# Counting in Trees for Free

Helmut Seidl[1], Thomas Schwentick[2⋆], Anca Muscholl[3], and Peter Habermehl[3]

[1] TU München, I2,
seidl@in.tum.de
[2] Philipps-Universität Marburg, FB Mathematik und Informatik
tick@informatik.uni-marburg.de
[3] LIAFA, Université Paris 7, 2, pl. Jussieu, F-75251 Paris

**Abstract.** It is known that MSO logic for ordered unranked trees is undecidable if Presburger constraints are allowed at children of nodes. We show here that a decidable logic is obtained if we use a modal fixpoint logic instead. We present a characterization of this logic by means of *deterministic* Presburger tree automata and show how it can be used to express numerical document queries. Surprisingly, the complexity of satisfiability for the extended logic is asymptotically the same as for the original fixpoint logic. The non-emptiness for Presburger tree automata (PTA) is PSPACE-complete, which is moderate given that it is already PSPACE-hard to test whether the complement of a regular expression is non-empty. We also identify a subclass of PTAs with a tractable non-emptiness problem. Further, to decide whether a tree $t$ satisfies a formula $\varphi$ is polynomial in the size of $\varphi$ and linear in the size of $t$.
A technical construction of independent interest is a *linear* time construction of a Presburger formula for the Parikh image of a regular language.

## 1 Introduction

In XML schema description languages as DTDs and XML Schema, the content of elements, i.e., the possible sequences of children elements of a node, is described mainly by regular expressions.[1] This is sufficient in very many cases. But often one is interested in expressing conditions on the frequency of occurrences of elements in the children sequence. When the order of elements is very constrained regular expressions still do the job, e.g. by $(\texttt{title author author}^+)$ one might express that there have to be at least two authors in a paper. If the order is not fixed, even simple conditions require complicated regular expressions. E.g., saying that there is exactly one title and there are at least two authors would require an expression like $(\texttt{title author author}^+)$ | $(\texttt{author}^+\ \texttt{title author}^+)$ | $(\texttt{author author}^+\ \texttt{title})$. It would be desirable to describe this condition simply by an expression like $|\texttt{title}| = 1 \wedge |\texttt{author}| \geq 2$. While these conditions do not go beyond the scope of regular expressions, others do. A simple example is $|\texttt{author}| \leq 2 \cdot |\texttt{title}|$.

⋆ Contact author. The support of this work by the DAAD and Egide under PROCOPE grant D/0205766 is kindly acknowledged.

[1] We view an XML document here and in the rest of the paper as a labeled, unranked, ordered tree.

J. Diaz et al. (Eds.): ICALP 2004, LNCS 3142, pp. 1136–1149, 2004.

Most of the existing theoretical work on XML schema languages has concentrated on regular tree languages. These languages can be described by tree automata [14,15] and a variety of other formalisms [16,8] including fixpoint formulas [13]. Typically, the interaction between the children of a node and the node itself are usually expressed in terms of regular expressions. Other work extended these formalisms to let them formulate (at least unary) queries. The resulting query facilities usually have the expressive power of monadic second-order (MSO) logic. Here, we study extensions of such formalisms by numerical conditions as above. In particular, we are interested in the main complexity questions.

The conditions we allow are Boolean combinations of regular expressions and Presburger formulas. Presburger formulas basically allow linear (in)equalities and expressions of the form $t \equiv c \pmod{n}$. A more detailed definition can be found in Section 2. Counting conditions in schema languages have been used, e.g., in [12].

In a previous paper [22] we considered *non-deterministic* tree automata with such extended conditions (PTAs). This kind of automata is not closed under complementation. Moreover, whereas their non-emptiness problem (whether an automaton accepts some tree) is decidable, the universality problem (whether it accepts all trees) is not. Consequently, MSO logic extended by such conditions has an undecidable satisfiability problem. In the present paper, we study two weaker formalisms. We consider a *fixpoint* logic instead of MSO logic and *deterministic* instead of non-deterministic PTAs. It turns out that these two formalisms define the same class of tree languages. Furthermore, their non-emptiness (resp., satisfiability) problem is decidable. Actually, it came as a surprise that the complexities of these problems are as low as one could hope for[2]:

- It is already PSPACE-hard to check whether the intersection of several regular expressions is empty. Therefore, the non-emptiness problem for PTAs is trivially PSPACE-hard. We prove that it is also in PSPACE. Additionally, we show that it becomes tractable when each precondition of the automaton is a disjunction of formuls $r \wedge f$, where $r$ is a regular expression and $f$ is an equation (with existentially quantified variables).
- Satisfiability for fixpoint formulas (without numerical conditions) is EXPTIME-complete. We show that the complexity does not increase when we add numerical conditions.
- The same complexities can be easily derived for the containment problem.
- Checking whether a tree $t$ is accepted by a Presburger tree automaton $A$ or a fixpoint formula $\phi$ can be decided in time $O(|t||A|)$ and $O(|t||\phi|^2)$, respectively.

Furthermore, we show how Presburger fixpoint formulas can be used to formulate unary queries. These queries can be evaluated in time which is linear in the size of the tree and polynomial in the size the formula. During our investigation we also studied the relationship between regular expressions and Presburger formulas. It is well-known that the Parikh image of each regular language (i.e., basically the set of symbol frequency vectors of words) can be expressed by a Presburger formula. We show that such a formula can be constructed very efficiently, in linear time, even from a non-deterministic finite automaton.

[2] Actually these complexities hold only with quantifier-free Presburger formulas. However, this does not restrict the expressivity of the logic.

The paper is organized as follows. In Section 2 we give the basic definitions for Presburger logic. Section 3 explains how to compute efficiently a Presburger formula from a regular string language. Section 4 introduces Presburger fixpoint formulas. In Section 5 we define Presburger automata and prove the equivalence with Presburger fixpoint formulas. Section 6 contains the complexity results. In Section 7 we define a query extension of Presburger fixpoint formulas and consider its evaluation complexity. We end with a conclusion.

