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. In [22], it was shown that MSO logic for ordered unranked trees becomes undecidable if Presburger constraints are allowed at children of nodes. We now show that a decidable logic is obtained if we use a a modal fixpoint logic instead. We present an automata theoretic characterization of this logic by means of *deterministic* Presburger tree automata (PTA) 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 logic. The non-emptiness for PTAs is in general 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 φ is polynomial in the size of φ 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 set of a finite automaton.

Keywords: Querying XML documents, fixpoint logic, Presburger arithmetic, automata.

⋆ Contact author. Address: Philipps-Universitt Marburg, Fachbereich Mathematik und Informatik, Hans-Meerwein-Strae, 35032 Marburg, Germany, tick@informatik.uni-marburg.de

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

1 Introduction

In XML schema description languages as DTDs and XML Schema, the content types 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 `(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 `(title author author`$^+$`) | (author`$^+$` title author`$^+$`) | (author author`$^+$` title)`. It would be desirable to describe this condition simply by an expression like $|\texttt{title}| = 1 \wedge |\texttt{author}| \geq 2$.

Whereas these conditions do not go beyond the scope of regular expressions, other simple ones do. E.g., it is of course not possible to express a condition like $|\texttt{author}| \leq 2 \cdot |\texttt{title}|$.

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]. In these formalisms the interaction between the children of a node and the node itself are usually expressed in terms of a regular expression. 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 logic.

In the present paper 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. It turned out that, whereas their non-emptiness problem (whether an automaton accepts some tree) is decidable, the universality problem (whether it accepts all trees) is not. Consequently, monadic second-order logic extended by such conditions has undecidable satisfiability. In the present paper, we study two weaker formalisms. We consider a fixpoint logic instead of monadic second-order logic and deterministic tree automata instead of non-deterministic tree automata. We refer to them as Presburger fixpoint formulas and Presburger tree automata, respectively.

It turns out that both define the same class of tree languages. Furthermore, their non-emptiness (resp., satisfiability) problem becomes 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 Presburger tree automata 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 ϕ can be decided in time $O(|t||A|)$ and $O(|t||\phi|^2)$, respectively.

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

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

Furthermore, we show how Presburger fixpoint formulas can be adapted to allow the formulation of unary queries. These queries can be evaluated time 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.

The paper is organized as follows. In Section 2 we give the basic definitions for Presburger logic. Section 3 explains how from an NFA a Presburger formula can be obtained. 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 short conclusion. Due to space restriction many proofs are given in an appendix.

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] (and their co-workers). 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, 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 [10]. The numbers n there, however, are hard-coded into the formula. Orderings on the successors is 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 for ordered trees, 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 automata. Seidl et al. in [22] on the other hand, prove that Presburger tree automata 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 Presburger automata.

2 Preliminaries

.

Presburger Logic is first-order logic over the structure $(\mathbb{N}, \leq, +)$. Given a formula f and an *assignment* σ mapping the variables of f to numbers, we write $\sigma \models f$ if f holds for σ (in the obvious sense) and call σ a solution of f.

For convenience, we use an extended language and make use of some abbreviations. We allow to write cx for $x + \cdots + x$ (c times) and we also 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 $x \equiv 0 \pmod 2$.

In this paper, 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 disjunct 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.*

Proof. Let f be a quantifier-free Presburger formula. We bring it into disjunctive normal form (DNF). Then we replace atomic and atomic negated formulas by equations, if necessary by introducing new existentially quantified variables. E.g., $t < c$ can be replaced by $\exists y_1(t + 1 + y_1 = c)$, $t \neq c$ by $\exists y_2(t + y_2 + 1 = c \vee t - y_2 - 1 = c)$. Further, $t \equiv c \pmod d$ can be replaced by $\exists y_3(t - dy_3 = c \vee t + dy_3 = c)$ and $t \not\equiv c \pmod d$ by

$$\exists y_4, y_5(t - dy_4 - y_5 = 0 \vee t + dy_4 - y_5 = 0) \wedge (y_5 < c \vee (y_5 > c \wedge y_5 < d)).$$

Note that the resulting formula is in general not in DNF, but it is free of negations and can be easily transformed into equation normal form. Note that, although the size of the formula might increase, the maximum number of conjuncts in a conjunction remains the same. $\square$

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 σ and the σ_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 in this paper can use conditions on the children of a node which are Boolean combinations of regular expressions and Presburger formulas. Whereas it is well-known that the Parikh image of a regular language (and even context-free language, [19]) is semilinear and thus can be described by a formula from Presburger arithmetic 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 even for NFA, a Presburger formula which describes the Parikh image of the corresponding language can be computed in linear time. In particular, the formula is of *linear* size.

