Numerical Document Queries

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

Thomas Schwentick  
Philips-Universität Marburg  
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

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 database [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.

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:

\[ x \in \text{Lab}_{\text{music}} \land x/\phi_1 \]

where \( \phi_1 \) is the formula

\[
\phi_1 \equiv [\text{Lab}_{\text{jazz}}] \geq [\text{Lab}_{\text{pop}}]
\]

Here, \([\text{Lab}_{\text{jazz}}]\) and \([\text{Lab}_{\text{pop}}]\) denote the numbers of children labeled with jazz and pop, respectively. For the second query we replace \( \phi_1 \) by \( \phi_2 \), where \( \phi_2 \) is the formula:

\[
\phi_2 \equiv [\text{Lab}_{\text{jazz}}] \geq [\text{Lab}_{\text{pop}}] + [\text{Lab}_{\text{music}}] + [\text{Lab}_{\text{class}}]
\]

Explicitly enumerating all flavors of music different from jazz may be awkward. Instead, we perhaps prefer to define an auxiliary predicate \( X \) by:

\[
\forall x. (x \in X) \leftrightarrow -(x \in \text{Lab}_{\text{jazz}})
\]

and then replace the formula \( \phi_2 \) with:

\[
\phi'_2 \equiv [\text{Lab}_{\text{jazz}}] \geq [X]
\]

As an operational counterpart of the extended logic we study bottom-up tree automata that are enhanced by Presburger

\footnote{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 \text{music} as a big jazz.}
<doc>
  <user>
    <name>...</name>
  </user>
  <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>
      <year>1997</year>
      <time>3325</time>
      <price>220</price>
    </french>
  </music>
  <video>
    <video>
      <images>...</images>
    </video>
  </video>
  <user/>
</doc>

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

*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 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 Nehren 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 satisfying 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 one global Presburger formula [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 $\mathbb{N}^*$ 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 multi-set union of two multi-sets $m_1, m_2$ is denoted by $m_1 + m_2$. Also, we write $a \in m$ if $a$ occurs in $m$ with non-zero multiplicity.

Let $\Sigma$ denote a finite alphabet. Given the above notion of multi-sets, we define the set $U_\Sigma$ of unordered trees $t$ (u-trees for short) over $\Sigma$ by the following grammar:

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

Presburger formulas $\phi$ are defined by the following grammar:

$$\phi ::= x = n \mid x + y = z \mid \phi \land \phi_2 \mid \exists x. \phi$$

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

$$\begin{align*}
\rho \models x = n & \quad \text{iff } \rho(x) = n \\
\rho \models x + y = z & \quad \text{iff } \rho(x) + \rho(y) = \rho(z) \\
\rho \models \phi \land \phi_2 & \quad \text{iff } \rho \models \phi \land \rho \models \phi_2 \\
\rho \models \exists x. \phi & \quad \text{iff } \rho \models \exists x. \phi \quad \text{for some } n \in \mathbb{N}
\end{align*}$$

In particular, a formula $\phi$ is satisfiable iff there is an assignment $\rho$ 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 $\nu$ 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

$$\{c + \sum_{i=0}^{k} x_i \rho_i \mid x_i \in \mathbb{N}\}$$

where $c$ and the $\rho_i$ are vectors over $\mathbb{N}$. Presburger formulas $\phi$ can be compiled into finite automata $A_\phi$ running on the binary representations of these vectors. The automaton $A_\phi$ checks whether a given tuple is contained in the semi-linear set corresponding to $\phi$. Practical verification tools based on such automata are studied, e.g., 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,
- $\Sigma$ is the finite alphabet of tree labels, and
- $\delta$ 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 $y_q$ represent the admissible multiplicities of the state $p$ on the children. As an example, consider the formula $\psi = \exists x. y_q + 2z = y_q$. This formula expresses that the number $y_q$ of sons labeled by the state $q$ is at least as big as $y_q$, 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 multi-set of u-subtrees of the root $a$) and $\delta(q, a) = \phi$. Then $t \models_A q$ iff there are multisets of u-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 $t' \models_S p$ ($p \in Q$);
- $\{y_q \mapsto n_q \mid p \in Q\} \models \phi$.

