Numerical Document Queries

Helmut Seidl
FB IV – Informatik
Universität Trier
D-54286 Trier, Germany
seidl@psi.uni-trier.de

Thomas Schwentick
Philipps-Universität Marburg
FB Mathematik und Informatik
D-35032 Marburg
tick@informatik.uni-marburg.de

Anca Muscholl
LIAFA, Université Paris 7
2 place Jussieu
75251 Paris, Cedex 05, France
anca@liafa.jussieu.fr

February 26, 2003

ABSTRACT

A query against a database behind a site like Napster may search, e.g., for all users who have downloaded more jazz titles than pop music titles. In order to express such queries, we extend classical monadic second-order logic by Presburger predicates *which pose numerical restrictions on the children (content) of an element node and provide a precise automata-theoretic characterization. While the* existential fragment *of the resulting logic is decidable, it turns out that satisfiability of the* full *logic is undecidable. Decidable satisfiability and a querying algorithm even with* linear *data complexity can be obtained if numerical constraints are only applied to those contents of elements where ordering is irrelevant. Finally, it is sketched how these techniques can be extended also to answer questions like, e.g., whether the total price of the jazz music downloaded so far exceeds a user's budget.*

Keywords: Querying XML documents, monadic second order logic, Presburger arithmetic, automata.

1. INTRODUCTION

Monadic Second-Order Logic has been identified as an important logic in the theory of XML schema and query languages. It captures the class of regular tree languages which is basically the class of XML documents that can be specified by a type definition in XML Schema. It has also been used to prove decidability of the type checking problem for an important part of XSLT transformations [18]. However, most of the work investigated MSO logic as a basis for XML querying. It turned out that it defines a very robust class of queries with a lot of equivalent characterizations by other query mechanisms like attributed grammars [23], automata [20, 1, 21] and datalog [12]. Although MSO logic is a robust and powerful query language it can not express all kinds of queries one might be interested in. For instance, queries in the cited papers are not allowed to compare data values that are located at different nodes of a tree (see [22] for a step in that direction). Also, there are no means to compute with data values or to count and compare the numbers of occurrences of different types of nodes.

In this paper, we therefore investigate in how far MSO-based querying can be extended to also allow for reasoning about numerical properties of XML documents. As an example consider a document containing music files shared by some peer-to-peer system as Napster, Gnutella etc. as exposed in Figure 1.[1]

For instance, we would like to query for users who prefer jazz over pop. Such a query can be expressed by asking for nodes labeled with "music" that have more sons labeled "jazz" than "pop". Querying for users who are extreme jazz fans can be expressed by requiring that the majority of the sons of a node labeled by "music" is labeled by "jazz".

In order to formulate such queries, we extend Monadic Second Order (MSO) Logic by Presburger arithmetic formulas constraining the children of a node (*Presburger constraints* for short). In this new *Presburger MSO* Logic, the first query can be expressed as:

$$x \in \mathsf{Lab}_{\mathtt{music}} \wedge x/\phi_1$$

where ϕ_1 is the formula

$$\phi_1 \equiv [\mathsf{Lab}_{\mathtt{jazz}}] \geq [\mathsf{Lab}_{\mathtt{pop}}]$$

Here, $[\mathsf{Lab}_{\mathtt{jazz}}]$ and $[\mathsf{Lab}_{\mathtt{pop}}]$ denote the numbers of children labeled with `jazz` and `pop`, respectively. For the second query we replace ϕ_1 by ϕ_2, where ϕ_2 is the formula:

$$\phi_2 \equiv [\mathsf{Lab}_{\mathtt{jazz}}] \geq [\mathsf{Lab}_{\mathtt{pop}}] + [\mathsf{Lab}_{\mathtt{french}}] + [\mathsf{Lab}_{\mathtt{classic}}]$$

As an operational counterpart of the extended logic we study bottom-up tree automata that are enhanced by *Presburger constraints*. Transitions from the children of a node to the node itself may depend on the frequencies of states at the children via a Presburger arithmetic condition, i.e., a formula involving addition.

We start our investigation by considering automata that *only* use Presburger constraints, i.e., automata that disregard the order of the children of a node and only use cardinalities of states. Technically speaking, we study in this

PODS 2003, June 9-12, 2003, San Diego, CA.

[1] It should be noted that in a realistic setting the type of music would likely be represented by an attribute and not by a separate tag for each type. But, of course, for the purpose of query processing we can interpret a tag with attribute *jazz* as a tag *jazz*.

```
<doc>
 <user>
  <name> ... </name>
   ...
  <music>
   <jazz>
    <album> Always let me go </album>
    <artist> Keith Jarrett </artist>
    <year> 2002 </year>
    <time> 3310 </time>
    <price> 240 </price>
   </jazz>
   <pop>
    <tit> Just my imagination</tit>
    <artist> The Cranberries </artist>
    <year> 2002 </year>
    <album> Stars </album>
    <time> 220 </time>
    <price> 20 </price>
   </pop>
   <french>
    <tit> Aux enfants de la chance </tit>
    <artist> Serge Gainsbourg </artist>
    <album> Serge Gainsbourg, vol. 3 </album>
    <time> 247 </time>
    <price> 16 </price>
   </french>
   <classic>
    <tit> The Seven Gates of Jerusalem </tit>
    <comp> Krzystof Penderecki </comp>
    <recorded> 1999 </recorded>
   <time> 3510 </time>
    <price> 262 </price>
   </classic>
   <jazz>
    <album> Kind of Blue </album>
    <artist> Miles Davis </artist>
    <year> 1997 </year>
    <time> 3325 </time>
    <price> 220 </price>
   </jazz>

  </music>
  <video>
      ...
  </video>
  <images>
      ...
  </images>
 </user>
</doc>
```

Figure 1: An example document containing information about music files downloaded by users.

part automata on unordered trees. It turns out that these automata are very well-behaved. They define a class of trees with very regular properties like various closure properties and equivalence with Presburger MSO logic. Further, these automata allow for effective static analysis. Emptiness and universality are decidable, from a nondeterministic automaton an equivalent deterministic automaton can be constructed. Last but not least, they allow to define a class of (unary) queries the evaluation of which has linear time data complexity.

Next, we study automata that are allowed to combine Presburger constraints with the common regular language constraints. It turns out that this leads to automata with less desirable properties. Although emptiness of such automata can still be decided, universality (whether an automaton accepts *all* trees) becomes undecidable. As we show that the nondeterministic automata of this type can be characterized by existential MSO logic we can conclude that MSO logic which takes into account the order of children becomes undecidable. Nevertheless, the word problem for these automata is decidable in polynomial time.

Often, however, and in particular in our example, some parts of a document can be considered as textual representations of information records. This means that inside certain elements, the ordering is not significant. We therefore investigate automata on *mixed* document trees, i.e., in which element tags either identify their content as ordered or as unordered. We further assume that, as in our example, numerical constraints only are applicable to such unordered element contents. Under these assumptions, we get the same kind of nice behavior as in the totally unordered case, mentioned above.

In many cases, one might not only be interested in cardinalities of tags but also in conditions to be fulfilled by the numbers that occur in a document. We sketch how our approach can also be used in this setting.

Related Work. Unordered document trees are closely related to the generalization of feature trees considered by Niehren and Podelski in [24] where they study the (classical) notion of recognizability and give a characterization of this notion by means of feature automata. No counting constraints are considered. Query languages for unordered trees have been proposed by Cardelli and Ghelli [5, 4, 6, 7] (and their co-workers). Their approach is based on first-order logic and fixpoint operators. Neither unbounded numerical values nor automata-theoretic characterizations or complexity issues are taken into account.

Kupferman, Sattler and Vardi study a μ-calculus with *graded* modalities where one can express, e.g., that a node has at least n successors satifying a certain property [14]. The numbers n there, however, are hard-coded into the formula. Klaedtke and Ruess consider automata on the unlabeled infinite binary tree, that have an accepting condition depending on a global Presburger constraint [13].

Extending the construction in [15], Lugiez and Dal Zilio have independently proposed automata models which are related to ours [16, 17]. The model closest to our investigations is studied in [17]: it essentially equals our second automaton model which allows to combine regular and Presburger constraints freely. For this model, Lugiez and Dal Zilio obtain comparable results concerning closure properties, membership tests and decidability of emptiness. In order to express properties of trees, they consider a *modal* logic similar in spirit to the proposal of Cardelli and Ghelli but without fixpoint operators or explicit quantification (outside Presburger sub-formulas). Although no precise characterization is given, their logic is strictly less powerful than the given automata model.