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

Proof. Let $A = (Q, \Sigma, \delta, F, q_0)$ be a non-deterministic finite string automaton (NFA). With an accepting run of A on a string w we can 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 that

- f is locally consistent, e.g., for each inner node the incoming flow equals the outgoing flow, and
- the subgraph induced by the states with a non-zero flow is connected. This is done by guessing numbers corresponding to the distance of nodes from s w.r.t. non-zero flow edges.

Details are given in the appendix. $\square$

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 Σ of element or node names, the set of all (ordered but unranked) trees t is given by:

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

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. Fixpoint formulas φ are constructed according to the following grammar:

$$\begin{array}{l} \varphi ::= \top \quad | \quad x \quad | \quad \mu\, x.\, \varphi \quad | \quad \varphi_1 \vee \varphi_2 \quad | \quad \varphi_1 \wedge \varphi_2 \quad | \quad a\langle F\rangle \quad | \quad *\langle F\rangle \\ F ::= r \quad | \quad \neg r \quad | \quad f \end{array}$$

Here, "$*$" denotes an arbitrary node label, 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 φ, φ a fixpoint formula, or a Presburger formula f with free variables $|\varphi|$ denoting the number of children satisfying φ. Essentially the same calculus is obtained if we enhance the *Sheaves logic* of Dal Zilio and Lugiez [11] with recursion.

In the following, we assume throughout that φ is a formula where all bound variables are distinct. Let Φ denote the set of all subformulas of φ plus $\top$ (the constant true). 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:

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

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

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

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 of formulas $\psi_1 \ldots \psi_k$ is (not) contained in the language $\mathcal{L}(r)$ of r. In case of a Presburger formula f, we collect for every formula ψ the number of children t_i satisfying ψ. Then u satisfies f iff the resulting variable assignment σ makes f true. Thus we have the rules:

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

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

Note that according to this rule for Presburger formulas, the same tree t_i may be counted several times, once for every ψ 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 $t = a\langle t_1 \ldots t_k\rangle$ 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 ψ' a subproof of $\vdash t_i : \psi'$ – whenever it exists. Moreover, we silently 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 φ is given by:

$$\mathcal{L}(\varphi) = \{t \in \mathcal{T}_\Sigma \mid \vdash t : \varphi\}$$

In particular, $\mathcal{L}(\top) = \mathcal{T}_\Sigma$ and $\mathcal{L}(\mu\, x.\, x) = \emptyset$. Using the convenient abbreviation "_" for $\top^*$, i.e., an arbitrary sequence of trees, 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, Σ, δ, T) where, as usual, Q, Σ, δ 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 δ 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. In case of PTA, such a pre-condition is a Boolean combination of regular expressions r over the state set Q and Presburger formulas f with free variables $|q|, q \in Q$. We define satisfaction relations $t \models_A q$ now for trees t and states q and $u \models p$ for sequences of states $u \in Q^*$ and pre-conditions p:

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

It should be noted 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 $\sigma(|q|)$. As opposed to this, the variables $|\psi|$ in fixpoint formulas count every subtree on which ψ 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 $\alpha \in Q^*$, $\alpha \models \delta(q, a)$ 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 ϕ, define the *closure* $\mathsf{cl}(B)$ as the least superset B' of B such that:

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

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 ϕ.

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 ϕ;

(2) $L = \mathcal{L}(A)$ for some deterministic PTA A.

Proof. (1) $\Rightarrow$ (2): Let ϕ be a Presburger fixpoint formula. We assume for simplicity that all regular expressions in ϕ are unnegated. We construct a PTA A as follows. Let Ψ denote the set of all subformulas of ϕ 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 ϕ.

Given a state $B \in Q$ and $a \in \Sigma$, we determine the pre-condition $\delta(B, a)$ as

$$\delta(B, a) = \bigwedge_{\psi \in B} \delta(\psi, a) \;\wedge\; \bigwedge_{\psi \notin B} \neg\delta(\psi, a)$$

where:

$$\begin{aligned} \delta(a\langle F\rangle, a) &= \bar{F} \\ \delta(*\langle F\rangle, a) &= \bar{F} \\ \delta(b\langle F\rangle, a) &= \mathsf{false} \qquad \text{if } a \neq b \end{aligned}$$

where $\bar{F}$ is constructed as follows. For a regular expression r, we obtain $\bar{r}$ from r by substituting $(B_1 \mid \ldots \mid B_m)$ for every occurrence of a formula ψ 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 $\sum_{\psi \in \mathsf{cl}(B)} |B|$ 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 σ is the Parikh image of $B_1 \cdots B_k$.

