Deterministic Automata for Unordered Trees

Adrien Boiret*†   Vincent Hugot ‡†   Joachim Niehren‡†   Ralf Treinen§

Automata for unordered unranked trees are relevant for defining schemas and queries for data trees in JSON or XML format. While the existing notions are well-investigated concerning expressiveness, they all lack a proper notion of determinism, which makes it difficult to distinguish subclasses of automata for which problems such as inclusion, equivalence, and minimization can be solved efficiently. In this paper, we propose and investigate different notions of "horizontal determinism", starting from automata for unranked trees in which the horizontal evaluation is performed by finite state automata. We show that a restriction to confluent horizontal evaluation leads to polynomial-time emptiness and universality, but still suffers from coNP-completeness of the emptiness of binary intersections. Finally, efficient algorithms can be obtained by imposing an order of horizontal evaluation globally for all automata in the class. Depending on the choice of the order, we obtain different classes of automata, each of which has the same expressiveness as Counting MSO.

1 Introduction

Logics and automata for unordered trees were studied in the last twenty years mostly for querying XML documents [12, 3, 18] and more recently for querying NOSQL databases [1]. They were already studied earlier, for modeling feature structures in computational linguistics [15] and records in programming languages [16, 9, 10].

In this paper, we shall consider unordered unranked data trees whose edges are labeled with strings over a finite alphabet, so that there are infinitely many such data values. For instance, we can consider a directory of a Linux file system as an unordered tree (when ignoring symbolic links and multiple hard links to files) given in JSON (the JavaScript Object Notation [11]), as for instance in Figure 1. This is a recent language-independent format for nested key-value stores, which already found much interest in Web browsers and for NOSQL databases such IBM's JAQL [2]. In this representation, we might want to verify that a LATEX repository contains exactly one main file, i.e. at most one file matching `*".tex"` whose content matches *"\documentclass"*. This property can be checked by formulæ from the Counting MSO fragment of Presburger MSO, but extended with regular expressions for matching data values:

$$\#(*".tex" : \{"\documentclass"* : \{\}\}) = 1$$

Alternatively, any formula of Presburger MSO can be expressed by a Presburger tree automaton [12, 3], if extended with regular expressions for matching data values.

The existing notions of tree automata for unordered trees are well-investigated concerning expressiveness (either Counting or Presburger MSO: CMSO or PMSO) [12, 3], and have the advantage that membership

---

*University of Lille 1, France

†Links (Inria Lille & LIFL, UMR CNRS 8022), France

‡Inria

§University Paris Diderot, PPS (UMR CNRS 7126), France

```
{ "file.tex" : {"\documentclass...":{}},
  "dir" : {
    "x.png" : {"<bin>":{}}, "y.png" : {"<bin>":{}},
  ...} ... }
```
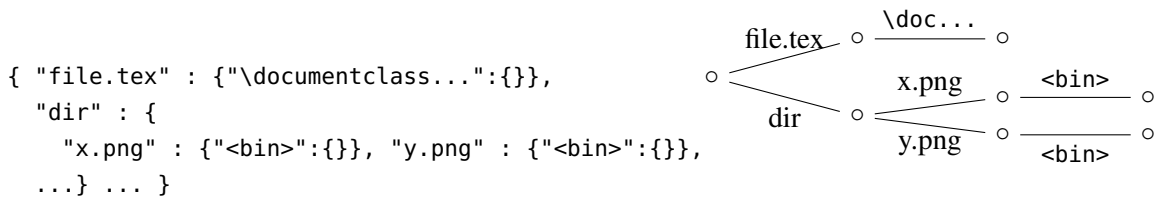

Figure 1: Unordered trees in JSON format, describing a typical file system.

can be tested in PTIME. When it comes to static analysis problems such as satisfiability, inclusion, or equivalence checking, they all lack a proper notion of determinism, which makes it difficult to distinguish subclasses of automata for which these problems can be solved efficiently. An exception is the class of feature automata [10] which have the same expressiveness as CMSO, but these have the disadvantage that they grow exponentially in size for testing simple patterns such as $\{$ *"a_1"* : $\{$ $\}$, *"a_2"* : $\{$ $\}$,...,*"a_n"* : $\{$ $\}\}$. The problem is that feature automata must be able to read the n different edge labels in all possible orders.