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

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

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 u-trees over $\{a, b\}$ where the internal nodes
are all labeled with $a$ and have at most as many u-subtrees with a $b$-leaf as without. One example u-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 $\delta$ can be represented in the table:

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$y_1 = 0$</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>$y_0 \geq y_1 &gt; 0$</td>
<td>$y_0 + y_1 = 0$</td>
</tr>
</tbody>
</table>

The state 0 is assumed by all u-trees without $b$ leaves while the state 1 is only assumed by u-trees containing $b$ leaves. Figure 3 shows a run of the automaton on $t$, i.e., an assignment mapping the nodes (u-subtree occurrences) of $t$ to states such that the pre-conditions in $\delta$ are locally satisfied at every node. In particular, we have: $t \models 1$.

---

**Figure 2**: An example u-tree $t$ from $L$.

**Figure 3**: An example run on the example u-tree $t$.

The main result of this section is:

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

**Proof.** Consider a Presburger u-tree automaton $A = (Q, \Sigma, \delta, F)$. Let us call a state $q \in Q$ reachable iff there is a u-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:

$$\left( \bigwedge_{p \in Q_1 \setminus \bigcup_{j < k} R^{(j)}} y_p = 0 \right) \land \left( \bigvee_{a \in \Sigma} \delta(q, a) \right)$$

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 u-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, $R^{(j)} = R^{(j+1)}$ 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 $L(A)$ is non-empty iff some state $f \in F$ is reachable, we conclude that emptiness for Presburger u-tree automata is decidable.

Given that all Presburger formulas $\phi$ in $\delta$ have already been compiled into automata $A_\phi$ (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 $\delta(q_1q_2, a)$ by the formula:

$$\bigwedge_{p_1 \in Q_1} y_{p_1} \land \bigwedge_{p_2 \in Q_2} y_{p_2} \land \delta_1(q_1, a) \land \delta_2(q_2, a) \land \left( \bigwedge_{p_1 \in Q_1, p_2 \in Q_2} y_{p_1p_2} = y_{p_1} \land \left( \bigwedge_{p_2 \in Q_2 \cap q_2} \sum_{p_1 \in Q_1} y_{p_1p_2} = y_{p_2} \right) \right)$$

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 $q_1q_2$ should be satisfiable iff each of the states $q_1$ and $q_2$ are. Thus, we construct the pre-condition for $q_1q_2$ and $\alpha$ on sequences $\alpha$ from $(Q_1 \times Q_2)^*$ as the conjunction of the pre-conditions for $q_1$ and $q_2$ on the projections of $\alpha$ 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: $\sum_{p_2 \in Q_2} y_{p_1p_2} = y_{p_1}$. Using the analogous formula also for $y_{p_2}$, $p_2 \in \Sigma$, we arrive at the stated formula. Accordingly, it is easy to prove that $t \models_A q_1q_2$ iff $t \models_A q_1$ and $t \models_A q_2$.

Thus, $L(A) = L(A_1) \cap L(A_2)$ 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 if for every $a \in \Sigma$ and every multiset of states $m = \sum_{q \in Q} n_q \cdot \{ q \}$, the following holds:

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

We can construct for the automaton $A$ a Presburger formula $\text{test}_A$ such that $A$ is deterministic iff $\text{test}_A$ is satisfiable. Therefore we have:

**Proposition 1.** It is decidable whether or not a Presburger u-tree automaton is deterministic.

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

**Proposition 2.** $\mathcal{L}(\overline{A}) = U_\Sigma \mathcal{L}(A)$.

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:

$$\left( \bigwedge_{q \in B} \psi_{q,a} \right) \land \left( \bigwedge_{q \notin B} \neg \psi_{q,a} \right)$$

