Counting in trees

Helmut Seidl[1]
Thomas Schwentick[2]
Anca Muscholl[3]

[1] Institut für Informatik, I2
TU München
Germany

[2] Lehrstuhl Informatik I
Universität Dortmund
Germany

[3] LaBRI
Université Bordeaux
France

Abstract

We consider automata and logics that allow to reason about numerical properties of unranked trees, expressed as Presburger constraints. We characterize non-deterministic automata by Presburger Monadic Second-Order logic, and deterministic automata by Presburger Fixpoint logic. We show how our results can be used in order to obtain efficient querying algorithms on XML trees.

1 Introduction

Tree automata and logics for finite trees have been considered since the seminal work of Thatcher and Wright [40] in the late sixties, with emphasis on *ranked* trees. More recently, research on semi-structured data and XML in particular, raised new questions about *unranked* trees, i.e., trees where the number of children of a node is not fixed *a priori*, [8, 23]. Trees in XML are unranked, labeled, and may occur in two versions, ordered or unordered, depending on whether the sequence of children of a node is ordered or not.

In XML schema description languages like DTDs and XML Schema, the possible sequences of types of children elements of a node are described by regular expressions. Thus, most of the existing theoretical work on XML query languages has concentrated on regular tree languages. These languages can be described by tree automata on unranked ordered trees (also known as *hedge automata*) [26, 27] and a variety of other formalisms [15, 25, 28]. In these formalisms the interaction between the children of a node and the node itself are usually expressed in terms of a regular language.

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 (MSO).

The regular framework is sufficient in many cases. But often one is interested in expressing conditions on the frequency of occurrences of elements in the children sequence. Consider as an example a document which contains music files shared by some peer-to-peer system as Napster, Gnutella etc. as described in Figure 1.[1]

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

One way of formulating such queries, is to extend the MSO logic by formulas of Presburger arithmetics constraining the children of a node (*Presburger constraints* for short). In this new *Presburger MSO* logic (PMSO) the first query can be expressed as:

$$x \in \mathsf{Lab}_{\mathtt{music}} \wedge x/\varphi_1 \,,$$

where φ_1 is the formula

$$\varphi_1 \equiv \#\mathsf{Lab}_{\mathtt{jazz}} \geq \#\mathsf{Lab}_{\mathtt{pop}} \,.$$

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

$$\varphi_2 \equiv \#\mathsf{Lab}_{\mathtt{jazz}} \geq \#\mathsf{Lab}_{\mathtt{pop}} + \#\mathsf{Lab}_{\mathtt{french}} + \#\mathsf{Lab}_{\mathtt{classic}} \,.$$

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

We start our investigation by considering automata that *only* use Presburger constraints, i.e., automata that disregard the order of the children of a node and only use cardinalities of states. Technically speaking, we study in this part automata on unordered trees. It turns out that these automata are very well-behaved. They define a class of tree languages with very regular properties like various closure properties and equivalence with PMSO