*Related work.* Unordered document trees are closely related to the generalization of feature trees considered by Niehren and Podelski in [17] 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 [2,1,3,4]. Their approach is based on first-order logic and fixpoint operators. An extension to numerical constraints has recently been proposed by Dal Zilio et al. [5]. Kupferman et al. study a $\mu$-calculus with *graded* modalities where one can express, e.g., that a node has at least $n$ successors satisfying a given property [10]. The numbers $n$, however, are hard-coded into the formula. Ordered successors are not considered. Klaedtke and Ruess consider automata on the unlabeled infinite binary tree, that have an accepting condition depending on a global Presburger constraint [9].

Our notion of Presburger Tree Automata, which combines both regular constraints on the children of nodes as well as numerical constraints given by Presburger formulas, has independently been introduced by Lugiez and Dal Zilio [11] and Seidl et al. [22]. In their paper, Lugiez and Dal Zilio indeed propose a modal logic for XML documents which they call *Sheaves logic*. This logic allows to reason about numerical properties of the contents of elements but still lacks recursion, i.e., fixpoint operators. Lugiez and Dal Zilio consider the satisfiability and the membership problem and they show that Sheaves logic formulas can be translated into deterministic PTAs. Seidl et al. in [22] on the other hand, prove that nondeterministic PTAs precisely correspond to the existential fragment of MSO logic on ordered trees enhanced with Presburger constraints on the children of nodes. As a technical result, they also show that *first-order* formulas can be translated into deterministic PTAs.

## 2 Preliminaries

*Presburger Logic* is the first-order logic over the structure $(\mathbb{N}, \leq, +)$. Given a formula $f$ and an *assignment* $\sigma$ mapping the variables of $f$ to numbers, we write $\sigma \models f$ if $f$ holds for $\sigma$ (in the obvious sense) and call $\sigma$ a solution of $f$. For convenience, we use an extended language. Thus, we write $cx$ for $x + \cdots + x$ ($c$ times). Also, we allow terms with negative coefficients as in $2y - 3x$. A typical Presburger formula is $\exists y\ (2y = x)$ stating that $x$ is even. It is well-known that the extension of Presburger logic by 0, 1 and the binary predicates $x \equiv y \pmod n$, for each constant $n$, has quantifier elimination, i.e., for each formula there is an equivalent quantifier-free formula [20]. E.g., the above formula can be written as $x \equiv 0 \pmod 2$. Here, we call quantifier-free formulas in the extended language with modulo predicates and equality over terms with integer coefficients *quantifier-free Presburger formulas*. We say that formulas of the form $\exists x_1, \ldots, x_k \bigvee_{i=1}^{m} f_i$, where

each $f_i$ is a conjunction of equations $t = c$ with a term $t$ and an integer constant $c$ are in *equation normal form*. Note that formulas in equation normal form do not contain any negations.

**Lemma 1.** *Every Presburger formula has an equivalent formula in equation normal form.*

It is well-known that sets of assignments which fulfill a given Presburger formula $f$ are equivalent to *semi-linear sets* [7]. A semi-linear set is a finite union of *linear sets* of the form $\{\sigma + \sum_{i=1}^{m} \sigma_i z_i \mid z_i \in \mathbb{N}\}$, where $\sigma$ and the $\sigma_i$ are assignments to a finite set of variables (using a fixed enumeration of the variables) or vectors from $\mathbb{N}^k$ for a given $k$.

The *Parikh image* of a word $w = a_1 \cdots a_k, a_j \in \Sigma$ is the assignment $\sigma \in \mathbb{N}^\Sigma$ which maps the variables $|a|, a \in \Sigma$, to the number of occurrences of the letter $a$ in $w$, i.e., $\sigma(|a|) = \#\{j \mid a = a_j\}$. Accordingly, the Parikh image of a set $L \subseteq \Sigma^*$ is the set of Parikh images of $w \in L$.

## 3 Regular String Languages and Presburger Formulas

The fixpoint formulas as well as the tree automata studied here use Boolean combinations of regular expressions and Presburger formulas as conditions on the children of nodes. Whereas it is well-known that the Parikh image of a regular (even context-free) language is semilinear [19] and thus can be described by a Presburger formula with free variables $|a|, a \in \Sigma$, it seems to be not quite as well-known how large the corresponding formula must be. In this section, we show that a Presburger formula for the Parikh image of the language of an NFA $A$ can be computed in linear time. In particular, the size of the formula is *linear* in the size $|A|$ of $A$ (which equals the number of states plus the number of transitions). For regular expressions we have another, direct linear-time construction.

**Theorem 1.** *For any NFA $A$, an existential Presburger formula $\varphi_A$ for the Parikh image of the language $\mathcal{L}(A)$ of $A$ can be constructed in time $\mathcal{O}(|A|)$.*

*Sketch of proof.* With an accepting run of an NFA $A$ on a string $w$ we associate a *flow* $f$ as follows: each transition $(p, a, q)$ of $A$ is labeled by the number of times it is taken in the computation. We construct a Presburger formula which checks two properties. First, the flow is locally consistent, e.g., for each inner node the incoming equals the outgoing flow. Secondly, the subgraph induced by the states with non-zero flow is connected. Here for each node, the distance is guessed from $s$ w.r.t. non-zero flow edges. □

