prec^*_{\mathrm{ch}} y \wedge \varphi)$ are ETL formulae;
– if $\varphi(x, X)$ is an ETL formula, then $\exists X\ (x \prec^*_{\mathrm{ch}} X \wedge \varphi)$ is an ETL formula. Here $x \prec^*_{\mathrm{ch}} X$ means that X only contains descendants of x. In this case φ cannot contain vertical path formulae (defined below).

Path formulae are defined as follows:

– if e is a regular expression over ETL formulae $\psi(u, v)$, then $e^{\downarrow}(x, y)$ is a (vertical path) ETL formula. The semantics is as follows: $T \models e^{\downarrow}(s, s')$ if there is a child-relation path $s = s_0, s_1, \ldots, s_n = s'$ in T and a sequence of ETL formulae $\psi_i(u, v)$, $i \leq n - 1$, such that $T \models \psi_i(s_i, s_{i+1})$ for each $i \leq n - 1$, and the sequence $\psi_0 \ldots \psi_{n-1}$ matches e.
– if e is a regular expression over ETL formulae $\psi(u, \bar{X})$, then $e^{\rightarrow}(x, \bar{X})$ is a (horizontal path) ETL formula. Then $T \models e^{\rightarrow}(s, \bar{X})$ if children $s \cdot i, i \leq k$ of s can be labeled with ETL formulae $\psi_i(u, \bar{X})$ such that $T \models \psi_i(s \cdot i, \bar{X})$ for all i, and the sequence $\psi_0 \ldots \psi_k$ matches e.

Theorem 3. (see [43]) *With respect to Boolean and unary queries,* ETL *and* MSO *are equally expressive. Furthermore, each* ETL *formula φ can be evaluated on a tree T in time $2^{O(\|\varphi\|)} \cdot \|T\|$.*

Monadic datalog Another approach to obtaining the full power of MSO while keeping the complexity low is based on database query language *datalog* (cf. [1]). A datalog program is a sequence of rules H:–$P_1, \ldots, P_k$ where H and all P_i's are atomic formulae. The predicate H is called the head of the rule, and every variable that appears in H is required to appear in one of the P_i's. Given a datalog program $\mathcal{P}$, predicates which appear as a head of some rule are called intensional, and other predicates are called extensional. If all intensional predicates are monadic (of the form $H(x)$), then $\mathcal{P}$ is a *monadic* datalog program. The semantics is a standard fixed-point semantics, see, e.g., [1]. An intensional unary predicate of a program $\mathcal{P}$ defines a unary query.

For extensional predicates, we shall need *Leaf, LastChild,* and *Root.* Given a tree domain D, they are interpreted as *Leaf* $= \{s \in D \mid \neg\exists s' \in D : s \prec_{\mathrm{ch}} s'\}$, *LastChild* $= \{s \cdot i \in D \mid s \cdot (i + 1) \notin D\}$ and *Root* $= \{\varepsilon\}$.

Theorem 4. (see [25]) *A unary query over unranked trees is definable in* MSO *iff it is definable in monadic datalog over extensional predicates $\prec_{\mathrm{fc}}$, $\prec_{\mathrm{ns}}$, Leaf, LastChild, Root, and $P_a, a \in \Sigma$. Furthermore, each monadic datalog query $(\mathcal{P}, H)$ can be evaluated on a tree T in time $O(\|\mathcal{P}\| \cdot \|T\|)$.*

μ-calculus Yet another way of getting a logic equivalent to MSO is suggested by a close connection between MSO and the modal μ-calculus L_μ on ranked trees, which can easily be extended to the unranked case by using the connection between ranked and unranked trees. It was shown in [22, 47] that every property of infinite binary trees definable in MSO is also be definable in L_μ. To deal with unranked trees, we shall define L_μ over Σ-labeled structures that have several binary relations $E_1, \ldots, E_m$, cf. [2]. Formulae of $L_\mu[E_1, \ldots, E_m]$ are given by

$$\varphi := a\ (a \in \Sigma) \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \Diamond(E_i)\varphi \mid \mu X\ \varphi(X),$$

where in $\mu X\ \varphi(X)$, the variable X must occur positively in φ. Given a tree T with domain D, $s \in D$, and a valuation v for free variables (each $v(X)$ is a subset of D), we define the semantics (omitting the rules for letters $a \in \Sigma$ and Boolean connectives) by

- $(T, v, s) \models X$ iff $s \in v(X)$.
- $(T, v, s) \models \Diamond(E_r)\varphi$ iff $(T, v, s') \models \varphi$ for some s' with $(s, s') \in E_r$.
- $(T, v, s) \models \mu X\, \varphi(X)$ iff s is in the least fixed point of the operator defined by φ.

An L_μ formula φ without free variables naturally defines a unary query on trees ($\{s \mid (T, s) \models \varphi\}$) and a Boolean query on trees (by checking if $(T, \varepsilon) \models \varphi$).

Using the translation into ranked trees, it is easy to show (see [3]):

Proposition 2. *The class of Boolean* MSO *queries on unranked trees is precisely the class of Boolean queries defined by* $L_\mu[\prec_{\mathrm{fc}}, \prec_{\mathrm{ns}}]$.

It is also possible to characterize unary MSO queries over unranked trees in terms of the *full* μ-calculus L_μ^{full} (cf. [59]) which adds backward modalities $\Diamond(E_i^-)\varphi$ with the semantics $(T, s) \models \Diamond(E_i^-)\varphi$ iff $(T, s') \models \varphi$ for some s' such that $(s', s) \in E_i$.