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

```
<doc>
 <user>
  <name> ... </name>
   ...
  <music>
   <jazz>
    <album> Always let me go </album>
    <artist> Keith Jarrett </artist>
    <year> 2002 </year>
    <time> 3310 </time>
    <price> 42 </price>
   </jazz>
   <french>
    <tit> Aux enfants de la chance </tit>
    <artist> Serge Gainsbourg </artist>
    <album> Serge Gainsbourg, vol. 3 </album>
    <time> 247 </time>
    <price> 16 </price>
   </french>
   <classic>
    <tit> The Seven Gates of Jerusalem </tit>
    <comp> Krzystof Penderecki </comp>
    <recorded> 1999 </recorded>
   <time> 3510 </time>
    <price> 43 </price>
   </classic>
   <jazz>
    <album> Kind of Blue </album>
    <artist> Miles Davis </artist>
    <year> 1997 </year>
    <time> 3325 </time>
    <price> 29 </price>
   </jazz>

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

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

logic. Further, these automata allow for effective static analysis. Emptiness and universality are decidable, and from any non-deterministic automaton an equivalent deterministic automaton can be constructed. Last but not least, they allow to define a class of (unary) queries the evaluation of which has linear time data complexity.

Next, we study automata that are allowed to combine Presburger constraints with the common regular language constraints (*Presburger tree automata, PTA*). It turns out that they have less desirable properties. Although emptiness of PTA can still be decided, universality (whether an automaton accepts *all* trees) becomes undecidable. As we show that the non-deterministic PTA can be characterized by existential PMSO logic, we can conclude that PMSO logic is undecidable. Nevertheless, the combined complexity of these automata is NP-complete, whereas the data complexity is polynomial time.

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

An alternative for the querying formalism enhanced by Presburger constraints is to replace the MSO logic by fixpoint logic. This Presburger fixpoint logic turns out to be decidable (EXPTIME-complete), and its combined complexity is polynomial time. Moreover, this logic has the same expressive power as deterministic PTA.

This paper is an extended version of [36, 38].

Overview. In Section 2 we define some basic Presburger logic notions. Section 3 studies unordered Presburger tree automata and logic. Section 4 studies basic algorithmic properties of Boolean combinations of regular expressions and Presburger conditions. In Section 5, ordered Presburger tree automata and logic are considered. Section 6 takes a quick look at the case where some unordered parts of a tree allow for Presburger constraints and the others for regular expressions. Section 7 studies Presburger fixpoint logic and its relation with Presburger tree automata. Finally, Section 8 shows how our framework can be used to express unary queries.

Related work. Unordered document trees are closely related to the generalization of feature trees considered by Niehren and Podelski in [29] where they study the (classical) notion of recognizability and give a characteriza-

tion of this notion by means of feature automata. No counting constraints are considered. A detailed study of automata over unranked trees has been initiated by Brüggeman-Klein, Murata and Wood [3].

Query languages for unordered trees have been proposed by Cardelli and Ghelli [5, 4, 6, 7] (and their co-workers). Their approach is based on first-order logic and fixpoint operators. An extension to numerical constraints has been proposed by Dal Zilio et al. [9]. 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 given property [19]. 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, which have an accepting condition depending on a global Presburger constraint [18].

Our notion of tree automata with combined Presburger and regular constraints has been introduced independently by Dal Zilio and Lugiez in [10]. In the latter paper, the authors also propose a modal logic for XML documents, called *Sheaves logic*. This logic allows to reason about numerical properties of the contents of elements but still lacks recursion, i.e., fixpoint operators. On the automata side they obtain comparable results concerning closure properties, membership tests and decidability of emptiness. Although no precise characterization is given, the Sheaves logic is strictly less powerful than the automata model. Recently, Demri and Lugiez proposed the extended modal logic EXML, which uses regular and Presburger constraints on the sequence of children (still without recursion) [11]. The logic EXML is shown to contain the Sheaves logic and to have an EXPSPACE satisfiability problem.

2 Preliminaries on Presburger Logic

Presburger logic is first-order logic with addition and the ordering relation over $\mathbb{N}$. It can express various decision questions such as solvability of systems of linear equations, integer programming, or verification questions. The decidability of Presburger logic was established by Presburger [34] by quantifier elimination. A doubly exponential non-deterministic lower bound was shown in [13]. Later, the precise complexity was shown to be LINA-TIME $2^{2^{O(n)}}$, namely doubly exponential alternating time with a linear number of alternations, [1]. A long line of research was devoted to the analysis of various decision procedures for this logic, based on quantifier elimination and automata. For instance, from a formula in prenex normal form one can construct automata of triply exponential size [17].

For complexity reasons it is quite common to consider either *quantifier-free* or *existential* Presburger formulas, since their satisfiability is in NP. Both use linear terms with integer coefficients, i.e., built according to the

syntax (with x a variable which is interpreted over $\mathbb{N}$):

$$t ::= 0 \mid 1 \mid \pm x \mid t_1 + t_2 \,.$$

Quantifier-free Presburger formulas are defined as the closure of atomic formulas of kind $t = 0$ and $t \equiv 0 \pmod d$ (with t a term and $d \in \mathbb{N}$ a constant) under the Boolean connectives. *Existential Presburger formulas* are defined as the closure of atomic formulas of kind $t = 0$ under the positive connectives $\vee, \wedge$ and existential quantification.

It is well known that each Presburger formula can be transformed into an equivalent quantifier-free formula [34]. In one more step, such a formula can be transformed into an existential formula in *normal form*, that is, into a formula of the form $\exists x_1, \ldots, x_k \bigvee_{i=1}^{m} \varphi_i$, where each disjunct φ_i is a conjunction of equations $t = 0$ with t a linear term (with integer coefficients):

Proposition 2.1. Every quantifier-free Presburger formula φ has an equivalent formula in existential normal form. This formula has at most exponentially many disjuncts, each of at most linear size (in $|\varphi|$).

Proof. Let φ be a quantifier-free Presburger formula. First we bring it into disjunctive normal form (DNF). Then we replace atomic and negated atomic formulas by equations, if necessary by introducing new existentially quantified variables. More precisely,

- $t < c$ can be replaced by $\exists y_1 (t + 1 + y_1 = c)$,
- $t \neq c$ by $\exists y_2 (t + y_2 + 1 = c \vee t - y_2 - 1 = c)$,
- $t \equiv c \pmod d$ by $\exists y_3 (t - dy_3 = c \vee t + dy_3 = c)$, and
- $t \not\equiv c \pmod d$ by

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

The resulting formula needs not to be in DNF yet, but it is free of negations and can be easily transformed into existential normal form. Note first that the DNF is of exponential size, but each disjunct contains at most $|\varphi|$ atoms. After replacing the atoms by equations, each conjunction is transformed into DNF. The size of each resulting disjunct is still linear, and the overall number of disjuncts remains exponential. Q.E.D.

Remark 2.2. Satisfiability of existential Presburger formulas is easily seen to be NP-complete. The upper bound is obtained by assuming w.l.o.g. that such a formula is in prenex form $\exists x_1, \ldots, x_k \, \psi$, with ψ a positive Boolean combination of equations $t = 0$, with t a linear term. It suffices to guess then a disjunct of the DNF of ψ, and test in NP whether a conjunction of such equations is satisfiable.

Given a formula φ and an *assignment* σ mapping the variables of φ to numbers, we write $\sigma \models \varphi$ if φ holds for σ (in the obvious sense) and call σ a *solution* of φ. It is well known that the set of solutions of any given Presburger formula is a *semi-linear set* [14]. A semi-linear set is a finite union of *linear sets*, i.e., sets of the form $\{\bar{c} + \sum_{i=1}^{m} x_i \bar{p}_i \mid x_i \in \mathbb{N}\}$, where $\bar{c}$ and the $\bar{p}_i$ are vectors from $\mathbb{N}^k$ for a given k.

The *Parikh image* of a word $w \in \Sigma^*$ is the assignment $\sigma \in \mathbb{N}^\Sigma$ with $\sigma(a)$ being the number of occurrences of the letter a in w, for each $a \in \Sigma$. Accordingly, the Parikh image of a set $L \subseteq \Sigma^*$ is the set of Parikh images of $w \in L$.

Given the alphabet Σ, $\mathcal{T}(\Sigma)$ stands for the set of *ordered, unranked trees over* Σ. A tree $t \in \mathcal{T}(\Sigma)$ with root label a and subtrees $t_1, \ldots, t_n$ will be denoted as $t = a\langle t_1, \ldots, t_n \rangle$.

3 Unordered Presburger Tree Automata

In this section we start with tree automata and logics that are *unordered*, i.e., they consider only the vertical parent-child order, but not the order between siblings. Technically speaking, we work on unordered trees, as considered for instance in [2, 9, 4, 5].

Given a finite set Q, we will consider a canonical set Y_Q of variables which are associated with the elements in Q. So, we define:

$$Y_Q = \{\#q \mid q \in Q\}\,.$$

An *unordered Presburger tree automaton* (u-PTA for short) is given by a tuple $\mathcal{A} = (Q, \Sigma, \delta, F)$ where:

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

Informally, u-PTA are bottom-up tree automata, with transitions controlled by quantifier-free Presburger formulas. A formula $\varphi = \delta(q, a)$ represents the *pre-condition* on the children of a node labeled by a for the transition into state q, where the value of the variable $\#p$ represents the number of children that are in state p. Formally, we introduce a satisfaction relation $t \models_{\mathcal{A}} q$ between trees $t \in \mathcal{T}(\Sigma)$ and states q which is defined as follows. Assume that $t = a\langle t_1, \ldots, t_n \rangle$, where a is the the label of the root, and $t_1, \ldots, t_n$ are the subtrees of the root, and let $\delta(q, a) = \varphi$. Then $t \models_{\mathcal{A}} q$ if $\{1, \ldots, n\}$ can be partitioned into $|Q|$ subsets I_p of cardinalities n_p ($p \in Q$), such that:

- $t_i \models_{\mathcal{A}} p$ for all $i \in I_p$,
- $\{\#p \mapsto n_p \mid p \in Q\} \models \varphi$.

The language $\mathcal{L}(\mathcal{A})$ of trees which are accepted by $\mathcal{A}$ is

$$\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\Sigma) \mid \exists\, f \in F : t \models_{\mathcal{A}} f\}\,.$$

As an example, consider the language of trees with labels in $\{a, b\}$, such that the internal nodes are all labeled by a and have at most as many subtrees with a b-leaf as ones without. A u-PTA for this language has two states, say q_0 and q_1, where state q_0 means that there is no b-leaf in the subtree, and state q_1 the converse. The transition relation is defined by $\delta(q_0, a) = (\#q_1 = 0)$, $\delta(q_0, b) = \mathit{false}$, $\delta(q_1, a) = (\#q_0 \geq \#q_1 > 0)$ and $\delta(q_1, b) = \text{leaf}$. Here, we use the Presburger constraint $\text{leaf} = (\sum_{i=0,1} \#q_i = 0)$, which is satisfied precisely at leaf nodes.

Note that u-PTA are defined as non-deterministic automata. A u-PTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ is called *deterministic* if for every $a \in \Sigma$ and every tuple $(n_p)_{p \in Q} \in \mathbb{N}^Q$, there is at most one state $q \in Q$ such that

$$\{\#p \mapsto n_p \mid p \in Q\} \models \delta(q, a)\,.$$

Remark 3.1. It is not too hard to verify whether a given u-PTA is deterministic. The precise complexity is NP-complete, since it amounts to check the satisfiability of quantifier-free Presburger formulas. The lower bound can be obtained by an obvious reduction from *Integer Linear Programming (ILP)*.

3.1 Closure and decidability results

The results of this section show that u-PTA enjoy several desirable properties, such as determinization and reasonable complexity.

Theorem 3.2. The non-emptiness problem for u-PTA is NP-complete.

Proof. Consider a u-PTA $\mathcal{A} = (Q, \Sigma, \delta, F)$. Let us call a state $q \in Q$ *reachable* iff there is a tree t with $t \models_{\mathcal{A}} q$. The algorithm guesses some final state $q \in F$, and checks that q is reachable. To this end, the algorithm guesses some $k \leq |Q|$ and a sequence $q_1, \ldots, q_k$ of states with $q_k = q$, and checks that, for each $1 \leq j \leq k$, the following formula is satisfiable:

$$\left(\bigwedge_{p \in Q \setminus \{q_i \mid i < j\}} \#p = 0 \right) \wedge \left(\bigvee_{a \in \Sigma} \delta(q_j, a) \right)\,.$$

Since each check can be done non-deterministically in polynomial time, the overall complexity is NP. Moreover, NP-hardness is again an immediate consequence of ILP, thus we conclude that non-emptiness of u-PTA is NP-complete. Q.E.D.

We show next that u-PTA are effectively closed under the Boolean operations (union, intersection and complement). In particular, we give a determinization construction for u-PTA.

Theorem 3.3. u-PTA are effectively closed under the Boolean operations and under renaming[2].

Proof. Closure under union and renaming is immediate. For intersection, assume that we are given automata $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, F_i)$, $i = 1, 2$. W.l.o.g. we assume that $Q_1 \cap Q_2 = \varnothing$. We proceed analogously to the standard construction of the product automaton for ordinary automata. Thus, we define the automaton $\mathcal{A} = (Q, \Sigma, \delta, F)$ as follows. We set $Q = Q_1 \times Q_2$, $F = F_1 \times F_2$ and define $\delta(\langle q_1, q_2 \rangle, a)$ by the formula below, where $\widetilde{\delta_1}$ and $\widetilde{\delta_2}$, resp., are obtained from δ_1, δ_2, resp., by replacing all variables $\#p$ by x_p ($p \in Q_1 \cup Q_2$):