## 4 Presburger Fixpoint Formulas

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 (or it is not known in advance). In other applications, though, which are more related to classical document processing the ordering matters. Since we cannot tell just from looking at a linearized textual representation of the document whether the ordering of children is irrelevant, we prefer to work with ordered trees only but allow the logic

to express properties of unordered documents. Thus, given an alphabet $\Sigma$ of element or node names, the set of all (ordered but unranked) trees $t$ is given by:

$$t \;::=\; a\langle t_1, \ldots, t_k\rangle\ , \quad a \in \Sigma, k \geq 0$$

We write $\mathcal{T}_\Sigma$ for the set of all such trees. We consider a calculus of fixpoint formulas which allows to express both regular and Presburger constraints on children of nodes. Presburger fixpoint formulas $\phi$ are constructed according to the following grammar:

$$\begin{array}{l}\phi \;::= \top \;\mid\; x \;\mid\; \mu\, x.\, \phi \;\mid\; \phi_1 \vee \phi_2 \;\mid\; \phi_1 \wedge \phi_2 \;\mid\; a\langle F\rangle \;\mid\; *\langle F\rangle \\ F \;::= r \;\mid\; \neg r \;\mid\; f\end{array}$$

Here, "$*$" denotes an arbitrary node label from $\Sigma$, and $F$ denotes a generic pre-condition on the children of a node. Such a pre-condition is either a regular expression $r$ over letters $\phi$, $\phi$ a fixpoint formula, or a Presburger formula $f$ with free variables $|\phi|$ denoting the number of children satisfying $\phi$. Essentially the same calculus is obtained if we enhance the *Sheaves logic* of Dal Zilio and Lugiez [11] with recursion.

In the sequel, we assume that $\phi$ is a formula where all bound variables are distinct. Let $\Phi$ denote the set of all subformulas of $\phi$ plus $\top$ (the constant true).[3] We consider assertions $t : \psi$, $t \in \mathcal{T}_\Sigma$, $\psi \in \Phi$. We write $\vdash t : \psi$ either if $\psi \equiv \top$ (every tree satisfies $\top$) or if the assertion $t : \psi$ can be derived from valid assertions by means of the following rules:

$$\frac{t : \psi \qquad \mu\, x.\psi \in \Phi}{t : x} \qquad \frac{t : \psi \qquad \mu\, x.\psi \in \Phi}{t : \mu\, x.\psi}$$

$$\frac{t : \psi_1 \qquad t : \psi_2}{t : \psi_1 \wedge \psi_2} \qquad \frac{t : \psi_i}{t : \psi_1 \vee \psi_2}$$

$$\frac{u : F}{a\langle u\rangle : a\langle F\rangle} \qquad \frac{u : F}{a\langle u\rangle : *\langle F\rangle}$$

Thus, besides assertions $t : \psi$, $t \in \mathcal{T}_\Sigma$, we additionally need auxiliary assertions $u : F$ where $u$ is a sequence of trees and $F$ is either a regular expression or a Presburger formula. A sequence $u = t_1 \ldots t_k$ satisfies a regular pre-condition $r$ (or $\neg r$) iff there are formulas $\psi_1, \ldots, \psi_k$ such that $t_i : \psi_i$ and the sequence $\psi_1 \ldots \psi_k$ is (not) contained in the language $\mathcal{L}(r)$ of $r$. In case of a Presburger formula $f$, we collect for every formula $\psi$ the number of children $t_i$ satisfying $\psi$ into an assignment $\sigma$. Then $u$ satisfies $f$ iff $\sigma \models f$. Thus we have:

$$\frac{t_i : \psi_i \quad (i = 1, \ldots, k) \qquad \psi_1 \ldots \psi_k \in \mathcal{L}(r)}{t_1 \ldots t_k : r} \qquad \frac{t_i : \psi_i \quad (i = 1, \ldots, k) \qquad \psi_1 \ldots \psi_k \notin \mathcal{L}(r)}{t_1 \ldots t_k : \neg r}$$

$$\frac{\sigma \models f \quad \text{where} \quad \sigma(|\psi|) = \#\{i \mid t_i : \psi\}}{t_1 \ldots t_k : f}$$

Note that according to this rule for Presburger formulas, the same tree $t_i$ may be counted several times, once for every $\psi$ such that $t_i : \psi$. A *proof* of an assertion $t : \psi$ consists of all rule applications to derive this assertion. In particular this means for $t = a\langle t_1 \ldots t_k\rangle$

[3] $\Phi$ also contains the subformulas of $\psi$ if $|\psi|$ occurs in $\phi$ and so on.

and $\psi = a\langle f \rangle$, $f$ a Presburger formula, that a proof of $t : \psi$ contains for every $i = 1, \ldots, k$, and every $\psi'$ a subproof of $\vdash\ t_i : \psi'$ – whenever it exists. Moreover, we assume that a proof always has tree-like structure. Thus, we may have several copies of a subproof for distinct occurrences of the same subtree within $t$. Finally, the language denoted by the formula $\phi$ is given by: $\mathcal{L}(\phi) = \{t \in \mathcal{T}_\Sigma \mid \vdash t : \phi\}$ . In particular, $\mathcal{L}(\top) = \mathcal{T}_\Sigma$ and $\mathcal{L}(\mu\, x.\, x) = \emptyset$. Using the convenient abbreviation "$\_$" for $\top^*$, we may write $\mu\, x.\, (a\langle \_ \rangle \vee *\langle \_\ x\ \_ \rangle)$ for the set of all trees with at least one inner node labeled $a$. Note that our fixpoint expressions do not provide an explicit notion of negation. However, we always can construct an equivalent expression with *guarded* fixpoints for which complementation is easy [23].