Proposition 3. (see [3]) *The class of unary* MSO *queries on unranked trees is precisely the class of queries defined by* $L_\mu^{\mathrm{full}}[\prec_{\mathrm{ch}}, \prec_{\mathrm{ns}}]$.

3.2 FO and its relatives

While much is known about FO on both finite and infinite strings, it has not been as extensively studied for trees until recently. Recall that over strings – which we can view as trees with only unary branching – FO defines precisely the star-free languages (cf. [58]), and over both finite and infinite strings FO has exactly the power of LTL [33].

In contrast, the natural analog of star-free expressions over binary trees captures not FO but MSO [48]. One well-known equivalent logical description of FO on binary trees is Hafer-Thomas's theorem [31] stating that over finite binary trees, $\mathrm{FO} = \mathrm{CTL}^\star$ ($\mathrm{CTL}^\star$ is a branching time temporal logic widely used in verification, cf. [16], and it will be defined shortly). Actually, the result of [31] shows that $\mathrm{CTL}^\star$ is equivalent to MSO with second-order quantification over paths only, but over finite trees this fragment of MSO is equivalent to FO.

The interest in logics over unranked trees whose power is equal to or subsumed by that of FO stems from the fact that navigational features of XPath can be described in FO. XPath [17] is a W3C standard for describing paths in XML documents. Thus, it is very natural to look for connections between XPath, FO on trees, and temporal logics, which are designed to talk about properties of paths.

Logics introduced in the context of studying XPath, and more generally, navigational properties of XML documents, can be roughly subdivided into two groups. Firstly, one may try to establish analogs of Kamp's theorem (stating that $\mathrm{FO} = \mathrm{LTL}$ over strings) for trees. Secondly, one can try extended Hafer-Thomas's theorem (the equivalence $\mathrm{FO} = \mathrm{CTL}^\star$) from binary to unranked trees.

XPath and temporal logics First, recall the syntax of LTL over alphabet Σ:

$$\varphi, \varphi' \ :=\ a, a \in \Sigma \ \mid\ \varphi \vee \varphi' \ \mid\ \neg\varphi \ \mid\ \mathbf{X}\varphi \ \mid\ \mathbf{X}^-\varphi \ \mid\ \varphi\mathbf{U}\varphi' \ \mid\ \varphi\mathbf{S}\varphi'.$$

Formulae of LTL are interpreted over finite or infinite strings over Σ. Given a string $s = a_0 a_1 \ldots$, the semantics is as follows: $(s,i) \models a$ iff $a_i = a$, $(s,i) \models \mathbf{X}\varphi$ ("next" φ) iff $(s,i+1) \models \varphi$; $(s,i) \models \mathbf{X}^{-}\varphi$ iff $(s,i-1) \models \varphi$; $(s,i) \models \varphi \mathbf{U} \varphi'$ (φ "until" φ') if there exists $j \geq i$ such that $(s,j) \models \varphi'$ and $(s,k) \models \varphi$ for all $i \leq k < j$, and the semantics of the dual $\varphi \mathbf{S} \varphi$ (φ "since" φ') is that there exists $j \leq i$ such that $(s,j) \models \varphi'$ and $(s,k) \models \varphi$ for all $j < k \leq i$. (Note: it is possible to avoid $\mathbf{X}$ and $\mathbf{X}^{-}$ by defining a strict semantics for $\mathbf{U}$ and $\mathbf{S}$, without requiring φ to be true in (s,i)).

A logic TL$^{\text{tree}}$ (*tree temporal logic*) is a minor extension of LTL:

$$\varphi, \varphi' \ := \ a, a \in \Sigma \ \mid \ \varphi \vee \varphi' \ \mid \ \neg\varphi \ \mid \ \mathbf{X}_{*}\varphi \ \mid \ \mathbf{X}^{-}_{*}\varphi \ \mid \ \varphi \mathbf{U}_{*} \varphi' \ \mid \ \varphi \mathbf{S}_{*} \varphi',$$

where $*$ is either 'ch' (child) or 'ns' (next sibling). We define the semantics with respect to a tree and a node in a tree: $(T,s) \models a$ iff $\lambda_T(s) = a$; $(T,s) \models \mathbf{X}_{\text{ch}}\varphi$ if $(T, s \cdot i) \models \varphi$ for some i; $(T,s) \models \mathbf{X}^{-}_{\text{ch}}\varphi$ if $(T,s') \models \varphi$ for the node s' such that $s' \prec_{\text{ch}} s$; $(T,s) \models \varphi \mathbf{U}_{\text{ch}} \varphi'$ if there is a node s' such that $s \prec^{*}_{\text{ch}} s'$, $(T,s') \models \varphi'$, and for all $s'' \neq s'$ satisfying $s \prec^{*}_{\text{ch}} s'' \prec^{*}_{\text{ch}} s'$ we have $(T,s'') \models \varphi$. The semantics of $\mathbf{S}_{\text{ch}}$ is defined by reversing the order in the semantics of $\mathbf{U}_{\text{ch}}$, and the semantics of $\mathbf{X}_{\text{ns}}$, $\mathbf{X}^{-}_{\text{ns}}$, $\mathbf{U}_{\text{ns}}$, and $\mathbf{S}_{\text{ns}}$ is the same by replacing the child relation with the next sibling relation.

As L_μ, the logic TL$^{\text{tree}}$ naturally defines unary and Boolean queries on trees.

