considered for publication in Logical Methods in Computer Science

LOGICS FOR UNRANKED TREES: AN OVERVIEW

LEONID LIBKIN

Department of Computer Science, University of Toronto, and School of Informatics, University of Edinburgh

E-mail address: libkin@cs.toronto.edu and libkin@inf.ed.ac.uk

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, monadic second-order logic, and various temporal and 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 [CG+02, Tho97]: 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. For example, the left tree in Figure 1 is a binary tree in which every non-leaf node has two children. In the second tree in Figure 1, however, different nodes have a different number of children. Although unranked trees have been considered in the 60s and 70s [PQ68, Tak75, Tha67], and are related to feature trees over an infinite set of features [Smo92] which are a particular kind of feature structures that have been investigated by computational linguists [Bla94, Car92, RK86], their systematic study was initiated by the development of XML (eXtensible Markup Language). XML

An earlier version of this paper appeared in the Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP 2005).

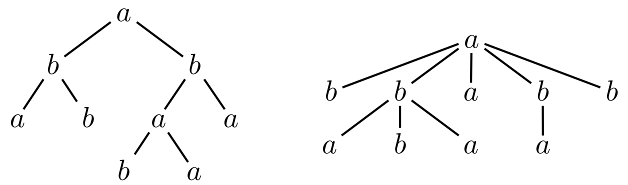


Figure 1: A ranked (binary) and an unranked tree

is a data format which has become the lingua franca for information exchange on the World Wide Web. In particular, XML data is typically modeled as labeled unranked trees [Nev02, Via01].

This connection has led to a renewed interest in logical and procedural formalisms for unranked trees. Since XML trees are used to exchange *data*, the usual database query language paradigms apply: one uses logical formalisms for expressing declarative queries, and procedural formalisms for evaluating those declarative queries. Logics over unranked trees defining a variety of query languages for them appeared in large numbers over the past 7–8 years, and they come in many flavors and shapes. What is common to them, however, 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.

The yardstick logic: Most formalisms are "inspired" by one of the two logics often used in the context of trees: *first-order logic* (FO), and *monadic second-order logic* (MSO) that extends FO by quantification over sets of nodes. Query languages and schema formalisms for XML tend to use MSO as the yardstick: for example, XML Document Type Definition (DTDs, or, more precisely, XSD – XML Schema Definition) are essentially equivalent to MSO sentences, and various languages for extraction of data from XML documents, although being syntactically very different, 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 closely related to FO and its fragments.

Arity of queries: Most commonly one considers Boolean or unary queries. Boolean queries are logical sentences and thus evaluate to *true* or *false*. For example, 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. For example, extracting sets of nodes, or evaluating XPath expressions relative to the root naturally give rise to unary queries.

Complexity of model-checking/query-evaluation: The model-checking problem asks whether a tree T satisfies a logical sentence φ, written $T \models \varphi$. If φ is an MSO sentence φ, it can be evaluated in linear time in the size of T, by converting φ 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-off is one of the central issues in dealing with logics over trees. Similar issues arise with evaluating formulae $\varphi(\bar{x})$ in trees, that is, finding tuples $\bar{s}$ of nodes such that $T \models \varphi(\bar{s})$.

Ordered vs. unordered trees: In the standard definition of unranked trees in the XML context, 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 unranked tree 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 [Cou90].

The paper is organized as follows. After we give basic definitions in Section 2, we move to logics for ordered trees. In Section 3 we deal with MSO-related logics, including syntactic restrictions of MSO, a datalog-based logic, and the μ-calculus. In Section 4 we turn to FO-related logics, present analogs of LTL and CTL* that have been studied for expressing navigational properties, and also look at conjunctive queries over trees. In Section 5 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. We also review ambient and feature logics over edge-labeled trees. In Section 6 we look at the model-theoretic approach. We consider an infinite first-order structure whose universe is the set of all unranked trees and obtain some well-known classes of trees by studying first-order definability (in the classic model-theoretic sense) over that structure.

2. Trees, logics, and automata

2.1. **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 $s = n_0 n_1 \ldots$ 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 of that element, etc. We shall write $s_1 \cdot s_2$ for the concatenation of strings s_1 and s_2, and ε for the empty string.

We now define some basic binary relations on $\mathbb{N}^*$. The *child relation* is

$$s \prec_{\text{ch}} s' \quad \Leftrightarrow \quad s' = s \cdot i \ \text{ for some } i \in \mathbb{N}.$$

The *next-sibling* relation is given by:

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

That is, s and s' are both children of the same $s_0 \in \mathbb{N}^*$, and s' is next after s in the natural ordering of siblings. We also use the *first child relation*: $s \prec_{\text{fc}} s \cdot 0$. These are shown in Figure 2.

We shall use * to denote the reflexive-transitive closure of a relation. Thus, $\prec_{\text{ch}}^*$ is the *descendant* relation (including self): $s \prec_{\text{ch}}^* s'$ iff s is a prefix of s' or $s = s'$. The transitive closure of the next-sibling relation, $\prec_{\text{ns}}^*$ is a linear ordering on siblings: $s \cdot i \prec_{\text{ns}}^* s \cdot j$ iff $i \leq j$.

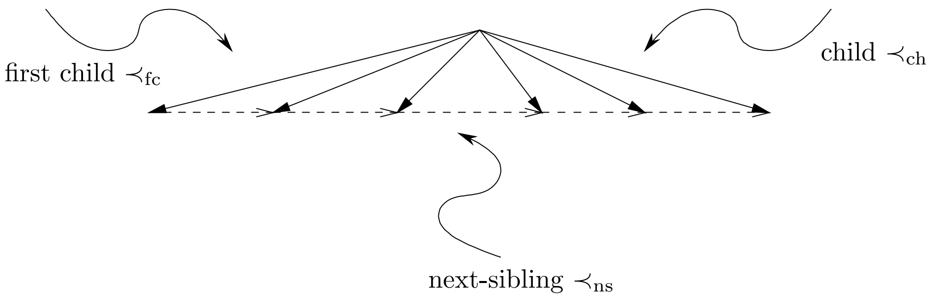


Figure 2: Basic relations for unranked trees

We shall be referring to younger/older siblings with respect to this ordering (the one of the form $s \cdot 0$ is the oldest).

A set $D \subseteq \mathbb{N}^*$ is called *prefix-closed* if whenever $s \in D$ and s' is a prefix of s (that is, $s' \prec^*_{\rm ch} s$), then $s' \in D$.

Definition 2.1 (Tree domain)**.** A *tree domain* D is a finite prefix-closed subset of $\mathbb{N}^*$ such that $s \cdot i \in D$ implies $s \cdot j \in D$ for all $j < i$.

Let Σ be a finite alphabet. We define trees as *structures* that consist of a universe and a number of predicates on the universe.

Definition 2.2 (Σ-trees)**.** An *ordered unranked tree* T is a structure

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

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

An *unordered* unranked tree is defined as a structure $\langle D, \prec^*_{\rm ch}, (P_a)_{a\in\Sigma}\rangle$, where $D, \prec^*_{\rm ch}$, and P_a's are as above.

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

Notice that when dealing with unranked trees we assume that each node has one label. Later we shall see a connection with temporal logics, where such a restriction on labeling is normally not imposed. However, one could always assume unique labeling in that case too, simply by collecting the set of all labels of a node (in this case the labeling alphabet becomes 2^Σ).

2.2. **First-order and monadic second-order logic.** We shall only consider relational vocabularies, that is, finite lists $(R_1, \ldots, R_m)$ of relation symbols, each R_i with an associated arity n_i. Over trees, relation symbols will be either binary (e.g., $\prec_{\rm ch}, \prec_{\rm ns}, \prec^*_{\rm ch}$) or unary (the 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 whose length equals the arity of R, 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 are evaluated on a structure, which consists of a universe and interpretations for relations. Quantifiers $\exists$ and $\forall$ range over the universe of the structure. For example, an FO formula

$$\varphi(x) \quad = \quad P_a(x) \wedge \exists y \exists z \big(x \prec^*_{\text{ch}} y \wedge y \prec^*_{\text{ns}} z \wedge P_b(y) \wedge P_c(z)\big)$$

is true for nodes s in a tree T that are labeled a, have a descendant labeled b, which in turn has a younger sibling labeled c.

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. As an example, an MSO formula $\varphi_{\text{odd}}(x, y)$ given by the conjunction of $x \prec^*_{\text{ch}} y$ and

$$\exists X \exists Y \left(\begin{array}{ll} & \forall z \Big(\big(x \prec^*_{\text{ch}} z \prec^*_{\text{ch}} y\big) \;\rightarrow\; \big(X(z) \leftrightarrow \neg Y(z)\big)\Big) \\ \wedge & \big(X(x) \wedge Y(y)\big) \\ \wedge & \forall z \forall v \; \Big(x \prec^*_{\text{ch}} z \prec_{\text{ch}} v \prec^*_{\text{ch}} y \rightarrow \big((X(z) \rightarrow Y(v)) \wedge (Y(z) \rightarrow X(v))\big)\Big) \end{array} \right)$$

says that y is a descendant of x and the path between them is of odd length. It says that there exist two sets, X and Y, that partition the path from x to y, such that $x \in X$, $y \in Y$, and the successor of each element in X is in Y, and the successor of each element in Y is in X. In the formula above, $x \prec^*_{\text{ch}} z \prec^*_{\text{ch}} y$ is of course an abbreviation for $(x \prec^*_{\text{ch}} z) \wedge (z \prec^*_{\text{ch}} y)$ and likewise for $x \prec^*_{\text{ch}} z \prec_{\text{ch}} v \prec^*_{\text{ch}} y$.

Note that the relations $\prec_{\text{ch}}$ and $\prec_{\text{ns}}$ are definable, even in FO, from $\prec^*_{\text{ch}}$ and $\prec^*_{\text{ns}}$: for example,

$$\neg(x = y) \wedge (x \prec^*_{\text{ch}} y) \wedge \forall z \big((x \prec^*_{\text{ch}} z) \wedge (z \prec^*_{\text{ch}} y) \rightarrow (x = z \vee y = z)\big)$$

defines the child relation from $\prec^*_{\text{ch}}$. In MSO one can define $\prec^*_{\text{ch}}$ from $\prec_{\text{ch}}$ by stating the existence of a path between two nodes (and likewise $\prec^*_{\text{ns}}$ from $\prec_{\text{ns}}$). However, it is well-known that in FO one *cannot* define $\prec^*_{\text{ch}}$ from $\prec_{\text{ch}}$ (cf. [Lib04]) and this is why we chose $\prec^*_{\text{ch}}$ and $\prec^*_{\text{ns}}$, rather than $\prec_{\text{ch}}$ and $\prec_{\text{ns}}$, as our basic relations. However, in all the results about MSO, we may assume that the basic relations are $\prec_{\text{ch}}$ and $\prec_{\text{ns}}$.