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 expressions. The semantics of such equation systems is an extension of the semantics for fixpoint expressions. 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 ϕ_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 the transformation $[.]_a$ takes a pre-condition and returns a fixpoint expression (without fixpoints) as follows:

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

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

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). The remaining details can be found in the appendix. $\square$

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, if the size of the alphabet is k, then the number m of occurring vectors is at most n^k.

Using Lemma 2, we obtain:

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

Proof. It remains to prove the upper bound. For that, let $A = (Q, \Sigma, \delta, T)$ be a PTA. Let n denote the size of A.

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 $(t_i = c_i)_i$ 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 this 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 minimal deterministic automaton A' for their product has at most 2^n states. By Lemma 2, the Parikh image of $L(A')$ 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 σ_0, σ_i are smaller than 2^n. Hence, a word fulfills $\delta(q,a)$ if and only if its Parikh image τ is in one of these linear sets and additionally fulfills the Presburger equations. This can be expressed by adding, for each $q \in Q$, the equation

$$|q| = \sigma_0(q) + \sum_{i=1}^{h} x_i \cdot \sigma_i(q).$$

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 PTA 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 Presburger automata is complete for* PSPACE. □

There is a special type of PTA with a tractable non-emptiness test which might be relevant for many practical cases. Its complexity is the same as that for traditional tree automata which have a single regular expression as precondition.

Theorem 4. *The non-emptiness problem can be solved in polynomial time for PTA, in which 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.*

The proof can be found in the appendix.

Allowing coefficients in binary notation makes this problem less tractable.

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

A proof sketch is given in the appendix.

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 μ (a similar result holds for model-checking μ-calculus against pushdown graphs, [25]). Testing whether such formulas are satisfiable by some tree is complete for EXPTIME. 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 formula is* EXPTIME*-complete.*

The proof follows a similar line as the one for Theorem 3. It is given in the appendix.

Next we show that membership for deterministic PTA as well as for fixpoint expressions 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 supposed to be deterministic, it suffices to compute bottom-up the state reached by each node of t. Since all Parikh vectors have entries at most $|t|$, every Presburger formula and thus, every precondition on a node with k children can be evaluated in time $\mathcal{O}(k \cdot |A|)$, which yields the claim. □

Theorem 8. *Given a tree* t *and a fixpoint formula* ϕ*, 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. For Presburger constraints we just need to count how many children satisfy a given subformula. □

7 The Query Language

Fixpoint expressions allow to express properties of (document) trees. Let us now show how an expressive querying language can be obtained 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 item containing "`Bartoli`" and having at least three reviews. In our fixpoint Presburger logic we can easily express that a tree contains a node satisfying a given property, without knowing at which depth this node occurs. For instance, the formula $\phi_1 = *\langle \_\ \texttt{Bartoli}\ \_ \rangle$ describes all elements containing "`Bartoli`". Note that in order to take properties of text contents into account, it (conceptually) suffices to consider each text character as a separate element node. We are not interested in the class of all these documents t, however, but for each such t in the sub-documents which satisfying the specific ϕ_1. Documents containing elements with the property ϕ_1 are described by the expression: $\mu\, x.(*\langle \_\ x\ \_ \rangle \ \vee\ \phi_1)$.

```
<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 </name> </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>
```

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

In order to indicate the sub-expression corresponding to the 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 expression then can be formulated as:

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

In order to obtain a query language, we therefore formally extend the language of Presburger fixpoint expressions by one extra case:

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

Accordingly, we add new axioms $\quad \vdash t : \bullet \quad$ for all trees t. A *match* s of a formula φ containing a subformula $\bullet$ is a proof for $t : \varphi$ containing the fact $s : \bullet$. We want to construct an algorithm to determine for a fixed query expression φ, 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 ψ' of φ such that $\vdash s : \psi'$. For that, we could construct, e.g., the deterministic PTA A for φ as considered in the last section. In order to deal with the special symbol $\bullet$ occurring in φ, 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 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.