Theorem 5. (see [38]) *A unary or Boolean query over unranked trees is definable in* FO *iff it is definable in* TL$^{\text{tree}}$.

In both CTL* and XPath formalisms there are two kinds of formulae: those evaluated in nodes of trees, and those evaluated on paths in trees (these are state and path formulae of CTL* and filter and location path expressions of XPath).

We now look at XPath-inspired logics, and present them using a slight modification of the syntax that keeps all the main XPath constructions and yet makes the connection with temporal logics more visible. The language CXPath [38] (*Conditional XPath*) is defined to have *node formulae* α and *path formulae* β given by:

$$\begin{array}{l} \alpha, \alpha' := a, a \in \Sigma \ \mid \ \neg\alpha \ \mid \ \alpha \vee \alpha' \ \mid \ \mathbf{E}\beta \\ \beta, \beta' := ?\alpha \ \mid \ \mathsf{step} \ \mid \ (\mathsf{step}/?\alpha)^{+} \ \mid \ \beta/\beta' \ \mid \ \beta \vee \beta' \end{array}$$

where step is one of the following: $\prec_{\text{ch}}$, $\prec^{-}_{\text{ch}}$, $\prec_{\text{ns}}$, or $\prec^{-}_{\text{ns}}$. Intuitively $\mathbf{E}\beta$ states the existence of a path starting in a given node and satisfying β, $?\alpha$ tests if α is true in the initial node of a path, and $/$ is the composition of paths.

Formally, given a tree T, we evaluate each node formula in a node s, and each path formula in a pair of nodes (s,s'). The main semantic rules are:

- $(T,s) \models \mathbf{E}\beta$ iff there is s' such that $(T,s,s') \models \beta$;
- $(T,s,s') \models ?\alpha$ iff $s = s'$ and $(T,s) \models \alpha$;
- $(T,s,s') \models \mathsf{step}$ iff $(s,s') \in \mathsf{step}$;
- $(T,s,s') \models \beta/\beta'$ iff for some s'' we have $(T,s,s'') \models \beta$ and $(T,s'',s') \models \beta'$;
- $(T,s,s') \models (\mathsf{step}/?\alpha)^{+}$ if there exists a sequence of nodes $s = s_0, s_1, \ldots, s_k = s'$, $k > 0$, such that each (s_i, s_{i+1}) is in step, and $(T, s_{i+1}) \models \alpha$ for each $i < k$.

The language Core_XPath [26] is obtained by only allowing step^+ as opposed to $(\mathsf{step}/?\alpha)^+$ in the definition of path formulae. Notice that since $\mathsf{step}^+ = (\mathsf{step}/?\mathit{true})$, where $\mathit{true} = \bigvee_{a\in\Sigma} a$, we have Core_XPath $\subseteq$ CXPath.

Core_XPath corresponds to XPath as defined by W3C [17], while CXPath represents an addition to XPath proposed by [38]. Node formulae of either CXPath or Core_XPath naturally define unary queries on trees. These can be characterized as follows.

Theorem 6. *a)* (see [38]) *The node formulae of* CXPath *have precisely the power of* FO *unary queries.*

b) (see [39]) *The node formulae of* Core_XPath *have precisely the power of unary* FO^2 *queries (that is,* FO *with two variables) in the vocabulary* $\prec_{\mathrm{ch}}, \prec^*_{\mathrm{ch}}, \prec_{\mathrm{ns}}, \prec^*_{\mathrm{ns}}$.

A $\mathrm{CTL}^\star$*-like logic* $\mathrm{CTL}^\star$ is a branching time temporal logic used in verification of reactive systems. Here we define it with past connectives, using the syntax close to that of [35]. In $\mathrm{CTL}^\star$, one also has node (normally called state) formulae and path formulae, but path formulae are evaluated on paths, not on arbitrary pairs of nodes.

We define $\mathrm{CTL}^\star_{\mathsf{past}}$ node formulae α, and child and sibling path formulae β_*, for $*$ being 'ch' or 'ns', as follows:

$$\begin{aligned}\alpha, \alpha' &:= a\ (a \in \Sigma) \mid \neg\alpha \mid \alpha \vee \alpha' \mid \mathbf{E}\beta_{\mathrm{ch}} \mid \mathbf{E}\beta_{\mathrm{ns}}\\ \beta_*, \beta'_* &:= \alpha \mid \neg\beta_* \mid \beta_* \vee \beta'_* \mid \mathbf{X}_*\beta_* \mid \mathbf{X}^-_*\beta_* \mid \beta_*\mathbf{U}_*\beta'_* \mid \beta_*\mathbf{S}_*\beta'_*\end{aligned}$$

The semantics is standard and omitted here. The following can be seen as an analog of the equivalence $\mathrm{FO} = \mathrm{CTL}^\star$ for finite binary trees [31].

Theorem 7. (see [3]) *A unary or Boolean query over unranked trees is definable in* FO *iff it is definable in* $\mathrm{CTL}^\star_{\mathsf{past}}$.

Conjunctive queries over unranked trees Conjunctive queries are a very important class of database queries: they correspond to the $\exists, \wedge$-fragment of FO. These are the same queries that can be expressed by selection, projection, and join in relational algebra, and thus they form the core of database queries. The complexity of evaluating a conjunctive query φ over a database $\mathcal{D}$ is in NP, in terms of both the size of φ and the size of $\mathcal{D}$. In fact, the problem is NP-hard, and there has been a large body of work on classifying tractable cases (see, e.g., [28, 30]).