In this paper, we introduce a general framework for defining classes of bottom-up automata for unranked unordered trees, that abstracts from the way in which properties of horizontal languages are specified. The problem is to find a good notion of "horizontal determinism", since there exists no order on the children of a node. Rather than using Presburger formulæ for describing labels of the outgoing edges of a node we shall use for this purpose finite automata that rewrite the labels of outgoing edges of a given node in an arbitrary order. Unfortunately, membership testing becomes NP-hard, since all orders must be inspected in the worst case. A first notion of horizontal determinism can then be defined by a restriction to confluent horiontal rewriting, so that the order of rewriting becomes irrelevant. For instance, one can test the above arity in the order "a_1",..., "a_n" or else in the inverse order (but not necessarily in all orders in contrast to feature automata). Our first positive result is that the restriction to confluent rewriting leads to polynomial-time membership, emptiness, and universality, as one might have hoped. However, the emptiness of binary intersections as well as inclusion still suffers from coNP-completeness, which might appear a little surprising, so confluence alone is not sufficient for efficiency.

A second notion of horizontal determinism can be obtained by imposing a fixed order on the horizontal evaluation, globally for all automata in the class. Depending on the choice of the order, we obtain different classes of automata but all of them have the same expressiveness, which is that of CMSO. We show that this leads to polynomial time membership, emptiness, universality, emptiness of binary intersections, equivalence, and inclusion problems.

Outline   In Section 2, we recall the notions of automata for ranked ordered trees, unordered data trees, and Presburger formulæ. In Section 3, we introduce a general framework for defining classes of bottom-up automata for unordered trees. In Section 4, we instantiate our framework for introducing alternating Presburger tree automata, In Section 5, we discuss alternating tree automata with horizontal rewriting, and in Section 6 the restriction to confluent rewriting. Automata for fixed-order rewriting are introduced in Section 7.

2 Preliminaries

2.1 Automata on Ranked Ordered Trees

We recall here the classical model of tree automata on ranked trees (cf. [4] for an introduction). A ranked signature is a set Σ of function symbols, each of which has a fixed arity $ar(f) \in \mathbb{N}$. A ranked tree is a term t with the abstract syntax $t ::= f(t_1, \ldots, t_n)$ where $n = ar(f)$.

Definition 1. An *alternating (bottom-up) tree automaton for ranked trees* is a tuple $B = (\Sigma, \mathbb{Q}, \mathbb{Q}_{\text{fin}}, \mathbb{R})$ where Σ is a finite ranked signature, $\mathbb{Q}$ a finite set of states, $\mathbb{Q}_{\text{fin}} \subseteq \mathbb{Q}$ the set of final states, and $\mathbb{R}$ a finite set of rules of the form $\psi \to q$ where ψ is a formula with the abstract syntax $\psi ::= f(q_1, \ldots, q_n) \mid \psi \wedge \psi' \mid \neg \psi$ for $f \in \Sigma$ of arity $n \in \mathbb{N}$ and $q, q_1, \ldots, q_n \in Q$. A *nondeterministic tree automaton for ranked ordered trees* is an alternating tree automaton, whose rules are of the form $f(q_1, \ldots, q_n) \to q$. A *deterministic (bottom-up) tree automaton* is a nondeterministic tree automaton in which no two rules share the same left-hand side.

The evaluator of a nondeterministic tree automaton is defined by $[\![f(t_1, \ldots, t_n)]\!]_B = \{q \mid q_1 \in [\![t_1]\!]_B, \ldots, q_n \in [\![t_n]\!]_B, (f(q_1, \ldots, q_n) \to q) \in \mathbb{R}\}$. The language defined by B is $\{t \mid [\![t]\!]_B \cap \mathbb{Q}_{\text{fin}} \neq \varnothing\}$. For instance, consider the set of Boolean formulas $t ::= true \mid false \mid and(t, t)$. The set of all valid Boolean formulas can the be defined by the deterministic tree automaton with state set $\mathbb{Q} = \{0, 1\}$, final states $\mathbb{Q}_{\text{fin}} = \{1\}$ and rules $true \to 1$, $false \to 0$, and $and(1, 1) \to 1$.