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 $y_q, p \in Q$, 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:

$$\exists \ y_q \cdot \delta(q, a) \land \exists \ y_{B,p} \cdot \left( \bigwedge_{B \subseteq Q \setminus q \subseteq B} y_{B,p} = y_p \right) \land \left( \bigwedge_{p \in Q} \bigwedge_{p \in B} y_{B,p} = y_q \right)$$

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 $\delta$. By theorem 3, we can w.l.o.g. assume that $A$ is deterministic. Also assume that all formulas $\phi$ in $\delta$ have been compiled into finite automata $A_\phi$ 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 subtree $t'$ of $t$ and every state $p$ of $A$ whether or not $t' \models_A p$. In total, this amounts to $O(n)$ tests of assertions $\rho \models \phi$ for formulas $\phi$ occurring in $\delta$ where $\rho(x) \leq n$ for all $x$ in the domain of $\rho$. The binary representations of the $\rho(x)$ altogether have size at most $O(n)$. Therefore, the automata $A_\phi$ can verify the assertions $\rho \models \phi$ in time $O(n)$. We conclude that the overall complexity is $O(n)$.

**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 $\Sigma$). An unordered context (u-context for short) is a u-tree $c \in U_\Sigma[\bullet]$ which contains exactly one occurrence of $\bullet$ at a leaf (the hole). Let $c[t]$ denote the u-tree which is obtained from $c$ by substituting $\bullet$ with $t$ (i.e., filling the hole). Note that for a given u-tree $t$, the set $C(t)$ of contexts such that $t = c[t]$ for suitable u-subtrees $t_i$ is in one-to-one correspondence with the set of nodes of $t$. Therefore, in the following we will no longer distinguish between contexts $c \in C(t)$ and nodes of $t$.

A (unary) query is a mapping $\mu$ 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 $c, p \models_A q, (p, q \in Q)$ if $c \models_A p, q$ where $A_p, \bullet = (Q, \Sigma \cup \{ \bullet \}, \delta_p, F)$ is obtained from $A$ by extending $\Sigma$ with $\bullet$ and defining:

$$\delta_p, \bullet(q', a) = \begin{cases} \delta(q', a) & \text{if } a \in \Sigma \\ \bigwedge_{q \in Q} \exists y_q = 0 & \text{if } a = \bullet \land q' = p \\ \text{false} & \text{if } a = \bullet \land q' \neq p \end{cases}$$

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 $t = c[t]$ for a context $c$ and $t, t_1 \in U_\Sigma$. Then $t \models_A q$ if $t_1 \models_A p$ and $c, p \models_A q$ for some $p \in Q$.

A (unary) Presburger pattern is a property of u-subtrees within u-trees. We define this property by means of a pair $(A, T)$ 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 $c \in C(t)$ is a match of the pattern $(A, T)$ 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 $(A, T)$ in a u-tree $t \in U_\Sigma$ of size $n$ is computable in time $O(n)$.
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:

\[
\begin{align*}
f &::=\ y < y' \mid y \in S \mid y/p \mid \psi_1 \land \psi_2 \mid \neg f \mid \exists y. f \mid \forall y. f \\
S &::= Y \mid \text{Lab}_a \\
p &::= t_1 = t_2 \mid t_1 + t_2 = t_3 \mid p_1 \land p_2 \mid \neg p \mid \exists x. p \\
t &::= [S] \mid x \mid n
\end{align*}
\]

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:

\[ \exists Y. Y = \{ y_1 \} \land y/\{Y\} = 1 \]