It remains to determine for every subtree (occurrence) s the subset $R(s) \subseteq \mathsf{cl}(S(s))$ containing all those ψ' which may occur in some proof of $t : \varphi$. Then s is a match iff $\bullet \in R(s)$. The subsets $R(s)$ are determined in a second topdown pass over the tree t. For a closed set of subformulas B, we introduce the auxiliary function core_B which takes a subformula ψ' of φ and returns the set of all subformulas in B which potentially contribute to any proof of ψ'. So, $\mathsf{core}_B(\psi') = \{\psi'\} \cup \mathsf{core}'(\psi')$ where $\mathsf{core}'_B(\bullet) = \mathsf{core}'_B(\top) = \emptyset$ and:

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

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: $R(t) = \mathsf{core}_B(\varphi)$ 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 all child trees s_i contribute to this assignment. Therefore, we simply have: $R_{\psi'}(s_i) = S(s_i)$ 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 φ in an input tree t can be computed in time linear in $|t|$. If φ is part of the input, the joint query complexity is $\mathcal{O}(|\varphi|^2 \cdot |t|)$.* □

8 Conclusion

We have enhanced a simple fixpoint logic for unranked trees with Presburger constraints. For the basic decision problems such as satifiability, membership and containment the resulting logic turned out to have comparable complexities as the fixpoint logic without Presburger constraints. Therfore, 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 PTAs 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)*, pages 1–22. LNCS 2028, Springer Verlag, 2001.
2. L. Cardelli and A. Gordon. Anytime, Anywhere: Modal Logics for Mobile Ambients. In *27th ACM Conf. on Principles of Programming Languages (POPL)*, pages 365–377, 2000.
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)*, pages 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. Silvano Dal-Zilio, Denis Lugiez, and Charles Meyssonnier. A Logic you can Count on. In *31st ACM Symp. on Principles of Programming Languages (POPL)*, pages 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*, pages 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 2001*, pages 17–28, 2002.
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 μ-Calculus. In *18th Int. Conf. on Automated Deduction (CADE)*, pages 423–437. LNCS 2392, Springer Verlag, 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)*, pages 246–263. LNCS 2706, Springer Verlag, 2003.
12. Wim Martens and Frank Neven. Typechecking top-down uniform unranked tree transducers. In *ICDT*, pages 64–78, 2003.
13. Andreas Neumann and Helmut Seidl. Locating Matches of Tree Patterns in Forests. Technical Report 98-08, University of Trier, 1998.
14. F. Neven. Automata, logic, and xml. In *16th International Workshop CSL*, pages 2–26. LNCS 2471, Springer, 2002.
15. F. Neven and T. Schwentick. Query Automata over Finite Trees. *Theoretical Computer Science (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 Int. Conf. on Theory and Practice of Software Development (TAPSOFT)*, pages 356–375. LNCS 668, Springer Verlag, 1993.
18. Christos 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. *Information Processing Letters (IPL)*, 54:57–60, 1994.
22. H. Seidl, T. Schwentick, and A. Muscholl. Numerical Document Queries. In *22nd ACM Conference on Principles of Database Systems (PODS)*, pages 155–166, 2003.
23. Helmut Seidl and Andreas Neumann. On Guarding Nested Fixpoints. In *Ann. Conf. of the European Association of Logic in Computer Science (CSL)*, pages 484–498. LNCS 1683, 1999.
24. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *STOC 73*, pages 1–9, 1973.
25. Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.

A Appendix

A.1 Proof of Theorem 1 in Section 3.

A *flow* for A is a triple (f, s, t), where s, t are states of A and f maps triples (p, a, q) with $q \in \delta(p, a)$ to natural numbers. We write:

$$\mathsf{in}_f(q) = \sum_{\substack{p \in Q, a \in \Sigma \\ q \in \delta(p,a)}} f(p, a, q) \quad \text{and} \quad \mathsf{out}_f(p) = \sum_{\substack{q \in Q, a \in \Sigma \\ q \in \delta(p,a)}} f(p, a, q)$$

A flow (f, s, t) is *consistent* if, for each $p \in Q$, at least one of the following holds.

- $\mathsf{in}_f(p) = \mathsf{out}_f(p)$,
- $p = s$ and $\mathsf{in}_f(p) = \mathsf{out}_f(p) - 1$, or
- $p = t$ and $\mathsf{in}_f(p) = \mathsf{out}_f(p) + 1$.

A state p *occurs* in (f, s, t) if $p \in \{s, t\}$ or $\mathsf{in}_f(p) > 0$. A flow is *connected*, if the undirected graph G which has the occurring states as vertices and edge set $\{\{p, q\} \mid f(p, a, q) > 0, \text{ for some } a \in \Sigma\}$ is connected.

Claim 1. A Parikh vector σ is in the Parikh image of $\mathcal{L}(A)$ if and only if there is a consistent and connected flow (f, s, t), such that

(a) $s = q_0$, $t \in F$, and
(b) for each $a \in \Sigma$, $\sigma(|a|) = \sum_{q \in \delta(p,a)} f(p, a, q)$.