In order to define an evaluator for more general alternating tree automata, we define the satisfaction relation $f(Q_1, \ldots, Q_n) \models \psi$ by $f(Q_1, \ldots, Q_n) \models g(q_1, \ldots, q_m)$ iff $m = n$, $g = f$, and $q_i \in Q_i$ for all $1 \leq i \leq n$, extended to negations and conjunctions as usual, i.e., $f(Q_1, \ldots, Q_n) \models \psi \wedge \psi'$ iff $f(Q_1, \ldots, Q_n) \models \psi$ and $f(Q_1, \ldots, Q_n) \models \psi'$, and $f(Q_1, \ldots, Q_n) \models \neg \psi$ iff not $f(Q_1, \ldots, Q_n) \models \psi$. As before, the language defined by B is $\{t \mid [\![t]\!]_B \cap \mathbb{Q}_{\text{fin}} \neq \varnothing\}$.

It is well known that alternating, nondeterministic, and deterministic tree automata can define the same classes of languages of ranked trees, which are those definable in MSO.

2.2 Unordered Unranked Data Trees

An *alphabet* is a finite set $\mathbb{A}$. A *data value* over $\mathbb{A}$ is a string in $\mathbb{A}^*$. We write $d_1 d_2$ for the concatenation of strings $d_1, d_2 \in \mathbb{A}^*$.

Let $\mathbb{N}$ be the set of natural numbers including 0. A *multiset* over a finite set D is a function $M : D \to \mathbb{N}$. The set of multisets over D is written $\mathbb{M}(D)$. As usual, we write $\{\!\!\{d_1, \ldots, d_n\}\!\!\}$ for the multiset in which each element of $d \in D$ has the same multiplicity as the number of occurrences of d within the brackets. Given a second set X, we use record notation for multisets over pairs in $D \times X$, i.e., we write $\{\!\!\{d_1 : x_1, \ldots, d_n : x_n\}\!\!\}$ instead of $\{\!\!\{(d_1, x_1), \ldots, (d_n, x_n)\}\!\!\}$ for any $d_i \in D$ and $x_i \in X$ – and sometimes use the same notations for isolated pairs $d : X$.

We define the set $\mathbb{T}$ of *unordered, edge–labelled data trees* (or simply *trees* in this paper) over data alphabet $\mathbb{A}$ inductively as the least set that contains all multisets $\{\!\!\{d_1 : t_1, \ldots, d_n : t_n\}\!\!\}$ such that $n \geqslant 0$, $d_1, \ldots, d_n \in \mathbb{A}^*$ and $t_1, \ldots, t_n \in \mathbb{T}$. Given a tree $t = \{\!\!\{d_1 : t_1, \ldots, d_n : t_n\}\!\!\}$, the multiset $\{\!\!\{d_1, \ldots, d_n\}\!\!\}$ is called the *arity* of t.

Figure 2: Drawings of $\{\!\!\{\, d_1 : \{\!\!\{\, d_3 : \{\!\!\{\,\}\!\!\} \,\}\!\!\}, d_1 : \{\!\!\{\, d_1 : \{\!\!\{\,\}\!\!\}, d_2 : \{\!\!\{\,\}\!\!\} \,\}\!\!\} \,\}\!\!\}$ with different edge orders.

We employ the usual graphic representation where $\{\!\!\{\, d_1 : \{\!\!\{\, d_3 : \{\!\!\{\,\}\!\!\} \,\}\!\!\}, d_1 : \{\!\!\{\, d_1 : \{\!\!\{\,\}\!\!\}, d_2 : \{\!\!\{\,\}\!\!\} \,\}\!\!\} \,\}\!\!\}$ is drawn as one of the many graphs in Figure 2 that differ only in the order in which the outgoing edges of nodes are drawn. Note that each node in a tree has a finite, but unbounded, number of sons.

We will use regular expressions as pattern for matching data values. A regular expression π has the following abstract syntax where $d \in \mathbb{A}^*$:

$$\pi ::= \text{``}d\text{''} \mid \pi\pi \mid \pi + \pi \mid \pi^* .$$

The set of regular expressions π is denoted by $\mathbb{E}_{\text{reg}}$. The semantics of a pattern $\pi \in \mathbb{E}_{\text{reg}}$ is a set of data values $[\![\pi]\!] \subseteq \mathbb{A}^*$ defined in the classical manner [6]. As syntactic sugar, we let $* \equiv (a_1 + \cdots + a_m)^*$, where $\mathbb{A} = \{a_1, \ldots, a_m\}$.