Our paper is organized as follows. We start in section 2 with the completely unordered case. Thus, we formally introduce *Presburger tree automata* running on unordered trees which can deal with numerical properties expressed by Presburger formulas. We study Presburger MSO logic. Then we extend the approach to ordered trees in section 3. In section 4, we study *mixed* trees. Finally, we sketch the treatment of numbers explicitly mentioned in a document in section 5.

2. UNORDERED TREES

In this section, we exemplify our techniques for the special case of unordered trees which seems of interest also in its own right.

For a base set A, let $\mathbb{N}^A$ denote the set of all multi-sets over A. An individual finite multi-set m which consists of the sequence of elements $a_1, \ldots, a_k$, $a_i \in A$ (not necessarily distinct), is denoted by:

$$m = \{a_1, \ldots, a_k\}$$

or:

$$m = \sum_{a \in A} v_a \cdot \{a\}$$

where $v_a \in \mathbb{N}$ is the *multiplicity* of the element a in the sequence. Note that multiplicities can also be 0. The set m is finite iff all but finitely many v_a are 0. In particular, the multiset union of two multisets m_1, m_2 is denoted by $m_1 + m_2$. Also, we write $a \in m$ if a occurs in m with non-zero multiplicity.

Let Σ denote a finite alphabet. Given the above notion of multisets, we define the set U_Σ of *unordered trees* t (u-trees for short) over Σ by the following grammar:

$$t \quad ::= \quad a\{t_1, \ldots, t_k\} \qquad (a \in \Sigma, k \geq 0)$$

Presburger formulas ϕ are defined by the following grammar:

$$\begin{array}{rcl} \phi & ::= & x = n \mid x + y = z \\ & & \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists x.\phi \end{array}$$

where x, y, z are variables and $n \in \mathbb{N}$ is a constant. As usual, we use abbreviations like $x \leq z$ for $\exists y.\ x + y = z$, $2x + y = z$ for $(\exists t.\ x + x = t \wedge t + y = z)$, false for $\exists x.\ x < 0$, $\phi_1 \vee \phi_2$ for $\neg(\neg\phi_1 \wedge \neg\phi_2)$ and $\forall x.\phi$ for $\neg\exists x.\neg\phi$. For a formula ϕ and an assignment ρ of (a superset of) the free variables of ϕ to naturals, we define the satisfaction relation $\rho \models \phi$ by:

$$\begin{array}{lll} \rho \models x = n & \text{iff} & \rho(x) = n \\ \rho \models x + y = z & \text{iff} & \rho(x) + \rho(y) = \rho(z) \\ \rho \models \phi_1 \wedge \phi_2 & \text{iff} & \rho \models \phi_1 \text{ and also } \rho \models \phi_2 \\ \rho \models \neg\phi & \text{iff} & \rho \not\models \phi \\ \rho \models \exists x.\phi & \text{iff} & \rho \oplus \{x \mapsto n\} \models \phi \text{ for some } n \in \mathbb{N} \end{array}$$

In particular, a formula ϕ is satisfiable iff there is an assignment ρ such that $\rho \models \phi$. It is well-known that satisfiability of Presburger arithmetic is decidable in doubly exponential space. For complexity results of corresponding decision procedures, we refer to [9, 8, 2]. Note that the set of vectors v satisfying a Presburger formula with free variables is a *semi-linear set* which can be effectively computed [10, 11]. Recall that a semi-linear set is a finite union of *linear sets*, i.e., sets of the form $\{barc + \sum_i x_i \bar{p_i} \mid x_i \in \mathbb{N}\}$, where $\bar{c}, \bar{p_i}$ are finitely many vectors over $\mathbb{N}$. Presburger formulas ϕ can be compiled into finite automata A_ϕ running on the binary representations of these vectors. The automaton A_ϕ checks whether a given tuple is contained in the semi-linear set corresponding to ϕ. Practical verification tools based on such automata, e.g., are studied by Wolper and Boigelot [28].

Given a finite set Q (of states), we will consider a canonical set Y_Q of variables which are indexed by the elements in Q. So, we define:

$$Y_Q = \{y_q \mid q \in Q\}$$

Presburger automata

A *Presburger u-tree automaton* is given by the tuple $A = (Q, \Sigma, \delta, F)$ where:

- Q is a finite set of states,
- $F \subseteq Q$ is the subset of accepting states,
- Σ is the finite alphabet of tree labels, and
- δ maps pairs (q, a) of states and labels to Presburger formulas with free variables from the set Y_Q.

The formula $\phi = \delta(q, a)$ represents the *pre-condition* on the children of a node labeled by a for the transition into state q where the possible values of the variable y_p represent the admissible multiplicities of the state p on the children. As an example, consider the formula $\psi = \exists z.\ y_p + 2z = y_q$. This formula expresses that the number y_q of sons labeled by the state q is at least as big as y_p, the respective number for state p, and that the difference is even. Formally, we introduce a satisfaction relation $t \models_A q$ between u-trees t and states q which is defined as follows. Assume that $t = aS$ (i.e., S is the – possibly empty– multiset of subtrees of the root labeled a) and $\delta(q, a) = \phi$. Then $t \models_A q$ iff there are multisets of trees S_p of cardinalities n_p, $p \in Q$, such that:

- $S = \sum_{p \in Q} S_p$;
- $t' \models_A p$ for all $t' \in S_p$ $(p \in Q)$;
- $\{y_p \mapsto n_p \mid p \in Q\} \models \phi$.

The language $\mathcal{L}(A)$ of u-trees which is *accepted* by the automaton A then is given by:

$$\mathcal{L}(A) = \{t \in U_\Sigma \mid \exists f \in F : t \models_A f\}$$

Note that Presburger automata are non-deterministic. We will view them as bottom-up automata in the following. In order to get an idea how these automata work, consider the language L of all trees over $\{a, b\}$ where the internal nodes are all labeled with a and have at most as many subtrees with a b-leaf as without. One example tree t from this language is depicted in fig. 2. An automaton for L needs two states, say 0 and 1 where the transition function δ can be represented in the table:

δ	a	b
0	$y_1 = 0$	false
1	$y_0 \geq y_1 > 0$	$y_0 + y_1 = 0$

The state 0 is assumed by all trees without b leaves while the state 1 is only assumed by trees containing b leaves. Figure

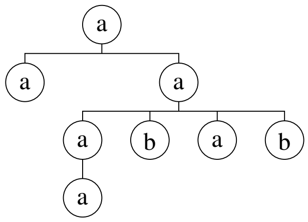

Figure 2: An example tree t from L.

3 shows a *run* of the automaton on t, i.e., an assignment mapping the nodes (subtree occurrences) of t to states such that the pre-conditions in δ are locally satisfied at every node. In particular, we have: $t \models 1$.

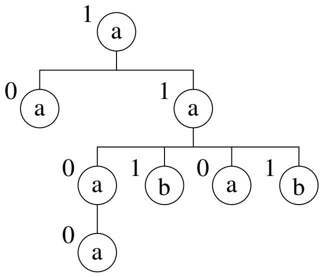

Figure 3: An example run on the example tree t.

The main result of this section is:

THEOREM 1. *Emptiness for Presburger u-tree automata is decidable.*

PROOF. Consider a Presburger tree automaton $A = (Q, \Sigma, \delta, F)$. Let us call a state $q \in Q$ *reachable* iff there is a tree t with $t \models_A q$. As for finite automata on ordered trees, we successively determine the set R of reachable states by:

$$R = \bigcup_{j \geq 0} R^{(j)}$$

where $R^{(0)} = \emptyset$. For $j > 0$, $q \in R^{(j)}$ iff the following formula is satisfiable:

$$\left(\bigwedge_{p \in Q \setminus R^{(j-1)}} y_p = 0 \right) \wedge \left(\bigvee_{a \in \Sigma} \delta(q, a) \right)$$

Since satisfiability of Presburger formulas is decidable, the sets $R^{(j)}$ are effectively computable. By induction on j, we prove:

Claim: For every $j \geq 0$, $q \in R^{(j)}$ iff there is a tree t of depth at most $j - 1$ such that $t \models_A q$.

In particular, the sequence of sets $R^{(j)}, j \geq 0$, is (not necessarily strictly) increasing. Moreover, $R^{(j)} = R^{(j+k)}$ for every $k \geq 0$ whenever $R^{(j)} = R^{(j+1)}$. Accordingly, $R = R^{(n)}$ with n denoting the number of states $|Q|$. Therefore, the set of all reachable states is computable. Since $\mathcal{L}(A)$ is nonempty iff some state $f \in F$ is reachable, we conclude that emptiness for Presburger tree automata is decidable. Given that all Presburger formulas ϕ in δ have already been compiled into automata A_ϕ (which now are part of the input), the emptiness test even runs in quadratic time. □

Closure Properties

Let $\mathcal{U}_\Sigma$ denote the set of languages that can be accepted by Presburger u-tree automata. We show next that $\mathcal{U}_\Sigma$ is closed under the Boolean operations union, intersection and complement. In particular, we define the notion of *deterministic* Presburger u-tree automata and construct for every Presburger u-tree automaton an equivalent deterministic one.