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

\[
\begin{align*}
t, \rho, \sigma &\models \phi
\end{align*}
\]

for the structure \( t \) together with valuations \( \rho, \sigma \) (for the sets of free set variables and free individual variables, respectively) is inductively defined as follows:

\[
\begin{align*}
t, \rho, \sigma &\models y < y' \quad \text{iff} \quad \sigma(y) < \sigma(y') \text{ holds in } t \\
t, \rho, \sigma &\models y \in \text{Lab}_a \quad \text{iff} \quad \sigma(y) \in \text{Lab}_a \text{ holds in } t \\
t, \rho, \sigma &\models y \in Y \quad \text{iff} \quad \sigma(y) \in \rho(Y) \\
t, \rho, \sigma &\models y/p \quad \text{iff} \quad m \models p \text{ where } \\
[B] = &\# \{ v \in D \mid \sigma(y) < v, v \in \text{Lab}_a \} \\
[Y] = &\# \{ v \in D \mid \sigma(y) < v, v \in \rho(Y) \} \\
t, \rho, \sigma &\models f_1 \land f_2 \quad \text{iff} \quad t, \rho, \sigma \models f_1 \text{ and } t, \rho, \sigma \models f_2 \\
t, \rho, \sigma &\models \neg f \quad \text{iff} \quad t, \rho, \sigma \not\models f \\
t, \rho, \sigma &\models \exists y. f \quad \text{iff} \quad t, \rho, \sigma \models \{ y \mapsto v \} \models f \text{ for some } v \in D \\
t, \rho, \sigma &\models \exists Y. f \quad \text{iff} \quad t, \rho, \sigma \models \{ Y \mapsto R \}, \sigma \models f \text{ for some } R \subseteq D
\end{align*}
\]

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

A language \( L \subseteq U_Y \) is unordered PMSO-definable iff there is a closed formula \( \phi \) 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 \( L \subseteq U_Y \) the following two statements are equivalent:

1. \( L \) is unordered PMSO-definable;
2. \( L = L(A) \) for some Presburger u-tree automaton \( A \).

The proof is analogous to the proof of the corresponding result for MSO-logic and tree automata over ordered trees.
Let us turn to the characterization of queries. An unordered PMOS-pattern is an unordered PMOS formula \( \phi \) with at most one free variable \( y \). A match of \( \phi \) in \( t \) is given by a node \( v \) such that
\[
t, 0, \{ y \mapsto v \} \models \phi
\]
A query \( \mu \) is unordered PMOS-definable iff there is an unordered PMOS-pattern \( \phi \) such that for every \( t, \mu(t) \) is the set of all matches of \( \phi \) in \( t \).

Theorem 6. For a query \( \mu \) the following two statements are equivalent:
1. \( \mu \) is unordered PMOS-definable;
2. \( \mu \) is definable by a Presburger pattern \( (A, T) \) for some Presburger v-tree automaton \( A \).

3. Extension to Ordered Trees

In many applications, e.g., where documents are automatically generated from databases as textural 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 \( \langle t_1, \ldots, t_n \rangle \) 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 \( z" \) (denoted by: \( y ; z \)) and correspondingly adjusting the definition of the satisfaction relation \( \models \).

Let us therefore see in how far Presburger v-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 \( \phi \) we define in the obvious way whether \( w \in Q^* \) matches \( \phi \) (i.e., \( w \models \phi \)). So for example, if \( \phi \) equals
\[
p \left( p \mid q \right)^* \wedge (y_0 = y_2)
\]
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 \( \phi \) there is a string \( w \) such that \( w \models \phi \).

Proof. Let \( \phi \) be a Presburger regular expression. First of all, \( \phi \) can be transformed into disjunctive normal form \( \bigvee_i \phi_i \), where each \( \phi_i \) 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 \( \bigvee_i (e_i \land \pi_i) \), where each \( e_i \) is a regular expression and each \( \pi_i \) a Presburger formula. Clearly, \( \phi \) is satisfiable by a string if and only if at least one disjunct \( e_i \land \pi_i \) is. To check satisfiability for the conjunction of a regular expression \( e \) and a Presburger formula \( \pi \), we compute a Presburger formula \( \pi_\varepsilon \) 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 each \( q \in \varnothing \) to each \( q \in Q \) [27]. An expression \( e \land \pi \) is satisfiable if and only if the Presburger formula \( \pi_\varepsilon \land \pi \) is satisfiable and we are done.