In the case of unranked trees, conjunctive queries are formulae of the form $\varphi(\bar{x}) = \exists\bar{y}\ R_1 \wedge \ldots \wedge R_k$, where each R_i is either $P_a(z)$ or $z \prec z'$, where z, z' are variables among $\bar{x}, \bar{y}$, and $\prec$ is one of $\prec_{\mathrm{ch}}, \prec^*_{\mathrm{ch}}, \prec_{\mathrm{ns}}$, or $\prec^*_{\mathrm{ns}}$. We write $\mathrm{CQ}(\prec_1, \ldots, \prec_m)$ to denote the class of conjunctive queries over unranked trees in which only unary predicates P_a and binary predicates among $\prec_i$ can be used.

Theorem 8. (see [27]) *The maximal tractable classes of queries* $\mathrm{CQ}(\prec_1, \ldots, \prec_m)$*, where all* $\prec_i$*'s are among* $\{\prec_{\mathrm{ch}}, \prec^*_{\mathrm{ch}}, \prec_{\mathrm{ns}}, \prec^*_{\mathrm{ns}}\}$*, are* $\mathrm{CQ}(\prec_{\mathrm{ch}}, \prec_{\mathrm{ns}}, \prec^*_{\mathrm{ns}})$ *and* $\mathrm{CQ}(\prec^*_{\mathrm{ch}})$*; all others are* NP*-hard.*

4 Unordered trees

In unordered trees, nodes can still have arbitrarily many children, but the sibling ordering $\prec_{\rm ns}$ is no longer available. Logics considered for unordered unranked trees typically introduce some form of *counting*, see [3, 19–21, 40, 46, 51, 53, 54].

An explanation for this comes from a modified notion of automata for unordered unranked trees. A *counting nondeterministic unranked tree automaton* is a tuple $\mathcal{A}_c = (Q, F, \delta)$, where Q is a set of states, and $F \subseteq Q$ is a set of final states. Let V_Q be the set of variables $\{v_q^k \mid q \in Q, k > 0\}$. Then the transition function δ maps each pair $(q, a) \in Q \times \Sigma$ into a Boolean function over V_Q. A *run* of $\mathcal{A}$ on an unordered tree T with domain D is a mapping $\rho_{\mathcal{A}_c} : D \to Q$ such that if $\rho_{\mathcal{A}_c}(s) = q$ for a node s labeled a, then the value of $\delta(q, a)$ is 1, where each variable $v_{q_i}^k$ is set to 1 if s has at least k children s' with $\rho_{\mathcal{A}_c}(s') = q_i$, and to 0 otherwise. A run is accepting if $\rho_{\mathcal{A}_c}(\varepsilon) \in F$, and the set of unordered trees accepted by $\mathcal{A}_c$ is denoted by $L_u(\mathcal{A}_c)$.

A *counting query automaton* $\mathcal{QA}_c$ is defined as (Q, F, δ, S) where $S \subseteq Q \times \Sigma$; it selects nodes s in a run ρ where $(\rho_{\mathcal{A}_c}(s), \lambda_T(s)) \in S$. The following appears not to have been stated explicitly, although it follows easily from results in [41, 44, 53].

Theorem 9. *a) A set of unordered unranked trees is* MSO*-definable iff it is of the form $L_u(\mathcal{A}_c)$ for a counting nondeterministic unranked tree automaton $\mathcal{A}_c$.*

b) A unary query over unordered unranked trees is MSO*-definable iff it is definable by a counting query automaton $\mathcal{QA}_c$.*

MSO *and* FO *over unordered trees* Define the *counting μ-calculus* C_μ (cf. [32]) as an extension of L_μ with formulae $\Diamond^{\geq k}(E)\varphi$. The semantics of $(T, s) \models \Diamond^{\geq k}(E)\varphi$ is as follows: there exist distinct elements $s_1, \ldots, s_k$ such that $(s, s_i) \in E$ and $(T, s_i) \models \varphi$ for every $1 \leq i \leq k$. The next result follows from [60], as was noticed in [32]:

Theorem 10. *Over unordered unranked trees,* MSO *and $C_\mu[\prec_{\rm ch}]$ have precisely the same power with respect to Boolean queries.*

For first-order logic, counting extensions of both the temporal logic $\mathrm{TL}^{\text{tree}}$ and $\mathrm{CTL}^\star$ give us analogs of Kamp's and Hafer-Thomas's theorems. Define $\mathrm{TL}^{\text{tree}}_{\text{count}}$ as a version of $\mathrm{TL}^{\text{tree}}$ in which only modalities for the child relation are used, but in addition we have formulae $\mathbf{X}^k_{\text{ch}}\varphi$, with the semantics that $(T, s) \models \mathbf{X}^k_{\text{ch}}\varphi$ iff there are at least k children s' of s such that $(T, s') \models \varphi$.

We also extend $\mathrm{CTL}^\star$ to a logic $\mathrm{CTL}^\star_{\rm count}$ in which we have new state formulae $\mathbf{EX}^k_{\text{ch}}\alpha$, where α is a state formula, with the same semantics as above.

Theorem 11. (see [40, 51]) *Over unordered unranked trees, the classes of Boolean queries expressed in* FO, $\mathrm{TL}^{\text{tree}}_{\text{count}}$, *and* $\mathrm{CTL}^\star_{\rm count}$ *over binary relation $\prec_{\rm ch}$, are the same.*

For unary queries, the equivalence $\mathrm{FO} = \mathrm{TL}^{\text{tree}}_{\text{count}}$ still holds [51], and FO can be shown to be equivalent to an extension of $\mathrm{CTL}^\star$ with both counting and the past [3].