$$\exists_{p_1 \in Q_1} x_{p_1} \,.\, \exists_{p_2 \in Q_2} x_{p_2} \,.\quad \widetilde{\delta_1}(q_1, a) \wedge \widetilde{\delta_2}(q_2, a) \quad \wedge$$
$$\left(\bigwedge_{p_1 \in Q_1} \sum_{p_2 \in Q_2} \#\langle p_1, p_2 \rangle = x_{p_1} \right) \wedge \left(\bigwedge_{p_2 \in Q_2} \sum_{p_1 \in Q_1} \#\langle p_1, p_2 \rangle = x_{p_2} \right) .$$

In addition, we use above the notation $\exists_{i \in I} x_i$, for some index set I, to denote the existential quantification over all variables x_i ($i \in I$). This is done for convenience only, since the above formula can be rewritten directly as a quantifier-free Presburger formula. It is easy to see that $t \models_{\mathcal{A}} \langle q_1, q_2 \rangle$ iff $t \models_{\mathcal{A}_1} q_1$ and $t \models_{\mathcal{A}_2} q_2$. Thus, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$, which completes the proof.

For closure under complement it suffices to know that u-PTA can be determinized, which is shown in the proposition below. Q.E.D.

Proposition 3.4. For every u-PTA $\mathcal{A}$, a deterministic u-PTA $\mathcal{A}'$ can be constructed such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Proof. The proof idea is similar to the power set construction for ordinary finite automata. Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ and $\mathcal{A}' = (Q', \Sigma, \delta', F')$, where $Q' = 2^Q$ and $F' = \{B \subseteq Q \mid F \cap B \neq \varnothing\}$. For each $B \subseteq Q$, $\delta'(B, a)$ is a formula with free variables from $Y_{Q'}$. It is given by:

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

Here, the formula $\psi_{q,a}$ should be true iff q is a potential successor state. In order to specify $\psi_{q,a}$, we refer to the auxiliary variables x_p ($p \in Q$), and also

[2] A renaming is a letter-to-letter morphism.

to auxiliary variables $x_{\langle B,p\rangle}$ ($B \subseteq Q$, $p \in B$). The variable $x_{\langle B,p\rangle}$ is meant to count all those children resulting in the state set B in $\mathcal{A}'$, for which we choose state $p \in B$ w.r.t. the $\mathcal{A}$-run. Using these auxiliary variables, $\psi_{q,a}$ is defined below, with $\widetilde{\delta}(q,a)$ as the formula $\delta(q,a)$ where each variable $\#p$ is replaced by x_p:

$$\exists_{p\in Q} x_p \,.\; \exists_{p\in B\subseteq Q} x_{\langle B,p\rangle} \,.\quad \widetilde{\delta}(q,a) \wedge \left(\bigwedge_{B\subseteq Q} \sum_{p\in B} x_{\langle B,p\rangle} = \#B\right) \wedge \left(\bigwedge_{p\in Q} \sum_{p\in B\subseteq Q} x_{\langle B,p\rangle} = x_p\right).$$

The transitions of the subset automaton can be transformed into quantifier-free formulas by quantifier elimination. Q.E.D.

As a corollary of Proposition 3.4, we also obtain:

Corollary 3.5. The universality problem for u-PTA is decidable.

The complexity upper bound for the universality problem that follows from Proposition 3.4 is 2-NEXPTIME. Note first that the transition formulas $\delta'(B,a)$ can be written as $\exists\bar{x}\forall\bar{y}\,\psi(\bar{x},\bar{y})$, with ψ quantifier-free. Using quantifier elimination we can first make $\forall\bar{y}\,\psi(\bar{x},\bar{y})$ quantifier-free, then rewrite $\delta'(B,a)$ as an existential Presburger formula. The first step is exponential in the size of the universal quantifier block (see e.g. [17]), hence we obtain an existential formula of doubly exponential size, for which satisfiability can be checked in 2-NEXPTIME.

Proposition 3.6. The combined complexity of u-PTA Model Checking is NP-complete. If the u-PTA is deterministic, the complexity is polynomial time. The data complexity is linear in the size of the input tree.

Proof. Assume we are given a u-PTA $\mathcal{A}$ and a tree t. We guess for each node of t a state of $\mathcal{A}$, and then check that the run is accepting: For each node we compute the Parikh image of the children states (note the entries have values $\leq |t|$). Then we need to check that the Presburger formulas are locally satisfied, for each node. Thus, the evaluation of all formulas at a node requires time $O(|\mathcal{A}| \cdot |t|)$ (using numbers in unary representation).

NP-hardness follows by a reduction from 0-1 ILP, where we ask whether a system S of linear equations $A\bar{x} = \bar{b}$ has a 0-1 solution, i.e. one with $\bar{x} \in \{0,1\}^m$. Given A and $\bar{b}$, we define a u-PTA with state space $\{p_1,\ldots,p_m,p,f\}$, final state f and transition relation $\delta(p,a) = \delta(p_i,a) =$ leaf for all i, and $\delta(f,c) = (\varphi \wedge \#f = 0)$, where φ is the conjunction of all equations in S (with x_i replaced by $\#p_i$) and of $0 \leq \#p_i \leq 1$ for all i. Clearly, the tree t

of depth 1 with root labeled c and m leaves labeled a satisfies $t \models_{\mathcal{A}} f$ iff S has a 0-1 solution.

If $\mathcal{A}$ is deterministic, then the tree can be evaluated in a bottom-up traversal in time $O(|t| \cdot |\mathcal{A}|)$. The data complexity follows immediately, using Proposition 3.4. Q.E.D.

3.2 Unordered Presburger logic

Unordered Presburger MSO logic (u-PMSO for short) is defined by extending monadic second-order logic (MSO) with Presburger predicates over the *children* of a node. As for u-PTA, the logic does not provide an ordering relation on siblings. A u-PMSO formula φ is given by the following grammar:

$$\begin{array}{lll} \varphi & ::= & y \in \mathsf{Lab}_a \mid y \in Y \mid \mathrm{Child}(y, y') \mid y/\psi \mid \\ & & \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \exists y.\, \varphi \mid \exists Y.\, \varphi \\ \psi & ::= & t_1 = t_2 \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \exists x.\, \psi_1 \\ t & ::= & 0 \mid 1 \mid \pm x \mid t_1 + t_2 \mid \#Y \end{array}$$

where $a \in \Sigma$, y, y' are first-order variables, Y is a second-order variable, and x is from a designated set of Presburger variables. The predicates $y \in \mathsf{Lab}_a$ and $\mathrm{Child}(y, y')$ are interpreted as usual, i.e. y is labeled by a, resp. y' is a child of y. The formula ψ in y/ψ is a quantifier-free Presburger formula. The interpretation of y/ψ is that the children of y satisfy the Presburger constraint ψ. A term $\#Y$ inside ψ is interpreted as the number of those children which are contained in Y. As usual, we write $t \models \varphi$, if t satisfies the (closed) u-PMSO formula φ.

Remark 3.7. We also allow derived predicates such as equality between variables such as $y_1 = y_2$, or $Y_1 = Y_2$, or equations $Y = \{y_1\}$. Note that the child predicate $\mathrm{Child}(y, y')$ is redundant, since it can be expressed as:

$$\exists Y.\ (Y = \{y'\} \wedge y/(\#Y = 1)) \ .$$

A tree language $L \subseteq \mathcal{T}(\Sigma)$ is u-PMSO-definable if there is a closed formula φ such that $L = \{t \mid t \models \varphi\}$.

Theorem 3.8 below states that u-PMSO-definable languages are precisely characterized by unordered Presburger tree automata. The proof is analogous to the corresponding result for ordinary tree automata (over ranked trees) [40], and uses in particular Theorem 3.3.

Theorem 3.8. A set of unranked trees is accepted by some u-PTA if and only if it is definable in u-PMSO.

4 Regular Expressions and Presburger Formulas

The automata considered in the next section, as well as the associated logics, use pre-conditions on the children of a node in form of Boolean combinations of regular expressions and Presburger formulas. A basic question is then the satisfiability of such conditions. Since the Parikh image of a regular language (and even context-free language, [33]) is semilinear, deciding emptiness can be reduced to computing the Parikh image of the regular expressions involved.

In [38] we showed that even for an NFA, an existential Presburger formula which describes the Parikh image of the corresponding language can be computed in linear time. Later, this result was extended to context-free grammars in [41] (see also [12] for a related approach).

Theorem 4.1. [41] Given a context-free grammar G, an existential Presburger formula for the Parikh image of $\mathcal{L}(G)$ can be computed in linear time.

A *Presburger regular expression* over Σ is a Boolean combination of regular expressions over Σ and quantifier-free Presburger formulas with variables only from the canonical set $Y_\Sigma = \{\#a \mid a \in \Sigma\}$.

Given a string $w \in \Sigma^*$ and a Presburger regular expression φ we define in the obvious way whether w matches φ (denoted as $w \models \varphi$). For example, if $\varphi = a(a+b)^* \wedge (\#a = \#b)$, then $w \models \varphi$ iff w contains only a's and b's, begins with an a and contains as many a's as b's. A Presburger regular expression φ is satisfiable if there exists some w with $w \models \varphi$. Before we show how to decide satisfiability for such expressions, we need the following property:

Proposition 4.2. Let $\mathcal{A}$ be a (non-deterministic) finite word automaton with n states and input alphabet of size k. Then the Parikh image of $\mathcal{L}(\mathcal{A})$ is a union of linear sets $\{\bar{c} + \sum_{i=1}^{m} x_i \cdot \bar{p}_i \mid x_i \in \mathbb{N}\}$ where each component of each vector $\bar{c}, \bar{p}_j \in \mathbb{N}^k$ is at most n.

In particular, if the size of the alphabet is k, then the number m of occurring vectors is at most $(n+1)^k$.

Proof. The proof is based on the following simple observation: any (accepting) path of $\mathcal{A}$ can be decomposed successively into loops of length at most n, and one (accepting) path of length at most n. Thus, we define each set of vectors $\bar{c}, \bar{p}_j \in \mathbb{N}^k$ by associating $\bar{c}$ with an accepting path λ_0 of length at most n and each $\bar{p}_i$ with a loop λ_i of length at most n, in such a way that the λ_j, $0 \leq j \leq m$, can be combined to a (accepting) path in $\mathcal{A}$. Specifically, it suffices to fix for each j the set of states that occur in λ_j in such a way that $\cup_{j=0}^{m} \lambda_j$ is connected. Q.E.D.

Proposition 4.3. The satisfiability problem for Presburger regular expressions is PSPACE-complete.

Proof. The lower bound is immediate, since 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 [39].

For the upper bound let φ be a Presburger regular expression of size n. First of all, we can assume w.l.o.g. that negations are used only as linear or modular inequations, or in form of negated regular expressions. The given expression φ is satisfiable iff some of the disjuncts in the DNF of φ is so. We can guess such a disjunct ψ in linear time. The formula ψ is a conjunction of regular expressions, negated regular expressions, linear (in)equations $t = 0$, $t \neq 0$ and modular (in)equations $t \equiv 0 \pmod d$, $t \not\equiv 0 \pmod d$.

We first show that ψ is satisfiable iff there exists some word w of exponential length with $w \models \psi$. Since the regular expressions in ψ all occur in φ, the sum of their sizes is at most n. The minimal automaton of each such (possibly negated) regular expression e (resp., $\neg e$) is of size $2^{|e|}$, hence the minimal automaton $\mathcal{A}_\psi$ of the intersection of all positive and negated regular expressions is of size 2^n .

By Proposition 4.2, the Parikh image of $\mathcal{L}(\mathcal{A}_\psi)$ is a union of linear sets $\{\bar{c} + \sum_{i=1}^{h} x_i \bar{p}_i \mid x_i \in \mathbb{N}\}$, where $h = O(2^{n \cdot |\Sigma|}) = O(2^{n^2})$ (as $|\Sigma| \leq n$) and the entries of the vectors $\bar{c}, \bar{p}_i$ are at most $O(2^n)$.

Now, a word $w \in \Sigma^*$ satisfies ψ iff its Parikh image is in one of these linear sets and additionally fulfills the remaining (Presburger) constraints. This can be expressed by adding, for each $a \in \Sigma$, the following equation:

$$\#a = \bar{c}(a) + \sum_{i=1}^{h} x_i \cdot \bar{p}_i(a)\,.$$

Let m be the number of Presburger constraints in ψ. By Proposition 2.1, the conjunction of these constraints is equivalent to a formula in existential normal form, with disjuncts of size $O(m)$. Thus, one has to check whether the Parikh image of w satisfies a system of $M = O(m) + |\Sigma| \leq O(n)$ equations with at most $N = |\Sigma| + O(m + 2^{n^2}) = O(n + 2^{n^2})$ variables, and coefficients of values bounded by $k = 2^n$. By a result of Papadimitriou [32] such a system has a solution with numbers bounded by

$$N \cdot (M \cdot k + 1)^{2M+4} = (O(n + 2^{n^2})) \cdot (O(n) \cdot 2^n + 1)^{O(n)} = 2^{O(n^2)}\,.$$

This shows that if some $w \models \psi$ exists, then there is some with length $2^{O(n^2)}$.

It remains to describe how to check the existence of w as above. We simply guess w symbol by symbol. For each regular expression e or $\neg e$ in ψ, we compute the set of states that can be reached in the non-deterministic automaton $\mathcal{A}_e$ for e when reading w. Further, for each $a \in \Sigma$ we count how often a occurs in w. All this can be done in polynomial space without storing w. A counter keeps track of the length of w. In the end, it can be checked whether the Parikh image of w satisfies all Presburger constraints. Q.E.D.

As Presburger regular expressions are closed under negation we immediately conclude that also universality is PSPACE-complete.

5 Presburger Tree Automata

In many applications, e.g., where documents are automatically generated from databases as textual representations of querying results, the element ordering on the children does not matter. In other applications, though, which are more related to classical document processing, this ordering is crucial. In this section, we extend our framework to automata and logics that take the sibling order into account. Naturally, we use then Presburger regular expressions as pre-conditions on children sequences.

We define a *Presburger tree automaton* for unranked trees (PTA for short) as a tuple $\mathcal{A} = (Q, \Sigma, \delta, F)$ where:

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

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

$$\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\Sigma) \mid \exists\, f \in F : t \models_{\mathcal{A}} f\}\,.$$