## 5 Presburger Automata

We recall the notion of a Presburger tree automaton (PTA) for ordered trees from [22, 11]. A *Presburger tree automaton* $A$ is a tuple $(Q, \Sigma, \delta, T)$ where, as usual, $Q$, $\Sigma$, $\delta$ and $T \subseteq Q$ are the finite set of states, the input alphabet, the transition relation and the set of accepting states of $A$, respectively. Here the transition relation $\delta$ is given by a mapping from $Q \times \Sigma$ to a pre-condition on the children of a node with label $a$ to reach $q$ in a bottom-up run over an input tree. For PTAs, pre-conditions are Boolean combinations of regular expressions $r$ over the state set $Q$ and Presburger formulas $f$ with free variables $|q|, q \in Q$. We define satisfaction relations $u \models p$ for $u \in Q^*$ and pre-conditions $p$ and $t \models_A q$ for $t \in \mathcal{T}_\Sigma, q \in Q$:

$$
\begin{array}{rl}
q_1 \ldots q_k \models r & \text{iff } q_1 \ldots q_k \in \mathcal{L}(r) \\
q_1 \ldots q_k \models f & \text{iff } \sigma \models f \text{ where } \sigma(|q|) = \#\{i \mid q_i = q\} \\
q_1 \ldots q_k \models p_1 \vee p_2 & \text{iff } q_1 \ldots q_k \models p_1 \text{ or } q_1 \ldots q_k \models p_2 \\
q_1 \ldots q_k \models p_1 \wedge p_2 & \text{iff } q_1 \ldots q_k \models p_1 \text{ and } q_1 \ldots q_k \models p_2 \\
q_1 \ldots q_k \models \neg p & \text{iff } q_1 \ldots q_k \not\models p \\
a\langle t_1 \ldots t_k \rangle \models_A q & \text{iff } t_i \models_A q_i \text{ for all } i \text{ and } q_1 \ldots q_k \models \delta(q, a),
\end{array}
$$

Note here that satisfaction of a Presburger pre-condition $f$ takes a different flavor than the corresponding definition for fixpoint formulas: In an automaton each subtree of a node takes only one state and thus contributes exactly once to the value of some $\sigma(|q|)$. Opposed to this, the variables $|\psi|$ in fixpoint formulas count every subtree on which $\psi$ holds, hence a subtree might contribute to the value of several (or no) variables.

The automaton $A$ is called *deterministic* iff for all $a \in \Sigma$ and all $u \in Q^*, u \models \delta(q, a)$ for exactly one $q \in Q$. In the proof that deterministic PTA and Presburger fixpoint formulas are equivalent we use the following notion. For a subset $B \subseteq \Phi$ of subformulas of $\phi$, define the *closure* $\mathsf{cl}(B)$ as the least superset $B'$ of $B$ such that:

- $\top \in B'$;
- If $\phi_1 \in B'$ and $\phi_2 \in B'$ then also $\phi_1 \wedge \phi_2 \in B'$, whenever $\phi_1 \wedge \phi_2 \in \Phi$;
- If $\phi_1 \in B'$ or $\phi_2 \in B'$ then also $\phi_1 \vee \phi_2 \in B'$, whenever $\phi_1 \vee \phi_2 \in \Phi$;
- If $\phi' \in B'$ then $\mu\, x.\phi' \in B'$ and $x \in B'$, whenever $\mu\, x.\phi' \in \Phi$.

Intuitively, the closure of a set $B$ of subformulas contains all subformulas which are implied by the formulas in $B$ and reachable by a (virtual) bottom-up traversal over an input tree constructing a proof for the fixpoint formula $\phi$.

**Theorem 2.** *For a tree language $L \subseteq \mathcal{T}_\Sigma$ the following statements are equivalent:*

*(1) $L = \mathcal{L}(\phi)$ for some fixpoint formula $\phi$;*
*(2) $L = \mathcal{L}(A)$ for some deterministic PTA $A$.*

*Proof.* (1) $\Rightarrow$ (2): Let $\phi$ be a Presburger fixpoint formula. We construct a PTA $A$ as follows. Let $\Psi$ denote the set of all subformulas of $\phi$ of the form $a\langle F\rangle$ or $*\langle F\rangle$. The set $Q$ of states of $A$ is given as the set of all subsets $B \subseteq \Psi$. The set $T$ of accepting states consists of all subsets $B$ such that $\phi \in \mathsf{cl}(B)$, i.e., whose closure contains the whole formula $\phi$. Given a state $B \in Q$ and $a \in \Sigma$, we determine the pre-condition $\delta(B, a)$ as

$$\delta(B,a) = \bigwedge_{\psi\in B} \delta_0(\psi, a) \ \wedge \ \bigwedge_{\psi\in\Psi\setminus B} \neg\delta_0(\psi, a)$$

where:

$$\begin{array}{ll} \delta_0(a\langle F\rangle, a) = \bar{F} & \\ \delta_0(*\langle F\rangle, a) = \bar{F} & \\ \delta_0(b\langle F\rangle, a) = \mathsf{false} & \text{if } a \neq b \end{array}$$

and $\bar{F}$ is constructed as follows. For a possibly negated regular expression $r$, we obtain $\bar{r}$ from $r$ by substituting $(B_1 \mid \ldots \mid B_m)$ for every occurrence of a formula $\psi$ if $\{B_1, \ldots, B_m\}$ is the set of all states $B$ such that $\psi \in \mathsf{cl}(B)$. For a Presburger formula $f$, let $\bar{f}$ be obtained from $f$ by substituting $\sum_{\psi\in\mathsf{cl}(B_i)} |B_i|$ for every occurrence of the free variable $|\psi|$. By construction, the resulting automaton is deterministic. We claim:

1. For every $\psi \in \Phi$, $\vdash t : \psi$ iff $t \models_A B$ for some $B \in Q$ with $\psi \in \mathsf{cl}(B)$;
2. $\vdash t_1 \ldots t_k : r$ iff $t_i \models_A B_i$ for some states $B_i$ such that $B_1 \ldots B_k \in \mathcal{L}(\bar{r})$;
3. $\vdash t_1 \ldots t_k : f$ iff $t_i \models_A B_i$ for some states $B_i$ such that $\sigma \models \bar{f}$ where $\sigma$ is the Parikh image of $B_1 \ldots B_k$.