Let w be a string which is accepted by A and let $\sigma(w)$ be its Parikh image. Let (f, s, t) be induced by an accepting run of A on w: $f(p, a, q)$ is the number of times A goes from state p to state q while reading symbol a, $s = q_0$ and t is the accepting state in which the run ends. It is straightforward that (f, s, t) is consistent, connected and fulfills (a) and (b).

For the opposite conclusion let (f, s, t) be a consistent and connected flow fulfilling (a) and (b). Let G be the multigraph induced by f as follows. The vertices of G are s, t and all p with $\mathsf{in}_f(p) > 0$. The number of edges from p to q is $\sum_{a \in \Sigma} f(p, a, q)$. As (f, s, t) is connected, G is connected as well. From consistency it follows that G is Eulerian, hence there is a path from s to t which traverses each edge exactly once. Obviously, using (a), such a path corresponds to an accepting run of A on a string w with $\sigma(w) = \sigma$. This finishes the proof of claim 1.

Claim 2. For each NFA A, an existential Presburger formula ψ_A with the following properties can be constructed in linear time in $|A|$:

- ψ_A has free variables $x_{(p,a,q)}$, where $p, q \in Q$, $a \in \Sigma$, $q \in \delta(p, a)$, and
- for every assignment ρ of natural numbers to these variables, $\rho \models \psi_A$ if and only if there are $s, t \in Q$ such that
  - $s = q_0$, $t \in F$, and
  - (f_ρ, s, t) is a connected and consistent flow for A, where $f_\rho(p, a, q) = \rho(x_{(p,a,q)})$ whenever $q \in \delta(p, a)$.

The formula ψ_A uses variables z_p for each p. The intended meaning is that $z_p = 1$ if $p = t \in F$, otherwise $z_p = 0$ (this avoids a quadratic blow-up). It contains the subformulas $\sum_{p \in F} z_p = 1$, $\sum_{p \notin F} z_p = 0$, as conjuncts. It is now straightforward to define another conjunct which checks that ρ together with the z_p corresponds to a consistent flow.

It only remains to be checked that the graph G is connected. It is easy to see that this is the case if and only we can label each node of Q by a natural number such that the following holds.

– s gets 0,
– each node of G besides s gets a number > 0,
– each node of G has a neighbour in G with a smaller number.

We can construct an existential Presburger formula of linear size with new variables u_p, for each $p \in Q$, which expresses these properties. Furthermore, it is straightforward that (f, s, t) is connected if and only if this formula holds. We take this formula as another conjunct of ψ_A. This completes the proof of claim 2.

The formula φ_A is now easily obtained as

$$\exists (x_{p,a,q})_{q\in\delta(p,a)} \;\; \psi_A \wedge \bigwedge_{a\in\Sigma} \sigma(|a|) = \sum_{p,q} x_{p,a,q},$$

where the quantification is over all variables $x_{p,a,q}$ with $q \in \delta(p, a)$. □

A.2 Remaining Proof of Theorem 2 in Section 5.

Assume $t_i \models_A q_i$ for $i = 1, \ldots, m$, and $\alpha = q_1 \ldots q_m \models p$. By induction on the structure of p, we verify that for every $a \in \Sigma$, $a\langle t_1 \ldots t_m\rangle : [p]_a$ where, by inductive hypothesis, we may assume that $t_i : x_{q_i}$ for $i = 1, \ldots, m$. Now, if p equals a regular expression r, then by assumption, $q_1 \ldots q_m \in \mathcal{L}(r)$. By definition, $[p]_a = a\langle r\{q \mapsto x_q \mid q \in Q\}\rangle$. Therefore, $x_{q_1} \ldots x_{q_m} \in \mathcal{L}(r\{q \mapsto x_q \mid q \in Q\})$ and hence $a\langle t_1 \ldots t_m\rangle : [p]_a$. If p equals a Presburger formula f, then the Parikh image of $q_1 \ldots q_m$ satisfies f. Let ρ denote the mapping defined by $\rho(|x_q|) = \#\{i \mid t_i : x_q\}$. Since the automaton A is deterministic, $t_i : q$ is provable for exactly one state q. Therefore, the number of occurrences of q in the sequence $q_1 \ldots q_m$ precisely equals $\rho(|x_q|)$. We conclude that $t_1 \ldots t_m \models f\{|q| \mapsto |x_q| \mid q \in Q\}$ and therefore also $a\langle t_1 \ldots t_m\rangle : [p]_a$. The cases $p \equiv p_1 \wedge p_2$ and $p \equiv p_1 \vee p_2$ are completely standard. For the converse direction assume $a\langle t_1 \ldots t_m\rangle : [p]_a$ for some $a \in \Sigma$. By inductive hypothesis for t_i, we already know that there are (unique) states q_i such that $t_i \models_A q_i$ and therefore also $t_i : x_{q_i}$, $i = 1, \ldots, m$. It remains to verify that $q_1 \ldots q_m \models p$. Again we perform an induction on the structure of p. Consider, e.g., the case where p equals a Presburger formula f. Then $[p]_a \equiv a\langle f\{|q| \mapsto |x_q| \mid q \in Q\}\rangle$. Since by assumption, $a\langle t_1 \ldots t_m\rangle : [p]_a$, $\rho \models f\{|q| \mapsto |x_q| \mid q \in Q\}$ for $\rho(|x_q|)\#\{i \mid t_i : x_q\}$, $q \in Q$. Since A is deterministic, $\rho(|x_q|)$ equals the number of occurrences of q in the sequence $q_1 \ldots q_m$. Therefore, $q_1 \ldots q_m \models f$.