A PTA $\mathcal{A}$ is called *deterministic* if for all $a \in \Sigma$ and all $w \in Q^*$, we have $w \models \delta(q, a)$ for at most one $q \in Q$.

Using Proposition 4.3, we obtain with a similar proof as for Theorem 3.2:

Theorem 5.1. The emptiness problem for PTA is PSPACE-complete.

Next we turn to the complexity of such automata.

Theorem 5.2. 1. The combined complexity of PTA Model Checking is NP-complete. If the PTA $\mathcal{A}$ is deterministic, it is $O(n \cdot |\mathcal{A}|)$, where n is the size of the input tree.

2. The data complexity of PTA is polynomial, $O(n^{k+1})$. The degree k of the polynomial is the number of states of the PTA.

Proof. Let $\mathcal{A}$ be a PTA with state set Q and t a tree. A non-deterministic algorithm guesses a run of $\mathcal{A}$ on t and checks the consistency at each node. Each consistency check amounts (1) to testing membership of a string $w \in Q^*$ of size $|t|$ for at most $|\mathcal{A}|$ many regular languages, represented by regular expressions (possibly negated), and (2) to evaluating at most $|\mathcal{A}|$ (quantifier-free) Presburger formulas on its Parikh image. All these checks can be done deterministically, in time $O(|t| \cdot |\mathcal{A}|)$. If $\mathcal{A}$ is deterministic, we perform the computation bottom-up deterministically. The NP lower bound for non-deterministic PTA follows already from the u-PTA case.

Towards (2.), suppose now that the PTA $\mathcal{A}$ is fixed and let Q be its set of states. We perform a bottom-up traversal of the input tree t, computing for each subtree t' the set of states $R = \{p \mid t' \models_{\mathcal{A}} p\} \subseteq Q$. Assume that $t' = a\langle t_1, \ldots, t_m\rangle$ and $R_i = \{p \mid t_i \models p\}$ have been already computed. Moreover, we can suppose that the Presburger regular expressions used in $\mathcal{A}$ are disjunctions of conjuncts $e_i \wedge \pi_i$ where for each e_i a deterministic automaton $\mathcal{B}_i$ is given, and each π_i is a Presburger formula. Then we may check for each $e_i \wedge \pi_i$ separately whether it is verified by $t_1, \ldots, t_m$.

To this end, let us now consider one conjunct $e \wedge \pi$, where the language of e is described by the finite automaton $\mathcal{B}$ with set of states P. Let the sets $V(i, s)$, $0 \leq i \leq m$, $s \in P$, contain all assignments $v : Y_Q \to \{0, \ldots, i\}$ verifying the following condition: there exists a sequence of states $\alpha = r_1 \cdots r_i$ with $r_j \in R_j$ for $j = 1, \ldots, i$ and with Parikh image v, such that state s can be reached from an initial state of $\mathcal{B}$ by reading α. Finally, let V be the union of all sets $V(m, f)$ where f is a final state of $\mathcal{B}$. Once V is computed, it simply remains to check whether $v \models \pi$ for some $v \in V$. Thus, assuming that the automaton $\mathcal{A}$ is of constant size, we spend $O((m+1)^{|Q|})$ time on the computation of the set of all successor states at the root of t'. Hence we can conclude that the overall runtime on a tree of size n is $O(n^{|Q|+1})$. Q.E.D.

PTA do not share the pleasant properties with u-PTA. In particular, it is a direct consequence of the following result that there is no determinization algorithm for PTA.

Theorem 5.3. The universality problem for PTA is undecidable.

Proof. The proof is a reduction from the accepting problem for 2-counter Minsky machines [24]. Given such an automaton $\mathcal{A}$ with state set Q we

construct a PTA $\mathcal{B}$ such that $\mathcal{A}$ does not accept the empty input if and only if $\mathcal{B}$ accepts all trees over the alphabet $Q \cup \{\#, \$, a, b\}$. In the construction we will concentrate on trees of depth 1 with frontier (i.e., leaf labels read from left to right) from $Qa^*b^*(\#Qa^*b^*)^*$. The root is labeled by \$. It is easy to construct a tree automaton (without Presburger constraints) which accepts all trees that are *not* of this special form. The union of this automaton with the automaton $\mathcal{B}'$ to be constructed in the remainder of the proof will be the automaton $\mathcal{B}$ we are looking for.

The PTA $\mathcal{B}'$ checks whether the frontier does *not* encode an accepting computation of the counter automaton $\mathcal{A}$. Here, a configuration of $\mathcal{A}$ with state q and counter contents n_1 and n_2, respectively, is encoded by the string $qa^{n_1}b^{n_2}$ and configurations are separated by #. The automaton $\mathcal{B}'$ checks whether one of the following cases arises:

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

We only describe how the latter case can be checked, as the first two cases are straightforward. The idea is that $\mathcal{B}'$ simply marks two consecutive configurations. A regular constraint can check that two such configurations are correctly marked, whereas a Presburger constraint ensures that the two configurations are indeed inconsistent with the transition function of $\mathcal{A}$.

Formally, the state set of $\mathcal{B}'$ equals $Q \cup \{\#, a, a', b, b', ?\}$. On each leaf the automaton can enter state ?. Further, it can enter state # on all leaves labeled #, and state q on all leaves with label $q \in Q$. For leaves with label a (b, resp.) it can also enter state a or a' (b or b', resp.)

The automaton $\mathcal{B}'$ enters an accepting state at the root if both conditions below hold:

- the states on the frontier form the sequence $?^*\#qa^*b^*\#q'a'^*b'^*\#?^*$ with $q, q' \in Q$, and
- the numbers of occurrences of a, b, a', b' are not consistent with respect to q, q' and the transition function of $\mathcal{A}$.

The first condition above is simply a regular constraint. The second one can be expressed by a conjunction of Presburger constraints, over all possible transitions of $\mathcal{A}$ leading from state q to state q'. For instance, for a transition that increases the first counter and leaves the second one

unchanged, the Presburger constraint requires that either the number of a' is not equal to the number of a plus 1, or the numbers of b and b' are not equal. Q.E.D.

5.1 Presburger MSO logic

Unordered Presburger MSO logic as defined in Section 3.2 is readily extended to take into account the sibling order, by adding the atomic predicate $\mathrm{Next}(y, y')$, with the meaning that y' is a *right sibling* of y. We denote this logic as PMSO. We now characterize Presburger tree automata by means of existential PMSO logic.

Theorem 5.4. A set of unranked trees is accepted by a PTA if and only if it can be described by a PMSO formula of the form $\exists X_1 \ldots \exists X_k . \varphi$ where φ contains no second-order quantifier.

Proof. Let $\mathcal{A}$ be a PTA with state set Q and transition relation δ. Without loss of generality we can assume that all Presburger regular expressions used in δ are disjunctions of expressions $e_i \wedge \pi_i$, where e_i is a regular expression over Q, and π_i is a quantifier-free Presburger formula. Furthermore, let, for each i, a finite automaton $\mathcal{B}_i$ for $\mathcal{L}(e_i)$ be given. From Büchi's Theorem it follows that each automaton $\mathcal{B}_i$ is equivalent to an existential MSO formula $\psi_i = \exists Y_1 \ldots \exists Y_l . \varphi_i$. Hence, we can construct a formula $\psi = \exists X_1 \cdots \exists X_k . \varphi$ in which some of the variables X_i are used to encode the states that $\mathcal{A}$ assumes and the remaining variables are those of the formulas ψ_i. The first-order part φ of ψ describes the consistency of the states between nodes of the input tree and their children, and uses the formulas φ_i.

For the converse we show first that every PMSO formula ψ containing no second-order quantifier can be evaluated by a *deterministic* PTA. The result is then immediate as a non-deterministic automaton can guess, for each node, those sets of $X_1, \ldots, X_k$ in which the node is contained. The proof proceeds by induction on the structure of ψ. The only case which is not entirely straightforward is the case of a formula $\psi = \exists x . \varphi(x)$. Let, by induction, $\mathcal{A}$ be an automaton over the alphabet $\Sigma \cup (\Sigma \times \{x\})$ for $\varphi(x)$. I.e., $\mathcal{A}$ accepts all trees t which have exactly one node v with a symbol (a, x) from $\Sigma \times \{x\}$ such that φ holds on t, if x is bound to v and the label of v is replaced by a.

Let Q be the set of states of $\mathcal{A}$. We construct a deterministic PTA $\mathcal{A}'$ for ψ as follows. The state set of $\mathcal{A}'$ is $Q \times 2^Q$. The intuitive meaning of a state $\langle q, X \rangle$ at a node v is the following. First, if x does not occur in the subtree rooted at v, then $\mathcal{A}$ assumes state q at v. Second, X is the set of states $\mathcal{A}$ can take if for one node of the subtree at v its label a is replaced by (a, x). We explain how the transitions of $\mathcal{A}'$ are defined. The mapping $\delta'(\langle q, X \rangle, a)$ is described by a Presburger regular expression

$e_{q,a} \wedge e_{X,a}$, where $e_{q,a}$ is obtained from $\delta(q,a)$ by replacing each occurrence of a state $r \in Q$ in a regular expression by $\bigcup_{S \subseteq Q} \langle r, S\rangle$ and each occurrence of $\#r$ in a Presburger formula by $\sum_{S \subseteq Q} \#\langle r, S\rangle$. The Presburger regular expression $e_{X,a}$ is of the form $\bigwedge_{p \in X}(e^1_{p,a} \vee e^2_{p,a}) \wedge \bigwedge_{p \notin X} \neg(e^1_{p,a} \vee e^2_{p,a})$. Here, $e^1_{p,a}$ expresses that $\mathcal{A}$ takes state p at v if the label of v is (a,x). Likewise, $e^2_{p,a}$ expresses that $\mathcal{A}$ takes state p at v (labeled by a) if the label b of some node below v is replaced by (b,x). The expression $e^1_{p,a}$ is obtained from $\delta(p,(a,x))$ in an analogous fashion as $e_{q,a}$ was obtained from $\delta(q,a)$.

It remains to describe the construction of $e^2_{p,a}$. The expression $e^2_{p,a}$ is obtained as a disjunction $\bigvee_{r \in Q} \bigvee_{r' \in S \subseteq Q} \delta(p,a)_{r,r',S}$. Here, for each choice of $S \subseteq Q$, $r \in Q$ and $r' \in S$, the Presburger regular expression $\delta(p,a)_{r,r',S}$ is satisfied by a sequence $\langle q_1, S_1\rangle \cdots \langle q_m, S_m\rangle$, $q_i \in Q$, $S_i \subseteq Q$, iff there is some $i \leq m$ with $q_i = r$, $S_i = S$ and $\delta(p,a)$ is satisfied by $q_1 \cdots q_{i-1} r' q_{i+1} \cdots q_m$.

The expression $\delta(p,a)_{r,r',S}$ is defined by replacing in $\delta(p,a)$ each regular expression e by $e_{r,r',S}$, and each Presburger formula π by $\pi_{r,r',S}$. We get $\pi_{r,r',S}$ as the conjunction of $\#\langle r, S\rangle > 0$ and the formula which is obtained from π by replacing $\#q$, for each $q \in Q$ with

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

The language L of a regular expression $e_{r,r',S}$ is given as:

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

Q.E.D.

Theorem 5.4 shows that existential PMSO logic is decidable. On the other hand we immediately obtain from Theorem 5.3:

Corollary 5.5. Satisfiability of PMSO formulas is undecidable.

6 Mixed Automata

In the previous section we have seen that in general we cannot expect decidability for all PMSO. Instead, we can restrict ourselves to automata and logics that work in a *mixed* mode, either pure regular or pure Presburger,

depending on the tag. Formally, we work on mixed trees, where the label of a node tells whether the ordering of its children matters or not. Recall from the introduction that this restriction naturally reflects a division of documents into parts which are made up from data records whose orderings are irrelevant and formatting parts where the ordering is significant. This classification is formalized by partitioning the finite alphabet Σ into subsets $\Sigma = \Sigma_0 + \Sigma_1$ where Σ_0 and Σ_1 consist of all labels of nodes with unordered and ordered children, respectively. Mixed trees in our sense correspond to terms with one associative symbol (for accumulating the ordered contents) and one associative and commutative symbol (for accumulating multi-sets). Languages of such trees, e.g., have been studied by Lugiez [20, 21] and Ohsaki [31, 30]. Note, however, that our formalism is slightly more specific as we rule out sequences of trees where unordered sections occur dispersed between ordered ones. Instead, the significance of order is already determined by the label of the parent node.

Mixed Presburger tree automata now subsume the ability of unordered Presburger automata to check Presburger formulas, as well as the ability of hedge automata to check containment in a regular set. Formally, $\delta(q, a)$ is a quantifier-free Presburger formula if $a \in \Sigma_0$, respectively a regular expression if $a \in \Sigma_1$. We call such an automaton a *mixed* PTA. Similarly to Theorem 3.2, we obtain:

Corollary 6.1. The emptiness problem for mixed PTA is NP-complete.

It turns out that the family of languages accepted by mixed PTA enjoys the same good closure properties as u-PTA. The proof of the theorem below follows the lines of Proposition 3.4 and is omitted:

Theorem 6.2. Mixed PTA are effectively closed under the Boolean operations. In particular, for every mixed PTA an equivalent deterministic mixed PTA can be constructed.

As for unordered and general PTA, respectively, we succeed to give a logical characterization of our automata model also in the mixed case. For that we use mixed PMSO logic, in which Presburger (regular, resp.) constraints can be applied only to the children of a node labeled with some element from Σ_0 (Σ_1, resp.). We therefore speak here of *mixed* PMSO-definable languages and queries. More formally, in a mixed PMSO-formula an atom $\mathrm{Next}(y, y')$ is allowed in a subformula φ occurring in a context $\mathrm{Child}(x, y) \wedge y \in \mathsf{Lab}_a \wedge \varphi$, where $a \in \Sigma_1$. Likewise a formula y/ψ is allowed in a subformula φ occurring in a context $y \in \mathsf{Lab}_a \wedge \varphi$, where $a \in \Sigma_0$. Mixed PMSO-definable queries are what we have considered in the introduction, by considering e.g. that the label `music` belongs to Σ_0. We obtain:

Theorem 6.3. A set of unranked trees is accepted by some mixed PTA iff it is mixed PMSO-definable.

We conclude that satisfiability of mixed PMSO-logic is decidable.