In particular, the first item of the claim implies that $\mathcal{L}(\phi) = \mathcal{L}(A)$.
(2) $\Rightarrow$ (1): For the reverse implication, consider a deterministic PTA $A = (Q, \Sigma, \delta, F)$. W.l.o.g. we may assume that no negation occurs in preconditions. We introduce one variable $x_q$ for every state $q \in Q$. For these variables, we construct an equation system $S_A$:

$$x_q \ = \ \psi_q \ , \quad q \in Q$$

where the right-hand sides are fixpoint formulas. The semantics of such equation systems is an extension of the semantics for fixpoint formulas. The only addition is a rule:

$$\frac{t : \psi}{t : x}$$

for every equation $x = \psi$. Thus, whenever a tree satisfies the right-hand side of an equation, then it also satisfies the variable to the left. The right-hand sides $\phi_q$ of the equation system $S_A$ are constructed from the right-hand sides $\delta(q, a), a \in \Sigma$, as follows:

$$\phi_q \ = \ \bigvee_{a\in\Sigma} [\delta(q,a)]_a$$

where $[.]_a$ takes a pre-condition and returns a fixpoint formula (without fixpoints):

$$\begin{array}{ll} [r]_a & = a\langle r\{q \mapsto x_q \mid q \in Q\}\rangle \\ [f]_a & = a\langle f\{|q| \mapsto |x_q| \mid q \in Q\}\rangle \\ [p_1 \vee p_2]_a & = [p_1]_a \vee [p_2]_a \\ [p_1 \wedge p_2]_a & = [p_1]_a \wedge [p_2]_a \end{array}$$

Thus, a regular expression $r$ over states $q$ is transformed by first substituting the states by the corresponding variables and then putting a node $a$ on top. A Presburger formula is transformed by first replacing the free $|q|$ with $|x_q|$, $q \in Q$, and again putting a node $a$ on top, whereas conjunctions and disjunctions are transformed by recursively proceeding to the involved conjuncts and disjuncts, respectively. By induction on the depth of terms $t, t_1, \ldots, t_m$ and pre-conditions $p$, we prove for every $q \in Q$ and $a \in \Sigma$:

(1) $t \models_A q$ iff $t : x_q$;
(2) $t_i \models_A q_i$ for $i = 1, \ldots, m$, with $q_1 \ldots q_m \models p$ iff $a\langle t_1 \ldots t_m\rangle : [p]_a$

The first claim then proves the correctness of the construction. The only non-trivial point in the proof of the claim is the inductive step for assertion (2). □

## 6 Complexity

In this section we study the complexity of decision problems related to Presburger automata and Presburger fixpoint formulas. The complexity of testing satisfiability of arbitrary Presburger formulas is prohibitively high, since the problem is hard for non-deterministic double exponential time [6]. As we are interested to study the interplay between regular expressions, Presburger formulas and tree automata, we assume for our complexity considerations that all Presburger formulas are given as *quantifier-free formulas* (possibly with modulo predicates and subtraction). Coefficients, like $c$ in $cx$, are in binary notation, except in theorem 4, which is a special case that can be dealt with efficiently. Recall also from Lemma 1 that any quantifier-free Presburger formula can be transformed into equation normal form. The DNF transformation might yield an exponential number of disjuncts. However, each disjunct can only contain a linear number of atoms. Further, the normalization does not increase the size of the occurring numbers.

First, we consider the non-emptiness problem for Presburger automata, i.e., given a PTA $A$ it has to be checked whether $A$ accepts at least one tree. It is already PSPACE-hard to decide whether a given set of regular expressions has a non-empty intersection or whether the complement of a single regular expression is non-empty[24]. Hence, the non-emptiness problem for PTA is PSPACE-hard. Surprisingly, it can also be solved in PSPACE. For that, the following observation about the representation of Parikh images of finite word automata turns out to be useful. It follows with a pumping argument, by observing that every path in the automaton can be decomposed into a union of simple cycles and one simple path.

**Lemma 2.** *Assume $A$ is a (non-deterministic) finite word automaton with $n$ states and input alphabet of size $k$. Then the Parikh image of $\mathcal{L}(A)$ is a union of linear sets $\{\sigma_0 + \sum_{i=1}^{m} x_i \cdot \sigma_i \mid x_i \geq 0\}$ where each component of each vector $\sigma_j \in \mathbb{N}^k$ is at most $n$.*

*In particular, the number $m$ of occurring vectors is at most $n^k$.*

**Theorem 3.** *The non-emptiness problem for (non-deterministic) PTAs is* PSPACE*-complete.*

*Proof.* It remains to prove the upper bound. For that, let $A = (Q, \Sigma, \delta, T)$ be a PTA of size $n$. We call a state $q$ of $A$ *reachable*, if there is a tree $t$ such that $t \models_A q$. We have to check whether there is a reachable state in $T$. The set $R$ of reachable states can be computed in a standard fashion as follows. First, the set $R$ consists of all states $q$ such that for some single-node tree $t$, we have $t \models q$. Then, given a set $R$ of reachable states, we obtain a (possibly larger) set $R'$ of reachable states $q$ by checking whether there is a string $w$ over $R$ and a symbol $a$, such that $w \models \delta(q, a)$. This process stops after at most $|Q|$ iterations. Hence, to get the desired upper bound, it suffices to show the following claim.

*Claim.* Given a PTA $A = (Q, \Sigma, \delta, T)$, $R \subseteq Q$, $q \in Q$, $a \in \Sigma$, it can be checked in space polynomial in $|A|$, whether there is a string $w \in R^*$ such that $w \models \delta(q, a)$.
The proof of the claim proceeds in two steps. First, we show that the length of the shortest word satisfying $\delta(q, a)$ has length at most $2^{p(n)}$, $p$ a suitably defined polynomial not depending on $A$. Second, we show that checking whether there exists some $w \in R^*$ of length at most $2^{p(n)}$ with $w \models \delta(q, a)$ can be checked in polynomial space.