THEOREM 2. *The family of languages $\mathcal{U}_\Sigma$ is effectively closed under:*

1. *union,*
2. *intersection,*
3. *complementation.*

PROOF. Here, we only consider the two latter operations. For the construction of an automaton for the intersection, assume that we are given automata $A_i = (Q_i, \Sigma, \delta_i, F_i)$, $i = 1, 2$. W.l.o.g. we assume that $Q_1 \cap Q_2 = \emptyset$. We proceed analogously to the standard construction of the product automaton for ordinary automata. Thus, we define the automaton $A = (Q, \Sigma, \delta, F)$ as follows. We set $Q = Q_1 \times Q_2$ and $F = F_1 \times F_2$ and define $\delta(q_1 q_2, a)$ by the formula:

$$\exists_{p_1 \in Q_1} y_{p_1} . \exists_{p_2 \in Q_2} y_{p_2} . \delta_1(q_1, a) \wedge \delta_2(q_2, a) \quad \wedge$$

$$\left(\bigwedge_{p_1 \in Q_1} \sum_{p_2 \in Q_2} y_{p_1 p_2} = y_{p_1} \right) \wedge \left(\bigwedge_{p_2 \in Q_2} \sum_{p_1 \in Q_1} y_{p_1 p_2} = y_{p_2} \right)$$

Here, we used the auxiliary variables $y_{p_1}, p_1 \in Q_1$ and $y_{p_2}, p_2 \in Q_2$. Moreover, we introduced the general auxiliary notation:

$$\exists_{i \in I} y_i .$$

(I some index set) to denote the existential quantification over all variables $y_i, i \in I$. The intuition behind this formula is quite simple: the pair of states $q_1 q_2$ should be satisfiable iff each of the states q_1 and q_2 are. Thus, we construct the precondition for $q_1 q_2$ and a on sequences α from $(Q_1 \times Q_2)^*$ as the conjunction of the preconditions for q_1 and q_2 on the projections of α onto the first and second components, respectively. The frequency y_{p_1} of the state $p_1 \in Q_1$ in the first projection then is given by: $\sum_{p_2 \in Q_2} y_{p_1 p_2} = y_{p_1}$. Using the analogous formula also for y_{p_2}, $p_2 \in Q_2$, we arrive at the stated formula. Accordingly, it is easy to prove that

$$t \models_A q_1 q_2 \quad \text{iff} \quad t \models_{A_1} q_1 \text{ and } t \models_{A_2} q_2$$

Thus, $\mathcal{L}(A) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$ which completes the proof. □

In order to prove closure under complement, we introduce the notion of deterministic automata. A Presburger automaton $A = (Q, \Sigma, \delta, F)$ is called *deterministic* iff for every $a \in \Sigma$ and every multiset of states $m = \sum_{p \in Q} n_p \cdot \{p\}$, the following holds:

- $\{y_p \mapsto n_p \mid p \in Q\} \models \delta(q, a)$ for some state $q \in Q$;
- $\{y_p \mapsto n_p \mid p \in Q\} \models \delta(q_i, a)$ for $i = 1, 2$ implies $q_1 = q_2$.

We can construct for the automaton A a Presburger formula test_A such that A is deterministic iff test_A is satisfiable. Therefore we have:

PROPOSITION 1. *It is decidable whether or not a Presburger u-tree automaton is deterministic.* $\square$

Given a deterministic Presburger u-tree automaton $A = (Q, \Sigma, \delta, F)$, we can construct the complement automaton $\bar{A} = (Q, \Sigma, \delta, Q \backslash F)$ simply by exchanging the accepting and non-accepting states, and we have:

PROPOSITION 2. $\mathcal{L}(\bar{A}) = U_\Sigma \backslash \mathcal{L}(A)$.

Therefore, it remains to construct for every Presburger u-tree automaton an equivalent deterministic Presburger u-tree automaton.

THEOREM 3. *For every Presburger u-tree automaton $A = (Q, \Sigma, \delta, F)$, a deterministic Presburger u-tree automaton A' can be constructed such that $\mathcal{L}(A) = \mathcal{L}(A')$.*

PROOF. The proof idea is similar to related constructions for ordinary finite automata. Let $A' = (Q', \Sigma, \delta', F')$ where $Q' = 2^Q$ and $F' = \{B \subseteq Q \mid F \cap B \neq \emptyset\}$ where $\delta'(B, a)$ is a formula with free variables from $Y_{Q'}$. It is given by:

$$\left(\bigwedge_{q \in B} \psi_{q,a}\right) \wedge \left(\bigwedge_{q \in Q \backslash B} \neg\psi_{q,a}\right)$$

Here, the formula $\psi_{q,a}$ should be true iff q is a possible successor state. In order to specify $\psi_{q,a}$, we refer to the auxiliary variables $y_p, p \in Q$, and also to auxiliary variables $y_{B,p}, B \subseteq Q, p \in B$. The variable $y_{B,p}$ is meant to count all those children resulting in the state set B which are assigned to the state $p \in B$. Using these auxiliary variables, $\psi_{q,a}$ is defined as:

$$\exists_{p \in Q} y_p . \quad \delta(q, a) \wedge \exists_{p \in B \subseteq Q} y_{B,p} .$$
$$\left(\bigwedge_{B \subseteq Q} \sum_{p \in B} y_{B,p} = y_B\right) \wedge \left(\bigwedge_{p \in Q} \sum_{p \in B \subseteq Q} y_{B,p} = y_p\right)$$

$\square$

As corollaries of theorem 3, we also obtain:

COROLLARY 3. *Universality for Presburger u-tree automata is decidable.*

COROLLARY 4.
1. *Given a fixed deterministic Presburger u-tree automaton A and some state q of A, it is decidable in linear time for a u-tree t whether or not $t \models_A q$.*
2. *In particular, the word problem for Presburger u-tree automata is decidable in linear time.*

PROOF. Assume we are given a Presburger u-tree automaton A with transition function δ. By theorem 3, we can w.l.o.g. assume that A is deterministic. Also assume that all formulas ϕ in δ have been compiled into finite automata A_ϕ running on tuples of of (binary) representations of the numbers. Let $n > 0$ denote the size of the input u-tree t. By a bottom-up traversal over t, we can determine for every u-subtree t' of t and every state p of A whether or not $t' \models_A p$. In total, this amounts to $\mathcal{O}(n)$ tests of assertions $\rho \models \phi$ for formulas ϕ occurring in δ where $\rho(x) \leq n$ for all x in the domain of ρ. The binary representations of the $\rho(x)$ altogether have size at most $\mathcal{O}(n)$. Therefore, the automata A_ϕ can verify the assertions $\rho \models \phi$ in time $\mathcal{O}(n)$. We conclude that the overall complexity is $\mathcal{O}(n)$. $\square$

Querying Unordered Trees

Now we consider Presburger automata as a facility to compute unary queries, i.e., to select a set of nodes in an unordered tree. Whether a node is selected is specified by an automaton A and a set T of states of A. The node v is in the output, if there is an accepting computation of A that obtains a state from T at v. We will see in the next subsection that this simple mechanism can compute all (unary) queries definable in (Presburger) MSO logic.

Let $\bullet$ denote a fresh symbol (not in Σ). An *unordered context* (u-context for short) is a u-tree $c \in U_{\Sigma \cup \{\bullet\}}$ which contains exactly one occurrence of $\bullet$ at a leaf (the *hole*). Let $c[t_1]$ denote the u-tree which is obtained from c by substituting $\bullet$ with t_1 (i.e., filling the hole). Note that for a given u-tree t, the set $\mathcal{C}(t)$ of contexts c such that $t = c[t_1]$ for suitable u-subtrees t_1 is in one-to-one correspondence with the set of nodes of t. Therefore, in the following we will no longer distinguish between contexts $c \in \mathcal{C}(t)$ and nodes of t.