7 Presburger Fixpoint Logic

As an alternative to monadic second-order logic, we consider in this section the extension of fixpoint logics with regular and Presburger constraints on children of nodes. Our fixpoint formulas φ are thus constructed according to the following grammar:

$$\begin{array}{lcl} \varphi & ::= & \top \mid x \mid \mu x.\varphi \\ & & \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \\ & & \mid a\langle F\rangle \mid *\langle F\rangle \\ F & ::= & e \mid \pi\,. \end{array}$$

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 e over letters φ, possibly negated, or a quantifier-free Presburger formula π with free variables $\#\varphi$, denoting the number of children satisfying φ (with φ a fixpoint formula).

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. We consider assertions $t : \psi$, with $t \in \mathcal{T}(\Sigma), \psi \in \Phi$. 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 e iff there are formulas $\psi_1, \ldots, \psi_k$ such that $t_i : \psi_i$ and the sequence of formulas $\psi_1 \cdots \psi_k$ is contained in the regular language $\mathcal{L}(e)$ described by e. In case of a Presburger formula π, we first collect for every formula ψ occurring in π the number n_ψ of children t_i satisfying ψ. Then u satisfies π iff the resulting assignment $\sigma = \{\#\psi \mapsto n_\psi \mid \psi \in \Phi\}$

satisfies $\sigma \models \pi$. Thus we have the rules:

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

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

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 $\psi = a\langle\pi\rangle$, π a Presburger formula, that a proof of $t : \psi$ contains for every $i = 1, \ldots, k$, and every ψ' occurring in π 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 $\mathcal{L}(\mu\, x.\, x) = \varnothing$. 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 (see, e.g., [37]). The free variable x occurs only guarded inside the formula φ if x occurs as a free variable only within the scope of elements a or $*$. The variable x, for example, occurs only guarded inside the formula $a\langle\_\ x\ \_\rangle \vee y$ while y does not. It turns out that guarded fixpoints are *unique*. More precisely, if x occurs only guarded in φ, then $\mu x.\, \varphi$ is semantically equivalent to $\nu x.\varphi$. Once greatest fixpoints are available, complementation is easy since then we can push negations inward. For example, we have: $t : \neg(\mu x.\, \varphi(x))$ iff $t : \nu x.\, \neg\varphi(\neg x)$.

In the subsequent proofs we will 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 $\varphi_1 \in B'$ and $\varphi_2 \in B'$ then also $\varphi_1 \wedge \varphi_2 \in B'$, whenever $\varphi_1 \wedge \varphi_2 \in \Phi$;
- If $\varphi_1 \in B'$ or $\varphi_2 \in B'$ then also $\varphi_1 \vee \varphi_2 \in B'$, whenever $\varphi_1 \vee \varphi_2 \in \Phi$;
- If $\varphi' \in B'$ then $\mu\, x.\varphi' \in B'$ and $x \in B'$, whenever $\mu\, x.\varphi' \in \Phi$.

Intuitively, the closure of a set B of subformulas contains precisely the subformulas which are implied by the formulas in B through the proof rules for fixpoint formulas. In particular, consider a given fixpoint formula, a tree t and let B be the set of all subformulas ψ of type $a\langle F\rangle$ and $*\langle F\rangle$ with $t : \psi$. Then, $\mathsf{cl}(B)$ is the set of *all* subformulas ψ with $t : \psi$.

Theorem 7.1. A set of trees is accepted by some deterministic PTA if and only if it satisfies some Presburger fixpoint formula.

Proof. Let φ be a Presburger fixpoint formula. We assume for simplicity that all regular expressions in φ are unnegated. We construct a PTA $\mathcal{A}$ 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 $\mathcal{A}$ is given as the set of all subsets $B \subseteq \Psi$. The set T of accepting states consists of all subsets B such that $\varphi \in \mathsf{cl}(B)$, i.e., whose closure contains the initial formula φ.

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

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

where:

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

where $\bar{F}$ is constructed as follows. For a regular expression e, we obtain $\bar{e}$ from e by substituting $B_1 + \cdots + B_m$ 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 π, let $\bar{\pi}$ be obtained from π 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 show for trees $t, t_1, \ldots, t_k$:

(1) $t \models_{\mathcal{A}} B$ iff $\mathsf{cl}(B) = \{\psi \in \Phi \mid \vdash t : \psi\}$;

(2) $\vdash t_1, \ldots, t_k : e$ iff $t_i \models_{\mathcal{A}} B_i$, $1 \le i \le k$, such that $B_1 \cdots B_k \in \mathcal{L}(\bar{e})$;

(3) $\vdash t_1, \ldots, t_k : \pi$ iff $t_i \models_{\mathcal{A}} B_i$, $1 \le i \le k$, such that the Parikh image of $B_1 \cdots B_k$ satisfies $\bar{\pi}$.

In particular, item (1) above implies that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.

The three claims above are shown inductively. Items (2) and (3) above are immediate for $k = 0$. For $k > 0$ they follow from the definition of $\bar{e}$ and $\bar{\pi}$, together with item (1). Suppose now that $t = a\langle t_1, \ldots, t_k\rangle$, $k \ge 0$. Then $t \models_{\mathcal{A}} B$ iff $t_i \models_{\mathcal{A}} B_i$ for some B_i, $1 \le i \le k$, such that:

- $B_1 \cdots B_k \in \mathcal{L}(\bar{e})$ iff $a\langle e\rangle$ or $*\langle e\rangle$ is in B,

- the Parikh image of $B_1 \cdots B_k$ satisfies $\bar{\pi}$ iff $a\langle\pi\rangle$ or $*\langle\pi\rangle$ is in B.

By induction, $\mathsf{cl}(B_i) = \{\psi \mid \vdash t_i : \psi\}$ for all i. Using items (2) and (3) we infer that $a\langle F\rangle$ or $*\langle F\rangle$ is in B iff $\vdash t_1, \ldots, t_k : F$, for all pre-conditions F. By the definition of $\mathsf{cl}(B)$ this is equivalent to $\mathsf{cl}(B) = \{\psi \mid \vdash t : \psi\}$.

For the reverse implication, consider a deterministic PTA $\mathcal{A} = (Q, \Sigma, \delta, F)$. W.l.o.g. we may assume that every pre-condition is a disjunction of conjunctions of regular expressions and Presburger formulas. We introduce one variable x_q for every state $q \in Q$. For these variables, we construct an equation system $S_{\mathcal{A}}$:

$$x_q \;=\; \varphi_q \;, \quad q \in Q$$

where the right-hand sides are defined as fixpoint expressions, but without allowing the μ operator. The semantics of such equation systems is an extension of the semantics for fixpoint expressions. The only addition is the rule:

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

for every equation $x = \varphi$. Thus, whenever a tree satisfies the right-hand side of an equation, then it also satisfies the variable to the left. The right-hand side φ_q for x_q in the equation system $S_{\mathcal{A}}$ is constructed from $\delta(q, a)$, $a \in \Sigma$, by:

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

where the transformation $[.]_a$ takes a pre-condition and returns a fixpoint expression (without fixpoints) as follows:

$$\begin{array}{lcl} [e]_a & = & a\langle e\{q \mapsto x_q \mid q \in Q\}\rangle\,, \\ [\pi]_a & = & a\langle \pi\{\#q \mapsto \#x_q \mid q \in Q\}\rangle\,, \\ [\varphi_1 \vee \varphi_2]_a & = & [\varphi_1]_a \vee [\varphi_2]_a\,, \\ [\varphi_1 \wedge \varphi_2]_a & = & [\varphi_1]_a \wedge [\varphi_2]_a\,. \end{array}$$

Thus, a regular expression 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 variables $\#q$ with $\#x_q$, and again putting a node a on top, whereas conjunctions and disjunctions are transformed recursively. By induction on the depth of terms $t, t_1, \ldots, t_k$ we prove for every $q \in Q$, $a \in \Sigma$ and right-hand side φ:

(1) $t \models_{\mathcal{A}} q$ iff $\vdash t : x_q$;

(2) $t_i \models_{\mathcal{A}} q_i$ for $1 \leq i \leq k$, with $q_1 \cdots q_k \models \varphi$ iff $\vdash a\langle t_1, \ldots, t_k\rangle : [\varphi]_a$.

The first claim then proves the correctness of the construction.

For the proof of the claims let us first assume that $t_i \models_{\mathcal{A}} q_i$ for all i, and $q_1 \cdots q_k \models \varphi$. We verify that for every $a \in \Sigma$, $a\langle t_1, \ldots, t_k\rangle : [\varphi]_a$ where, by inductive hypothesis, we may assume that $\vdash t_i : x_{q_i}$ for all i. If $\varphi = e$ is a regular expression, then by assumption, $q_1 \cdots q_k \in \mathcal{L}(e)$. By definition, $[\varphi]_a = a\langle e\{q \mapsto x_q \mid q \in Q\}\rangle$. Therefore, $x_{q_1} \cdots x_{q_k} \in \mathcal{L}(e\{q \mapsto x_q \mid q \in Q\})$ and hence $a\langle t_1, \ldots, t_k\rangle : [e]_a$. If $\varphi = \pi$ equals a Presburger formula, then the Parikh image of $x_{q_1} \cdots x_{q_k}$ satisfies $\pi\{\#q \mapsto \#x_q \mid q \in Q\}$. Let ρ denote the mapping defined by $\rho(\#x_q) = |\{i \mid \vdash t_i : x_q\}|$. Since the automaton $\mathcal{A}$ is deterministic, $t_i : x_q$ is provable for exactly one state q. Therefore, the number of occurrences of q in the sequence $q_1, \ldots, q_k$ precisely equals $\rho(\#x_q)$. We conclude that $t_1, \ldots, t_k : \pi\{\#q \mapsto \#x_q \mid q \in Q\}$ and therefore also $a\langle t_1, \ldots, t_k\rangle : [\pi]_a$. The cases $\varphi = \varphi_1 \wedge \varphi_2$ and $\varphi = \varphi_1 \vee \varphi_2$ are completely standard.

For the converse direction assume $a\langle t_1, \ldots, t_k\rangle : [\varphi]_a$ for some $a \in \Sigma$. By inductive hypothesis for t_i, we already know that there are (unique) states q_i such that $t_i \models_{\mathcal{A}} q_i$ and therefore also $\vdash t_i : x_{q_i}$, for all i. It remains to verify that $q_1 \cdots q_k \models \varphi$. If $\varphi = e$ is a regular expression, then $x_{q_1} \cdots x_{q_k} \in \mathcal{L}(e\{q \mapsto x_q \mid q \in Q\})$, thus $q_1 \cdots q_k \models \varphi$. If $\varphi = \pi$ equals a Presburger formula, then $[\varphi]_a = a\langle \pi\{\#q \mapsto \#x_q \mid q \in Q\}\rangle$. Since by assumption, $a\langle t_1, \ldots, t_k\rangle : [\varphi]_a$, we obtain $\rho \models \pi\{\#q \mapsto \#x_q \mid q \in Q\}$ for the assignment $\rho(\#x_q) = |\{i \mid \vdash t_i : x_q\}|$, $q \in Q$. Since $\mathcal{A}$ is deterministic, $\rho(\#x_q)$ equals the number of occurrences of q in the sequence $q_1, \ldots, q_k$. Therefore, $q_1 \cdots q_k \models \pi$. The case where $\varphi = \varphi_1 \vee \varphi_2$ or $\varphi = \varphi_1 \wedge \varphi_2$ are dealt with recursively.

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

$$\varphi_{\mathcal{A}} \;=\; \bigvee_{q \in F} \bar{\varphi}_q \;.$$

Q.E.D.

In the remainder of this section we turn to the complexity of Presburger fixpoint logic. Concerning satisfiability, Theorem 7.1 provides an EXPSPACE upper bound. The theorem below shows that this can be improved to EXPTIME, which is as good as we can hope for, since satisfiability of fixpoint

formulas (without Presburger conditions) over binary trees is EXPTIME-complete (a similar result holds for model-checking μ-calculus against push-down graphs, [42]).

Theorem 7.2. The satisfiability problem for Presburger fixpoint formulas is EXPTIME-complete.

Proof. The lower bound is obtained, e.g., by encoding the accepting runs of an alternating polynomial space Turing machine through a binary tree.

It remains to prove the exponential upper bound. Let φ be a Presburger fixpoint formula. We denote by Ψ the set of its subformulas of type $a\langle F\rangle$ or $*\langle F\rangle$, and by 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, actually we will need here only polynomial space.

Let X denote a set of obtainable subsets of Ψ. We show that, given the fixpoint formula φ of size n 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, X is not part of the input, since it might be of exponential size. We can imagine X as stored on a separate tape, and our PSPACE algorithm will access non-deterministically this tape.

A set 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 exists a sequence of sets $w = B_1 \dots B_h$ with $B_i \in X$ for all i, 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.

Consider first a formula of type $a\langle e\rangle$ ($*\langle e\rangle$, resp.), with e regular expression. By the definition of the closure of sets of formulas from Ψ, it is immediate that t satisfies $a\langle e\rangle$ ($*\langle e\rangle$, resp.) iff $w \in \mathcal{L}(\bar{e})$, where $\bar{e}$ is obtained from e by replacing every formula ψ with the disjunction of all $B' \in X$ with $\psi \in \mathsf{cl}(B')$. Likewise for $a\langle \neg e\rangle$ ($*\langle \neg e\rangle$, resp.).

For formulas $a\langle \pi\rangle, *\langle \pi\rangle$, with π Presburger formula, we first need the following definition. Let H denote the mapping which takes an assignment

$\sigma : X \to \mathbb{N}$ and computes an assignment $\tau : \Phi \to \mathbb{N}$ by

$$\tau(\psi) = \sum_{B' \in X \text{ with } \psi \in \mathsf{cl}(B')} \sigma(B') \,.$$

The tree $t = a\langle t(B_1), \cdots, t(B_h)\rangle$ (with $w = B_1 \dots B_h$) satisfies the formula $a\langle\pi\rangle$ ($*\langle\pi\rangle$, resp.) iff $H(\mathrm{Par}(w))$ satisfies π, where $\mathrm{Par}(w)$ denotes the Parikh vector of $w \in X^*$. The reader should recall here that with the fixpoint semantics a subtree can be counted several times, once for each formula it satisfies.

As in the proof of Proposition 4.3, we will show the following:

Claim 7.3. If there exists a string which simultaneously verifies all formulas of type $a\langle F\rangle$ or $*\langle F\rangle$ in B, and falsifies all such formulas outside B, then there exists one 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. We successively guess subsets $B' \subseteq X$ (in polynomial space). For each such B', we simulate the evaluations of the non-deterministic automata corresponding to all regular expressions e occurring in $a\langle F\rangle \in \Psi$ or $*\langle F\rangle \in \Psi$. Of course, in order to do so, we need to check each time whether a subformula $\varphi' \in \Phi$ is in $\mathsf{cl}(B')$. All these simulations are done in PSPACE. During this process, we maintain an occurrence vector τ indexed by subformulas $\varphi' \in \Phi$. Whenever a set B' is processed, we increment in τ the values of all φ' contained in $\mathsf{cl}(B')$. 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 $a\langle e\rangle \in B$ and $*\langle e\rangle \in B$ and invalidating those outside B, we verify that τ satisfies the formula