In the introduction, we mentioned that we are mostly interested (in this survey) in Boolean and unary queries. A Boolean query over trees is just a set of trees closed under isomorphism (that is, a query cannot distinguish between two isomorphic trees). A unary query $\mathcal{Q}$ is a mapping that associates with each tree T a subset $\mathcal{Q}(T)$ of its domain. Again, a query is required to be closed under isomorphism.

Definition 2.3 (Definability in logic)**.** Given a logic $\mathcal{L}$, we say that a Boolean query (that is, a set $\mathcal{T}$ of trees) 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 $\mathcal{Q}$ 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.

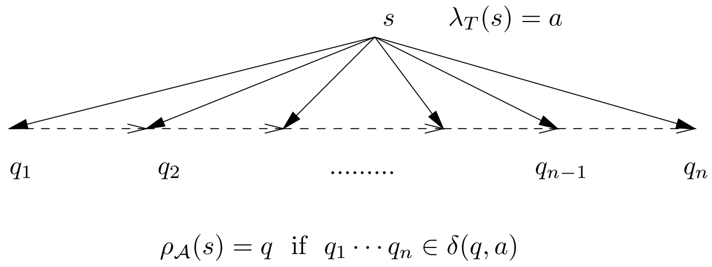


Figure 3: Run of an unranked tree automaton

2.3. **Unranked tree automata.** A *nondeterministic unranked tree automaton, NUTA* [Tha67, BMW01], 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 $\delta(\rho_{\mathcal{A}}(s), a)$.

This is illustrated in Figure 3. 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*.

There could be different representations of NUTAs, depending on how regular expressions over Q are represented. These issues are discussed in [Nev02, MN05].

2.4. **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). It is common to define (binary) tree automata with both initial and final states, using the initial states to avoid conditions $\varepsilon \in \delta(q, a)$ imposed in the runs of NUTAs. That is, 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 $Q \times Q \times \Sigma \to 2^Q$. In this case 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 $\rho_{\mathcal{A}_b}(s) \in \delta(\rho_{\mathcal{A}_b}(s \cdot 0), \rho_{\mathcal{A}_b}(s \cdot 1), a)$. As before, 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 of $\mathcal{A}_b$. Sets of trees of this form are regular sets (of binary trees).

There is a well-known regularity-preserving translation between unranked and ranked trees. It was first used in [Rab69] 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 $s = s_0 \cdot i$,

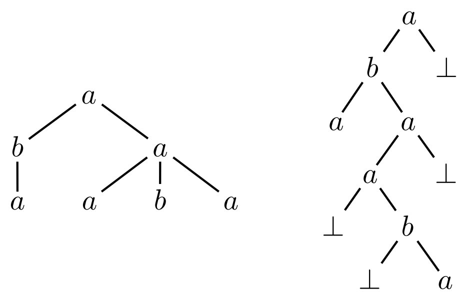


Figure 4: A unranked tree T and its translation $\mathcal{R}(T)$

then $\mathcal{R}(s \cdot 0) = s' \cdot 0$ and $\mathcal{R}(s_0 \cdot (i+1)) = s' \cdot 1$. Or, equivalently, $\mathcal{R}(\varepsilon) = \varepsilon$, and if $\mathcal{R}(s) = s'$, then $\mathcal{R}(s \cdot i) = s' \cdot 0 \cdot 1^i$.

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 $\bot \notin \Sigma$. An example is shown in Figure 4.

The following is a folklore result.

Lemma 2.4. 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 conversely, for every NTA $\mathcal{A}_b$ there is an NUTA $\mathcal{A}$ such that the above holds.

Moreover, $\mathcal{A}_b$ can be constructed from $\mathcal{A}$ very fast, in DLOGSPACE [GK+05].

Other regularity-preserving translations from unranked trees to binary trees exist. For example, [CNT04] views unranked trees as built from labeled nodes by means of a binary operation $T@T'$ that attaches T' at the new youngest child of the root of T. This immediately yields a binary tree representation and an automaton construction, and of course an analog of Lemma 2.4 holds.

3. Ordered trees: MSO and its relatives

In the next two sections we only deal with ordered unranked trees.

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 [Nev99].

Theorem 3.1. *A set of unranked trees is regular iff it is definable in* MSO*.*

When restricted to strings and binary trees, this corresponds to well-known results by Büchi [Büc60] saying that MSO equals regular languages over strings, and by Thatcher, Wright [TW68], and Doner [Don70], saying that MSO equals regular (binary) tree languages.

There is also a close connection between automata, MSO, and the common formalism for describing schemas for XML documents 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. We write $\textsc{Sat}(d)$ for the set of trees that satisfy d.

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. In fact, DTDs (that is, sets of the form $\textsc{Sat}(d)$) are closed under neither unions nor complement, which makes DTDs unsuitable for capturing a logic with disjunction and negation.

However, a slight extension of DTDs does capture MSO. 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'. We say that a Σ-labeled tree T satisfies (Σ', d', g) if there is a Σ'-labeled tree T' that satisfies d' such that $T = g(T')$ (more formally, T is obtained by replacing each label a in T' by $g(a)$). We write $\textsc{Sat}(\Sigma', d', g)$ for the set of trees that satisfy (Σ', d', g).

The following was established in [Tha67] and then restated using the DTD terminology in [PV00, Via01].

Proposition 3.2. A set of unranked trees is MSO definable iff it is of the form $\textsc{Sat}(\Sigma', d', g)$ for some extended DTD (Σ', d', g).

Theorem 3.1 talks about MSO sentences, but it can be extended to unary MSO queries using the concept of *query automata* [NS02]. 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 two unary queries $\mathcal{Q}^{\exists}_{\mathcal{QA}}$ and $\mathcal{Q}^{\forall}_{\mathcal{QA}}$ on unranked trees:

Existential semantics query: : $s \in \mathcal{Q}^{\exists}_{\mathcal{QA}}(T)$ iff $(\rho_{\mathcal{A}}(s), \lambda_T(s)) \in S$ for some accepting run $\rho_{\mathcal{A}}$.

Universal semantics query: : $s \in \mathcal{Q}^{\forall}_{\mathcal{QA}}(T)$ iff $(\rho_{\mathcal{A}}(s), \lambda_T(s)) \in S$ for every accepting run $\rho_{\mathcal{A}}$.

Theorem 3.3. (see [NS02, Nev99, FGK03]) *For a unary query $\mathcal{Q}$ on unranked trees, the following are equivalent:*

(1) *$\mathcal{Q}$ is definable in* MSO*;*
(2) *$\mathcal{Q}$ is of the form $\mathcal{Q}^{\exists}_{\mathcal{QA}}$ for some query automaton $\mathcal{QA}$;*
(3) *$\mathcal{Q}$ is of the form $\mathcal{Q}^{\forall}_{\mathcal{QA}}$ for some query automaton $\mathcal{QA}$.*

Query automata, just as usual tree automata, have a deterministic counterpart; however, in the deterministic version, two passes over the tree are required. See [NS02] for details.

Theorems 3.1 and 3.3 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 $\|\varphi\|$ [SM02]. An even stronger result of [FG02] 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 [FG02] 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 [Kam68] and admits a model-checking algorithm with running time $2^{O(\|\varphi\|)} \cdot |s|$.

3.1. **Logic ETL.** The first logic for unranked trees that has the power of MSO and model-checking complexity matching that of LTL appeared in [NS00] 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 $\prec_{\rm ch}$ and $\prec^*_{\rm ch}$ and are *not* allowed to use the next-sibling relation $\prec^*_{\rm ns}$. The formulae of ETL are then closed under Boolean combinations, *guarded quantification*, and *path formulae*. The rules for guarded quantification are as follows:

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

Path formulae are defined below, and illustrated in Figure 5.

- if e is a regular expression over ETL formulae of the form $\psi(u, v)$, 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 $\psi_i(u, v)$, $i \le n-1$, such that $T \models \psi_i(s_i, s_{i+1})$ for each $i \le n-1$, and the sequence $\psi_0 \ldots \psi_{n-1}$ matches e.
- if e is a regular expression over ETL formulae of the form $\psi(u, \bar{X})$, then $e^\rightarrow(x, \bar{X})$ is a (horizontal path) ETL formula. Then $T \models e^\rightarrow(s, \bar{X})$ if children $s \cdot i, i \le k$ 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.

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

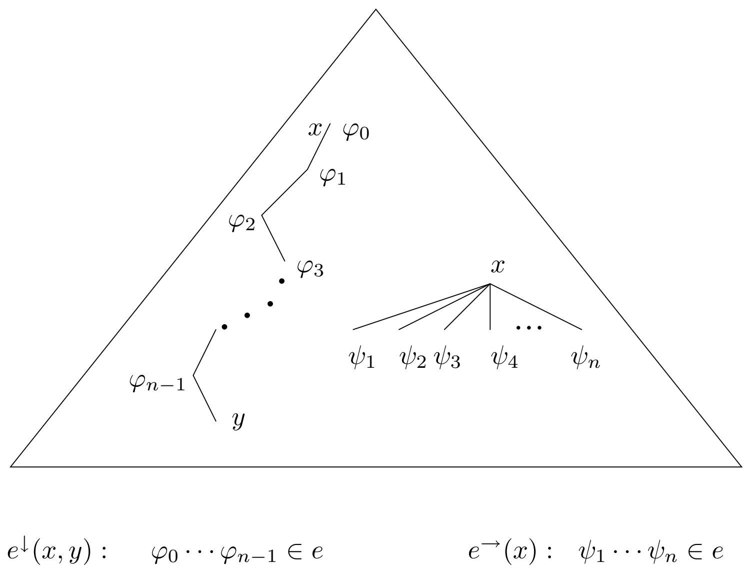


Figure 5: The semantics of path formulae of ETL

We also define a slight syntactic modification ETL$^\circ$ of ETL, in which the closure under Boolean connectives is replaced by a rule that formulae are closed under taking Boolean combinations which are in DNF: that is, if φ_{ij}'s are ETL$^\circ$ formulae, then $\bigvee_i \bigwedge_j \varphi'_{ij}$ is an ETL$^\circ$ formula, where each φ'_{ij} is either φ_{ij} or $\neg\varphi_{ij}$. Clearly the expressiveness of ETL$^\circ$ is exactly the same as the expressiveness of ETL.