A *(unary) query* is a mapping μ from u-trees to subsets of nodes. The nodes in $\mu(t)$ are also called *matches*. In the following, we present a class of queries which is definable by means of Presburger u-tree automata. For this, we extend the definition of $\models_A$ to contexts by defining $c, p \models_A q$, $(p, q \in Q)$ iff $c \models_{A_{p,\bullet}} q$ where $A_{p,\bullet} = (Q, \Sigma \cup \{\bullet\}, \delta_{p,\bullet}, F)$ is obtained from A by extending Σ with $\bullet$ and defining:

$$\delta_{p,\bullet}(q', a) = \begin{cases} \delta(q', a) & \text{if } a \in \Sigma \\ \bigwedge_{p' \in Q} y_{p'} = 0 & \text{if } a = \bullet \wedge q' = p \\ \mathsf{false} & \text{if } a = \bullet \wedge q' \neq p \end{cases}$$

Thus, the automaton $A_{p,\bullet}$ behaves like A but additionally lets the hole satisfy p. Obviously, we have:

PROPOSITION 5. *Let $A = (Q, \Sigma, \delta, F)$ be a Presburger u-tree automaton and $t = c[t_1]$ for a context c and $t, t_1 \in U_\Sigma$. Then $t \models_A q$ iff $t_1 \models_A p$ and $c, p \models_A q$ for some $p \in Q$.* $\square$

A (unary) *Presburger* pattern is a property of u-subtrees within u-trees. We define this property by means of a pair $\langle A, T\rangle$ where $A = (Q, \Sigma, \delta, F)$ is a Presburger u-tree automaton and $T \subseteq Q$ is a set of states. Let $t \in U_\Sigma$. A u-context $c \in \mathcal{C}(t)$ is a *match* of the pattern $\langle A, T\rangle$ in t iff $t = c[t_1]$ where $t_1 \models_A q$ and $c, q \models_A f$ for some $q \in T$ and $f \in F$. We get:

THEOREM 4. *The set of matches of a fixed Presburger pattern $\langle A, T\rangle$ in a u-tree $t \in U_\Sigma$ of size n is computable in time $\mathcal{O}(n)$.*

PROOF. Let $A = (Q, \Sigma, \delta, F)$. We proceed in two passes over the input u-tree t of size $n > 0$. In the first pass, we determine for every u-subtree t_1 of t the set:

$$B(t_1) = \{p \in Q \mid t_1 \models_A p\}$$

Let A' denote the deterministic automaton as constructed in the proof of theorem 3. Then we know that for every $t' \in U_\Sigma$, $t' \models_{A'} B$ iff $B = \{p \in Q \mid t' \models_A p\}$. Therefore, the sets $B(t_1)$ can be determined by one bottom-up run of A' on t. According to corollary 4, this first pass can be performed in time $\mathcal{O}(n)$.

In the second pass, we determine for each u-context $c \in \mathcal{C}(t)$, the set:

$$D(c) = \{p \in B(t_1) \mid \exists f \in F : c, p \models_A f\}$$

where t_1 is the sub-tree of t with $t = c[t_1]$. Given the sets $D(c)$, the matches of the pattern are determined as the set of all u-contexts c where $T \cap D(c) \neq \emptyset$.

In order to determine the sets $D(c)$, we proceed topdown over t. Assume that we are given a u-context c in t where $t = c[aS]$ for some $a \in \Sigma$ and multi-set S containing a u-subtree t_1, i.e., $S = \{t_1\} + S'$. Then we may proceed from the father node c to the son c_1 which is defined as the context $c_1 = c[a(\{\bullet\}+S')]$. Remark that now $t = c_1[t_1]$. Let $B_1 = B(t_1)$. Assume that we have already determined the value $D(c)$ and now want to determine the corresponding set for c_1. For $B \subseteq Q$, let n_B denote the number of u-trees $t' \in S$ such that $t' \models_{A'} B$. Let ρ denote the variable environment defined by:

$$\{y_B \mapsto n_B \mid B \subseteq Q\}$$

We claim:

$$D(c_1) = \{q_1 \in B(t_1) \mid \rho \models \bigvee_{q \in D(c)} \psi_{q,q_1}\}$$

where the formula ψ_{q,q_1} is given by:

$$\exists_{p \in Q} y_p \,.\, \delta(q,a) \wedge \exists_{p \in B \subseteq Q} y_{B,p} \,.\; y_{B_1,q_1} > 0 \wedge$$
$$\left(\bigwedge_{B \subseteq Q} \sum_{p \in B} y_{B,p} = y_B \right) \wedge \left(\bigwedge_{p \in Q} \sum_{B, p \in B} y_{B,p} = y_p \right)$$

Intuitively, formula ψ_{q,q_1} expresses that there is an assignment mapping the children t' to states $q \in B(t')$ such that t_1 receives q_1 and the pre-condition $\delta(q,a)$ is satisfied. Since satisfiability of Presburger formulas is decidable, we conclude that the sets $D(c)$ are computable. In total, our algorithm amounts to $\mathcal{O}(n)$ tests of assertions $\rho \models \phi$ for formulas ϕ which only depend on the automaton A and variable environments ρ where $\rho(x) \leq n$ for all x in the domain of ρ. As in the proof of corollary 4, we compile the formulas ϕ into finite automata A_ϕ running on the tuple of binary representations of the numbers $\rho(x)$. Note that we can cluster together all subtrees t_1 of c which agree in their sets $B(t_1)$ of reachable states. We conclude that the total length of the numbers $\rho(x)$ can be bounded by $\mathcal{O}(n)$. Therefore, the overall complexity of the second pass is linear as well. This completes the proof. $\square$

Presburger MSO Logic for Unordered Trees

We define unordered Presburger MSO (PMSO) logic by extending MSO logic with Presburger predicates on children. As we consider unordered trees only, the logic does not provide an ordering relation on brothers. More precisely, a PMSO formula f is given by the following grammar:

$$\begin{array}{lll}
f & ::= & y < y' \mid y \in S \mid y/p \\
 & & \mid f_1 \wedge f_2 \mid \neg f \mid \exists y.\, f \mid \exists Y.\, f \\
S & ::= & Y \mid \mathsf{Lab}_a \\
p & ::= & t_1 = t_2 \mid t_1 + t_2 = t_3 \\
 & & \mid p_1 \wedge p_2 \mid \neg p \mid \exists x.\, p \\
t & ::= & [S] \mid x \mid n
\end{array}$$

where $y < y'$ expresses that y is the father of y', x is from a designated set of Presburger variables, and the formulas p of y/p are Presburger-closed, i.e., do not contain free occurrences of variables x. Intuitively, the assertion y/p means that the children of y satisfy the constraint p where a term $[S]$ inside p is interpreted as the number of those children which are contained in S.

As usual, we also allow derived predicates such as equality between variables such as $y_1 = y_2$ or $Y_1 = Y_2$ or equations $Y = \{y_1\}$.

Remark 1. *The MSO predicate $y < y_1$ is expressible by means of the new form of atomic predicates, namely by:*

$$\exists\, Y.\, Y = \{y_1\} \;\wedge\; y/([Y] = 1)$$

In order to define a satisfiability relation "$\models$", we view a tree $t \in U_\Sigma$ as the structure $t = \langle D, (\mathsf{Lab}_a)_{a \in \Sigma}, < \rangle$ where D is the underlying set of nodes, Lab_a is the subset of nodes labeled with a, and "$<$" is the father relation. The satisfaction relation

$$t, \rho, \sigma \models \phi$$

for the structure t together with valuations ρ, σ (for the sets of free set variables and free individuum variables, respectively) is inductively defined as follows:

$$\begin{array}{lll}
t,\rho,\sigma \models y < y' & \text{iff} & \sigma(y) < \sigma(y') \text{ holds in } t \\
t,\rho,\sigma \models y \in \mathsf{Lab}_a & \text{iff} & \sigma(y) \in \mathsf{Lab}_a \text{ holds in } t \\
t,\rho,\sigma \models y \in Y & \text{iff} & \sigma(y) \in \rho(Y) \\
t,\rho,\sigma \models y/p & \text{iff} & m \models p \text{ where} \\
 & m\,[\mathsf{Lab}_a] & = \; \#\{v \in D \mid \sigma(y) < v, v \in \mathsf{Lab}_a\} \\
 & m\,[Y] & = \; \#\{v \in D \mid \sigma(y) < v, v \in \rho(Y)\} \\
t,\rho,\sigma \models f_1 \wedge f_2 & \text{iff} & t,\rho,\sigma \models f_1 \text{ and } t,\rho,\sigma \models f_2 \\
t,\rho,\sigma \models \neg f & \text{iff} & t,\rho,\sigma \not\models f \\
t,\rho,\sigma \models \exists\, y.\, f & \text{iff} & t,\rho,\sigma \oplus \{y \mapsto v\} \models f \text{ for some } v \in D \\
t,\rho,\sigma \models \exists\, Y.\, f & \text{iff} & t,\rho \oplus \{Y \mapsto R\},\sigma \models f \text{ for some } R \subseteq D
\end{array}$$

If the formula f is closed, we also write: $t \models f$ instead of $t, \emptyset, \emptyset \models f$.