The precondition $\delta(q, a)$ can be written in disjunctive normal form. Each disjunct is a conjunction of regular expressions $r_1, \ldots, r_k$, negated regular expressions $\neg r'_1, \ldots, \neg r'_l$, and $m \leq n$ Presburger equations of the form $t_i = c_i$ over variables $|q|, q \in Q$, and possibly other, free variables (at most $5n$). The formula $\delta(q, a)$ has a model if and only if one of these disjuncts has a model. Since the regular expressions all occur in $A$, the sum of their sizes is less than $n$. Let $A_1, \ldots, A_k$ and $A'_1, \ldots, A'_l$ be the corresponding non-deterministic automata. Then the natural deterministic automaton $A'$ for their product has at most $2^n$ states. By Lemma 2, the Parikh image of $\mathcal{L}(A')$ is a union of linear sets $\{\sigma_0 + \sum_{i=1}^{h} x_i \sigma_i \mid x_i \in \mathbb{N}\}$, where $h < 2^{n \cdot |Q|} \leq 2^{n^2}$ and the entries of the vectors $\sigma_0, \sigma_i$ are smaller than $2^n$. Hence, a word fulfills the disjunct iff its Parikh image $\tau$ fulfills the Presburger equations and is in one of these linear sets. The latter can be expressed by:

$$|q| = \sigma_0(q) + \textstyle\sum_{i=1}^{h} x_i \cdot \sigma_i(q) \quad , \; q \in Q \; .$$

Together we have $M = m + |Q| \leq 2n$ equations with at most $N = 6n + 2^{n^2}$ variables and coefficients of values bounded by $a = 2^n$. By a result of Papadimitriou [18] such a system has a solution with numbers bounded by

$$N \cdot (M \cdot a + 1)^{2M+4} = (6n + 2^{n^2}) \cdot (2n2^n + 1)^{4n+4} = 2^{O(n^2)}$$

This proves the first step, for some polynomial $p(n) = O(n^2)$.

It remains to describe the algorithm which checks whether a string $w$ of size $2^{p(n)}$ over $Q$ exists such that $w \models \delta(q, a)$. The algorithm is non-deterministic. It simply guesses $w$ symbol by symbol. For each regular expression $r$ in $\delta(q, a)$, it computes the set of states that can be reached by the corresponding automaton $A_r$ when reading $w$. Further, for each $q' \in Q$ it counts how often $q'$ occurs in $w$. All this can be done in polynomial space without actually storing $w$. A counter keeps track of the length of $w$. In the end, it can be checked whether $w \models \delta(q, a)$. By Savitch's theorem this non-

deterministic polynomial space algorithm can be turned into a deterministic one still using polynomial space. □

Since our PTAs are deterministic and thus complementable by exchanging the sets of accepting and non-accepting states, we obtain as an immediate consequence:

**Corollary 1.** *The containment problem for deterministic PTAs is* PSPACE*-complete.* □

For certain PTAs non-emptiness can be tested in polynomial time — actually with the same complexity as for tree automata with single regular expressions as preconditions.

**Theorem 4.** *Non-emptiness can be decided in polynomial time for PTAs where every precondition is of the form $\bigvee_{i=1}^{k}(r_i \wedge f_i)$, with regular expressions $r_i$ and Presburger formulas $f_i$ in equation normal form with only one equation and coefficients represented in unary.*

Allowing coefficients in binary notation makes this problem less tractable.

**Theorem 5.** *The non-emptiness problem for PTAs as in Theorem 4 but with coefficients represented in binary is* NP*-complete.*

Now we turn to the related problem of deciding satisfiability for Presburger fixpoint formulas. Here, an EXPTIME lower bound is given by the same problem for fixpoint formulas without Presburger subformulas. The lower bound is achieved already by formulas with only one occurrence of $\mu$ (a similar result holds for model-checking $\mu$-calculus against pushdown graphs, [25]). Testing whether such formulas are satisfiable by some tree is EXPTIME-complete. Again, it turns out that adding Presburger formulas does not increase the complexity, i.e., we get the following result.

**Theorem 6.** *Satisfiability for Presburger fixpoint formulas is* EXPTIME*-complete.*

The proof follows a similar line as the one for Theorem 3. Next we show that membership for deterministic PTA as well as for fixpoint formulas can be solved efficiently. This means that properties expressed by deterministic PTA are indeed of practical use:

**Theorem 7.** *Given a tree $t$ and a deterministic PTA $A$, it can be checked in time $\mathcal{O}(|t| \cdot |A|)$ whether $t \in \mathcal{L}(A)$.*

*Proof.* Since the PTA is deterministic, it suffices to compute bottom-up the state reached by each node of $t$. Since every Presburger formula and thus, every precondition on a node with $k$ children can be evaluated in time $\mathcal{O}(k \cdot |A|)$, the claim follows. □

**Theorem 8.** *Given a tree $t$ and a fixpoint formula $\phi$, it can be checked in time $\mathcal{O}(|t| \cdot |\phi|^2)$ whether $t \models \phi$.*

*Proof.* We compute bottom-up the set of formulas satisfied by each subtree. For each node we have to simulate the NFA corresponding to regular expressions $r$, by keeping the set of reachable states of the NFA. Since each NFA is of size at most $|\phi|$, each such simulation costs at most $\mathcal{O}(|\phi|^2)$. For Presburger constraints we just need to count how many children satisfy a given subformula, which can be done in $\mathcal{O}(|\phi|)$. □