Extensions and more powerful counting Consider now a scenario in which we deal with unordered trees, but in our formulae we can refer to some arbitrary ordering on siblings: after all, in any encoding of a tree, siblings will come in some order. Of course we do not want any particular order to affect the truth value, so we want our formulae, even if they use an ordering, to be independent of a particular ordering that was used.

This is the standard setting of *order-invariance*, an important concept in finite model theory, cf. [36]. We say that an MSO sentence φ over vocabulary including $\prec^*_{\mathrm{ch}}$ and $\prec^*_{\mathrm{ns}}$ is $\prec_{\mathrm{ns}}$-*invariant* if for every unordered tree T and every two expansions $T^{\prec^1_{\mathrm{ns}}}$ and $T^{\prec^2_{\mathrm{ns}}}$ with sibling-orderings $\prec^1_{\mathrm{ns}}$ and $\prec^2_{\mathrm{ns}}$ we have $T^{\prec^1_{\mathrm{ns}}} \models \varphi \Leftrightarrow T^{\prec^2_{\mathrm{ns}}} \models \varphi$. A $\prec_{\mathrm{ns}}$-invariant sentence defines a Boolean query on unordered trees.

We now define $\mathrm{MSO}_{\mathrm{mod}}$ [19] as an extension of MSO with *modulo quantifiers*: for each set variable X, and $k > 1$, we have set new formulae $Q_k(X)$ which are true iff the cardinality of X is congruent to 0 modulo k.

Theorem 12. (see [20]) *Over unordered unranked trees, $\prec_{\mathrm{ns}}$-invariant Boolean queries are precisely the Boolean queries definable in* $\mathrm{MSO}_{\mathrm{mod}}$.

Further extensions in terms of arithmetic power have been considered [53, 54]. Recall that Presburger arithmetic refers to the FO theory of the structure $\langle \mathbb{N}, + \rangle$. Define *Presburger* MSO, or PMSO, as an extension of MSO over unordered trees with the following rule: if $\varphi(\bar{x}, y, \bar{X})$ is a PMSO formula and $\alpha(\bar{v})$ a Presburger arithmetic formula with $|\bar{X}| = |\bar{v}| = n$, then $[\varphi/\alpha](\bar{x}, y, \bar{X})$ is a PMSO formula. Given valuation $\bar{s}, s_0, \bar{S}$ for free variables, with $\bar{S} = (S_1, \ldots, S_n)$, let m_i be the cardinality of $\{s' \mid s_0 \prec_{\mathrm{ch}} s' \text{ and } s' \in S_i\}$. Then $[\varphi/\alpha](\bar{s}, s_0, \bar{S})$ is true iff $\alpha(m_1, \ldots, m_n)$ is true.

It is easy to see that $\mathrm{MSO} \subsetneq \mathrm{MSO}_{\mathrm{mod}} \subsetneq \mathrm{PMSO}$ over unordered trees. Still, PMSO is captured by a decidable automaton model.

Define Presburger unordered tree automata just as counting automata except that δ maps pairs from $Q \times \Sigma$ into Presburger formulae over v_q, for $q \in Q$. We interpret v_q as the number of children in state q, and a transition is enabled if the corresponding Presburger formula is true in this interpretation.

Theorem 13. (see [53]) *Presburger unordered tree automata and* PMSO *are equivalent. Furthermore, both emptiness and universality are decidable for Presburger unordered tree automata.*

Further extensions with counting have been considered for fixed-point logics [54] and the μ-calculus with modulo-quantifiers [3].

Edge-labeled unordered trees There are several areas where edge-labeled trees play a prominent and role, and traditionally logical formalisms have been designed for such data. For example, there are feature logics, used extensively in computational linguistics [15], or spatial logics used for describing networks and mobile agents [14]: in both cases one deals with unordered edge-labeled trees.

In the setting of feature trees, one has an infinite set of features $\mathcal{F}$, and in an unordered unranked tree every edge is labeled by an element $f \in \mathcal{F}$ such that each node s has at most one outgoing edge labeled f for each $f \in \mathcal{F}$. Furthermore, nodes may be labeled by elements of some alphabet Σ, as before. It is thus natural to model feature trees as structures $\langle D, (E_f)_{f \in \mathcal{F}}, (P_a)_{a \in \Sigma} \rangle$ such that the union of all E_f's forms the

child relation of a tree, and no node has two outgoing E_f-edges. In the context of computational linguistics, one commonly used [5] logic for feature trees is the propositional modal logic that, in the context of feature structures (not necessarily trees), is also often supplemented with path-equivalence [50], as well as regular expressions [34].

Ambient logics are modal logics for trees that have been proposed in the context of mobile computation [14] and later adapted for tree-represented data [12, 13]. One views trees as edge-labeled and defines them by the grammar

$$T, T' \;:=\; \Lambda \;\mid\; T|T' \;\mid\; a[T],\ a \in \Sigma,$$

with the equivalences that $|$ is commutative and associative, and that $T|\Lambda \equiv T$. Here Λ is the empty tree, $|$ is the parallel composition, and $a[T]$ adds an a-labeled edge on top of T. If we extend $\equiv$ to a congruence in the natural way, then every tree is equivalent to one of the form $a_1[T_1]|\ldots|a_m[T_m]$, which is viewed as a tree whose root has m outgoing edges labeled $a_1, \ldots, a_m$, with subtrees rooted at its children being $T_1, \ldots, T_m$.