The complexity of the described decision procedure crucially depends on the complexity of constructing the formula \( \pi_\varepsilon \). 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;
- \( \delta \) maps pairs \( (q, a) \) of states and labels from \( \Sigma \) to Presburger regular expressions \( \phi \) over \( Q \).

Accordingly, we introduce an extended satisfaction relation between ordered trees \( t \) and states \( q \) by defining for \( t = (t_1, \ldots, t_n) \) and \( \delta(q, a) = \phi \) if \( t = \varepsilon \) or if there are states \( p_1, \ldots, p_l \in Q \) such that \( t_j \models \phi \) for all \( j \) and \( p_1 \ldots p_l \models \phi \). The language \( L(A) \subseteq T_Q \) which is accepted by the automaton \( A \) then is given by:
\[
L(A) = \{ t \in T_Q \mid \exists f \in F : t \models \phi \}
\]
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 a state \( q \), 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 \( t' = a(t_1, \ldots, t_n) \) and \( R_t = \{ p \mid t \models p \} \) have been already computed. Moreover, we can suppose that the Presburger regular expressions used in \( A \) are disjunctions of conjuncts \( e_i \land \pi_i \), where each \( e_i \) is a regular expression and \( \pi_i \) a Presburger formula. Then we may check for each \( e_i \land \pi_i \) separately whether it is verified by \( t_1, \ldots, t_n \). So now consider one conjunct \( e \land \pi \). 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 \( v : Y_Q \to \{ 1, \ldots, \} \) verifying the following condition: there exists a sequence of states \( \alpha = r_1 \ldots r_k \) with \( r_k \in P_k \) 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 $\alpha$. 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 $O(n^{3+1})$ on the computation of the set of all successor states at the root node of $A$. Hence, the overall runtime on a tree of size $m$ is $O(n^{3+1})$.

Clearly, the upper bound complexity 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 $c \in T_{n,1} \{\bullet\}$ that contains exactly one occurrence of the fresh symbol $\bullet$. Accordingly, we generalize the definition of $|= A$ to contexts by defining $c, p |= A q$. $(p, q \in Q)$, iff $c |= A_{\bullet} q$ where $A_{\bullet} = (Q, \Sigma \cup \{\bullet\}, \delta_{\bullet}, F)$ is the automaton obtained from $A$ by extending $\Sigma$ with $\bullet$ and extending $\delta$ to $\delta_{\bullet}$, where $\delta_{\bullet}(q', a) = \delta(q', a)$ whenever $a \in \Sigma$ and

$$\delta_{\bullet}(q', a) = \begin{cases} e & \text{if } q' = p \\ 0 & \text{if } q' \neq p \end{cases}$$

A (monadic) Presburger pattern now is a property of subtrees within trees. We define this property by means of a pair $(A, T)$ where $A = (Q, \Sigma, \Delta, F)$ is a Presburger tree automaton and $T \subseteq Q$ is a set of states. Let $t \in T_T$. A context $c \in C(t)$ is a match of the pattern $(A, T)$ 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 $(A, T)$ in a tree $t \in T_T$ 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. \(\square\)

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 MSO formula of the form $\exists X_1 \ldots \exists X_k \varphi$ where $\varphi$ is first-order.

**Proof.** Only-if: Let $A$ be a Presburger tree automaton and $\delta$ the transition relation of $A$. Without loss of generality we can assume that all Presburger regular expressions used in $A$ are disjunctions of expressions $e \land \pi$, where $e$ is a regular expression and $\pi$ is a Presburger formula. From Büchi’s Theorem it follows that each regular expression $e$ can be expressed by an existential MSO formula $\varphi_e = \exists Y_1 \ldots \exists Y_i \varphi_e$ (on strings). Hence, we can construct a formula $\varphi = \exists X_1 \ldots X_k \varphi$ 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 $\varphi_e$. The first-order part of $\varphi$ describes the consistency of the states between nodes of the input tree and their children by using the formulas $\varphi_e$.