Theorem 3.4. (see [NS00]) *With respect to Boolean and unary queries,* ETL *and* MSO *are equally expressive. Furthermore, each* ETL$^\circ$ *formula* φ *can be evaluated on a tree* T *in time* $2^{O(\|\varphi\|)} \cdot \|T\|$.

ETL formulae can thus be evaluated in linear time in the size of the tree, and double exponential time in $\|\varphi\|$, by converting Boolean combinations into DNF. It is not known if ETL itself admits a $2^{O(\|\varphi\|)} \cdot \|T\|$ model-checking algorithm.

3.2. **Monadic datalog.** Another approach to obtaining the full power of MSO while keeping the complexity low is based on the database query language *datalog* (cf. [AHV95]); it was proposed in [GK04, GK02]. A datalog program can be viewed as a prolog program without function symbols. Datalog is often used to extend expressiveness of database queries beyond FO.

A datalog program consists of a sequence of rules

$$H \;:-\; P_1, \ldots, P_k,$$

where H and all P_i's are atoms: that is, atomic formulae of the form $E(\bar{x})$. The predicate H is called the head of the rule, and $P_1, \ldots, P_k$ are called its body. Every variable that appears in the head is required to appear in the body. 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 (that is, of the form $H(x)$), then $\mathcal{P}$ is a *monadic* datalog program.

Given a datalog program $\mathcal{P}$ with extensional predicates $P_1, \ldots, P_m$ and intensional predicates $H_1, \ldots, H_\ell$, and a structure $\mathcal{D} = \langle D, P_1^{\mathcal{D}}, \ldots, P_m^{\mathcal{D}} \rangle$ that interprets each p-ary predicate P_i as $P_i^{\mathcal{D}} \subseteq D^p$, we define $\mathcal{P}(\mathcal{D})$ as the least fixed point of the *immediate consequence* operator. This operator takes a structure $\mathcal{H}' = \langle D, H_1', \ldots, H_\ell' \rangle$ and produces another structure $\mathcal{H}'' = \langle D, H_1'', \ldots, H_\ell'' \rangle$ such that a tuple $\bar{a}$ is in H_i'' if it is in H_i' or there is a rule $H_i(\bar{x})\text{:-}R_1(\bar{x}, \bar{y}), \ldots, R_s(\bar{x}, \bar{y})$ and a tuple $\bar{b}$ such that for each extensional predicate R_i, the fact $R_i(\bar{a}, \bar{b})$ is true in $\mathcal{D}$, and for each intensional predicate R_i, the fact $R_i(\bar{a}, \bar{b})$ is true in $\mathcal{H}'$.

A monadic datalog query is a pair $(\mathcal{P}, H)$ where $\mathcal{P}$ is a monadic datalog program, and H is an intensional predicate. The value of H in $\mathcal{P}(\mathcal{D})$ is the output of this program on $\mathcal{D}$.

We consider three unary predicates on unranked tree domains: *Leaf*, *LastChild*, and *Root*. Given a tree domain D, they are interpreted as

$$\begin{array}{rcl} \mathit{Leaf} & = & \{s \in D \mid \neg\exists s' \in D : s \prec_{\mathrm{ch}} s'\}, \\ \mathit{LastChild} & = & \{s \cdot i \in D \mid s \cdot (i+1) \notin D\}, \\ \mathit{Root} & = & \{\varepsilon\}. \end{array}$$

Theorem 3.5. (see [GK04]) *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* $O(\|\mathcal{P}\| \cdot \|T\|)$.

There are two proofs of this result in [GK04]: one codes query automata in monadic datalog, and the other one uses the standard reduction to ranked trees and the composition method (cf. [HT87]) for MSO games.

3.3. **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 [EJ91, Niw88] that every property of infinite binary trees definable in MSO is also 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. [AN01]. Formulae of L_μ 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 structure T with domain D, $s \in D$, and a valuation v for free variables (such that each $v(X)$ is a subset of D), we define the semantics by

- $(T, v, s) \models a$ iff s is labeled a.

- $(T, v, s) \models \varphi \vee \varphi'$ iff $(T, v, s) \models \varphi$ or $(T, v, s) \models \varphi'$.
- $(T, v, s) \models \neg\varphi$ iff $(T, v, s) \models \varphi$ is false.
- $(T, v, s) \models X$ 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; in other words, if
$$s \in \bigcap \{P \mid \{s' \mid (T, v[P/X], s') \models \varphi\} \subseteq P\},$$
where $v[P/X]$ extends the valuation v by $v(X) = P$.

We shall list explicitly binary relations E_i, writing $L_\mu[E_1, \ldots, E_m]$ to refer L_μ formulae that only use those relations. 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 (or direct coding of automata), it is easy to show the following (see [BL05]):

Proposition 3.6. The class of Boolean MSO queries on unranked trees is precisely the class of Boolean queries defined by $L_\mu[\prec_{\rm fc}, \prec_{\rm ns}]$.

If we consider unranked trees as structures with relations $\prec_{\rm fc}$ and $\prec_{\rm ns}$, then they are acyclic, and hence the complexity of model checking is $O(\|\varphi\|^2 \cdot \|T\|)$ [Mat02]. Furthermore, results of [Mat02] tell us that one can strengthen Proposition 3.6: MSO equals alternation-free L_μ over $\prec_{\rm fc}, \prec_{\rm ns}$. For alternation-free L_μ formulae over unranked trees the complexity of model-checking further reduces to $O(\|\varphi\| \cdot \|T\|)$, matching the complexity of monadic datalog.

It is also possible to characterize unary MSO queries over unranked trees in terms of the *full* μ-calculus $L_\mu^{\rm full}$ (cf. [Var98]) 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 3.7. (see [BL05]) The class of unary MSO queries on unranked trees is precisely the class of queries defined by $L_\mu^{\rm full}[\prec_{\rm ch}, \prec_{\rm ns}]$.

There are other fixed-point constructions that have been shown to capture the power of automata and MSO over unranked trees; see, e.g. [NS98].

4. Ordered trees: FO and its relatives

We continue dealing with ordered trees, but now we move to logics closely related to first-order, as opposed to monadic second-order.

While a lot is known about FO on both finite and infinite strings, it has not been nearly as extensively studied for trees until very recently. Recall that over strings – which we can view as trees with only unary branching – FO defines precisely the star-free languages (cf. [Tho97]), and over both finite and infinite strings FO has exactly the power of LTL [Kam68]. It can further be characterized by aperiodicity of the syntactic monoid (cf. [Str94]).

In contrast, the natural analog of star-free expressions over binary trees captures not FO but MSO [PT93]. Algebraic characterizations of FO-definable classes of binary trees have been obtained very recently [BS05, BW04, EW03], with [BS05] showing that FO-definability (without the descendant relation) is decidable for regular tree languages. One well-known equivalent logical description of FO on binary trees is Hafer-Thomas's theorem [HT87] stating that over finite binary trees, FO = CTL* (CTL* is a branching time temporal logic widely used in verification, cf. [CGP99], and it will be defined shortly). Actually, the result of [HT87] shows that CTL* 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 [CD99] is a W3C standard for describing paths in XML documents. For example, an XPath expression

$$//a[//b]/c$$

produces the c-labeled children of a-labeled nodes having a b-labeled descendant. Here // denotes descendant, / denotes child, and [ ] is a node test. The expression above looks for a-nodes (descendants of the root) in which the test $[//b]$ is true (the existence of a node labeled b) and from there it proceeds to children of such nodes labeled c. While this is the syntax one typically finds in the literature on XPath, here we shall use a different syntax, highlighting connections with temporal logics.

In this section we shall 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 FO = LTL over strings) for trees. Secondly, one can try to extend Hafer-Thomas's theorem (the equivalence FO = CTL*) from binary to unranked trees.

Both ways are possible, and in both cases we get FO completeness results, stating that some temporal logics have precisely the power of unary FO queries.

4.1. **XPath and temporal logics.** We start with LTL-like 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: a formula is evaluated in a position in a string. Given a string $s = a_0a_1\ldots$, we define the semantics 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 $i \leq k < j$;

- 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 that 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).

We now consider a logic $\mathrm{TL}^{\mathrm{tree}}$ (*tree temporal logic*, cf. [Mar05, Sch92]) defined as follows:

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

where $*$ is either 'ch' (child) or 'ns' (next sibling). We define the semantics with respect to a tree T and a node s in T:

- $(T, s) \models a$ iff $\lambda_T(s) = a$;
- $(T, s) \models \mathbf{X}_{\mathrm{ch}} \varphi$ if $(T, s \cdot i) \models \varphi$ for some i;
- $(T, s) \models \mathbf{X}_{\mathrm{ch}}^- \varphi$ if $(T, s') \models \varphi$ for the node s' such that $s' \prec_{\mathrm{ch}} s$;
- $(T, s) \models \varphi \mathbf{U}_{\mathrm{ch}} \varphi'$ if there is a node s' such that $s \prec_{\mathrm{ch}}^* s'$, $(T, s') \models \varphi'$, and for all $s'' \neq s'$ satisfying $s \prec_{\mathrm{ch}}^* s'' \prec_{\mathrm{ch}}^* s'$ we have $(T, s'') \models \varphi$.

The semantics of $\mathbf{S}_{\mathrm{ch}}$ is defined by reversing the order in the semantics of $\mathbf{U}_{\mathrm{ch}}$, and the semantics of $\mathbf{X}_{\mathrm{ns}}, \mathbf{X}_{\mathrm{ns}}^-, \mathbf{U}_{\mathrm{ns}}$, and $\mathbf{S}_{\mathrm{ns}}$ is the same by replacing the child relation with the next sibling relation.