A language $L \subseteq U_\Sigma$ is unordered PMSO-definable iff there is a closed formula ϕ such that $L = \{t \mid t \models \phi\}$.

Theorem 5 states that unordered PMSO-definable languages are precisely characterized by Presburger u-tree automata.

Theorem 5. *For a language $L \subseteq U_\Sigma$ the following two statements are equivalent:*

1. *L is unordered PMSO-definable;*
2. *$L = \mathcal{L}(A)$ for some Presburger u-tree automaton A.* $\square$

The proof is analogous to the proof of the corresponding result for MSO-logic and tree automata over ordered trees. [3] Let us turn to the characterization of queries. An unordered PMSO-pattern is an unordered PMSO formula ϕ with at most one free variable y. A match of ϕ in t is given by a node v such that

$$t, \emptyset, \{y \mapsto v\} \models \phi$$

A query μ is unordered PMSO-definable iff there is an unordered PMSO-pattern ϕ such that for every t, $\mu(t)$ is the set of all matches of ϕ in t.

Theorem 6. *For a query μ the following two statements are equivalent:*

1. *μ is unordered PMSO-definable;*
2. *μ is definable by a Presburger pattern $\langle A, T\rangle$ for some Presburger u-tree automaton A.* $\square$

3. EXTENSION TO ORDERED TREES

In many applications, e.g., where documents are automatically generated from databases as textual representations of querying results, the element ordering on the children does not matter. In other applications, though, which are more related to classical document processing the ordering matters, and we envision applications where document trees contain both ordered and unordered regions. In this section, we therefore extend our framework to ordered trees.

Obviously, every ordered sequence $\langle t_1, \ldots, t_n \rangle$ can be considered as (one representation of) the unordered sequence $\{t_1, \ldots, t_n\}$ consisting of the same elements. In particular, every ordered tree can be considered as (one representation of) an unordered tree as well. Accordingly, Presburger MSO logic as defined for unordered trees in section 2 is readily extended to ordered trees by adding the atomic predicate "y is *left sibling* of y'" and correspondingly adjusting the definition of the satisfaction relation $\models$. Let us therefore see in how far Presburger u-tree automata can also be extended to ordered trees (which we simply call *trees* for short). As the children of a node form an ordered sequence, we now need a more general pre-condition than just a Presburger formula for the number of occurrences of states. Let Q be an alphabet (of states). A *Presburger regular expression* over Q is a Boolean combination of regular expressions over Q and Presburger formulas having free variables only from the canonical set Y_Q.

Given a string w and a Presburger regular expression ϕ we define in the obvious way whether $w \in Q^*$ matches ϕ (i.e., $w \models \phi$). So, if ϕ equals, e.g.,

$$p\,(p \mid q)^* \wedge (y_p = y_q)$$

then $w \models \phi$ iff w contains only p's and q's, begins with a p and contains equally many occurrences of p and q.

LEMMA 6. *It is decidable whether for a Presburger regular expression ϕ there is a string w such that $w \models \phi$.*

PROOF. Let ϕ be a Presburger regular expression. First of all, ϕ can be transformed into disjunctive normal form $\bigvee_i \bigwedge_j \phi_{ij}$, where each ϕ_{ij} is either a regular expression or a Presburger formula. By combining regular expressions and Presburger formulas, respectively, we arrive at an expression of the form $\bigvee_i (e_i \wedge \pi_i)$, where each e_i is a regular expression and each π_i is a Presburger formula. Clearly, ϕ is satisfiable by a string if and only if at least one disjunct $e_i \wedge \pi_i$ is. To check satisfiability for the conjunction of a regular expression e and a Presburger formula π we compute a Presburger formula π_e which describes the Parikh images of words in $L(e)$, the language defined by e. Here, the *Parikh image* of a word w over some alphabet Q assigns the number of occurrences of q in w to each $q \in Q$ [27]. An expression $e \wedge \pi$ is satisfiable if and only if the Presburger formula $\pi_e \wedge \pi$ is satisfiable and we are done. □

The complexity of the described decision procedure crucially depends on the complexity of constructing the formula π_e. Constructions based on semi-linear sets may produce rather long formulas [10, 11, 27]. It is an interesting question whether efficient constructions are possible. As Presburger regular expressions are closed under negation we immediately conclude that also universality is decidable, i.e., whether an expression matches all strings.

Lemma 6 encourages us to generalize the notion of Presburger tree automata. Thus, we define a *Presburger tree automaton* for ordered trees as a tuple $A = (Q, \Sigma, \delta, F)$ where:

- Q is a finite set of states;
- $F \subseteq Q$ is the subset of accepting states;
- δ maps pairs (q, a) of states and labels from Σ to Presburger regular expressions ϕ over Q.

Accordingly, we introduce an extended satisfaction relation between ordered trees t and states q by defining for $t = a\langle t_1 \ldots t_l \rangle$ and $\delta(q, a) = \phi$, $t \models_A q$ iff there are states $p_1, \ldots, p_l \in Q$ such that $t_j \models_A p_j$ for all t_j and $p_1 \ldots p_l \models \phi$. The language $\mathcal{L}(A) \subseteq T_\Sigma$ which is *accepted* by the automaton A then is given by:

$$\mathcal{L}(A) = \{t \in T_\Sigma \mid \exists\, f \in F : t \models_A f\}$$

We obtain:

THEOREM 7. *Emptiness for Presburger tree automata is decidable.*

PROOF. Follows almost immediately from lemma 6. □

Next we show that membership and hence, also querying for a fixed Presburger tree automaton can be solved in polynomial time.

THEOREM 8. *Given a fixed Presburger tree automaton A and some state q of A, it is decidable in polynomial time for a tree t whether or not $t \models_A q$.*

PROOF. Let Q be the states set of A. We perform a bottom-up traversal of the input tree t, computing for each subtree t' the set of states $R = \{p \mid t' \models p\} \subseteq Q$. Assume that $t' = a\langle t_1 \ldots t_n \rangle$ and $R_i = \{p \mid t_i \models p\}$ have been already computed. Moreover, we can suppose that the Presburger regular expressions used in A are disjunctions of conjuncts $e_i \wedge \pi_i$ where each e_i is a regular expression and π_i is a Presburger formula. Then we may check for each $e_i \wedge \pi_i$ separately whether it is verified by $t_1 \ldots t_n$. So now consider one conjunct $e \wedge \pi$. With e, we associate an equivalent finite automaton B with set of states P. Then we successively compute the sets $V(i, s)$, $1 \leq i \leq n, s \in P$, of assignments $v : Y_Q \to \{1, \ldots, i\}$ verifying the following condition: there exists a sequence of states $\alpha = r_1 \cdots r_i$ with $r_k \in R_k$ for $k = 1, \ldots, i$ and with Parikh image v, such that state s can be reached from an initial state of B by reading α. Finally, we consider the union V of all sets $V(n, f)$ where f is a final state of B. Now it simply remains to check whether $v \models \pi$ for any $v \in V$. Thus, assuming that the automaton A is of constant size, we spend time $\mathcal{O}(n^{|Q|+1})$ on the computation of the set of all successor states at the root node of t_1. Hence, the overall runtime on a tree of size m is $\mathcal{O}(m^{|Q|+1})$. □

Clearly, the upper complexity bound in theorem 8 is not as encouraging as one might have wished. It remains an interesting question, though, whether practical examples indeed exhibit this worst-case behavior. A generally better upper bound, however, can be obtained for *deterministic* automata where membership is decidable in linear time [17]. Generalizing the notion of a u-context from section 2, we define a *context* as a tree $c \in T_{\Sigma \cup \{\bullet\}}$ that contains exactly one occurrence of the fresh symbol $\bullet$. Accordingly, we generalize the definition of $\models_A$ to contexts by defining $c, p \models_A q$, $(p, q \in Q)$, iff $c \models_{A_{p,\bullet}} q$ where $A_{p,\bullet} = (Q, \Sigma \cup \{\bullet\}, \delta_{p,\bullet}, F)$ is the automaton obtained from A by extending Σ with $\bullet$ and extending δ to $\delta_{p,\bullet}$ where $\delta_{p,\bullet}(q', a) = \delta(q', a)$ whenever $a \in \Sigma$ and

$$\delta_{p,\bullet}(q', \bullet) = \begin{cases} \epsilon & \text{if } q' = p \\ \emptyset & \text{if } q' \neq p \end{cases}$$

A (monadic) *Presburger* pattern now is a property of subtrees within trees. We define this property by means of a pair $\langle A, T\rangle$ where $A = (Q, \Sigma, \delta, F)$ is a Presburger tree automaton and $T \subseteq Q$ is a set of states. Let $t \in T_\Sigma$. A context $c \in \mathcal{C}(t)$ is a *match* of the pattern $\langle A, T\rangle$ in t iff $t = c[t_1]$ where $t_1 \models_A q$ and $c, q \models f$ for some $q \in T$ and $f \in F$. We have:

THEOREM 9. *The set of matches of a fixed Presburger pattern $\langle A, T\rangle$ in a tree $t \in T_\Sigma$ of size n is computable in polynomial time.*

PROOF. Assume we have marked the root node of one subtree t_1 of t. Assume further that we have modified A in such a way that the marked node always receives a state in T. Then the modified tree is accepted iff t_1 is a match. Since there are only n different nodes to be marked, the theorem follows from theorem 8. □

We now characterize Presburger tree automata by means of a fragment of ordered Presburger MSO logic.