## 7 The Query Language

Fixpoint formulas allow to express properties of (document) trees. We now construct an expressive querying language which still allows for efficient algorithms to collect all matches in a tree. In the example shown in Figure 7 we might ask for all items containing "`Bartoli`". A second query could ask for items containing "`Bartoli`" and having at least three reviews. with at least three reviews, Presburger fixpoint formulas easily can express that a tree contains a node (at unkown depth) satisfying a given property. E.g., the formula $\phi_1 = *\langle \_ \texttt{"Bartoli"} \_ \rangle$ describes all elements containing "`Bartoli`". Note that text contents can be taken into account by (conceptually) considering each text character as a separate element node. We are interested in the class of all documents containing sub-documents satisfying the specific property $\phi_1$. These are described by: $\mu\, x.(*\langle \_\ x\ \_ \rangle \ \vee \ \phi_1)$.

```
<music> ...
   <classical> ...
      <opera>
         <title> The Salieri Album </title>
         <composer> Bartoli </composer>
         <review> ... </review>
         <review> ... </review>
         <review> ... </review></opera>
      <opera>
         <title> The No. 1 Opera Album </title>
         <composer> Puccini ; Verdi </composer>
         <performer> Bartoli ; Pavarotti </performer>
         <review> ... </review></opera>
   </classical> ...
</music>
<dvd> ...
   <music dvd>
      <opera>
         <title> Rossini - La Cenerentola </title>
         <performer> Bartoli </performer>
         <review> ... </review>
         <review> ... </review></opera> ...
   </music dvd>
</dvd>
```

**Fig. 1.** Part of an example document containing information about items sold by a store.

In order to indicate the sub-formula corresponding to requested sub-documents, we introduce an extra marker "•". Thus, we specify the query as $\psi_1 = \mu x.(*\langle \_\ x\ \_ \rangle \ \vee \ (\bullet \wedge \phi_1))$. Accordingly for the second query, we describe the set of all elements containing at least three reviews by: $\phi_2 = *\langle |\texttt{review}| \geq 3 \rangle$. The query formula then can be formulated as:

$$\psi_2 = \mu x.(*\langle \_\ x\ \_ \rangle \ \vee \ (\bullet \wedge \phi_1 \wedge \phi_2))$$

Thus, a query language is obtained by extending Presburger fixpoint formulas by one case:

$$\phi \quad ::= \quad \ldots \quad | \quad \bullet \quad | \quad \ldots$$

Accordingly, we add new axioms $\vdash t : \bullet$ for all trees $t$. A *match* $s$ of a formula $\phi$ containing a subformula $\bullet$ is a proof for $t : \phi$ containing the fact $s : \bullet$. We want to construct an algorithm to determine for a fixed query formula $\phi$, all matches inside a document tree $t$. We first observe that we can determine in linear time for every subtree $s$ of $t$ the set of all subformulas $\psi'$ of $\phi$ such that $\vdash s : \psi'$. For that, we could construct, e.g., the deterministic PTA $A$ for $\phi$ as considered in the last section. In order to deal with the special symbol $\bullet$ occurring in $\phi$, we extend the notion of closure of states by adding the formula $\bullet$. The rest of the construction we leave unchanged. Let then $S(s)$ denote the unique state of $A$ with $s \models_A S(s)$. By construction, $\psi' \in \mathsf{cl}(S(s))$ iff $\vdash s : \psi'$. Moreover, all these sets can be determined by a single run of $A$ over the tree $t$, i.e., in linear time.

In a second topdown pass over the tree $t$, we determine for every subtree (occurrence) $s$ the subset $R(s) \subseteq \mathsf{cl}(S(s))$ containing all those $\psi'$ which may occur in some proof of $t : \phi$. Then $s$ is a match iff $\bullet \in R(s)$. For a closed set of subformulas $B$, we introduce the auxiliary function $\mathsf{core}_B$ which takes a subformula $\psi'$ of $\phi$ and returns the set of subformulas in $B$ which potentially contribute to proofs of $\psi'$. So, $\mathsf{core}_B(\psi') = \{\psi'\} \cup \mathsf{core'}_B(\psi')$ where $\mathsf{core'}_B(\bullet) = \mathsf{core'}_B(\top) = \mathsf{core'}_B(a\langle F\rangle) = \mathsf{core'}_B(*\langle F\rangle) = \emptyset$ and:

$$\begin{array}{ll}
\mathsf{core'}_B(\mu\, x.\psi') & = \mathsf{core}_B(\psi') \\
\mathsf{core'}_B(x) & = \mathsf{core}_B(\psi') \qquad \text{if } \mu\, x.\psi' \in B \\
\mathsf{core'}_B(\psi_1 \wedge \psi_2) & = \mathsf{core}(\psi_1)_B \cup \mathsf{core}_B(\psi_2) \\
\mathsf{core'}_B(\psi_1 \vee \psi_2) & = \begin{cases} \mathsf{core}(\psi_i) & \text{if } \psi_{3-i} \notin B \\ \mathsf{core}(\psi_1) \cup \mathsf{core}(\psi_2) & \text{otherwise} \end{cases}
\end{array}$$

Moreover, we set: $\mathsf{core}_B(R) = \bigcup_{\psi \in R} \mathsf{core}_B(\psi)$ for $R \subseteq B$.