$\mathrm{TL}^{\mathrm{tree}}$ naturally defines unary queries on trees, and it also defines Boolean queries: we say that $T \models \varphi$ if $(T, \varepsilon) \models \varphi$.

Theorem 4.1. (see [Mar05]) *A unary or Boolean query over unranked trees is definable in* FO *iff it is definable in* $\mathrm{TL}^{\mathrm{tree}}$*.*

In $\mathrm{CTL}^\star$-like logics, there are two kinds of formulae: those evaluated in nodes of trees, and those evaluated on paths in trees. This is similar to the situation with XPath, which has filter expressions evaluated on nodes, and location path expressions, which are evaluated on paths in XML trees. We shall now present two logics: $\mathrm{CTL}^\star$ with the past, in the spirit of [KP95], and a CTL-like reformulation of XPath, as presented in [Mar05].

We start with 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 [Mar05] (Conditional XPath) is defined to have *node formulae* α and *path formulae* β given by:

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

where `step` is one of the following: $\prec_{\mathrm{ch}}$, $\prec_{\mathrm{ch}}^-$, $\prec_{\mathrm{ns}}$, or $\prec_{\mathrm{ns}}^-$. 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 (that is, we define $(T, s) \models \alpha$), and each path formula in two nodes (that is, $(T, s, s') \models \beta$). The semantics is then as follows (we omit the rules for Boolean connectives):

- $(T, s) \models a$ iff $\lambda_T(s) = a$;
- $(T, s) \models \mathbf{E}\beta$ iff there is s' such that $(T, s, s') \models \beta$;
- $(T, s, s') \models ?\alpha$ iff $s = s'$ and $(T, s) \models \alpha$;
- $(T, s, s') \models \texttt{step}$ iff $(s, s') \in \texttt{step}$;
- $(T, s, s') \models \beta/\beta'$ iff for some s'' we have $(T, s, s'') \models \beta$ and $(T, s'', s') \models \beta'$;
- $(T, s, s') \models (\texttt{step}/?\alpha)^+$ 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 $\texttt{step}$, and $(T, s_{i+1}) \models \alpha$ for each $i < k$.

The language Core_XPath [GK+05, GKP05] is obtained by only allowing $\texttt{step}^+$ as opposed to $(\texttt{step}/?\alpha)^+$ in the definition of path formulae. Notice that since $\texttt{step}^+ = (\texttt{step}/?\mathit{true})^+$, where $\mathit{true} = \bigvee_{a \in \Sigma} a$, we have Core_XPath $\subseteq$ CXPath.

The earlier example of an XPath expression $(//a[//b]/c)$ can be represented in this syntax by a node formula $c \wedge \mathbf{E}(\prec_{\mathrm{ch}}^{-} /?a/ \prec_{\mathrm{ch}}^{+} /?b)$ saying that a node is labeled c, and there is a path that starts by going to its parent, finding a there, and then going to a descendant of that a and finding a b.

Core_XPath corresponds to XPath as defined by the W3C [CD99], while CXPath represents an addition to XPath proposed by [Mar05]. This addition is essentially the "until" operator of temporal logic: for example, to represent the strict version of until (that is, to say that in the next element of a path $a\mathbf{U}b$ holds), one could write $\prec_{\mathrm{ch}} /?b \ \vee \ (\prec_{\mathrm{ch}} /?a)^+/ \prec_{\mathrm{ch}} /?b$.

Node formulae of either CXPath or Core_XPath naturally define unary queries on trees. These can be characterized as follows.

Theorem 4.2. *a)* (see [Mar05]) *The node formulae of* CXPath *have precisely the power of* FO *unary queries.*

b) (see [MdR04]) *The node formulae of* Core_XPath *have precisely the power of unary* FO^2 *queries (that is,* FO *with two variables) in the vocabulary* $\prec_{\mathrm{ch}}, \prec_{\mathrm{ch}}^*, \prec_{\mathrm{ns}}, \prec_{\mathrm{ns}}^*$.

Part a) of Theorem 4.2 can also be extended to formulae in two free variables, see [Mar05].

4.2. **A** $\mathrm{CTL}^\star$**-like logic.** The logics CTL (computation tree logic) and $\mathrm{CTL}^\star$ are branching time temporal logics used in verification of reactive systems. They are normally defined without past connectives, but here we use the syntax close to that of [KP95] to make it possible to reason about the past. In these logics, one also has node (usually called state) formulae and path formulae, but path formulae are evaluated on paths, not on arbitrary pairs of nodes.

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

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

Given a tree, a child-path π_{ch} is a sequence of nodes on a path from the root to a leaf, and a sibling-path is a sequence π_{ns} of nodes of the form $s \cdot 0, \ldots, s \cdot (n-1)$ for a node s with n children. We define the semantics of node formulae with respect to a node in a tree, and of path formulae with respect to a path and a node on the path (i.e., we define the notion of $(T, \pi_*, s) \models \beta_*$, for $*$ being 'ch' or 'ns').

- $(T, s) \models \mathbf{E}\beta_*$ if there exists a path π_* such that $s \in \pi_*$ and $(T, \pi_*, s) \models \beta_*$;
- $(T, \pi_{\mathrm{ch}}, s) \models \mathbf{X}_{\mathrm{ch}}\beta$ if $(T, \pi_{\mathrm{ch}}, s') \models \beta$, where s' is the child of s on path π_{ch};
- $(T, \pi_{\mathrm{ch}}, s) \models \mathbf{X}^-_{\mathrm{ch}}\beta$ if $(T, \pi_{\mathrm{ch}}, s') \models \beta$ where s' is the parent of s on π_{ch};
- $(T, \pi_{\mathrm{ch}}, s) \models \beta\mathbf{U}_{\mathrm{ch}}\beta'$ if for some $s' \neq s$ such that $s' \in \pi_{\mathrm{ch}}$ and $s \prec^*_{\mathrm{ch}} s'$, we have $(T, \pi_{\mathrm{ch}}, s') \models \beta'$, and for all $s \prec^*_{\mathrm{ch}} s'' \prec^*_{\mathrm{ch}} s', s'' \neq s'$, we have $(T, \pi_{\mathrm{ch}}, s'') \models \beta$.

The definitions for $\mathbf{S}_{\mathrm{ch}}$ and for sibling-paths are analogous.

The following can be seen as an analog of the equivalence FO = $\mathrm{CTL}^\star$ for finite binary trees [HT87]. While the proof the connection between ranked and unranked tree, the straightforward translation from the binary tree fails because paths over translations of unranked trees may change direction between child and sibling-paths arbitrarily many times.

Theorem 4.3. (see [BL05]) *A unary or Boolean query over unranked trees is definable in* FO *iff it is definable in* $\mathrm{CTL}^\star_{\mathrm{past}}$.

4.3. **Extensions of** FO **and regular languages.** Over strings, FO falls short of all regular languages, as it defines precisely the star-free ones. However, using arbitrary regular expressions is often convenient in the context of navigating in XML documents.

Given a class $\mathcal{C}$ of regular expressions, define $\mathrm{FO}(\mathcal{C})^*$ as an extension of FO with the rules: (i) if e is a regular expression in $\mathcal{C}$ over $\mathrm{FO}(\mathcal{C})^*$ formulae $\psi(u, v)$, then $e^{\downarrow}(x, y)$ is a formula, and (ii) if e is a regular in $\mathcal{C}$ over $\mathrm{FO}(\mathcal{C})^*$ formulae $\psi(u)$, then $e^{\rightarrow}(x)$ is a formula. The semantics is the same as for the case of ETL. If formulae ψ are restricted to be Boolean combinations of atomic formulae P_a, $a \in \Sigma$, we obtain the logic $\mathrm{FO}(\mathcal{C})$.

Let StarFree be the class of star-free expressions, and Reg the class of all regular expressions.

Theorem 4.4. (see [NS00]) *a)* FO(StarFree) = FO(StarFree)* = FO.

b) FO(Reg) $\subsetneq$ FO(Reg)* $\subsetneq$ MSO.

For more on FO(Reg) and FO(Reg)* and their connections with fragments of MSO such as the path logic [Tho87], see [BLN06, NS00].

4.4. **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. Their complexity had been studied extensively. In general, 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., [GLS01, GSS01]).

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 $\prec_{\text{ch}}, \prec^*_{\text{ch}}, \prec_{\text{ns}}$, or $\prec^*_{\text{ns}}$. We write $\text{CQ}(\prec_1, \ldots, \prec_m)$ 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.

If we restrict ourselves to classes of conjunctive queries that use at most two binary predicates, then there is a complete classification for the complexity of query evaluation on unranked trees.

Theorem 4.5. (see [GKS04]) *The maximal tractable classes of queries* $\text{CQ}(\prec_1, \ldots, \prec_m)$*, where all* $\prec_i$*'s are among* $\{\prec_{\text{ch}}, \prec^*_{\text{ch}}, \prec_{\text{ns}}, \prec^*_{\text{ns}}\}$*, are* $\text{CQ}(\prec_{\text{ch}}, \prec_{\text{ns}}, \prec^*_{\text{ns}})$ *and* $\text{CQ}(\prec^*_{\text{ch}})$*; all others are* NP*-hard.*

In fact, [GKS04] provided a more general (but rather technical) criterion for checking when evaluation is in PTIME, and that condition can be used for other relations present in a query.

Conjunctive queries can also be used to capture all FO over unranked tree, even if more than one free variable is used, assuming path formulae of CXPath can be used as atomic predicates. More precisely, every FO formula $\varphi(\bar{x})$ over unranked trees is equivalent to a union of conjunctive queries whose atomic predicates are $\beta(x, x')$, where β ranges over path formulae of CXPath [Mar05].

5. Unordered trees

In unordered trees, nodes can still have arbitrarily many children, but the sibling ordering $\prec_{\text{ns}}$ is no longer available. That is, we view trees as structures

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

where D is a tree domain, $\prec^*_{\text{ch}}$ is the descendant relation, and P_a's define the labels on D. Logics considered for unordered unranked trees typically introduce some form of *counting*, see [BL05, Cou90, Cou91, DLM04, MR03, NP93, Sch92, SSM03, SS+04].