If we show first that every first-order ordered MSO formula $\psi$ 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 $\psi$. 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 $\Sigma \cup \{x\}$ 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 $\varphi$ 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 $\psi$ as follows. The state set of $A_1$ is $Q \times 2^{S}$. The intuitive meaning of a state $(q, X)$ at a node $v$ in the following. If $x$ did not occur in the subtree rooted at $v$, then $A_1$ 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 a Presburger regular expression $e_q \land e_x$, where $e_q$ is obtained from $\delta(q, a)$ by replacing each occurrence of a state $r \in Q$ in a regular expression by $\bigcup_{r \subseteq Q} (r, S)$ and each occurrence of $y_q$ in a Presburger formula by $\sum_{y \subseteq Q} y_{(v, S)}$.

The regular expression $e_x$ is of the form $\bigwedge_{S \subseteq Q} (e_x \lor e_q \land c_q)$.

This, $e_x$ expresses that $A_1$ would take state $p$ at $v$ if the label of $v$ was $(a, x)$. Likewise, $c_q$ expresses that $A_1$ would take state $p$ at $v$ if the label $b$ of a suitable node below $v$ was replaced by $(b, x)$. Therefore, $e_x$ 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 $e_q$. Let $\delta(p, a)$ be a disjunction of conjunctions of the form $e \land \pi$ where $e$ is a regular expression and $\pi$ is a Presburger formula. The expression $e_x$ is obtained by replacing each $e \land \pi$ with a disjunction $\bigvee_{r \subseteq Q} (e_x \lor e_q \land c_q)$.

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

- $\sum_{y \subseteq Q} y_{(v, S)} > 0$ if $q \neq \{r', r''\}$ or $q = r = r'$. 
- $\sum_{y \subseteq Q} y_{(v, S)} > 0$ if $q = r$ and $r \neq r'$, and
- $\sum_{y \subseteq Q} y_{(v, S)} > 0$ if $q = r = r'$. 

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

$$L = \{(q_1, S_1) \ldots (q_n, S_n) \mid \exists i : (q_i, S_i) = (r, S) \land q_i \neq q_i-1q_i+1 \ldots q_n \in L(e)\}$$

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

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 $A_0$ such that $A$ does not halt on the empty input if and only if $A_0$ accepts all trees over the alphabet $Q \cup \{ \#, \emptyset, a, b \}$. 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 $r_0 a^* b^*(\# r_1 a^* b)^*$, where $r_0 = q_0 \cdots q_m$ 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 $A_2$ 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 contexts $n_1$ and $n_2$, respectively, is encoded by the string $qa^{n_2+b^{-1}}$ 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 $\# q a^* b^*$, 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 $Q \cup \{ q_1, q_2, q_0, q_1, q_2 \}$. In the first stage, when moving from the leaves to nodes of level 1 the automaton can enter state $q_1$ from each state on the leaves. Further, it can enter state $q_2$ from all leaves labeled $\#$ and state $q$, for each leaf with label $q \in Q$. For leaves with label $a$ it enters state $q_0$ or $q_1$. Accordingly, it may enter $q_2$ or $q_0$ from $b$.

It enters an accepting state if

- the string of states of the nodes of the first level is of the form $q_1 q_1 q_2 q_1 q_1 q_2 q_1 q_1 q_2$ with $q_1, q_2 \in Q$, and
- the numbers of the states $q_1, q_2, q_1, q_2$ 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.