There were several similar logics proposed in [11–14, 21]. Here we consider the logic from [11] whose formulae are given by

$$\varphi, \varphi' := \bot \mid \Lambda \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \varphi|\varphi' \mid \varphi \triangleright \varphi' \mid a[\varphi] \mid \varphi@a, \quad a \in \Sigma.$$

The semantics is as follows: $\bot$ is *false*; Λ is only true in a tree equivalent to Λ, $T \models \varphi_1|\varphi_2$ iff $T \equiv T_1|T_2$ with $T_i \models \varphi_i$, $i = 1, 2$; $T \models \varphi \triangleright \varphi'$ if for every $T' \models \varphi$ we have $T|T' \models \varphi'$; $T \models a[\varphi]$ iff $T \equiv a[T']$ with $T' \models \varphi$, and $T \models \varphi@a$ iff $a[T] \models \varphi$.

The study of ambient logics for trees took a different path compared to other logics seen in this survey; in particular, the focus was on type systems for tree languages and thus on proof systems for logics, rather than model-checking, its complexity, automata models, and comparison with other logics.

However, the ambient logic above does not take us outside of the MSO expressiveness: this can be seen by going from edge-labeled trees to node-labeled ones. The translation is simple: the label of each edge (x, y) becomes the label of y. The root will have a special label *Root* that cannot occur as a label of any other node. The only modification in the logic is that now we have formulae Λ_a for $a \in \Sigma$, which are true in a singleton-tree labeled a. The resulting logic is easily translated into MSO. For example, $\varphi|\varphi'$ states that the children of the root can be partitioned into two sets, X and X', such that the subtree that contains all the X-children satisfies φ and the subtree that contains all the X'-children satisfies φ'. For $\varphi \triangleright \varphi'$, one can consider $\neg(\varphi \triangleright \varphi')$ saying that there exists a tree T' such that $T' \models \varphi$ and $T|T' \models \neg\varphi'$, and use nondeterministic counting automata to guess this tree T'.

5 Automatic structures

In this section we look at a different kind of logics for unranked trees, using the standard approach of model theory. Let $\textsc{Tree}(\Sigma)$ be the set of all Σ-labeled unranked trees. We consider structures of the form $\mathfrak{M} = \langle \textsc{Tree}(\Sigma), \Omega \rangle$ where Ω is a set of relation, constant, and function symbols.

Let $\mathsf{Def}_n(\mathfrak{M})$ be the family of *n-dimensional definable sets* over $\mathfrak{M}$: that is, sets of the form $\{\bar{T} \in \textsc{Tree}(\Sigma)^n \mid \mathfrak{M} \models \varphi(\bar{T})\}$, where $\varphi(x_1, \ldots, x_n)$ is an FO formula in

the vocabulary Ω. We shall be looking at structures $\mathfrak{M}$ so that definable sets would be relations definable in MSO or other logics. In particular, such relations will be given by automata, and thus structures $\mathfrak{M}$ of this kind are called *automatic structures*.

Following known automatic structures for strings [4, 6], we introduce several predicates on trees: the extension predicate, node tests, and domain equality. For two trees T_1 and T_2 with domains D_1 and D_2, we say that T_2 is an *extension* of T_1, written $T_1 \preceq T_2$, if $D_1 \subseteq D_2$, and the labeling function of T_2 agrees with the labeling function of T_1 on D_1. It will actually be more convenient to work with two extension relations: extension on the right $\preceq_\rightarrow$ and extension down $\preceq_\downarrow$. For $T_1 \preceq_\rightarrow T_2$, we require that every $s \in D_2 - D_1$ be of the form $s' \cdot i$ when $s' \cdot j \in D_1$ for some $j < i$. For $T_1 \preceq_\downarrow T_2$, we require that every $s \in D_2 - D_1$ have a prefix s' which is a leaf of T_1. Define L_a to be true in a tree T if the rightmost node is labeled a. Finally, $T_1 \approx_{\mathrm{dom}} T_2$ iff $D_1 = D_2$.

Now we have the following structures:

$$\begin{aligned} \mathfrak{T}_{\text{univ}} &= \langle \textsc{Tree}(\Sigma),\ \preceq_\rightarrow,\ \preceq_\downarrow,\ (L_a)_{a\in\Sigma},\ \approx_{\mathrm{dom}}\rangle \\ \mathfrak{T} &= \langle \textsc{Tree}(\Sigma),\ \preceq_\rightarrow,\ \preceq_\downarrow,\ (L_a)_{a\in\Sigma}\rangle \end{aligned}$$

Theorem 14. (see [37]) *a) For every $n \geq 1$, $\mathsf{Def}_n(\mathfrak{T}_{\text{univ}})$ is precisely the class of regular n-ary relations over $\textsc{Tree}(\Sigma)$.*

b) $\mathsf{Def}_1(\mathfrak{T}) = \mathsf{Def}_1(\mathfrak{T}_{\text{univ}})$ is the class of regular unranked tree languages, but for every $n > 1$, $\mathsf{Def}_n(\mathfrak{T}) \subsetneq \mathsf{Def}_n(\mathfrak{T}_{\text{univ}})$.