A simple explanation for this comes from a modified notion of unranked tree automata and query automata for unordered unranked trees. A *counting nondeterministic unranked tree* *automaton* is a tuple $\mathcal{A}_c = (Q, F, \delta)$, where, as before, 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), for $q \in Q$ and $a \in \Sigma$, into a Boolean function over V_Q. A *run* of $\mathcal{A}$ on an unordered tree T with domain D is then 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 $\mathcal{A}_c$ (that is, trees for which there is an accepting run) 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 $(\rho_{\mathcal{A}_c}(s), \lambda_T(s)) \in S$. As before, it can be given both existential and universal semantics.

The following appears not to have been stated explicitly, although it follows easily from results in [Nev99, NS02, SSM03].

Theorem 5.1. *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* $\mathcal{QA}_c$ *under either existential or universal semantics.*

5.1. MSO **and** FO **over unordered trees.** Now we look at several alternative characterizations of MSO and FO over unordered unranked trees that exploit the counting connection.

Define the *counting μ-calculus* C_μ (cf. [JL01]) 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 [Wal02], as was noticed in [JL01]:

Theorem 5.2. *Over unordered unranked trees,* MSO *and* $C_\mu[\prec_{\mathrm{ch}}]$ *have precisely the same power with respect to Boolean queries.*

In fact, it is not hard to show that MSO can be translated into alternation-free C_μ, and thus evaluated with complexity $O(\|T\| \cdot \|\varphi\|)$, where φ is an alternation-free C_μ formula.

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

We also extend $\mathrm{CTL}^\star$ with counting. In this counting extension $\mathrm{CTL}^\star_{\mathrm{count}}$, we have new state formulae $\mathbf{EX}^k_{\mathrm{ch}}\alpha$, where α is a state formula, with the same semantics as above.

Theorem 5.3. (see [MR03, Sch92]) *Over unordered unranked trees, the classes of Boolean queries expressed in* FO*,* $\mathrm{TL}^{\mathrm{tree}}_{\mathrm{count}}$*, and* $\mathrm{CTL}^\star_{\mathrm{count}}$ *over binary relation* $\prec_{\mathrm{ch}}$*, are the same.*

For unary queries, the equivalence $\mathrm{FO} = \mathrm{TL}^{\mathrm{tree}}_{\mathrm{count}}$ still holds [Sch92], and FO can be shown to be equivalent to an extension of $\mathrm{CTL}^\star$ with both counting and the past [BL05, Rab02].

Adding counting does not increase the complexity of model-checking in temporal logics, which is $2^{O(\|\varphi\|)} \cdot \|T\|$, cf. [CGP99].

Unordered fragments of XPath have also been looked at in the literature. For example, [BFK03] showed that the restriction of positive (no negation) Core_XPath that only uses $\prec_{\text{ch}}$ and $\prec^*_{\text{ch}}$ is equivalent to existential positive FO formulae over the vocabulary that includes both $\prec_{\text{ch}}$ and $\prec^*_{\text{ch}}$.

5.2. **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*, a very important concept in finite model theory, cf. [Lib04]. We say that an MSO sentence φ over vocabulary including $\prec^*_{\text{ch}}$ and $\prec^*_{\text{ns}}$ is $\prec_{\text{ns}}$*-invariant* if for any unordered tree T and any two expansions $T^{\prec^1_{\text{ns}}}$ and $T^{\prec^2_{\text{ns}}}$ with sibling-orderings $\prec^1_{\text{ns}}$ and $\prec^2_{\text{ns}}$ we have $T^{\prec^1_{\text{ns}}} \models \varphi \Leftrightarrow T^{\prec^2_{\text{ns}}} \models \varphi$. Any $\prec_{\text{ns}}$-invariant sentence defines a Boolean query on unordered trees.

We now define MSO_{mod} [Cou90] 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 5.4. (see [Cou91]) *Over unordered unranked trees, $\prec_{\text{ns}}$-invariant Boolean queries are precisely the Boolean queries definable in* MSO_{mod}.

Further extensions in terms of arithmetic power have been considered in [SSM03, SS+04]. Recall that Presburger arithmetic refers to the FO theory of the structure $\langle \mathbb{N}, + \rangle$, and it is known that this structure admits quantifier elimination in the vocabulary $(+, <, 0, 1, (\sim_k)_{k\in\mathbb{N}})$ where $n \sim_k m$ iff $n - m = 0 (\text{mod } k)$. We next define *Presburger* MSO, called 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 number of children of s_0 that belong to S_i, that is, the cardinality of the set $\{s' \mid s_0 \prec_{\text{ch}} s' \text{ and } s' \in S_i\}$. 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 $\text{MSO} \subsetneq \text{MSO}_{\text{mod}} \subsetneq \text{PMSO}$ 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. That is, in a run ρ of such an automaton, if $\rho(s) = q$, the label of s is a and $\delta(q, a) = \chi(v_{q_1}, \ldots, v_{q_m})$, then $\chi(n_1, \ldots, n_m)$ is true, where n_i is the number of children s' of s such that $\rho(s') = q_i$.

Theorem 5.5. (see [SSM03]) *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 [SS+04] and the μ-calculus with modulo-quantifiers [BL05].

5.3. **Edge-labeled unordered trees.** While in the early days of tree-based data models there was some debate as to whether labels should be on edges or nodes, the arrival of XML seems to have settled that dispute. Nonetheless, there are several areas where edge-labeled trees play a prominent and role, and traditionally logical formalisms have been designed for such data. First, there are logics for *feature trees*, which are a special case of feature structures used extensively in computational linguistics [Car92]. Second, in recent work on spatial logics, used for describing networks and mobile agents [CG00], one looks at modal logics over 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 logic for feature trees [Bla94] is the propositional modal logic that, in the context of feature structures (not necessarily trees), is also often supplemented with path-equivalence, stating that from a certain node, one can reach another node following two different paths. This is the setting of the Kasper-Rounds logic [RK86]. Over trees, however, path-equivalence is the same as equality of paths. A more powerful logic proposed in [Kel93] combined the Kasper-Rounds logic with the propositional dynamic logic. Its formulae are defined by

$$\varphi, \varphi' \ := \ a, \ a \in \Sigma \ \mid \ \varphi \vee \varphi' \ \mid \ \neg\varphi \ \mid \ \Diamond(e)\varphi \ \mid \ e \approx e',$$

where e, e' are regular expressions over $\mathcal{F}$. Formulae are evaluated in nodes of a feature tree T. We have $(T, s_0) \models \Diamond(e)\varphi$ if there is a path $(s_0, s_1) \in E_{f_0}, (s_1, s_2) \in E_{f_1}, \ldots, (s_{n-1}, s_n) \in E_{f_{n-1}}$ such that $(T, s_n) \models \varphi$ and $f_0 f_1 \ldots f_{n-1}$ is a word in the language denoted by e. Furthermore, $(T, s) \models e \approx e'$ if there is a node s' that can be reached from s by a word in e as well as a word in e'. This semantics is normally considered over graphs, but over trees this is equivalent to saying that there is a node reachable by an expression in the language denoted by $e \cap e'$. That is, $e \approx e'$ is equivalent to $\Diamond(e \cap e')\mathit{true}$, and thus the Kasper-Rounds logic is effectively a reachability logic over trees.

The reader is referred to [Kel93] for computational linguistics applications of this logic. In terms of expressiveness it is clearly contained in MSO, and if all expressions e, e' are star-free, then in FO as well, as long as we have the descendant relation.

Automata for feature trees, based on the algebraic approach to recognizability [Cou90], were considered in [NP93] (which also showed that over flat feature trees the automaton model coincides with a simple counting logic).

5.4. **An ambient logic for trees.** Ambient logics are modal logics for trees that have been proposed in the context of mobile computation [CG00] and later adapted for tree-represented data [Car01, CG01]. One views trees as edge-labeled and defines them by the grammar

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

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 [CCG03, Car01, CG01, CG00, DLM04]. Here we consider the logic from [CCG03] whose formulae are given by

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

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' with $T' \models \varphi$ we have $T|T' \models \varphi'$;
- $T \models a[\varphi]$ iff $T \equiv a[T']$ with $T' \models \varphi$;
- $T \models \varphi@a$ iff $a[T] \models \varphi$.

Variations appear in the literature, e.g. with the Kleene star in [DLM04] and recursion in [CG01].

The study of ambient logics for trees took a very 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. Several lines of work closely resemble those for node-labeled trees: e.g., [DLM04] introduced Presburger conditions on children, defined an automaton model, and proved decidability, similarly to [SSM03, SS+04].

However, the ambient logic 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'.

Since moving labels from edges to nodes and back can be defined in MSO, we see that the ambient logic is embedded into MSO. However, to the best of the author's knowledge, this direction has never been seriously pursued, and the exact relationship between ambient logics and other logics described in this survey is still not well understood.

6. Automatic structures

In this section we look at a different kind of logics for unranked trees, using the standard approach of model theory. So far we represented each tree as a structure and looked at definability over that structure. Now we want to consider structures whose universe is the set of *all* trees. Definability over such structures allows us to describe sets of trees and, more generally, relations over trees. Choosing the right operations on trees, we shall find structures where definable sets are precisely the regular languages. Such structures are very convenient for proving that certain properties of trees are regular, as it is sometimes easier to define properties logically than to construct automata for them.

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, interpreted over $\textsc{Tree}(\Sigma)$.

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 a first-order 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*.

6.1. Automatic structures on strings.

Before we move to trees, we first survey automatic structures over strings, cf. [BG00, BL+03]. In this case we consider structures of the form $\langle \Sigma^*, \Omega \rangle$. Our first example has the following relations in Ω:

- $\prec$ is a binary relation; $s \prec s'$ is true iff s is a prefix of s';
- L_a, $a \in \Sigma$, is a unary relation; $L_a(s)$ is true iff the last symbol of s is a;
- el is a binary relation; $\mathsf{el}(s, s')$ is true iff $|s| = |s'|$.