2.3   Descriptor Classes and Presburger Formulæ

Definition 2 (Descriptor Class). A *descriptor class* for a set $\mathcal{M}$ of models is a tuple $\langle \mathbb{H}, \models, |\cdot|, c \rangle$ where $\mathbb{H}$ is a set of *descriptors*, $\models$ a subset of $\mathcal{M} \times \mathbb{H}$, $|\delta| \in \mathbb{N}$ the *size* of a descriptor $\delta \in \mathbb{H}$, and $c \in \mathbb{N}$ the *cost* of the class.

As a first example, any subset $\mathbb{E} \subseteq \mathbb{E}_{\text{reg}}$ of regular expressions over our data alphabet $\mathbb{A}$ can be seen as a descriptor class selecting words in $\mathbb{A}^*$: satisfaction is defined as $d \models \pi$ iff $d \in [\![\pi]\!]$, the size $|\pi|$ of a pattern π is the number of its symbols, and the cost of the class is $c = 0$.

We recall the definition of propositional Presburger formulæ, which serve to specify properties of multisets. The logic is parametrized by a set X over which the multisets are constructed and a descriptor class $\langle \mathbb{F}, \models, |\cdot|, c \rangle$ providing descriptors for elements of X that we shall call *filters*.

Presburger formulæ ψ are built from filters as follows. One first constructs counting expressions ν, which are either constants $n \in \mathbb{N}$, sums $\nu + \nu'$, or counters for filters $\#\phi$ where $\phi \in \mathbb{F}$. A counter $\#\phi$ sums up the multiplicities of all elements of the multiset satisfying ϕ:

$$
\begin{aligned}
\nu &::= \quad n \mid \#\phi \mid \nu + \nu \\
\psi &::= \quad \nu \leq \nu' \mid \nu \equiv_m \nu' \mid \psi \wedge \psi' \mid \neg\psi
\end{aligned}
$$

An atomic Presburger formula $\nu \leq \nu'$ or $\nu \equiv_m \nu'$ compares the values of two counting expressions. General Presburger formulæ are constructed from atomic Presburger formulæ and the usual Boolean

operators from propositional logic. Given a multiset M over X, the semantics $[\![v]\!]^M \in \mathbb{N}$ is defined as usual:

$$[\![n]\!]^M = n, \qquad [\![v+v']\!]^M = [\![v]\!]^M + [\![v']\!]^M, \qquad [\![\#\phi]\!]^M = \sum_{x \models \phi} M(x)$$

We say that M satisfies the atomic formula $v \leq v'$ if $[\![v]\!]^M \leq [\![v']\!]^M$ and similarly M satisfies $v \equiv_m v'$ if $[\![v]\!]^M = [\![v']\!]^M \mod m$. This extends to Boolean combinations in the usual way. In this case we write $M \models \phi$.

Presburger formulæ with filters in $\mathbb{F}$ define a descriptor class for multisets over X. The size of a Presburger formula is the sum of the number of its symbols, excepting filters, plus the sizes $|\phi|$ of all occurrences of filters ϕ in the formula. The cost of the class of Presburger formulæ is the cost of its class of filters.

3 Automata for Unordered Trees

We start with abstract classes of bottom-up automata for unordered unranked trees, which generalize on alternating tree automata as well as on nondeterministic tree automata. This will enable us to introduce alternating Presburger automata (in Section 4) and alternating tree automata with horizontal rewriting (in Section 5) as concrete instances.

3.1 Automata for Unordered Trees

We fix Π, a countable set of *properties*. We develop a parametrized framework of automata, in which one can freely choose a descriptor class for matching arities that are decorated with sets of properties, which will also be sets of our automata's states.

Definition 3. A *horizontal descriptor class* $\mathbb{H}$ is a descriptor class for multisets over $\mathbb{A}^* \times \wp(\Pi)$.

The *support* $\mathrm{Supp}(h)$ of a horizontal descriptor $h \in \mathbb{H}$ is the set of all properties that h actually deals with; it is defined as the least subset $\mathbb{Q}$ of Π such that for any $i = 1..n$, $d_i \in \mathbb{A}^*$ and $Q_i, Q'_i \subseteq \Pi$ such that $Q'_i \cap \mathbb{Q} = Q_i \cap \mathbb{Q}$, $\{\!| d_1 : Q_1, \ldots, d_n : Q_n |\!\} \models h \iff \{\!| d_1 : Q'_1, \ldots, d_n : Q'_n |\!\} \models h$.