Working with $\mathfrak{T}_{\text{univ}}$ makes it easy to write rather complicated properties of tree languages, and then Theorem 14 implies that those languages are regular. For example, if $X \subseteq \textsc{Tree}(\Sigma)$ is regular, then the set of trees T such that all their extensions can be extended on the right to a tree in X is regular. Indeed, this is easy to write in FO over $\mathfrak{T}_{\text{univ}}$, if we have a membership test for X, which is definable by Theorem 14. Also, conversions from formulae to automata are effective for both $\mathfrak{T}$ and $\mathfrak{T}_{\text{univ}}$, which implies decidability of their theories.

Other logics over unranked trees can be naturally represented over these structures: for example, Boolean FO queries are precisely sets of trees definable over $\mathfrak{T}$ if quantification is restricted to single branches [37].

A different view of unranked trees We conclude by presenting a different view of unranked trees and a different structure for them that makes it easy to talk about about their extensions in which new children may be inserted between existing ones. For example, if we have a tree T with domain $D = \{\varepsilon, 0, 1\}$, and we want to add more children of the root, they would have to be added on the right, e.g, we may have an extension with domain $\{\varepsilon, 0, 1, 2, 3\}$. But what if we want to add a child on the left of 0, and two children between 1 and 2? Intuitively, we need a new tree domain $\{\varepsilon, -1, 0, \frac{1}{3}, \frac{2}{3}, 1\}$ then. We now capture this situation and present a different automatic structure that makes it easy to derive that certain relations on trees are regular.

A *rational unranked tree domain* is a finite prefix-closed subset of $\mathbb{Q}^*$. Relation $\prec^*_{\mathrm{ch}}$ is defined for rational domains just as before, and relation $\prec^*_{\mathrm{ns}}$ is now given by $s \cdot r \prec^*_{\mathrm{ns}} s \cdot r'$ iff $r \leq r'$. Then an unranked tree T over a rational unranked tree domain is, as before, a structure $T = \langle D, \prec^*_{\mathrm{ch}}, \prec^*_{\mathrm{ns}}, (P_a)_{a\in\Sigma}\rangle$.

Let $\text{TREE}_{\mathbb{Q}}(\Sigma)$ be the set of all unranked trees with rational unranked tree domains. Note that different elements of $\text{TREE}_{\mathbb{Q}}(\Sigma)$ may be isomorphic as trees; we denote this isomorphism relation by $\cong$.

Define the extension relation $\preceq$ over trees in $\text{TREE}_{\mathbb{Q}}(\Sigma)$ as before. A *branch* is a tree $T \in \text{TREE}_{\mathbb{Q}}(\Sigma)$ such that the set $\{T' \mid T' \preceq T\}$ is linearly ordered by $\preceq$. It follows from the definition of rational unranked tree domains that the domain of a branch consists of all the prefixes of some string $s \in \mathbb{Q}^*$. Let $L_a(T)$ be true iff T is a branch whose leaf is labeled a, and let $T_1 <_{\text{lex}} T_2$ be true iff T_1 and T_2 are branches with leaves s_1 and s_2, and $s_1 <_{\text{lex}} s_2$. We then define the structure

$$\mathfrak{T}^{\mathbb{Q}}_{\text{univ}} \;=\; \langle \text{TREE}_{\mathbb{Q}}(\Sigma),\, \preceq,\, <_{\text{lex}},\, \approx_{\text{dom}},\, (L_a)_{a\in\Sigma}\rangle.$$

Proposition 4. *The structure $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$ is interpretable in $\mathfrak{T}_{\text{univ}}$. Furthermore, there is a definable subset of the image of $\text{TREE}_{\mathbb{Q}}(\Sigma)$ that contains exactly one representative of each $\cong$-equivalence class.*

That is, under the mapping $\iota : \text{TREE}_{\mathbb{Q}}(\Sigma)/\cong \to \text{TREE}(\Sigma)$, definable sets over $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$ become precisely the regular tree languages. Hence, expressing properties of unranked trees in first-order logic over $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$ allows us to conclude easily that certain tree languages are regular, and thus MSO-definable.

6 Other directions and conclusions

We present very briefly some directions for future work (for more detailed discussion, see the full version).

Among problems that need to be addressed are the following: (a) How does one compare different logics over unranked trees? One way is in terms of their succinctness [29]. (b) Connection between ambient logics and other logics presented there is not yet adequately understood. (c) We do not know much about logics over string representations of trees (which occur naturally, for example, in streaming XML applications [52]). (d) Nor do we know much about handling data values which are present in XML trees. Some early results were reported in [45, 8], complemented recently by a nice decidability result that works on strings with data values [7].

Acknowledgments. I am grateful to Cristiana Chitic, Christoph Koch, Maarten Marx, Frank Neven, Joachim Niehren, Gerald Penn, Thomas Schwentick, and Luc Segoufin for their comments.

References

1. S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*, Addison Wesley, 1995.
2. A. Arnold, D. Niwinski. *Rudiments of μ-calculus*. Elsevier, 2001.
3. P. Barceló, L. Libkin. Temporal logics over unranked trees. In *LICS'05*.
4. M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. Definable relations and first-order query languages over strings. *J. ACM*, 50 (2003), 694–751.
5. P. Blackburn. Structures, languages and translations: the structural approach to feature logic. In *Constraints, Language and Computation*, AP, 1994, pages 1–27.
6. A. Blumensath and E. Grädel. Automatic structures. In *LICS'00*, pages 51–62.
7. M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, L. Segoufin. Two-variable logic on words with data. 2005.