Let $\mathfrak{S}_{\text{univ}}$ be the structure $\langle \Sigma^*, \prec, (L_a)_{a \in \Sigma}, \mathsf{el} \rangle$. Then $\mathfrak{S}_{\text{univ}}$ is the *universal automatic structure*: that is, relations $\mathsf{Def}_n(\mathfrak{S}_{\text{univ}})$ are precisely the regular relations. Following a standard definition – see, e.g., [FS93] – we say that a relation $S \subseteq (\Sigma^*)^n$ is *regular* iff there is an automaton $\mathcal{A}$ over alphabet $(\Sigma \cup \{\#\})^n$ that accepts precisely the strings $[\bar{s}]$, for $\bar{s} = (s_1, \ldots, s_n) \in S$. The length of $[\bar{s}]$ is $\max_i |s_i|$, and the jth symbol of $[\bar{s}]$ is a tuple $(\sigma_1, \ldots, \sigma_n)$, where σ_i is the jth symbol of s_i if $|s_i| \leq j$, and $\#$ otherwise.

Thus, $\mathsf{Def}_1(\mathfrak{S}_{\text{univ}})$ contains exactly the regular languages over Σ. Furthermore, the conversion of formulae over $\mathfrak{S}_{\text{univ}}$ to automata is effective [BG00] and the theory of $\mathfrak{S}_{\text{univ}}$ is decidable. In fact the theory of every structure that is interpretable in $\mathfrak{S}_{\text{univ}}$ is thus decidable.

As an example, consider the structure $\langle \mathbb{Q}, < \rangle$. Since it is isomorphic to $\langle \{0,1\}^*1, <_{\text{lex}} \rangle$, where $<_{\text{lex}}$ is the lexicographic ordering (which is easily definable in $\mathfrak{S}_{\text{univ}}$), we obtain the well-known decidability of $\langle \mathbb{Q}, < \rangle$.

A restriction of $\mathfrak{S}_{\text{univ}}$ that does not have the equal length predicate, that is, $\mathfrak{S} = \langle \Sigma^*, \prec, (L_a)_{a \in \Sigma} \rangle$ is known to be strictly weaker that $\mathfrak{S}_{\text{univ}}$ in every dimension: in particular, el is not in $\mathsf{Def}_2(\mathfrak{S})$, and $\mathsf{Def}_1(\mathfrak{S})$ is precisely the class of star-free languages [BL+03].

Notice that both the empty string ε and functions $g_a(s) = s \cdot a$ are definable in $\mathfrak{S}$, and hence another well-known theory interpretable in $\mathfrak{S}$ and $\mathfrak{S}_{\text{univ}}$ is that of unary term algebras. However, it is known that for binary term algebras, adding relations like $\prec$ results in undecidable theories [MNT98, Ven87]. In particular, if we want to keep an analog of the $\prec$-relation (which is MSO-definable), we cannot introduce an operation like the $|$ operation in the ambient logic, and still have a decidable theory.

6.2. **Automatic structures on trees.** To get structures over $\text{TREE}(\Sigma)$ that define regular languages and relations[2], we find natural analogs of $\prec$, L_a, and el for trees. 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 $\preceq_\rightarrow$:** : 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 $j < i$.
- **Extension down $\preceq_\downarrow$:** : 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.

Clearly $T_1 \preceq T_2$ iff there is T' such that $T_1 \preceq_\rightarrow T'$ and $T' \preceq_\downarrow T_2$, so in terms of definability we do not lose anything by using $\preceq_\rightarrow$ and $\preceq_\downarrow$ instead of $\preceq$.

We define L_a to be true in a tree T if the rightmost node is labeled a. That is, the node $s \in D$ which is the largest with respect to $<_{\text{lex}}$ is labeled a. For the analog of el, recall that in the standard representation of strings as first-order structures, the domain is an initial segment of $\mathbb{N}$, corresponding to the length of the string. Hence, $\mathsf{el}(s_1, s_2)$ means that if strings are represented as structures, their domains are the same. We thus introduce a predicate $\approx_{\text{dom}}$ such that $T_1 \approx_{\text{dom}} T_2$ iff $D_1 = D_2$ (there D_i is the domain of T_i).

Now we define analogs of $\mathfrak{S}_{\text{univ}}$ and $\mathfrak{S}$:

$$\begin{array}{rcl} \mathfrak{T}_{\text{univ}} & = & \langle \text{TREE}(\Sigma),\ \preceq_\rightarrow,\ \preceq_\downarrow,\ (L_a)_{a\in\Sigma},\ \approx_{\text{dom}} \rangle \\ \mathfrak{T} & = & \langle \text{TREE}(\Sigma),\ \preceq_\rightarrow,\ \preceq_\downarrow,\ (L_a)_{a\in\Sigma} \rangle \end{array}$$

[2]The notion of regular relations for trees is obtained in the same way as for strings

Theorem 6.1. (see [BLN06]) *a) For every $n \geq 1$, $\mathsf{Def}_n(\mathfrak{T}_{\text{univ}})$ is precisely the class of regular n-ary relations over* $\text{TREE}(\Sigma)$.

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

Notice the difference with the string case, where removing el (domain equality) resulting in a smaller class of one-dimensional definable sets: star-free languages. On the other hand, even over binary trees, the notions of star-free and regular coincide [PT93].

Working with $\mathfrak{T}_{\text{univ}}$ makes it easy to write rather complicated properties of tree languages, and then Theorem 6.1 implies that those languages are regular. For example, if $X \subseteq \text{TREE}(\Sigma)$ 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 defined as $\varphi(T) = \forall T'\big(T \preceq T' \to \exists T''(T' \preceq_{\to} T'' \wedge \alpha_X(T''))\big)$, where α_X defines X (by Theorem 6.1, we know such α_X exists). Then Theorem 6.1 again tells us that φ defines a regular language. Furthermore, the 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. Consider, for example, a restriction of first-order logic over $\mathfrak{T}$ or $\mathfrak{T}_{\text{univ}}$ in which all quantification is over *branches*. A branch is a tree T such that the set $\{T' \mid T' \preceq T\}$ is linearly ordered by $\preceq$. Let Def_1^{η} be the class of sets of trees (equivalently, Boolean queries over trees) definable in this restriction.

Proposition 6.2. (see [BLN06]) $\mathsf{Def}_1^{\eta}(\mathfrak{T})$ is precisely the class of FO-definable Boolean queries over unranked trees, and $\mathsf{Def}_1^{\eta}(\mathfrak{T}_{\text{univ}})$ is the class of Boolean queries definable in a restriction of MSO in which quantification is allowed only over sets linearly ordered by $\prec^*_{\text{ch}}$ or by $\prec^*_{\text{ns}}$.

For more results of this type, see [BLN06].

6.3. **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 $\prec^*_{\text{ch}}$ is defined for rational domains just as before, and relation $\prec^*_{\text{ns}}$ is now given by $s \cdot r \prec^*_{\text{ns}} s \cdot r'$ iff $r \leq r'$. Then an unranked tree T over a rational unranked tree domain is, as before, a structure $T = \langle D, \prec^*_{\text{ch}}, \prec^*_{\text{ns}}, (P_a)_{a\in\Sigma}\rangle$.

Let $\textsc{Tree}_{\mathbb{Q}}(\Sigma)$ be the set of all unranked trees with rational unranked tree domains. Note that different trees in $\textsc{Tree}_{\mathbb{Q}}(\Sigma)$ may be isomorphic; we denote this isomorphism relation by $\cong$. There is a natural one-to-one correspondence between $\textsc{Tree}_{\mathbb{Q}}(\Sigma)/\cong$ and $\textsc{Tree}(\Sigma)$.

We define the extension relation $\preceq$ over trees in $\textsc{Tree}_{\mathbb{Q}}(\Sigma)$ as before. A *branch*, again, is a tree $T \in \textsc{Tree}_{\mathbb{Q}}(\Sigma)$ 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 $s \in \mathbb{Q}^*$; i.e., it is completely determined by s, which is its unique leaf. 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

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

In this structure it is much easier to reason about tree extensions that allow one to insert nodes between existing ones, and not only on the right or under the leaves. But what about definable sets and relations over $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$? It turns out that they are all regular. More precisely, we can interpret $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$ in $\mathfrak{T}_{\text{univ}}$: that is, find a set $X \in \mathsf{Def}_1(\mathfrak{T}_{\text{univ}})$, binary relations $R_1, R_2, R_3 \in \mathsf{Def}_2(\mathfrak{T}_{\text{univ}})$ and sets $Y_a \in \mathsf{Def}_1(\mathfrak{T}_{\text{univ}})$, $a \in \Sigma$, such that $\langle X, R_1, R_2, R_3, (Y_a)_{a\in\Sigma}\rangle$ is isomorphic to $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$. That is, we have:

Proposition 6.3. 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 $\textsc{Tree}_{\mathbb{Q}}(\Sigma)$ that contains exactly one representative of each $\cong$-equivalence class.

That is, under the mapping $\iota : \textsc{Tree}_{\mathbb{Q}}(\Sigma)/\cong \to \textsc{Tree}(\Sigma)$, definable sets and relations over $\mathfrak{T}^{\mathbb{Q}}_{\text{univ}}$ become precisely the regular tree languages (and relations). 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.

7. Other directions and conclusions

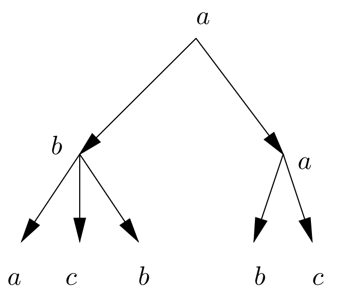
We present here a somewhat random sample of other directions that work on logics for unranked trees has taken or may take in the future. We concentrate on streaming applications, and then briefly describe other directions.

Streaming XML documents. A typical XML document is a sequence of matching opening and closing tags, with some data between then. For example, the sequence of opening and closing tags corresponding to a tree is shown in Figure 6. Thus, an XML tree naturally has a string representation. For example, for the tree in Figure 6, such a representation is

$$aba\bar{a}c\bar{c}b\bar{b}\bar{b}ab\bar{b}c\bar{c}\bar{a}\bar{a},$$

where we use a label, say a, for the opening tag `<a>`, and $\bar{a}$ for the closing tag `</a>`. More generally, for an ordered unranked tree T we define inductively its string representation $\text{str}(T)$:

- if T is a single node labeled a, then $\text{str}(T) = a\bar{a}$




```
<a>
  <b>
    <a></a>
    <c></c>
    <b></b>
  </b>
  <a>
    <b></b>
    <c></c>
  </a>
</a>
```

Figure 6: An XML document as a tree and as a sequence of tags

- if T has a root labeled a, with n children $s_0 \prec_{\rm ns} \ldots \prec_{\rm ns} s_{n-1}$, such that T_i is the subtree rooted at s_i, $i < n$, then $\mathrm{str}(T) = a\ \mathrm{str}(T_0)\ \ldots\ \mathrm{str}(T_{n-1})\ \bar{a}$.