Definition 4 (AUTS). An *alternating bottom-up automaton for unordered unranked data trees* (AUT) is a tuple $A = \langle \mathbb{A}, \mathbb{Q}, \mathbb{Q}_{\text{fin}}, \mathbb{H}, \mathbb{R} \rangle$ where $\mathbb{Q} \subseteq \Pi$ is the finite set of *(vertical) states*, $\mathbb{Q}_{\text{fin}} \subseteq \mathbb{Q}$ the subset of final states, $\mathbb{H}$ is a horizontal descriptor class, and $\mathbb{R} \subseteq \mathbb{H} \times \mathbb{Q}$, such that for all $(h, q) \in \mathbb{R}$, $\mathrm{Supp}(h) \subseteq \mathbb{Q}$, is the finite set of *(vertical) transition rules*.

We shall write $h \to q$ if $(h, q) \in \mathbb{R}$. Any automaton A evaluates any tree with data alphabet $\mathbb{A}$ to a set of states. This set is defined by induction on the structure of trees such that for all $n \geq 0$, data values $d_1, \ldots, d_n \in \mathbb{A}^*$ and trees $t_1, \ldots, t_n \in \mathbb{T}$:

$$[\![\{\!| d_1 : t_1, \ldots, d_n : t_n |\!\}]\!]_A = \{ q \mid \{\!| d_1 : [\![t_1]\!]_A, \ldots, d_n : [\![t_n]\!]_A |\!\} \models h, \ h \to q \}.$$

Alternation requires to consider all states assigned to subtrees when applying a transition rule, and not only one of them nondeterministically. The *language accepted by A* is defined as $\mathscr{L}(A) = \{ t \in \mathbb{T} \mid [\![t]\!]_A \cap \mathbb{Q}_{\text{fin}} \neq \varnothing \}$. The size $|A|$ is the sum of the number of states $\#\mathbb{Q}$, the size $\sum_{h \to q} 1 + |h|$, and the cost of the descriptor class $\mathbb{H}$.

Definition 5. The *class* $\text{AUT}(\mathbb{H})$ is the set of all AUT whose horizontal descriptor class is $\mathbb{H}$.

As a first example, we consider the horizontal descriptor class $\mathbb{H}^{ar}$, which tests arity constraints. An arity constraint has the form $\{|\,"d_1":q_1,\ldots,"d_n":q_n\,|\}$, where $n\geq 0$, $q_1,\ldots,q_n\in\Pi$ and $d_1,\ldots,d_n\in\mathbb{A}^*$. It is satisfied by all multisets of the form $M+\{|\,"d_1":Q_1,\ldots,"d_n":Q_n\,|\}$ such that $q_i\in Q_i$ and $(d_i,Q)\notin M$ for all $1\leq i\leq n$ and any Q. The size of an arity constraint is the number of its symbols. The cost of any descriptor class $\mathbb{H}^{ar}$ is 0.

As a second example, we consider the richer class of horizontal descriptors $\mathbb{H}^{ar,\wedge,\neg}$ which, besides arity constraints, supports Boolean operators, i.e. the formulæ ψ of $\mathbb{H}^{ar,\wedge,\neg}$ are given by the following abstract syntax, where all $q_i\in Q$ and $d_i\in\mathbb{A}^*$:

$$\psi ::= \{|\,"d_1":q_1,\ldots,"d_n":q_n\,|\} \mid \psi\wedge\psi' \mid \neg\psi\,.$$

The automata from the classes $\text{AUT}(\mathbb{H}^{ar})$ and $\text{AUT}(\mathbb{H}^{ar,\wedge,\neg})$ show how easy it is to translate the notions of ranked automata into the unordered framework. More precisely:

Proposition 6 (*Encoding Automata for Ranked Ordered Trees*). *There exists an encoding $[\![\cdot]\!]$ of ranked ordered trees into unordered trees, and of alternating ranked ordered tree automata into $\text{AUT}(\mathbb{H}^{ar,\wedge,\neg})$, such that for any automaton B on ranked ordered trees we have $[\![\mathscr{L}(B)]\!]=\mathscr{L}([\![B]\!])$. Furthermore, if B is non-deterministic, then $[\![B]\!]\in\text{AUT}(\mathbb{H}^{ar})$.*