THEOREM 10. *A set of ordered trees is accepted by a Presburger tree automaton if and only if it can be described by an ordered PMSO formula of the form $\exists X_1 \ldots \exists X_k.\, \varphi$ where φ is first-order.*

PROOF. *only-if:* Let A be a Presburger tree automaton and δ the transition relation of A. Without loss of generality we can assume that all Presburger regular expressions used in A are disjunctions of expressions $e \wedge \pi$, where e is a regular expression and π is a Presburger formula. From Büchi's Theorem it follows that each regular expression e can be expressed by an existential MSO formula $\psi_e = \exists Y_1 \ldots \exists Y_l.\, \varphi_e$ (on strings). Hence, we can construct a formula $\psi = \exists X_1 \ldots X_k.\, \varphi$ in which some of the variables X_i are used to encode the states that A assumes and the remaining variables are those of the formulas ψ_e. The first-order part φ of ψ describes the consistency of the states between nodes of the input tree and their children by using the formulas φ_e.

if: We show first that every first-order ordered PMSO formula ψ can be evaluated by a deterministic Presburger tree automaton. The result is then immediate as a non-deterministic automaton can guess, for each node, those sets of $X_1, \ldots, X_k$ in which the node is contained. The proof proceeds by induction on the structure of ψ. The only case which is not entirely straightforward is the case of a formula $\psi = \exists x.\, \varphi(x)$. Let, by induction, A be an automaton over the alphabet $\Sigma \cup (\Sigma \times \{x\})$ for $\varphi(x)$. I.e., A accepts all trees t which have exactly one node v with a symbol (a, x) from $\Sigma \times \{x\}$ such that φ holds on t, if x is bound to v and the label of v is replaced by a.

Let Q be the set of states of A. We construct a deterministic Presburger tree automaton A_1 for ψ as follows. The state set of A_1 is $Q \times 2^Q$. The intuitive meaning of a state (q, X) at a node v is the following. If x did not occur in the subtree rooted at v, then A would take state q at v. X is the set of states A can take if for one node of the subtree at v a label a is replaced by (a, x). We explain how the mappings $\delta_1((q, X), a)$ of A_1 are defined. $\delta_1((q, X), a)$ is described by a Presburger regular expression $e_q \wedge e_X$, where e_q is obtained from $\delta(q, a)$ by replacing each occurrence of a state $r \in Q$ in a regular expression by $\bigcup_{S \subseteq Q}(r, S)$ and each occurrence of y_r in a Presburger formula by $\sum_{S \subseteq Q} y_{(r,S)}$. The Presburger regular expression e_X is of the form $\bigwedge_{p \in X}(e_p^1 \vee e_p^2) \wedge \bigwedge_{p \notin X} \neg(e_p^1 \vee e_p^2)$. Here, e_p^1 expresses that A would take state p at v if the label of v was (a, x). Likewise, e_p^2 expresses that A would take state p at v if the label b of a suitable node below v was replaced by (b, x). Therefore, e_p^1 is obtained from $\delta(p, (a, x))$ in an analogous fashion as e_q was obtained from $\delta(q, a)$.

It remains to describe the construction of e_p^2. Let $\delta(p, a)$ be a disjunction of conjuncts of the form $e \wedge \pi$ where e is a regular expression and π is a Presburger formula. The expression e_p^2 is obtained by replacing each $e \wedge \pi$ with a disjunction $\bigvee_{r \in Q} \bigvee_{r' \in S \subseteq Q} (e_{r,r',S} \wedge \pi_{r,r',S})$. Here, for each choice of $S \subseteq Q$, $r \in Q$ and $r' \in S$, the expression $e_{r,r',S} \wedge \pi_{r,r',S}$ is satisfied by a sequence $(q_1, S_1) \cdots (q_m, S_m)$, $q_i \in Q$, $S_i \subseteq Q$, if there is some $i \leq m$ with $q_i = r$, $S_i = S$ and $e \wedge \pi$ holds for the string $q_1 \cdots q_{i-1} r' q_{i+1} \cdots q_m$.

We get $\pi_{r,r',S}$ as the conjunction of $(y_{(r,S)} > 0)$ and the formula which is obtained from π by replacing y_q, for each $q \in Q$ with

- $\sum_{S' \subseteq Q} y_{(q,S')}$, if $q \notin \{r, r'\}$ or $q = r = r'$,
- $(\sum_{S' \subseteq Q} y_{(q,S')}) - 1$, if $q = r$ and $r \neq r'$, and
- $(\sum_{S' \subseteq Q} y_{(q,S')}) + 1$, if $q = r'$ and $r \neq r'$.

The language to be described by $e_{r,r',S}$ is given as:

$$L = \{(q_1, S_1) \ldots (q_m, S_m) \mid \exists\, i\ :\ (q_i, S_i) = (r, S) \wedge q_1 \ldots q_{i-1} r' q_{i+1} \ldots q_n \in L(e)\}$$

Clearly, if $L(e)$ is regular, then L is regular as well and hence can be described by a regular expression. □

Theorem 10 is interesting in its own right. It implies that we can answer all *existential* ordered Presburger MSO queries with polynomial data complexity. In particular, all monadic *first-order* queries can be answered. On the other hand, it turns out that in general it is impossible to tell whether a given (non-deterministic) Presburger tree automaton accepts *all* ordered trees.

THEOREM 11. *Universality for Presburger tree automata is undecidable.*

PROOF. The proof is a reduction from the Halting Problem for 2-counter-automata with empty input [19]. Given such an automaton A with state set Q we construct a Presburger tree automaton A_1 such that A does not halt on the empty input if and only if A_1 accepts all trees over the alphabet $Q \cup \{\#, \$, a, b\}$. In the construction we will concentrate on trees of a special shape. They are of depth 2 and nodes on the first level (between the root and the leaves) have exactly one child. I.e., the trees are balanced and only the root might have more than one child. The pattern of the labels of the leaves, from left to right, is of the form $r_Q a^* b^* (\# r_Q a^* b^*)^*$, where $r_Q = q_0 | \cdots | q_k$ represents the set of states of A. The root is labeled with \$. All nodes of level 1 are labeled by #. It is easy to construct a Presburger tree automaton[2] A_2 which accepts all trees that are *not* of this special form. The union of A_2 with the automaton A_3 to be constructed in the remainder of the proof will be the automaton A_1 we are looking for.

A_3 checks whether the leaf string of the input tree does *not* encode an accepting computation of the counter automaton A. Here, a configuration of A with state q and counter contents n_1 and n_2, respectively, is encoded by the string $qa^{n_1}b^{n_2}$ and configurations are separated by #. A_3 checks whether

- this string does not start with $q_0\#$, where q_0 is the initial state of A,
- this string does not end by a string of the form $\#qa^*b^*$, where q is an accepting state of A, or
- there are two successive configurations that are not consistent with the transition function of A.

We only describe how the latter can be checked, as the first two tasks are straightforward. To this end, the state set of A_3 equals $Q \cup \{q_\#, q_a, q'_a, q_b, q'_b, q_?\}$. In the first stage, when moving from the leaves to nodes of level 1 the automaton can enter state $q_?$ from each state on the leaves. Further, it can enter state $q_\#$ from all leaves labeled # and state q, for each leaf with label $q \in Q$. For leaves with label a it enters state q_a or q'_a. Accordingly, it may enter q_b or q'_b from b.