If an XML document T is transmitted as a stream, then the object we work with is precisely $\mathrm{str}(T)$. Furthermore, we may not have the whole string $\mathrm{str}(T)$ available, or may need to compute some of its properties without looking at the whole string (for instance, a device receiving the stream may have memory limitations and cannot store the entire stream). One possible model for this scenario was proposed in [SV02]: in this model, one processes the stream $\mathrm{str}(T)$ by using a finite *string* automaton. It is natural to ask then what kinds of properties of trees can be recognized by finite automata that run on their streamed representations. More precisely, one is interested in tree languages of the form

$$L^{\rm str}_{\mathcal{A}} \ = \ \{T \mid \mathrm{str}(T) \text{ is accepted by } \mathcal{A}\},$$

where $\mathcal{A}$ is a string automaton.

This question has been primarily addressed in the context of DTD validation. Namely, given a DTD d, is it possible to find an automaton $\mathcal{A}_d$ such that

$$L^{\rm str}_{\mathcal{A}_d} \ = \ \textsc{Sat}(d)?$$

In general, the answer is negative, as was shown in [SV02]. We now sketch a very simple proof of this. Consider the following DTD d_1:

$$a \to ab \ \mid \ ca \ \mid \ \varepsilon, \qquad b \to \varepsilon, \qquad c \to \varepsilon.$$

Suppose $\textsc{Sat}(d_1) = L^{\rm str}_{\mathcal{A}}$ for some $\mathcal{A}$. The regular language given by $\mathcal{A}$ is definable in MSO, say by a sentence of quantifier rank r. Choose numbers n and k so that a^n and a^{n+k} cannot be distinguished by MSO sentences of quantifier rank r, and consider two strings:

$$\begin{array}{lcl} s_1 & = & a^n \quad (ac\bar{c})^n \quad a\bar{a} \quad \bar{a}^n \quad (b\bar{b}\bar{a})^n \\ s_2 & = & a^{n+k} \quad (ac\bar{c})^n \quad a\bar{a} \quad \bar{a}^{n+k} \quad (b\bar{b}\bar{a})^n \end{array}$$

which in turn (by a standard composition argument, see, e.g., [Lib04, Tho97]) cannot be distinguished by $\mathcal{A}$. One clearly has $s_1 = \mathrm{str}(T_1)$ for some $T_1 \in \mathrm{SAT}(d_1)$, and $s_2 = \mathrm{str}(T_2)$ for a tree $T_2 \in \mathrm{SAT}(d_2) - \mathrm{SAT}(d_1)$, where d_2 is

$$a \to a \mid ab \mid ca \mid \varepsilon, \qquad b \to \varepsilon, \qquad c \to \varepsilon,$$

which contradicts the assumption $\mathrm{SAT}(d_1) = L^{\mathrm{str}}_{\mathcal{A}}$.

While [SV02] provides many results on streamed validation of DTDs, the problem of characterizing DTDs that can be checked by finite automata over streamed representations remains open. Such a characterization can be found for MSO-definable properties as follows. Given an MSO sentence φ over ordered unranked trees, we say that φ is streamable if $\{T \mid T \models \varphi\}$ is of the form $L^{\mathrm{str}}_{\mathcal{A}}$ for some finite string automaton $\mathcal{A}$.

Let s be a node in a tree T; define $\mathrm{rma}(s)$ (the right-most ancestor) to be the smallest prefix of s such that each node s' with $\mathrm{rma}(s) \prec s' \preceq s$ is the largest in the $\prec^*_{\mathrm{ns}}$ ordering. This naturally defines a string of labels, by collecting all labels of nodes between $\mathrm{rma}(s)$ and s. We denote this string by $\mathrm{rms}(s)$. For example, if s is the rightmost node in the tree shown in Fig. 6, then $\mathrm{rms}(s) = aac$. Finally, for each regular language L over strings, we write $U^{\mathrm{rms}}_L(s)$ iff $\mathrm{rms}(s) \in L$.

The following is due to Segoufin and the author.

Proposition 7.1. An MSO sentence φ over ordered unranked trees is streamable iff it is expressible in MSO over the vocabulary that includes $\prec_{\mathrm{fc}}$, $(P_a)_{a\in\Sigma}$, and U^{rms}_L, where L ranges over regular languages.

However, the decidability of checking whether an MSO sentence belongs to the fragment of Proposition 7.1 remains open.

Some recent results on processing queries over streaming data (especially XPath queries) can be found in [BY+05, GKS05].

7.1. **Future directions and open questions.**

(1) This survey has concentrated primarily on Boolean and unary queries. While these are sufficient in many applications, there are formalisms that require more general n-ary queries. For example, the core expressions of XQuery can be seen as rearranging arbitrary n-tuples of nodes selected from a tree as another tree. The logical study of XQuery is just beginning [Koc05], and there are several papers that show how to extend results from logics that define Boolean and unary queries to arbitrary n-ary queries. For example, [Sch00] does it for queries definable in FO(Reg)-like logics. Using a similar approach, [ABL06] shows how to combine temporal logics over trees to define n-ary queries. An extension of unranked tree automata to n-ary queries is presented in [NP+05].

(2) While we have a number of logics that provide a declarative approach to expressing properties of trees and yet match (or are close to) the complexity of the procedural automata formalism, it is not really understood what causes certain logics to have such a nice behavior. There must be some intrinsic properties of logics that lead

to good model-checking algorithms (in a way similar to, say, finite- or tree-model properties being an explanation for decidability).

(3) Closely related to the first item is the issue of succinctness of logics, measured as the size of formulae needed to express certain properties. Initial investigation on the issue of succinctness for logics on ranked trees was done in [GS03] and some logics have been shown to be much more succinct than others, but more needs to be done. In view of the standard translation between ranked and unranked trees, it is likely that results for binary trees will be sufficient.

(4) The connection between FO, MSO, temporal logics and logics used in the programming languages and computational linguistics communities must be understood. The focus was quite different, as we mentioned earlier: for example, many questions about the complexity and expressiveness of ambient logics are unresolved. Some very recent results in this direction are reported in [BTT05].

(5) XML trees in addition to labels have data values associated with some nodes (typically attribute values or `PCDATA` values). Adding values from a potentially infinite set and just equality over them immediately leads to undecidable formalisms. This is observed, in particular, in the study of XML constraints. Some typically considered constraints include keys and foreign keys, that arise naturally when relational data is converted into XML. Keys say that a certain sequence of attributes identifies a node uniquely. A key is unary if it consists of one attribute (for example, a unique id would be a unary key, while a pair (firstname,lastname) can be a key consisting of two attributes). A foreign key states that a sequence of attributes of each node labeled by a_1 should also occur as a sequence of attributes of some other node labeled a_2.

XML specifications may consist of DTDs together with constraints. However, their interaction could be quite complicated. In fact, [FL02] showed that it is undecidable whether a specification that consists of a DTD and a set of keys and foreign keys is consistent. However, if all keys and foreign keys are unary, then consistency checking is NP-complete.

It would be nice to find a purely logical explanation for this type of results. Decidability restrictions studied in [NSV01, BPT01] are very weak for this purpose. However, a recent line of results shows much more promise. Consider trees that can carry data values, and assume that we can test them for equality, that is, we have a binary relation $\sim$ that is true if two nodes in a tree have the same data values. Then FO^2 over such trees with the $\sim$ relation and the successor relation is decidable [B+06a]. Here FO^2 refers to FO with two variables. Notice that for expressing unary constraints two variables suffice. It is open whether the descendant can be added while preserving decidability; the only resolved case is that of strings, where indeed FO^2 over the successor relation, the linear ordering, and the $\sim$ relation is decidable [B+06b].

Acknowledgement

I am grateful to Cristiana Chitic, Christoph Koch, Maarten Marx, Frank Neven, Joachim Niehren, Gerald Penn, Thomas Schwentick, Luc Segoufin, Anthony Widjaja To, and the referees for their comments on the paper. I also thank Luc Segoufin for his permission to include Proposition 7.1 in the survey. This work was supported by grants from NSERC and CITO.

References

[ABL06] M. Arenas, P. Barceló, L. Libkin. Combining temporal logics for querying XML documents. Manuscript, 2006.

[AHV95] S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*, Addison Wesley, 1995.

[AN01] A. Arnold, D. Niwinski. *Rudiments of μ-calculus*. Studies in Logic and the Foundations of Mathematics 146, Elsevier, 2001.

[B+06a] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, L. Segoufin. Two-variable logic on data trees and XML reasoning. In *ACM Symp. on Principles of Database Systems*, 2006, to appear.

[B+06b] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, L. Segoufin. Two-variable logic on words with data. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2006, to appear.

[BFK03] M. Benedikt, W. Fan, G. Kuper. Structural properties of XPath fragments. *Theoretical Computer Science*, 336 (2005), 3–31.

[BG00] A. Blumensath and E. Grädel. Automatic structures. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2000, pages 51–62.

[BL05] P. Barceló, L. Libkin. Temporal logics over unranked trees. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2005, pages 31–40.

[Bla94] P. Blackburn. Structures, languages and translations: the structural approach to feature logic. In *Constraints, Language and Computation*, edited by C. Rupp, M. Rosner and R. Johnson, Academic Press, 1994, pages 1–27.

[BLN06] M. Benedikt, L. Libkin, F. Neven. Logical definability and query languages over ranked and unranked trees. *ACM Trans. on Comput. Logic*, 2006, to appear.

[BL+03] M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. Definable relations and first-order query languages over strings. *Journal of the ACM*, 50 (2003), 694–751.

[BMW01] A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets: Version 1, 2001. Technical Report HKUST-TCSC-2001-0, The Hongkong University of Science and Technology, 2001.

[BPT01] P. Bouyer, A. Petit, D. Thérien. An algebraic characterization of data and timed languages. In *Int. Conf. on Concurrency Theory 2001*, pages 248–261.