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

The latter can be done in polynomial time (recall that each Presburger formula π is quantifier-free). This algorithm uses only space polynomial in n, therefore it can be executed in deterministic exponential time — which we wanted to prove.

It remains to show the claim above. Recall first that we defined the regular expressions $\bar{e}$ over the alphabet X by replacing each subformula φ' of φ by the disjunction of all $B' \in X$ with $\varphi' \in \mathsf{cl}(B')$. Now, we first construct an automaton $\mathcal{B}$ for the intersection of the regular expressions $\bar{e}$ (resp. $\neg\bar{e}$) occurring in formulas from B. This automaton has at most 2^n states, and its alphabet is of size 2^n. By Proposition 4.2, the Parikh image of the accepted language is a finite union $\mathrm{Par}(\mathcal{L}(\mathcal{B})) = L_1 \cup \cdots \cup L_m$ of linear

sets L_r of the form $\{\bar{c} + \sum_{i=1}^{h} x_i \cdot \bar{p}_i \mid x_i \geq 0\}$, where the entries of each vector $\bar{c}, \bar{p}_j$ are bounded by 2^n — whereas their number $h \leq (2^n+1)^{2^n}$ might be doubly exponentially large. Recall however, that for the satisfiability of the Presburger formulas π occurring in formulas $a\langle\pi\rangle, *\langle\pi\rangle$ contained in B, we are not interested in the Parikh image $\mathrm{Par}(\mathcal{L}(\mathcal{B}))$ of the words accepted by $\mathcal{B}$ itself, but in the image of $\mathrm{Par}(\mathcal{L}(\mathcal{B}))$ under H. By definition, $H(\mathrm{Par}(\mathcal{L}(\mathcal{B}))) = H(L_1) \cup \cdots \cup H(L_m)$. Moreover, for every linear set of the form $L = \{\bar{c} + \sum_{i=1}^{h} x_i \cdot \bar{p}_i \mid x_i \geq 0\}$, the image $H(L)$ is given by $H(L) = \{\tau_0 + \sum_{i=1}^{h} x_i \cdot \tau_i \mid x_i \geq 0\}$ where $\tau_0 = H(\bar{c})$, $\tau_j = H(\bar{p}_j)$, $j = 1, \ldots, h$. This implies that each component in a vector τ_j is obtained by the sum of at most 2^n entries of the vectors $\bar{c}, \bar{p}_j$. Therefore, all entries of the τ_j are bounded by $2^n \cdot 2^n = 2^{2n}$. The crucial point is that the vectors τ_j now only have at most n entries (instead of 2^n for $\bar{c}, \bar{p}_j$). 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 Proposition 4.3. A linear set L contained in the Parikh image $\mathrm{Par}(\mathcal{L}(\mathcal{B}))$ of $\mathcal{B}$ gives rise to a linear set $H(L)$ contained in $H(\mathrm{Par}(\mathcal{L}(\mathcal{B})))$, 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 $O(n)$ many equations obtained from the Presburger formulas from B. That is, as in Proposition 4.3 we consider a disjunct of the DNF of each formula π occurring in some Presburger formula form B (resp., with $\neg\pi$ occurring outside B), and we eliminate inequations and modulo equations using Proposition 2.1. Once again applying Papadimitriou's estimation [32], we obtain that the entries of a minimal solution $\tau \in H(\mathrm{Par}(\mathcal{L}(\mathcal{B}))) \cap S$, with

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

are bounded by $2^{O(n^2)}$. Clearly, we have $\tau \in H(\mathrm{Par}(\mathcal{L}(\mathcal{B}))) \cap S$ iff there is some string $w \in \mathcal{L}(\mathcal{B})$ such that $H(\mathrm{Par}(w))$ satisfies S. Recall that by construction, $\top$ is contained in $\mathsf{cl}(B')$ for *every* subset $B' \subseteq \Psi$. Therefore, $H(\mathrm{Par}(w))(\top)$ 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 w and thus the claim. Q.E.D.

We finish this section with the following

Proposition 7.4. Given a tree t and a Presburger fixpoint formula φ, it can be checked in time $O(|t| \cdot |\varphi|^2)$ whether $t \models \varphi$.

Proof. We compute bottom-up the set of subformulas of φ that are satisfied by each subtree. For each subtree $t' = a\langle t_1, \ldots, t_k\rangle$ we simulate first the

NFA corresponding to regular expressions e ($\neg e$, resp.) occurring in preconditions $a\langle\ldots\rangle$ and $*\langle\ldots\rangle$, by keeping the set of reachable states of the NFA. Since each NFA is of size at most $|\varphi|$, each such simulation costs at most $O(k \cdot |\varphi|^2)$. For Presburger constraints $a\langle\pi\rangle, *\langle\pi\rangle$ we just need to count how many children satisfy a given subformula occurring in π, which can be done in $O(k \cdot |\varphi|)$, and to evaluate linear (in)equations and modular (in)equations. The last check is done in $O(|\varphi|^2)$. Finally, we compute $\mathsf{cl}(B)$ in $O(|\varphi|)$, with $B \subseteq \Psi$ the set of all $a\langle F\rangle$ or $*\langle F\rangle$ satisfied by $a\langle t_1, \ldots, t_k\rangle$.

Q.E.D.

8 Querying Unranked Trees

Presburger automata or logics can be used as a facility to express *unary queries*, i.e., to select a set of nodes in a document tree. We start this section with automata-based queries, and consider in Subsection 8.1 queries based on fixpoint logics, which exhibit a much better complexity than PTA-based queries.

With automata-based querying, a tree node is selected via an automaton $\mathcal{A}$ and a set T of states of $\mathcal{A}$. The node v is in the output, if there is an accepting computation of $\mathcal{A}$ that obtains a state from T at v. By the equivalence between Presburger automata and Presburger MSO logic (Thms. 3.8, 5.4, 7.1), this simple mechanism allows to express all (unary) queries definable in Presburger MSO logic.

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

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

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

Thus, the automaton $\mathcal{A}_{p,\bullet}$ behaves like $\mathcal{A}$ but additionally labels the hole by p. We have:

Proposition 8.1. Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a PTA and $t = c[t']$ for a context c and $t, t' \in \mathcal{T}(\Sigma)$. Then $t \models_{\mathcal{A}} q$ iff $t' \models_{\mathcal{A}} p$ and $c, p \models_{\mathcal{A}} q$ for some $p \in Q$.

A (unary) *Presburger* pattern is a property of nodes of trees from $\mathcal{T}(\Sigma)$. We define this property by means of a pair $\langle \mathcal{A}, T \rangle$ where $\mathcal{A} = (Q, \Sigma, \delta, F)$ is a PTA (resp., a u-PTA or mixed PTA) and $T \subseteq Q$ is a set of states. Let $t \in \mathcal{T}(\Sigma)$. A context $c \in \mathcal{C}(t)$ is a *match* of the pattern $\langle \mathcal{A}, T \rangle$ in t iff $t = c[t']$ where $t' \models_{\mathcal{A}} q$ and $c, q \models_{\mathcal{A}} f$ for some $q \in T$ and $f \in F$.

We consider first mixed queries, with unordered ones as a special case. Whenever we speak about the complexity of the querying problem below, we mean the complexity of the following decision problem: given a query R, a tree t and a node v of t, is $v \in R(t)$?

Theorem 8.2. Let $\mathcal{A}$ be mixed PTA. The set of matches of a fixed Presburger pattern $\langle \mathcal{A}, T \rangle$, in a tree $t \in \mathcal{T}(\Sigma)$ of size n is computable in time $O(n)$. If the pattern is part of the input, the joint query complexity is NP-complete.

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, F)$. We proceed in two passes over the input tree t. In the first pass, we determine for every subtree t' of t the set of states:

$$B(t') = \{p \in Q \mid t' \models_{\mathcal{A}} p\} .$$

Let $\mathcal{A}'$ denote the deterministic automaton constructed from the mixed PTA $\mathcal{A}$ as in the proof of Theorem 6.2. Then we know that for every $t' \in \mathcal{T}(\Sigma)$, $t' \models_{\mathcal{A}'} B$ iff $B = \{p \in Q \mid t' \models_{\mathcal{A}} p\}$. Therefore, the sets $B(t')$ (over all subtrees t') can be determined by one bottom-up run of $\mathcal{A}'$ on t. According to Proposition 3.6, this first pass can be performed in linear time.

In the second pass, we determine for each context $c \in \mathcal{C}(t)$ with $t = c[t']$, the set of states:

$$D(c) = \{p \in B(t') \mid \exists f \in F : c, p \models_{\mathcal{A}} f\} .$$

Given the sets $D(c)$, the matches of the pattern are determined as the set of all contexts c where $T \cap D(c) \neq \varnothing$.

In order to determine the sets $D(c)$, we proceed top-down over t. For the root context c we set $D(c) = B(t) \cap F$. Assume that we are given a context c in t where $t = c[a\langle t_1, \ldots, t_k \rangle]$ for some $a \in \Sigma$ and subtrees t_i. Then we may proceed from the father node c to the son c_i which is defined as the context $c_i = c[a\langle t_1, \ldots, t_{i-1}, \bullet, \ldots, t_k \rangle]$. Remark that now $t = c_i[t_i]$. Let $B_i = B(t_i)$. Assume that we have already determined the value $D(c)$ and now want to determine the corresponding set for c_i.

Suppose first that the tag a is unordered, $a \in \Sigma_0$. For $B \subseteq Q$, let n_B denote the number of trees t_j, $1 \leq j \leq k$, such that $t_j \models_{\mathcal{A}'} B$. Let ρ denote the variable environment defined by:

$$\{x_B \mapsto n_B \mid B \subseteq Q\} .$$

We claim:

$$D(c_i) = \{q' \in B(t_i) \mid \rho \models \bigvee_{q \in D(c)} \psi_{q,q'}\}$$

where the formula $\psi_{q,q'}$ is given by:

$$\exists_{p \in Q} \#p \,.\, \exists_{p \in B \subseteq Q} x_{\langle B,p\rangle} \,.\, \delta(q,a) \;\wedge\; x_{\langle B_i,q'\rangle} > 0 \;\wedge \left(\bigwedge_{B \subseteq Q} \sum_{p \in B} x_{\langle B,p\rangle} = x_B\right) \wedge \left(\bigwedge_{p \in Q} \sum_{B,p \in B} x_{\langle B,p\rangle} = \#p\right) .$$

Intuitively, formula $\psi_{q,q'}$ expresses that there is an assignment mapping the children t_j to states $q \in B(t_j)$ such that t_i receives q' and the Presburger pre-condition $\delta(q,a)$ is satisfied. Since satisfiability of Presburger formulas is decidable, we conclude that the sets $D(c_i)$ are computable.

The total complexity of our algorithm in this part consists, for each node v labeled in Σ_0, in a test of an assertion $\rho \models \varphi$. Here, the formula φ only depends on the fixed automaton $\mathcal{A}$, and the variable environment ρ is such that $\rho(x_{\langle B,p\rangle}) \leq k$ for all $x_{\langle B,p\rangle}$ in the domain of ρ, with k denoting the number of children of v. Each formula φ can be transformed into a quantifier-free formula, which is evaluated in time $O(k)$ on numbers in unary representation. Since the sum of all k is bounded by n, the total complexity is in $O(n)$.

In the case where $a \in \Sigma_1$ we have:

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

Given a (non-deterministic) finite automaton $\mathcal{B}$ for $\delta(q,a)$, all sets $D_q(i)$, $i = 1,\ldots,k$, can be computed in time $O(k)$ as follows: by one left-to-right pass we compute at each position the set of reachable states of $\mathcal{B}$; in a second, right-to-left pass we compute at each position the set of states from which we can reach a final state of $\mathcal{B}$. With this information we compute all sets $D_q(i)$ in a final pass in $O(k)$.

Therefore, the overall complexity of the second pass is linear as well. This completes the proof in the case where the pattern is fixed.

For the joint complexity, consider first the upper bound. The first pass can be done deterministically in polynomial time, by computing bottom-up the reachable states at each node. For the top-down pass, we solve at each node an existential Presburger formula, which is done in NP. The lower bound follows from Proposition 3.6. Q.E.D.

As a special case of the querying algorithm in the proof of Theorem 8.2, we obtain a linear time querying algorithm for (fixed) queries on *classical* ordered trees (i.e., trees with $\Sigma_0 = \varnothing$).

We now consider ordered queries, i.e., queries stated as Presburger patterns $\langle \mathcal{A}, T \rangle$ where $\mathcal{A}$ is a PTA.

Theorem 8.3. The set of matches of a fixed Presburger pattern $\langle \mathcal{A}, T \rangle$, with $\mathcal{A}$ PTA, in a tree from $\mathcal{T}(\Sigma)$ is computable in polynomial time. If the pattern is part of the input, the joint query complexity is NP-complete.

Proof. Assume we have marked the root node of one subtree t' of t. Assume further that we have modified $\mathcal{A}$ in such a way that the marked node always receives a state in T. Then the modified tree is accepted iff t' is a match. Since there are only n different nodes to be marked, the theorem follows from Theorem 5.2.

For the joint query complexity we can implement easily the 2-pass approach of Theorem 8.2 in NP. The lower bound follows from the combined complexity of PTA. Q.E.D.

Let us turn to queries specified through Presburger MSO. A mixed PMSO-pattern is a mixed PMSO formula φ with at most one free variable y. A match of φ in t at a node v means that t together with the assignment of v to the free variable y satisfies φ. A query R is *mixed PMSO-definable* iff there is a mixed PMSO-pattern φ such that for every t, $R(t)$ is the set of all matches of φ in t. Replacing mixed PMSO by *existential* PMSO, we get *existential PMSO-definable* queries.

Theorem 8.4. For a query R the following statements hold:

1. R is mixed PMSO-definable iff R is definable by a Presburger pattern $\langle \mathcal{A}, T \rangle$ for some mixed PTA $\mathcal{A}$.
2. R is existential PMSO-definable iff R is definable by a Presburger pattern $\langle \mathcal{A}, T \rangle$ for some PTA $\mathcal{A}$.

As compared to PTA-based queries, it is worth noting that the joint query complexity of mixed PMSO-definable and existential PMSO-definable queries is PSPACE-complete. Both arguments for the upper and the lower bound use that alternating polynomial time is equivalent to PSPACE.

8.1 Presburger fixpoint queries

In this section we focus on unary queries expressed in Presburger fixpoint logic. Compared to PTA, fixpoint logic allows for very efficient algorithms – linear time for fixed queries and polynomial time for the joint query complexity.

In order to get an intuition about the expressive power of Presburger fixpoint logic, consider the example document shown in Figure 2. There we might first ask for all elements (tree nodes) containing "`Bartoli`". A second query could ask for elements containing "`Bartoli`" and having at least