It enters an accepting state if

- the string of states of the nodes of the first level is of the form $q_?^* q_\# q q_a^* q_b^* q_\# q' q_{a'}^* q_{b'}^* q_\# q_?^*$ with $q, q' \in Q$, and
- the numbers of the states q_a, q_b, q'_a, q'_b are not consistent with respect to q, q' and the transition function of A.

This can be expressed by a disjunction over all possible pairs $(q, q') \in Q \times Q$. Each disjunct consists of a regular expression expressing the first condition and a Presburger formula for the latter condition. It should be clear that the automaton constructed in this way has the desired properties. □

By Theorem 10, the language of ordered trees defined by a Presburger tree automaton is definable by an ordered PMSO formula – and so is its complement, since ordered PMSO logic is trivially closed under complementation. Therefore, we immediately obtain from Theorem 11:

PROPOSITION 7. *Satisfiability for ordered PMSO formulas is undecidable.* □

For a comparison, we note that the situation here dramatically differs from the unordered case. Since Presburger u-tree automata are effectively closed under complement, their universality problem is decidable.

[2]Actually, A_2 does not use any Presburger formulas.

4. MIXED TREES

In the previous section we have seen that in general we cannot expect decidability for all ordered PMSO. Instead, we restrict ourselves to *mixed* ordered/unordered document trees In these trees, the label of a node tells whether the ordering of its children matters or not. Recall from the introduction that this restriction naturally reflects a division of documents into parts which are made up from data records whose orderings are irrelevant and formating parts where the ordering is significant. This classification is formalized by partitioning the finite alphabet Σ into subsets $\Sigma = \Sigma_0 + \Sigma_1$ where Σ_0 and Σ_1 consist of all labels of nodes with unordered and ordered children, respectively. The set M_Σ of all mixed trees (*m-trees* for short) over Σ is now given by the grammar:

$$\begin{array}{lcll} t & ::= & a\{t_1, \ldots, t_k\} & (a \in \Sigma_0, k \geq 0) \\ & | & b\langle t_1, \ldots, t_k \rangle & (b \in \Sigma_1, k \geq 0) \end{array}$$

Mixed trees in our sense correspond to terms with one associative symbol "." (for accumulating the ordered contents) and one associative and commutative symbol "⊕" (for accumulating multi-sets). Languages of such trees, e.g., have been studied Lugiez [15, 16] and Ohsaki [25, 26]. Note, however, that our formalism is slightly more specific as we rule out sequences of trees where unordered sections occur dispersed between ordered ones. Instead, the significance of order is already determined by the label of the ancestor.

Presburger tree automata for mixed trees now should subsume the ability of Presburger automata for unordered trees to check Presburger formulas on unordered sequences of children as well as the ability of automata for ordered trees to check containment in a regular set for ordered sequences. Thus, we use Presburger conditions in transitions for labels from Σ_0 and regular expressions in transitions for labels from Σ_1 only. We call such an automaton *Presburger m-tree automaton.* As a corollary of theorem 7, we obtain:

COROLLARY 8. *Emptiness for Presburger m-tree automata is decidable.* □

It turns out that the family of languages accepted by Presburger m-tree automata enjoys much better closure properties than Presburger automata for ordered trees. In fact, they are not only closed under union and intersection, but also under complement.

THEOREM 12. *The family of languages accepted by Presburger m-tree automata is effectively closed under:*

1. *union,*
2. *intersection,*
3. *complementation.* □

The reason for closure under complement is that for Presburger m-tree automata, we again have available a determinization procedure:

THEOREM 13. *For every Presburger m-tree automaton A, a deterministic Presburger m-tree automaton A' can be constructed such that $\mathcal{L}(A) = \mathcal{L}(A')$.* □

We call a pair of a Presburger m-tree automaton A and a subset T of states of A *mixed* Presburger pattern. We obtain:

THEOREM 14. *The set of matches of a fixed mixed Presburger pattern $\langle A, T \rangle$ in a m-tree $t \in M_\Sigma$ of size n is computable in time $\mathcal{O}(n)$.*

The stated complexity is better than the corresponding complexity for general Presburger patterns where the exponent of the upper bound depended on the number of states of the automaton. Here, the upper bound is linear.

PROOF. The base idea to achieve this result is the same as in the *unordered* case. Let $A = (Q, \Sigma, \delta, F)$. We proceed in two passes over the input t of size $n > 0$. In the first pass, we determine for every m-subtree t_1 of t the set:

$$B(t_1) = \{p \in Q \mid t_1 \models_A p\}$$

Let A' denote the deterministic automaton as constructed in section 2. Then we know that for every $t' \in M_\Sigma$, $t \models_{A'} B$ iff $B = \{p \in Q \mid t' \models_A p\}$. Therefore, the sets $B(t_1)$ can be determined by one bottom-up run of A' on t. According to the proof of corollary 4, this can be performed in time $\mathcal{O}(n)$.

In the second pass, we determine for each m-context $c \in \mathcal{C}(t)$, the set:

$$D(c) = \{p \in B(t_1) \mid \exists f \in F : c, p \models_A f\}$$

where t_1 is the sub-tree of t with $t = c[t_1]$. Given the sets $D(c)$, the matches of the pattern are determined as the set of all m-contexts c where $T \cap D(c) \neq \emptyset$.

In order to determine the sets $D(c)$, we again proceed topdown over t. The crucial new construction which we have to provide is for processing nodes with labels from Σ_1. So assume that we are given a m-context c in t where $t = c[a\langle t_1, \ldots, t_k \rangle]$ for some $a \in \Sigma_1$ and m-trees t_i. Then we may proceed from the father node c to the son c_i which is defined as the m-context $c_i = c[a\langle t_1, \ldots, t_{i-1}, \bullet, t_{i+1}, \ldots, t_k \rangle]$. Let $B_j = B(t_j)$ for $j = 1, \ldots, k$. Assume that we have already determined the value $D(c)$ and now want to determine the corresponding set for c_i. Obviously, we have:

$$\begin{array}{rcll} D(c_i) & = & \bigcup\{D_q(i) \mid q \in D(c)\} & \text{where} \\ D_q(i) & = & \{p_i \in B_i \mid \forall j \neq i\ \exists p_j \in B_j : p_1 \ldots p_k \in \delta_1(q, a)\} & \end{array}$$

Given a (non-deterministic) finite automaton $A_{q,a}$ for $\delta_1(q, a)$, all sets $D_q(i)$, $i = 1, \ldots, k$, can be computed by one left-to-right and one right-to-left pass of $A_{q,a}$ over the children $c_1, \ldots, c_k$ of the current node c. We conclude that all sets $D(c)$ are computable. Given a fixed pattern, we furthermore conclude that the nodes with ordered children incur computational cost $\mathcal{O}(n)$ only. In total, our algorithm has time complexity $\mathcal{O}(n)$. This completes the proof. □

Note that, as a special case of the querying algorithm in the proof of theorem 14, we obtain a linear time querying algorithm for *classical* ordered trees (i.e., trees with $\Sigma_0 = \emptyset$).

As for unordered and ordered trees, respectively, we succeed to give a logical characterization of our automata model also in the mixed case. For that, we use ordered PMSO logic. Now, however, formulas are interpreted over mixed trees only. In particular, we assume that Presburger constraints only can be applied to the children of a node labeled with some element from Σ_0. For a distinction, we therefore speak now of *mixed* PMSO-definable languages and queries (instead of ordered PMSO-definable ones over mixed trees). These mixed PMSO-definable queries are what we have considered in the introduction. We obtain:

THEOREM 15. *For a language $L \subseteq M_\Sigma$ the following two statements are equivalent:*

1. *L is mixed PMSO-definable;*
2. *$L = \mathcal{L}(A)$ for some Presburger m-tree automaton A.* □

THEOREM 16. *For a query μ the following two statements are equivalent:*

1. *μ is mixed PMSO-definable;*
2. *μ is definable by a Presburger pattern $\langle A, T \rangle$ where A is a Presburger m-tree automaton.* □

In particular, we conclude that satisfiability of mixed PMSO-logic is decidable. Even more important, mixed PMSO-queries can effectively be computed over mixed trees where the data complexity is linear.

5. CONCLUSION AND FURTHER EXTENSIONS

In summary, we have shown how Monadic Second Order logic can be extended by Presburger constraints on children. This extension allows to formulate properties of subdocuments depending on numerical properties like those in our toy example in Figure 1. Based on automata characterizations, we derived algorithms which compute the set of matches of a given monadic query in polynomial time. In fact, for unordered or mixed trees we even obtained linear querying algorithms.

But of course, besides reasoning about the numbers of children with certain properties it is also interesting to compute with the numbers that are *explicitly* mentioned in the document. So, a user might be interested to reason about *how much* music she gets for a certain amount of money or about the total quantity of music she has downloaded. In this section, we explain how our techniques can be extended to handle such queries as well.