To the equation system S_A, we then apply *Gaussian elimination*. Thus, we take any equation $x_q = \phi_q$ where ϕ_q possibly contains free occurrences of x_q and replace it with $x_q = \mu\, x_q.\phi_q$. Then we replace all free occurrences of x_q in all other right-hand sides $\phi_{q'}, q' \neq q$, with the new fixpoint formula $\mu\, x_q.\phi_q$. The resulting system still is equivalent to the original one but does no longer contain free occurrences of x_q in right-hand sides. This we iteratively perform for every state q'. Eventually, we arrive for each $q \in Q$ at an equation $x_q = \bar{\phi}_q$ where $\bar{\phi}_q$ is a closed fixpoint expression which denotes the set $\{t \in \mathcal{T}_\Sigma \mid t \models_A q\}$. Thus, the desired expression ϕ can be chosen as:

$$\phi \;\equiv\; \bigvee_{q\in F} \bar{\phi}_q$$

This completes the proof. □

A.3 Proof sketch of Theorem 4 in Section 6.

Given such a PTA the general inductive strategy as in the proof of Theorem 3 can be used to compute the set of reachable states. It remains to show that we can check in polynomial time whether there is an i and a string $w \in R^*$ with $w \models r_i \wedge f_i$, where R is the set of states which are already known as reachable. We can do this by constructing a one-counter automaton

C (with integer counter) which accepts exactly all strings w with $w \models r_i \wedge f_i$. In order to construct C we first bring f_i into a form

$$a_0 + \sum_{a \in R} c_a|a| + \sum_{i=1}^{m} b_i z_i = 0,$$

where each c_a, b_i is an integer (in unary notation) and each z_i is existentially quantified. C reads w and and checks whether it fulfils r_i. Simultaneously, it maintains in its counter the number $a_0 + \sum_{a \in R} c_a|a|$, where each $|a|$ is the number of a's seen so far. If it reads a then it adds c_a to the counter. At the end of the computation it adds an arbitrary multiple of b_i to the counter, for each i. It accepts with counter zero.

Testing whether $L(C) \neq \emptyset$ can then be done in polynomial time.

A.4 Proof sketch of Theorem 5 in Section 6.

Upper bound: We can check whether there is a string $w \models r \wedge f$ as follows. The Parikh image of $L(r)$ is the union of linear sets with coefficients of polynomial size (in the size of r). By guessing a linear set, verifying that it is contained in the Parikh image of $L(r)$ and then substituting the variables $|a|$ in f by linear linear representation we get a single equation with a solution of polynomila length [18].
Lower bound: There is a reduction from Exactly 1-in-3 SAT. Given a positive 3-SAT formula φ with n clauses, we construct a regular expression as follows. For each variable x of φ, let $m(x) = i_1 \cdots i_k$ be the string of numbers i_j such that x occurs in C_{i_j}. Let r be the concatenation of all the $m(x)$ in any order. Now, φ has an assignment which picks exactly one literal from each clause, if and only if $L(r)$ contains a string in which each letter from $\{1, \ldots, n\}$ occurs exactly once. This can be checked by the equation $\sum_{i=1}^{n} 4^i|i| = \sum_{i=1}^{n} 4^i$. □

A.5 Proof of Theorem 6 in Section 6.