[BS05] M. Benedikt, L. Segoufin. Regular tree languages definable in FO. In *Symp. on Theor. Aspects of Comp. Sci. 2005*, pages 327–339.

[BTT05] I. Boneva, J.-M. Talbot, S. Tison. Expressiveness of a spatial logic for trees. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2005.

[Büc60] J.R. Büchi. Weak second-order arithmetic and finite automata. *Zeit. Math. Logik Grundl. Math.* 6 (1960), 66–92.

[BW04] M. Bojanczyk, I. Walukiewicz. Characterizing EF and EX tree logics. In *Int. Conf. on Concurrency Theory 2004*, pages 131–145.

[BY+05] Z. Bar-Yossef, M. Fontoura, V. Josifovski. Buffering in query evaluation over XML streams. In *ACM Symp. on Principles of Database Systems 2005*, pages 216–227.

[Car92] B. Carpenter. *The Logic of Typed Feature Structures*. Cambridge, 1992.

[Car01] L. Cardelli. Describing semistructured data. *SIGMOD Record* 30 (2001), 80–85.

[CCG03] C. Calcagno, L. Cardelli, A. Gordon. Deciding validity in a spatial logic for trees. *J. Functional Programming*, 15 (2005), 543–572.

[CD99] J. Clark and S. DeRose. XML Path Language (XPath). W3C Recommendation, Nov. 1999. http://www.w3.org/TR/xpath.

[CG00] L. Cardelli, A. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Principles of Progr. Lang. 2000*, pages 365–377.

[CG01] L. Cardelli, G. Ghelli. A query language based on the ambient logic. In *Europ. Symp. on Progr. 2001*, pages 1–22.

[CG+02] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata: Techniques and Applications*. Available at www.grappa.univ-lille3.fr/tata. October 2002.

[CGP99] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[CNT04] J. Carme, J. Niehren, M. Tommasi. Querying unranked trees with stepwise tree automata. In *Rewriting Techniques and Applications 2004*, pages 105–118.

[Cou90] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85 (1990), 12–75.

[Cou91] B. Courcelle. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theoretical Computer Science* 80 (1991), 153–202.

[DLM04] S. Dal-Zilio, D. Lugiez, C. Meyssonnier. A logic you can count on. In *POPL 2004*, pages 135–146.

[Don70] J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[EJ91] E. A. Emerson, C. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS 1991*, pages 368–377.

[EW03] Z. Ésik, P. Weil. On logically defined recognizable tree languages. In *Found. of Software Tech. and Theor. Comp. Sci. 2003*, pages 195–207.

[FG02] M. Frick, M. Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130 (2004), 3–31.

[FGK03] M. Frick, M. Grohe, C. Koch. Query evaluation on compressed trees. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2003, pages 188-197.

[FL02] W. Fan and L. Libkin. On XML integrity constraints in the presence of DTDs. *Journal of the ACM*, 49 (2002), 368-406.

[FS93] C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Computer Science* 108 (1993), 45–82.

[GK02] G. Gottlob, C. Koch. Monadic queries over tree-structured data. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2002, pages 189–202.

[GK04] G. Gottlob, C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *Journal of the ACM* 51 (2004), 74–113.

[GKP05] G. Gottlob, C. Koch, R. Pichler, Efficient algorithms for processing XPath queries. *ACM Trans. on Database Systems*, 30 (2005), 444–491.

[GK+05] G. Gottlob, C. Koch, R. Pichler, and L. Segoufin. The complexity of XPath query evaluation and XML typing. *Journal of the ACM*, 52 (2005), 284–335.

[GKS04] G. Gottlob, C. Koch, K. Schulz. Conjunctive queries over trees. In *ACM Symp. on Principles of Database Systems 2004*, pages 189–200.

[GKS05] M. Grohe, C. Koch, N. Schweikardt. Tight lower bounds for query processing on streaming and external memory data. In *Proc. Intl. Colloq. Automata, Lang., and Progr.*, Springer 2005, pages 1076–1088.

[GLS01] G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 48 (2001), 431–498.

[GS03] M. Grohe, N. Schweikardt. Comparing the succinctness of monadic query languages over finite trees. In *CSL 2003*, pages 226–240.

[GSS01] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Symp. on Theory of Comput. 2001*, pages 657–666.

[HT87] T. Hafer, W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. *Int. Colloq. on Automata, Languages, and Programming 1987*, pages 269–279.

[JL01] D. Janin, G. Lenzi. Relating levels of the mu-calculus hierarchy and levels of the monadic hierarchy. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 2001, pages 347–356.

[Kam68] H.W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD Thesis, UCLA, 1968.

[Kel93] B. Keller. *Feature Logics, Infinitary Descriptions and Grammar.* CSLI Press, 1993.

[Koc05] C. Koch. On the complexity of nonrecursive XQuery and functional query languages on complex values. In *ACM Symp. on Principles of Database Systems 2005*, pages 84–97.

[KP95] O. Kupferman, A. Pnueli. Once and for all. In *Proc. IEEE Symp. on Logic in Comp. Sci.*, 1995, pages 25–35.

[Lib04] L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.

[Mar05] M. Marx. Conditional XPath. *ACM Trans. on Database Systems*, 30 (2005), 929–959.

[Mat02] R. Mateescu. Local model-checking of modal mu-calculus on acyclic labeled transition systems. In *Tools and Algorithms for Construction and Analysis of Systems*, 2002, pages 281–295.

[MdR04] M. Marx and M. de Rijke. Semantic characterizations of XPath. In *TDM workshop on XML Databases and Information Retrieval*, 2004.

[MN05] W. Martens, J. Niehren. Minimizing tree automata for unranked trees. In *Proc. Database Progr. Lang.*, Springer, 2005, pages 232–246.

[MNT98] M. Müller, J. Niehren, R. Treinen. The first-order theory of ordering constraints over feature trees. *Discrete Mathematics and Theoretical Computer Science*, 4 (2001), 193–234.

[MR03] F. Moller, A. Rabinovich. Counting on CTL*: on the expressive power of monadic path logic. *Information and Computation*, 184 (2003), 147-159.

[Nev99] F. Neven. *Design and Analysis of Query Languages for Structured Documents.* PhD Thesis, U. Limburg, 1999.

[Nev02] F. Neven. Automata, logic, and XML. In *CSL 2002*, pages 2–26.

[Niw88] D. Niwinski. Fixed points vs. infinite generation. In *Proc. IEEE Symp. on Logic in Comp. Sci.* 1988, pages 402–409.

[NP93] J. Niehren, A. Podelski. Feature automata and recognizable sets of feature trees. In *TAPSOFT 1993*, pages 356–375.

[NP+05] J. Niehren, L. Planque, J.-M. Talbot, S. Tison. N-ary queries by tree automata. In *Proc. Database Progr. Lang.*, Springer, 2005, pages 217–231.

[NS98] A. Neumann, H. Seidl. Locating matches of tree patterns in forests. In *Found. of Software Tech. and Theor. Comp. Sci. 1998*, pages 134–145.

[NS00] F. Neven, Th. Schwentick. Expressive and efficient pattern languages for tree-structured data. In *ACM Symp. on Principles of Database Systems 2000*, pages 145–156.

[NS02] F. Neven, Th. Schwentick. Query automata over finite trees. *Theor. Comput. Sci.* 275 (2002), 633–674.

[NSV01] F. Neven, Th. Schwentick, V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5 (2004), 403–435.

[PQ68] C. Pair and A. Quere. Définition et etude des bilangages réguliers. *Information and Control*, 13(6):565–593, 1968.

[PT93] A. Potthoff, W. Thomas. Regular tree languages without unary symbols are star-free. In *Fundamentals of Computation Theory 1993*, pages 396–405.

[PV00] Y. Papakonstantinou, V. Vianu. DTD inference for views of XML data. In *Proc. ACM Symp. on Principles of Database Systems*, 2000, pages 35–46.

[Rab69] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.* 141 (1969), 1–35.

[Rab02] A. Rabinovich. Expressive power of temporal logics. In *Int. Conf. on Concurrency Theory 2002*, pages 57–75.

[RK86] 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.

[Sch92] B.-H. Schlingloff. Expressive completeness of temporal logic of trees. *Journal of Applied Non-Classical Logics* 2 (1992), 157–180.

[Sch00] T. Schwentick. On diving in trees. In *Proc. Math. Found. Comp. Sci.*, Springer 2000, pages 660–669.

[Smo92] G. Smolka. Feature-constraint logics for unification grammars. *J. Log. Progr.* 12 (1992), 51–87.

[SM02] 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.

[SSM03] H. Seidl, Th. Schwentick, A. Muscholl. Numerical document queries. In *ACM Symp. on Principles of Database Systems 2003*, pages 155–166.

[SS+04] H. Seidl, Th. Schwentick, A. Muscholl, P. Habermehl. Counting in trees for free. In *Int. Colloq. on Automata, Languages, and Programming 2004*, pages 1136–1149.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.

[SV02] L. Segoufin, V. Vianu. Validating streaming XML documents. In *ACM Symp. on Principles of Database Systems 2002*, pages 53–64.

[Tak75] M. Takahashi. Generalizations of regular sets and their application to a study of context-free languages. *Information and Control*, 27(1):1–36, 1975.

[Tha67] J.W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *J. Comput. Syst. Sci.* 1 (1967), 317–322.

[Tho87] W. Thomas. On chain logic, path logic, and first-order logic over infinite trees. In *Proc. IEEE Symp. on Logic in Comput. Sci.*, 1987, pages 245–256.

[Tho97] W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Vol. 3*, Springer-Verlag, 1997, pages 389–455.

[TW68] 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.

[Var98] M. Y. Vardi. Reasoning about the past with two-way automata. In *Int. Colloq. on Automata, Languages, and Programming 1998*, pages 628–641.

[Ven87] K. Venkataraman. Decidability of the purely existential fragment of the theory of term algebras. *Journal of the ACM* 34 (1987), 492–510.

[Via01] V. Vianu. A web Odyssey: from Codd to XML. In *ACM Symp. on Principles of Database Systems*, 2001, pages 1–15.

[Wal02] I. Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science* 275 (2002), 311–346.

Department of Computer Science, University of Toronto, and School of Informatics, University of Edinburgh

E-mail address: `libkin@cs.toronto.edu` and `libkin@inf.ed.ac.uk`