8. P. Bouyer, A. Petit, D. Thérien. An algebraic characterization of data and timed languages. In *CONCUR 2001*, pages 248–261.
9. A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets: Version 1, 2001. HKUST Tech. Report.
10. J.R. Büchi. Weak second-order arithmetic and finite automata. *Zeit. Math. Logik Grundl. Math.* 6 (1960), 66–92.
11. C. Calcagno, L. Cardelli, A. Gordon. Deciding validity in a spatial logic for trees. *J. Funct. Progr.*, to appear.
12. L. Cardelli. Describing semistructured data. *SIGMOD Record* 30 (2001), 80–85.
13. L. Cardelli, G. Ghelli. A query language based on the ambient logic. In *ESOP 2001*, pages 1–22.
14. L. Cardelli, A. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *POPL 2000*, pages 365–377.
15. B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge, 1992.
16. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.
17. J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation, Nov. 1999. www.w3.org/TR/xpath.
18. H. Comon et al. *Tree Automata: Techniques and Applications*. Available at www.grappa.univ-lille3.fr/tata. October 2002.
19. B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inf.&Comput.* 85 (1990), 12–75.
20. B. Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *TCS* 80 (1991), 153–202.
21. S. Dal-Zilio, D. Lugiez, C. Meyssonnier. A logic you can count on. In *POPL 2004*, pages 135–146.
22. E. A. Emerson, C. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS 1991*, pages 368–377.
23. M. Frick, M. Grohe. The complexity of first-order and monadic second-order logic revisited. In *LICS 2002*, 215–224.
24. M. Frick, M. Grohe, C. Koch. Query evaluation on compressed trees. In *LICS 2003*, pages 188-197.
25. G. Gottlob, C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM* 51 (2004), 74–113.
26. G. Gottlob, C. Koch, R. Pichler, and L. Segoufin. The complexity of XPath query evaluation and XML typing. *J. ACM*, 2005, to appear.
27. G. Gottlob, C. Koch, K. Schulz. Conjunctive queries over trees. In *PODS 2004*, pages 189–200.
28. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *J. ACM*, 48 (2001), 431–498.
29. M. Grohe, N. Schweikardt. Comparing the succinctness of monadic query languages over finite trees. In *CSL 2003*, pages 226–240.
30. M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *STOC 2001*, pages 657–666.
31. T. Hafer, W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. *ICALP 1987*, pages 269–279.
32. D. Janin, G. Lenzi. Relating levels of the mu-calculus hierarchy and levels of the monadic hierarchy. In *LICS 2001*, pages 347–356.
33. H.W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD Thesis, UCLA, 1968.
34. B. Keller. *Feature Logics, Infinitary Descriptions and Grammar*. CSLI Press, 1993.
35. O. Kupferman, A. Pnueli. Once and for all. In *LICS'95*, pages 25–35.
36. L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.
37. L. Libkin, F. Neven. Logical definability and query languages over unranked trees. In *LICS 2003*, pages 178–187.
38. M. Marx. Conditional XPath, the first order complete XPath dialect. In *PODS 2004*, pages 13–22.
39. M. Marx and M. de Rijke. Semantic characterizations of XPath. In *TDM Workshop on XML Databases and Information Retrieval*, 2004.
40. F. Moller, A. Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Information and Computation*, 184 (2003), 147-159.
41. F. Neven. *Design and Analysis of Query Languages for Structured Documents*. PhD Thesis, U. Limburg, 1999.
42. F. Neven. Automata, logic, and XML. In *CSL 2002*, pages 2–26.
43. F. Neven, Th. Schwentick. Expressive and efficient pattern languages for tree-structured data. In *PODS 2000*, pages 145–156. Corrigendum at http://www.mathematik.uni-marburg.de/~tick/
44. F. Neven, Th. Schwentick. Query automata over finite trees. *Theor. Comput. Sci.* 275 (2002), 633–674.
45. F. Neven, Th. Schwentick, V. Vianu. Towards regular languages over infinite alphabets. In *MFCS 2001*, pages 560–572.
46. J. Niehren, A. Podelski. Feature automata and recognizable sets of feature trees. *TAPSOFT 1993*, pages 356–375.
47. D. Niwinski. Fixed points vs. infinite generation. In *LICS 1988*, pages 402–409.
48. A. Potthoff, W. Thomas. Regular tree languages without unary symbols are star-free. In *FCT 1993*, pages 396–405.
49. M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. AMS* 141 (1969), 1–35.
50. W. C. Rounds, R. Kasper. A logical semantics for feature structures. In *24th Annual Meeting of the Assoc. for Computational Linguistics*, 1986, pages 257–266.
51. B.-H. Schlingloff. Expressive completeness of temporal logic of trees. *Journal of Applied Non-Classical Logics* 2 (1992), 157–180.
52. L. Segoufin, V. Vianu. Validating streaming XML documents. In *PODS 2002*, pages 53–64.
53. H. Seidl, Th. Schwentick, A. Muscholl. Numerical document queries. In *PODS 2003*, 155–166.
54. H. Seidl, Th. Schwentick, A. Muscholl, P. Habermehl. Counting in trees for free. In *ICALP 2004*, pages 1136–1149.
55. L. Stockmeyer and A. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM*, 49 (2002), 753–784.
56. J.W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *JCSS* 1 (1967), 317–322.
57. J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.
58. W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Vol. 3*, Springer-Verlag, 1997.
59. M. Y. Vardi. Reasoning about the past with two-way automata. In *ICALP 1998*, pages 628–641.
60. I. Walukiewicz. Monadic second-order logic on tree-like structures. *TCS* 275 (2002), 311–346.