**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 formatting parts where the ordering is significant. This classification is formalized by partitioning the finite alphabet $\Sigma$ into subsets $\Sigma_n = \Sigma_0 + \Sigma_1$ where $\Sigma_0$ and $\Sigma_1$ consist of all labels of nodes with unordered and ordered children, respectively. The set $M$ of all mixed trees (m-trees for short) over $\Sigma$ is now given by the grammar:

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

Mixed trees in our sense correspond to terms with one associative symbol "*" (for accumulating the ordered contexts) and one associative and commutative symbol "\#" (for accumulating multi-sets). Languages of such trees, e.g., have been studied by Ungg [5, 6] and Ohnuki [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 $\Sigma_0$ and regular expressions in transitions for labels from $\Sigma_1$ 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 $L(A) = L(A')$. □

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 $(A, T)$ in a m-tree $t \in M_C$ of size $n$ is computable in time $O(n)$.

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 basic 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 $n$. In the first pass, we determine for every m-subtree $t_i$ of $t$ the set:

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

Let $A'$ denote the deterministic automaton as constructed in section 2. Then we know that for every $t' \in M_C$, $t' \models_A B$ iff $B = \{ p \in Q \mid t' \models_A p \}$. Therefore, the sets $B(t_i)$ 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 $O(n)$.

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

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

where $t_i$ is the m-subtree of $t$ with $t = c[t_i]$. 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 $\Sigma_1$. So assume that we are given a m-context $c$ in $t$ where $t = c[a(t_1, \ldots, t_k)]$ 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 $c_i = c[a(t_1, \ldots, t_{i-1}, \ast, t_{i+1}, \ldots, t_k)]$. Let $B_i = B(t_i)$ 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:

$$D(c_i) = \bigcup_{j \neq i} \{ p \in D(c) \mid q \in D(c) \}$$

$$D_i(t) = \{ p_i \in B_i \mid \exists j \neq i : p_j \in B_j \} : p_1 \ldots p_k \in \delta_1(q, a) \}$$

Given a (non-deterministic) finite automaton $A_{s,a}$ for $\delta_1(q, a)$, all sets $D_i(t)$, $i = 1, \ldots, k$, can be computed by one left-to-right and one right-to-left pass of $A_{s,a}$ 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 $O(n)$ only. In total, our algorithm has time complexity $O(n)$. 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 $\Sigma_0$. 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 $L \subseteq M_C$ the following two statements are equivalent:

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

**Theorem 16.** For a query $\mu$ the following two statements are equivalent:

1. $\mu$ is mixed PMSO-definable;
2. $\mu$ is definable by a Presburger pattern $(A, T)$ 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 query algorithms.

But of course, besides reasoning about the number 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 (time) tags represent the number of seconds of a piece. Hence, we are interested in all entries where the number in the (time) tag exceeds six times the number in the (price) tag. In an extended logic we express this as follows.

$$x \in \text{Lab}_{jazz} \land x/\phi$$

where:

$$\phi \equiv \left[ \text{time} \right] > 6 \cdot \left[ \text{price} \right]$$
As the example shows, we refer to the number within a tag \( \langle \text{time} \rangle \) by \([\text{time}]\). 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 \( \langle \text{time} \rangle \) with number \( n \) as \( n \) occurrences of a tag \( \langle \#\text{time} \rangle \). 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 \( \langle \text{time} \rangle \)- and \( \langle \text{price} \rangle \)-tags.

```xml
<user>
  ...
  <jazz>
    <time> 3310 </time>
    <price> 240 </price>
    <time> 220 </time>
    <price> 16 </price>
    <time> 3510 </time>
    <price> 262 </price>
    <time> 3525 </time>
    <price> 220 </price>
  </jazz>
  ...
</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 \([\text{time}]\) as the sum of numbers within tags \( \langle \text{time} \rangle \). For instance, the set of all users who downloaded at least 2 hours of jazz music could be formulated as follows.

\[
x \in \text{Lab}_{\text{user}} \land \exists y. \; x < y \land y \in \text{Lab}_{\text{jazz}} \land y/\phi
\]

where:

\[
\phi \equiv [\text{time}] \geq 1200
\]

By replacing \( \phi \) with \([\text{time}] > 6 \cdot [\text{price}]\) 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