```
<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 2. Part of a document with music items.

three reviews. In the fixpoint Presburger logic we can express that a tree contains a node satisfying a given property, without knowing at which depth this node occurs. For instance, the formula $\varphi_1 = *\langle \_ \texttt{Bartoli} \_ \rangle$ describes all nodes containing "`Bartoli`". Note that in order to take properties of text contents into account, it (conceptually) suffices to consider each text as a tag. We are not interested in the class of all these documents t, however, but for each such t in the subdocuments which satisfy the specific formula φ_1. Documents containing elements with the property φ_1 are described by the expression: $\mu\, x.(*\langle \_\ x\ \_ \rangle \vee \varphi_1)$. In order to indicate the subformula corresponding to the requested subdocuments, we introduce the extra marker "$\bullet$". Thus, we specify the query as $\psi_1 = \mu x.(*\langle \_\ x\ \_ \rangle \vee (\bullet \wedge \varphi_1))$. Accordingly for the second query, we describe the set of all elements containing at

least three reviews by: $\varphi_2 = *\langle \#\texttt{review} \geq 3 \rangle$. The query formula then can be formulated as:

$$\psi_2 = \mu x.(*\langle \_\ x\ \_ \rangle \ \vee\ (\bullet \wedge \varphi_1 \wedge \varphi_2))\,.$$