Let us first consider a query which asks for all entities of jazz music where the price per minute is less than 10 (Cent). We assume that the numbers within the $\langle\texttt{time}\rangle$ tags represent the number of seconds of a piece. Hence, we are interested in all entries where the number in the $\langle\texttt{time}\rangle$ tag exceeds six times the number in the $\langle\texttt{price}\rangle$ tag. In an extended logic we express this as follows.

$$x \in \mathsf{Lab}_{\mathtt{jazz}} \wedge x/\phi$$

where:

$$\phi \quad \equiv \quad [\texttt{time}] > 6 \cdot [\texttt{price}]$$

As the example shows, we refer to the number within a tag $\langle\texttt{time}\rangle$ by $[\texttt{time}]$. In principle, such queries can be handled in a similar way as the queries we have considered so far. Concerning the semantics we can simply view a tag $\langle\texttt{time}\rangle$ with number n as n occurrences of a tag $\langle\texttt{\#time}\rangle$. It is straightforward that all the decidability results go through.

The approach of viewing numbers as multiplicities works fine for naturals (greater than 0). We can generalize it, though, to arbitrary integer multiplicities of leaf nodes (which are not queried themselves). Since satisfiability of Presburger formulas is also decidable over $\mathbb{Z}$, the same techniques can be applied.

One might suspect that dealing with numbers in decimal representation as opposed to the unary representation of multiplicities by tags results in an exponential blow-up for query evaluation. Fortunately this is not the case. Recall that in the proof of Theorem 4 we evaluate Presburger formulas by finite automata operating on the *binary representations* of multiplicities. Therefore, query evaluation is still possible in linear time.

Now let us consider a second example query. Assume that by some transformation process from the initial document of Figure 1 a document is created, as depicted in Figure 4 which, for each user and each music style, consists of a list of the ⟨time⟩- and ⟨price⟩-tags.

```
<user>
 ...
 <jazz>
  <time> 3310 </time>
  <price> 240 </price>
  <time> 220 </time>
  <price> 20 </price>
  <time> 247 </time>
  <price> 16 </price>
  <time> 3510 </time>
  <price> 262 </price>
  <time> 3325 </time>
  <price> 220 </price>
 </jazz>
 <pop>
  ...
 </pop>
 ...
</user>
```

Figure 4: A transformed document containing information about time and price of music files downloaded by a user.

An obvious question is how queries referring to numbers can be formulated against this document. The previous approach suggests to interpret the expression [time] as the *sum* of numbers within tags ⟨time⟩. For instance, the set of all users who downloaded at least 2 hours of jazz music could be formulated as follows.

$$x \in \mathsf{Lab}_{\mathtt{user}} \wedge \exists y.\, x < y \wedge y \in \mathsf{Lab}_{\mathtt{jazz}} \wedge y/\phi$$

where:

$$\phi \quad \equiv \quad [\mathtt{time}] \geq 7200$$

By replacing ϕ with $[\mathtt{time}] > 6 \cdot [\mathtt{price}]$ we could as well select all users who consumed jazz music with an overall price of less than 10 Cent per minute.

Of course in a practical setting, it would be interesting to allow *aggregate* functions (besides summing of values) such as `min`, `max` or `average`. It is straightforward to incorporate these additional features into Presburger formulas of automata. Concerning query evaluation the additional effort then only depends on the cost of the computation of the aggregate functions. Determinization is still possible in Presburger u-tree and m-tree automata. Whether Emptiness or Universality remain decidable depends on the specific properties of the aggregate functions.

A closer investigation of the properties of aggregate functions with respect to the questions we considered here remains to be carried out in future work.

6. REFERENCES

[1] A. Berlea and H. Seidl. Binary Queries. In *Extreme Markup Languages 2002 Conference, Montreal*, August 2002.

[2] L. Berman. The Complexity of Logical Theories. *Theoretical Computer Science (TCS)*, 11:71–77, 1980.

[3] J.R. Büchi. Weak Second-order Arithmetic and Finite Automata. *Math. Logik. Grund. Math.*, 6:66–92, 1960.

[4] L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In *10th European Symposium on Programming (ESOP)*, pages 1–22. LNCS 2028, Springer Verlag, 2001.

[5] L. Cardelli and A. Gordon. Anytime, Anywhere: Modal Logics for Mobile Ambients. In *27th ACM Conf. on Principles of Programming Languages (POPL)*, pages 365–377, 2000.

[6] G. Conforti, O. Ferrara, and G. Ghelli. TQL Algebra and its Implementation (Extended Abstract). In *IFIP Int. Conf. on Theoretical Computer Science (IFIP TCS)*, pages 422–434, 2002.

[7] G. Conforti, G. Ghelli, A. Albano, D. Colazzo, P. Manghi, and C. Sartiani. The Query Language TQL. In *5th Int. Workshop on the Web and Databases (WebDB)*, 2002.

[8] J. Ferrante and C.W. Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer Verlag, 1979.

[9] M.J. Fischer and M.O. Rabin. Superexponential Complexity of Presburger Arithmetic. In *AMS Symp. on the Complexity of Computational Computational Processes. Vol. 7*, pages 27–41, 1974.

[10] S. Ginsburg and E.H. Spanier. Bounded ALGOL-like Languages. *Trans. Amer. Math. Soc.*, 113:333–368, 1964.

[11] S. Ginsburg and E.H. Spanier. Semigroups, Presburger Formulas and Languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[12] G. Gottlob and C. Koch. Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. In *PODS 2001*, pages 17–28, 2002.

[13] F. Klaedtke and H. Ruess. Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science at Freiburg University, 2002.

[14] O. Kupferman, U. Sattler, and M.Y. Vardi. The Complexity of the Graded μ-Calculus. In *18th Int. Conf. on Automated Deduction (CADE)*, pages 423–437. LNCS 2392, Springer Verlag, 2002.

[15] D. Lugiez. A Good Class of Automata and Application to Inductive Theorem Proving. In *25th Int. Coll. on Automata, Languages and Programming (ICALP)*, pages 409–420. LNCS 1443, Springer Verlag, 1998.

[16] D. Lugiez and S. Dal Zilio. Multitrees Automata, Presburger's Constraints and Tree Logics. Technical Report 08-2002, Laboratoire d'Informatique Fondamentale de Marseille, 2002.

[17] D. Lugiez and S. Dal Zilio. XML Schema, Tree Logic and Sheaves Automata. Technical Report RR-4631, INRIA, 2002.

[18] T. Milo, D. Suciu, and V. Vianu. Typechecking for XML Transformers. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages

11–22, Dallas, TX, 2000.

[19] M. Minsky. Recursive Unsolvability of Post's Problem of Tag and Other Topics in the Theory of Turing Machines. *Ann. of Math.*, 74:437–455, 1961.

[20] A. Neumann and H. Seidl. Locating Matches of Tree Patterns in Forests. In V. Arvind and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, LNCS, pages 134–145. Springer Verlag, 1998.

[21] F. Neven and T. Schwentick. Query Automata over Finite Trees. *Theoretical Computer Science (TCS)*, 275(1-2):633–674, 2002.

[22] F. Neven, T. Schwentick, and V. Vianu. Towards Regular Languages over Infinite Alphabets. In *MFCS 2001*, pages 560–572, 2001.

[23] F. Neven and J. Van den Bussche. Expressiveness of Structured Document Query Languages Based on Attribute Grammars. *Journal of the ACM*, 49(1):56–100, 2002.

[24] J. Niehren and A. Podelski. Feature Automata and Recognizable Sets of Feature Trees. In *4th Int. Conf. on Theory and Practice of Software Development (TAPSOFT)*, pages 356–375. LNCS 668, Springer Verlag, 1993.

[25] H. Ohsaki. Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories. In *15th Computer Science Logic (CSL)*, pages 539–553. LNCS 2142, Springer Verlag, 2001.

[26] H. Ohsaki and T. Takai. Decidability and Closure Properties of Equational Tree Languages. In *13th Int. Conf. on Rewriting Techniques and Applications (RTA)*, pages 114–128. LNCS 2378, Springer Verlag, 2002.

[27] R. Parikh. On Context-Free Languages. *Journal of the ACM (JACM)*, 13(4):570–581, 1966.

[28] P. Wolper and B. Boigelot. On the Construction of Automata from Linear Arithmetic Constraints. In *Tools and Algorithms for Construction and Analysis of Systems, 6th Ann. Conf. (TACAS)*, pages 1–19. LNCS 1785, Springer Verlag, 2000.