3.2 Complexity

Proposition 7 (*Membership*). *Let C be a class of automata such that for all $\langle\mathbb{A},\mathbb{Q},\mathbb{Q}_{fin},\mathbb{H},\mathbb{R}\rangle\in C$ and $h\in\mathbb{H}$, whether $\{|\,d_1:Q_1,\ldots,d_n:Q_n\,|\}\models h$ can be decided in time $O(\sum_{m=1}^n|d_m|\,|h|)$, for any $d_1,\ldots,d_n\in\mathbb{A}^*$, and finite sets $Q_1,\ldots,Q_n\subseteq\Pi$. In this case, membership $t\in\mathscr{L}(A)$ for trees $t\in\mathbb{T}$ and automata $A\in C$ can be decided in time $O(|t|\,|A|)$.*

A descriptor class $\mathbb{H}$ is closed by the boolean operation $\circledast$ if for every $h,h'\in\mathbb{H}$, there is $h\circledast h'\in\mathbb{H}$ such that $M\models h\circledast h'$ iff $(M\models h)\circledast(M\models h')$.

Proposition 8 (*Emptiness*). *Let $\mathbb{H}$ be such that (1) for any $h\in\mathbb{H}$, whether $\exists M:M\models h$ is decidable in time $O(g(|h|))$, and (2) $\mathbb{H}$ is closed by all boolean operations in linear time, and (3) for any $\mathbb{Q}\subseteq\Pi$ and $S\subseteq\wp(\mathbb{Q})$, there exists $\text{all}_S\in\mathbb{H}$ of size $O(2^{\#\mathbb{Q}})$ that is satisfied exactly by all multisets over $\mathbb{A}^*\times S$. In this case, whether $\mathscr{L}(A)=\varnothing$ can be decided in time $O(2^{2\cdot\#\mathbb{Q}}\cdot g(2^{\#\mathbb{Q}}+|A|))$ for all automata $A\in\text{AUT}(\mathbb{H})$ of states $\mathbb{Q}$.*

Proof. We perform a vertical reachability algorithm on sets of simultaneously reachable states. Each step involves testing all state subsets, and each test is exponential. There are at most an exponential number of steps, as each reachable subset remains reachable throughout. $\square$

Note that under the conditions of that proposition, the boolean closure properties for the automata, and the decidability of universality, disjointness, equivalence, and inclusion follow naturally, some technicalities notwithstanding. In a nutshell, one must be careful any time two descriptors acting on different sets of states must interact or relate to one another. We also need more sophisticated notions of boolean closure, for families of descriptor classes. Those details are more tedious than difficult, and are left out of this paper.

3.3 Vertical Determinism

We next introduce the notion of vertical determinism, which is the "standard" view of determinism in bottom-up automata: that is to say, trees are evaluated in at most one state. Like in the ranked case, vertical determinism is necessary in order to obtain good complexities for static analysis problems. It is not sufficient, however: all the classes which we consider in the next sections define filters (see Sec. 2.3[p192]) that can manipulate regular patterns and properties, by conjunction, disjunction or negation. If all regular patterns were allowed, testing satisfiability of such filters would be PSPACE-hard (by emptiness of intersection of regular languages). And if the automata were alternating, NP-hardness would be hard to avoid, as sets of properties are tested, which can encode variable assignments, for instance (SAT problem). To get reasonable complexity and reasonable expressive power, one must therefore combine vertical determinism and restrictions on the patterns one can test.

Fortunately, provided that $\mathbb{H}$ satisfies the boolean closure properties, any $\text{AUT}(\mathbb{H})$ can be transformed into an equivalent, vertically deterministic $\text{AUT}(\mathbb{H})$.

Definition 9 (Vertical Determinism). An AUT A is *vertically deterministic* if $\max_{t \in \mathbb{T}}(\#[\![t]\!]_A) = 1$.

Proposition 10 (*Vertical Determinisation*). *For any $A \in \text{AUT}(\mathbb{H})$, an equivalent vertically deterministic $B \in \text{AUT}(\mathbb{H})$ can be constructed in time $O(2^{2^{|A|}})$, provided that $\mathbb{H}$ is closed in linear time under conjunction and negation.*

4 Alternating Presburger Tree Automata

We introduce *alternating Presburger automata* for unordered unranked data trees ($\text{AUT}^{\#}$s), by instantiating the horizontal descriptors of AUT's by propositional Presburger formulæ, and present expressiveness and complexity results.