As already mentioned the lower bound follows from the lower bound for fixpoint formulas without Presburger formulas. The proof may proceed along the same lines as the proof of EXPTIME-hardness in [21], i.e., we consider an alternating linear space-bounded Turing machine M. W.l.o.g., we may assume that M uses precisely n tape cells, that existential and universal states strictly alternate and the initial state is existential, that all universal states have precisely two successor configurations, and that no transitions are possible in accepting states. Then one configuration $(q, a_1 \ldots a_{i-1}\#a_i \ldots a_n)$ (q a state of M, a_j tape symbols and $\#$ marking the position of the head) can be represented as a segment in a unary tree: $a_1\langle \ldots a_{i-1}\langle q\langle a_i\langle \ldots a_n\langle \ldots \rangle \ldots \rangle\rangle\rangle \ldots \rangle$. These segments then can be composed to represent alternating computation trees of M. Thus, it suffices to construct in polynomial time, a fixpoint expression ϕ_M such that $t \models \phi_M$ if and only if t represents an accepting computation of M on a given input $w = w_1 \ldots w_n$. Indeed, such a formula can be constructed as:

$$\phi_M \;\equiv\; \mathsf{init}_w \wedge \big(\mu\, x.\, \mathsf{acc} \vee \big(\mathsf{correct}_\exists \wedge *^{n+1}\langle \mathsf{acc} \vee \big(\mathsf{correct}_\forall \wedge *^{n+1}\langle x, x\rangle\big)\rangle\big)\big)$$

where $\mathsf{init}_w \equiv q_0\langle w_1\langle \ldots w_n\langle \top \rangle \ldots \rangle\rangle$ (q_0 the initial state of M) is a closed fixpoint formula describing the initial configuration of M; acc describes the set of all trees $a_1\langle \ldots a_{i-1}\langle q\langle a_i\langle \ldots a_n\langle\rangle \ldots \rangle\rangle\rangle \ldots \rangle$ where q is an accepting state; $\mathsf{correct}_\exists$ describes the set of all trees starting with two configurations connected by an existential transition of M; and finally, $\mathsf{correct}_\forall$ describes the set of all trees starting with three configurations connected by a universal transition of M. Each of these formulas can be written down in polynomial time.

Interestingly enough, neither of the formulas $\mathsf{init}, \mathsf{acc}, \mathsf{correct}_\exists$ or $\mathsf{correct}_\forall$ contain fixpoints. Thus, satisfiability even for formulas with just one occurrence of μ is hard for EXPTIME.

It remains to prove the exponential upper bound. The general idea is related to the proof of Theorem 3. Let φ be a Presburger fixpoint formula. Let Ψ denote the set of its subformulas of the types $a\langle F\rangle$ and $*\langle F\rangle$ and Φ the set of *all* subformulas.

We call a subset $B \subseteq \Psi$ *obtainable* if there is a tree t such that, for each $\psi \in \Psi$, $\vdash t : \psi$ if and only if $\psi \in B$. In this case, we call t a *witness for* B and denote t by $t(B)$.

We compute in an inductive fashion the set of all obtainable sets $B \subseteq \Psi$. First, we compute the set X_0 of sets that are obtainable by some one-node tree t. Given X_i, we let X_{i+1} be the set of sets that are in X_i or are obtainable by a tree consisting of a root the subtrees of which are witnesses for the sets in X_i. As this process is monotonic it ends after at most $2^{|\Psi|}$ iterations, i.e., an exponential number of steps.

It therefore suffices to prove that each step takes no more than exponential time as well. Let X denote a set of obtainable subsets of Ψ. We show that, given a fixpoint formula ϕ of size n together with a test for membership in X, and a set $B \subseteq \Psi$, it can be checked in space polynomial in n, whether B is obtainable by a tree with subtrees which are witnesses for sets in X.

Of course, B is only obtainable if there is some symbol a such that all formulas in B are either of the form $a\langle F\rangle$ or $*\langle F\rangle$. Accordingly we must check whether there is a string $w = B_1 \cdots B_h$ over the alphabet X such that the tree $t = a\langle t(B_1) \cdots t(B_h)\rangle$ makes all formulas in B true and all others false. Let H denote the mapping which takes an assignment $\sigma : X \to \mathbb{N}$ and computes an assignment $\tau : \Phi \to \mathbb{N}$ by

$$\tau(\phi') = \sum_{\phi' \in \mathsf{cl}(B'), B' \in X} \sigma(B')$$

Then t satisfies the formula $a\langle r\rangle$, r a regular expression, iff $w \in \mathcal{L}(\bar{r})$ where $\bar{r}$ is obtained from r by replacing every formula ϕ' with the disjunction of all $B' \in X$, $\phi' \in \mathsf{cl}(B')$. Likewise for $a\langle \neg r\rangle$. Moreover, t satisfies the formula $a\langle f\rangle$, f a Presburger formula, iff $H(\pi(w))$ satisfies f.