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

$$\varphi \quad ::= \quad \ldots \mid \bullet \mid \ldots\,.$$

Accordingly, we add new axioms $\quad \vdash t : \bullet \quad$ for all trees t. A *match* t' of a formula φ containing a subformula $\bullet$ is a proof for $t : \varphi$ containing the fact $t' : \bullet$. We want to construct an algorithm to determine for a fixed query formula φ, all matches inside a document tree t. We first observe that we can determine in time $O(|t|)$ for every subtree t' of t the set of all subformulas ψ of φ such that $\vdash t' : \psi$. For that, we can do as in Proposition 7.4 a bottom-up pass on t. 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 is unchanged. Let then $S(t')$ denote the set of subformulas ψ of type $a\langle F \rangle$, $*\langle F \rangle$ such that $t' : \psi$. By construction, $\psi \in \mathsf{cl}(S(t'))$ iff $\vdash t' : \psi$, for *every* subformula ψ of φ.

It remains to determine for every subtree t' of t the subset $R(t') \subseteq \mathsf{cl}(S(t'))$ containing all those ψ which may occur in some proof of $t : \varphi$. Then t' is a match iff $\bullet \in R(t')$. The subsets $R(t')$ are determined in a second pass over the tree t, in a top-down manner. 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 (including ψ). Let $\mathsf{core}'_B(\psi) = \mathsf{core}_B(\psi) \setminus \{\psi\}$. So, $\mathsf{core}'_B(\bullet) = \mathsf{core}'_B(\top) = \varnothing$, and

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

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

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 $R(t')$ for the subtree $t' = a\langle t_1 \ldots t_k \rangle$. Let $R' = R(t') \cap S(t')$ denote the set of subformulas in $R(t')$ of the form $a\langle F \rangle$ or $*\langle F \rangle$. Then $R(t_i) = \bigcup_{\psi \in R'} R_\psi(t_i)$, where $R_\psi(t_i)$ equals the set of formulas from $\mathsf{cl}(S(t_i))$ which

may have occurred in a proof of $t' : \psi$. Let $B_i = \mathsf{cl}(S(t_i))$ be the set of all subformulas that are valid at t_i. If $\psi = a\langle\pi\rangle$ or $\psi = *\langle\pi\rangle$ for a Presburger formula π, then we must compute the assignment to the variables of π. In fact, *all* subformulas from B_i contribute to this assignment. Therefore, we simply have $R_\psi(t_i) = B_i$ in this case. On the other hand, if $\psi = a\langle e\rangle$ or $\psi = *\langle e\rangle$ for a regular expression e, then $R_\psi(t_i) = \mathsf{core}_{B_i}(R_i)$ where

$$R_i = \{\psi_i \mid \exists\, \psi_1 \dots \psi_k \in \mathcal{L}(e) : \forall\, j : \psi_j \in B_j\}\,.$$

The set R_i denotes all subformulas provable for t_i which may contribute to the validation of e. According to this definition, the sets $R_\psi(t_i)$, $i = 1, \dots, k$ can jointly be computed by a left-to-right followed by a right-to-left pass of a finite (string) automaton for e over the children of t'. The case of negated regular expressions is treated analogously. Summarizing we conclude:

Theorem 8.5. Let φ be a fixed query in Presburger fixpoint logic. Then the set of matches of 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 $O(|\varphi|^2 \cdot |t|)$.

9 Conclusion

We have considered extensions of logics and automata over unranked trees by arithmetical Presburger constraints. Our motivation comes from XML, where one is interested in expressing properties of such trees that go beyond regular languages, such as numerical constraints. We showed that fixpoint logic extended by Presburger constraints has particularly pleasant properties, namely good expressiveness, complexity which does not increase with the additional Presburger part, and joint querying complexity which is polynomial.

Some of our results raise open problems. The universality problem for u-PTA is one of them: we have a 2-NEXPTIME upper bound, and as lower bound only EXPTIME. Another issue is the data complexity for general PTA: can we improve the bound or is it inherently difficult (w.r.t. fixed parameter complexity, with the size of the PTA as parameter)? Finally, it would be interesting to see whether the automata and logics can be enhanced by more general arithmetical constraints, like for instance the semi-polynomial or semi-quadratic sets considered in [16].

Acknowledgement: We thank the referee for his/her careful reading and the various remarks and suggestions that helped improving the paper.

References

[1] L. Berman. The Complexity of Logical Theories. *Theor. Comp. Sci.* 11:71-77, 1980.

[2] I. Boneva and J.M. Talbot. Automata and Logics for Unranked and Unordered Trees. In *16th Int. Conf. on Rewriting Techniques and Applications (RTA)*, pages 500–515. LNCS 3467, 2005.

[3] A. Brüggeman-Klein, M. Murata and D. Wood. Regular Tree Languages over Non-ranked Alphabets. Unpublished manuscript, 1998.

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

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

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

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

[8] J. Cristau, Ch. Löding and W. Thomas. Deterministic Automata on Unranked Trees. In *15th International Symposium on Fundamentals of Computation Theory (FCT)*, pages 68-79. LNCS 3623, 2005.

[9] S. Dal Zilio, D. Lugiez and C. Meyssonnier. A Logic You Can Count on. In *31st ACM Symp. on Principles of Programming Languages (POPL)*, pages 135–146, 2004.

[10] S. Dal Zilio and D. Lugiez. XML Schema, Tree Logic and Sheaves Automata. In *14th Int. Conf. on Rewriting Techniques and Applications (RTA)*, pages 246–263. LNCS 2706, 2003.

[11] S. Demri and D. Lugiez. Complexity of Modal Logics with Presburger Constraints. Research Report LSV-06-15, LSV, ENS Cachan, 2006.

[12] J. Esparza. Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes. *Fundam. Inform.* 31(1):13-25, 1997.

[13] 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.

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

[15] G. Gottlob and C. Koch. Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. In *20th ACM Conference on Principles of Database Systems (PODS)*, pages 17–28, 2002.

[16] W. Karianto, A. Krieg and W. Thomas. On Intersection Problems for Polynomially Generated Sets. *Int. Conf. on Automata, Languages and Programming (ICALP)*, pages 516–527. LNCS 4052, 2006.

[17] F. Klaedtke. On the Automata Size for Presburger Arithmetic. In *19th IEEE Symposium on Logic in Computer Science (LICS)*, pages 110-119, 2004. Journal version in *ACM Transactions on Computational Logic*, to appear.

[18] 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.

[19] 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, 2002.

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

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

[22] W. Martens and F. Neven. Typechecking Top-down Uniform Unranked Tree Transducers. In *Database Theory - ICDT 2003, 9th International Conference*, pages 64–78, 2003.

[23] W. Martens and J. Niehren. Minimizing Tree Automata for Unranked Trees. In *Database Programming Languages, 10th International Symposium (DBPL)*, LNCS 3774, 2005.

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

[25] A. Neumann and H. Seidl. Locating Matches of Tree Patterns in Forests. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, LNCS 1530, pages 134–145, 1998.

[26] F. Neven. Automata, Logic, and XML. In *16th International Workshop CSL*, pages 2–26. LNCS 2471, Springer, 2002.

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

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

[29] 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.

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

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

[32] C.H. Papadimitriou. On the Complexity of Integer Programming. *J. ACM*, 28(4):765–768, 1981.

[33] R. J. Parikh. On Context-free Languages. *J. ACM*, 13(4):570–581, 1966.

[34] 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*, 12:92-101, 1991. English translation of the original paper from 1929.

[35] H. Seidl. Haskell Overloading is DEXPTIME Complete. *Information Processing Letters (IPL)*, 54:57–60, 1994.

[36] H. Seidl, T. Schwentick and A. Muscholl. Numerical Document Queries. In *22nd ACM Conference on Principles of Database Systems (PODS)*, pages 155–166, 2003.

[37] H. Seidl and A. Neumann. On Guarding Nested Fixpoints. In *Ann. Conf. of the European Association of Logic in Computer Science (CSL)*, pages 484–498. LNCS 1683, 1999.

[38] H. Seidl, Th. Schwentick, A. Muscholl and P. Habermehl. Counting in Trees for Free. In *Int. Conf. on Automata, Languages and Programming (ICALP)*, pages 1136–1149. LNCS 3142, 2004.

[39] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time: Preliminary Report. In *STOC*, pages 1–9, 1973.

[40] J.W. Thatcher and J.B. Wright. Generalized Finite Automata with an Application to a Decision Problem of Second-order Logic. *Math. Syst. Theory*, 2:57-82, 1968.

[41] K. N. Verma, H. Seidl and Th. Schwentick. On the Complexity of Equational Horn Clauses. In *CADE*, pages 337–352, LNCS 3632, 2005.

[42] I. Walukiewicz. Pushdown Processes: Games and Model-checking. *Information and Computation*, 164(2):234–263, 2001.