The second pass over $t$ starts at the root of $t$. There, we have: $R(t) = \mathsf{core}_B(\phi)$ for $B = \mathsf{cl}(S(t))$. Now assume we have already computed the set $R(s)$ for the occurrence $s$ of a subtree $a\langle s_1 \ldots s_k\rangle$. Let $R' = R(s) \cap \Psi$ denote the set of subformulas in $R(s)$ of the form $a\langle F\rangle$ or $*\langle F\rangle$. Then $R(s_i) = \bigcup_{\psi' \in R'} R_{\psi'}(i)$ where $R_{\psi'}(i)$ equals the set of subformulas for the $i$-th child of $s$ which may occur at $s_i$ in a proof of $s : \psi'$. If $\psi' = a\langle f\rangle$ or $\psi' = *\langle f\rangle$ for a Presburger formula $f$, then we must compute the assignment to the global variables of $f$. In fact, *all* valid sub-formulas at child trees $s_i$ contribute to this assignment. Therefore, $R_{\psi'}(s_i) = S(s_i)$ for all $i$. On the other hand, if $\psi' = a\langle r\rangle$ or $\psi' = *\langle r\rangle$ for a regular expression $r$, then $R_{\psi'}(s_i) = \mathsf{core}_{B_i}(R_i)$ where $B_i = \mathsf{cl}(S(s_i))$ and

$$R_i = \{\psi_i \mid \exists\, \psi_1 \ldots \psi_k \in \mathcal{L}(r) : \forall\, j : \psi_j \in S(s_j)\}$$

The set $R_i$ denotes all subformulas provable for $s_i$ which may contribute to the validation of $r$. From these, we take all the formulas $a\langle F\rangle$ or $*\langle F\rangle$ in $S(s_i)$ which may contribute to a proof of these. According to this definition, the sets $R_{\psi'}(s_i)$, $i = 1, \ldots, k$ can jointly be computed by a left-to-right followed by a right-to-left pass of a finite (string) automaton for $r$ over the children of $s$. The case of negated regular expressions is treated analogously. Summarizing we conclude:

**Theorem 9.** *The set of matches of a fixpoint query $\phi$ in an input tree $t$ can be computed in time linear in $|t|$. If $\phi$ is part of the input, the joint query complexity is $\mathcal{O}(|\phi|^2 \cdot |t|)$.* □

## 8 Conclusion

We have enhanced a simple fixpoint logic for unranked trees with Presburger constraints. For the basic decision problems such as satisfiability, membership and containment the resulting logic turned out to have comparable complexities to the fixpoint logic without Presburger constraints. Therefore, our logic is a promising candidate for a smooth enhancement of classical Schema and querying languages for XML documents.

It remains a challenging engineering problem to obtain an implementation of the new logic which behaves well on practical examples. Also, we would like to know more about the complexity of the satisfiability problem for other restrictions on the transition function of a PTA or the fixpoint formula to obtain further useful classes with efficient algorithms.

Since the class of tree languages defined by deterministic PTA is a strict superclass of the regular tree languages, we would also like to see other characterizations of this class.

## References

1. L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In *10th European Symposium on Programming (ESOP)*, 1–22. LNCS 2028, 2001.
2. L. Cardelli and A. Gordon. Anytime, Anywhere: Modal Logics for Mobile Ambients. In *27th ACM Conf. on Principles of Programming Languages (POPL)*, 365–377, 2000.
3. G. Conforti, O. Ferrara, and G. Ghelli. TQL Algebra and its Implementation (Extended Abstract). In *IFIP Int. Conf. on Theoretical Computer Science (IFIP TCS)*, 422–434, 2002.
4. 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.
5. S. Dal-Zilio, D. Lugiez, and C. Meyssonnier. A Logic you can Count on. In *31st ACM Symp. on Principles of Programming Languages (POPL)*, 135–146, 2004.
6. M.J. Fischer and M.O. Rabin. Superexponential Complexity of Presburger Arithmetic. In *AMS Symp. on the Complexity of Computational Computational Processes. Vol. 7*, 27–41, 1974.
7. S. Ginsburg and E.H. Spanier. Semigroups, Presburger Formulas and Languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
8. G. Gottlob and C. Koch. Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. In *PODS*, 17–28, 2001.
9. F. Klaedtke and H. Ruess. Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints. Technical Report 177, Institute of CS at Freiburg University, 2002.
10. O. Kupferman, U. Sattler, and M.Y. Vardi. The Complexity of the Graded $\mu$-Calculus. In *18th Int. Conf. on Automated Deduction (CADE)*, 423–437. LNCS 2392, 2002.
11. D. Lugiez and S. Dal Zilio. XML Schema, Tree Logic and Sheaves Automata. In *14th Int. Conf. on Rewriting Techniques and Applications (RTA)*, 246–263. LNCS 2706, 2003.
12. W. Martens and F. Neven. Typechecking Top-down Uniform Unranked Tree Transducers. In *ICDT*, 64–78, 2003.

13. A. Neumann and H. Seidl. Locating Matches of Tree Patterns in Forests. TR 98-08, Trier, 1998.
14. F. Neven. Automata, Logic, and XML. In *16th Int. Workshop CSL*, 2–26. LNCS 2471, 2002.
15. F. Neven and T. Schwentick. Query Automata over Finite Trees. *TCS*, 275(1-2):633–674, 2002.
16. 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.
17. J. Niehren and A. Podelski. Feature Automata and Recognizable Sets of Feature Trees. In *4th TAPSOFT*, 356–375. LNCS 668, 1993.
18. C.H. Papadimitriou. On the Complexity of Integer Programming. *J. ACM*, 28(4):765–768, 1981.
19. R. J. Parikh. On Context-free Languages. *J. of the ACM*, 13(4):570–581, 1966.
20. M. Presburger. On the Completeness of a Certain System of Arithmetic of Whole Numbers in which Addition Occurs as the only Operation. *Hist. Philos. Logic*, 12:225–233, 1991. English translation of the original paper from 1929.
21. H. Seidl. Haskell Overloading is DEXPTIME Complete. *Inf. Proc. Lett. (IPL)*, 54:57–60, 1994.
22. H. Seidl, T. Schwentick, A. Muscholl. Numerical Document Queries. In *PODS*, 155–166, 2003.
23. H. Seidl and A. Neumann. On Guarding Nested Fixpoints. In *Ann. Conf. of the European Association of Logic in Computer Science (CSL)*, 484–498. LNCS 1683, 1999.
24. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC*, 1–9, 1973.
25. Igor Walukiewicz. Pushdown Processes: Games and Model-Checking. *Information and Computation*, 164(2):234–263, 2001.