As in the proof of Theorem 3, we claim that if such a string w exists which simultaneously verifies the formulas $a\langle F\rangle \in B$ or $*\langle F\rangle \in B$ and falsifies all other such formulas potentially occurring as pre-conditions of a node label a or $*$, then there exists one such witness string w' whose length is bounded by $2^{p(n)}$ for some polynomial p.

We first show how the statement of the theorem follows from this claim. Recall that for Theorem 3, the string w of bounded length consisted of individual states of the PTA (polynomially many) whereas now we use *sets* of subformulas as letters of which there might be *exponentially many*. We successively guess subsets $B' \subseteq \Psi$ which are in X (this can be done by means of the oracle for X). For each such B', we simulate the evaluations of the nondeterministic automata corresponding to the regular expressions r occuring in $a\langle F\rangle \in \Psi$ or $*\langle F\rangle \in \Psi$. During this evaluation, we maintain an occurrence vector τ indexed by subformulas ϕ' of φ. Whenever a set B' is processed, we increment in τ the values of all ϕ' contained in the closure $\mathsf{cl}(B')$. Since each letter B' may have incremented each entry of τ at most by 1, the assignment τ can always be represented in polynomial space. Once we have guessed a sequence of length at most $2^{p(n)}$ verifying the formulas $a\langle F\rangle \in B$ and $*\langle F\rangle \in B$ that are based on regular expressions, we verify that τ satisfies the formula

$$\Big(\bigwedge_{a\langle f\rangle \in B \vee *\langle f\rangle \in B} f\Big) \wedge \Big(\bigwedge_{a\langle f\rangle \notin B \wedge *\langle f\rangle \notin B} \neg f\Big)$$

The latter can be done even in polynomial time. Since this algorithm uses only extra space polynomial in n, it can be executed in deterministic exponential time — which we wanted to prove.

It remains to prove the existence of a witness string of length at most $2^{p(n)}$. Let F denote a Boolean combination of regular expressions over sub-formulas of φ and quantifier-free Presburger formulas with free variables of the form $|\phi'|$, ϕ' a subformula of φ whose size is bounded by n (the size of the fixpoint formula φ). As in Theorem 3 we start by constructing an automaton A for the regular expressions occurring in F. This automaton has an input alphabet of size at most 2^n and at most 2^n states. Therefore by Lemma 2, the Parikh image of the accepted language is a finite union $\pi(\mathcal{L}(A)) = L_1 \cup \ldots \cup L_m$ of linear sets L_r of the form: $\{\sigma_0 + \sum_{i=1}^{h} x_i \cdot \sigma_i \mid x_i \geq 0\}$ where the entries of each σ_j are bounded by 2^n — whereas the number $h \leq 2^{n \cdot 2^n}$ might be doubly exponentially large. Recall however, that for additional satisfiability of the Presburger formulas contained in F, we are not interested in the Parikh image of the words accepted by A itself but in the image of the Parikh image under H. By definition, $H(\pi(\mathcal{L}(A))) = H(L_1) \cup \cdots \cup H(L_m)$. Moreover, for $L = \{\sigma_0 + \sum_{i=1}^{h} x_i \cdot \sigma_i \mid x_i \geq 0\}$, the set $H(L)$ is given by $H(L) = \{\tau_0 + \sum_{i=1}^{h} x_i \cdot \tau_i \mid x_i \geq 0\}$ where $\tau_j = H(\sigma_j)$, $j = 0, \ldots, h$. This implies that each component in a vector τ_j is obtained by the sum of at most 2^n entries of σ_j. Therefore, all entries of the τ_j are bounded by $2^n \cdot 2^n = 2^{2n}$. Moreover, the vectors τ_j now only have at most n entries. Accordingly, only $(2^{2n})^n = 2^{2n^2}$ of the τ_j can be distinct and therefore necessary to describe $H(L)$. Thus, now we may proceed along the same lines as in the proof of Theorem 3. A linear set contained in the Parikh image of w thus gives rise to a linear set containing $H(\pi(w))$ which in turn gives rise to at most n extra equations in 2^{2n^2} variables with coefficients bounded by 2^{2n}. These are to be added to the linear many equations obtained from the Presburger formulas which, after removal of inequalities and (mod d) equivalences may have a linear number of variables. Thus once again applying Papadimitriou's estimation, we obtain that the entries of $\tau = H(\pi(w))$ of a witness w are bounded by $2^{\mathcal{O}(n^2)}$. Recall that by construction, $\top$ is contained in $\mathsf{cl}(B')$ for *every* subset $B' \subseteq \Psi$. Therefore, $H(\pi(w))(\top)$ precisely equals the length of w. Thus, the upper bound on the entries of τ proves the desired upper bound on the length of a shortest witness. This completes the proof. □