We now define the descriptor class $\mathbb{H}^{\#}$ of propositional Presbuger formulæ. We first fix a descriptor class $\mathbb{E}$, subclass of $\mathbb{E}_{\text{reg}}$, for words in $\mathbb{A}^*$. We then define the filters ϕ with the following syntax, where π is a descriptor of $\mathbb{E}$ and $q \in \Pi$:

$$\phi ::= \pi \mid q \mid \phi \wedge \phi \mid \neg \phi \ .$$

The semantics is defined as follows, for $(d, Q) \in \mathbb{A}^* \times \wp(\Pi)$: $(d, Q) \models q$ iff $q \in Q$, $(d, Q) \models \pi$ iff $d \models \pi$. The inductive cases are as usual. The size of a filter $|\phi|$ is the number of its symbols plus $|\pi|$ for all occurences of π. The cost of the filter class is the cost of the pattern class, i.e. 0.

Definition 11 (AUT$^{\#}$: Alternating Presburger Tree Automata). The class AUT$^{\#}$ of *alternating bottom-up Presburger automaton for unordered unranked trees* is defined as $\text{AUT}(\mathbb{H}^{\#})$.

Alternating Presburger automata take into account the fact that a tree may be recognized in several states. For instance, $\{\!\!\{ d_1 : \{q_1, q_2\}, d_2 : \{q_2, q_3\} \}\!\!\} \models \#q_1 + \#q_2 = 3$. This allows in general to obtain more concise automata than in case of the Presburger tree automata of [12, 3] which are non-deterministic, which is to say that acceptance is based on the notion of a run that assigns a single state to each tree, even when this is done in a non-deterministic way.

As a consequence, $\text{AUT}^{\#}$ do not directly capture all Presburger tree automata from [3], but only those that are alternation-free. From the viewpoint of expressiveness, this is good enough, since vertically deterministic $\text{AUT}^{\#}$s capture PMSO already, as we shall see in Theorem 13.

Example 12. Let us illustrate the above by showing an $\text{AUT}^{\#}$ checking some basic cleanness criteria for a LaTeX document directory. We require that the files produced by compilation, i.e. all files whose name matches `*.dvi`, `*.pdf`, `*.aux`, are absent from the directory. Furthermore, all `*.tex` must be simple files, and exactly one among them must be a valid TeX main document. There is no restriction on the subdirectories. We note $\pi_{\text{main}} = \text{"}\backslash documentclass\text{"}*$, $\pi_{\text{cmp}} = *\text{"}.dvi\text{"} + *\text{"}.pdf\text{"} + *\text{"}.aux\text{"}$. We have the following rules:

$$
\begin{aligned}
\#(*) = 0 &\rightarrow q_{\text{leaf}} \\
\#(\pi_{\text{main}} \wedge q_{\text{leaf}}) = 1 \wedge \#(*) = 1 &\rightarrow q_{\text{main}} \\
\#(q_{\text{leaf}}) = 1 \wedge \#(*) = 1 &\rightarrow q_{\text{file}} \\
\#(*\text{".\textit{tex}"} \wedge q_{\text{main}}) = 1 \wedge \#(*\text{".\textit{tex}"} \wedge \neg q_{\text{file}}) = 0 \wedge \#(\pi_{\text{cmp}}) = 0 &\rightarrow q_{\text{ok}}
\end{aligned}
$$

State q_{leaf} is assigned to leaf nodes, and state q_{file} to all nodes representing files, i.e. nodes with exactly one outgoing edge, whose data value is the file's content and whose target is a leaf node. State q_{main} is assigned to all nodes with one outgoing edge labeled by the content of a main LaTeX file, i.e., a string matching "$\backslash documentclass$". State q_{ok} accepts only clean LaTeX repositories.

Note that the properties tested by q_{main} and q_{file} are not mutually exclusive. To make this automaton alternation-free, we would have to force q_{file} to specifically test that its only data value doesn't match "$\backslash documentclass$". Here, alternation facilitates specification.

We now present a logical characterization of the expressiveness of $\text{AUT}^{\#}$'s, showing that they capture PMSO. We assume a possibly infinite set $\mathscr{Q}$ of set variables ranged over by q. Let ψ range over propositional Presburger formulæ with predicates in $\mathscr{Q}$, which describe multisets over $\mathbb{A}^* \times 2^{\mathscr{Q}}$. The formulæ α and β of PMSO are then defined by:

$$
\begin{aligned}
\text{node sets} \quad & \alpha ::= \psi \mid q \mid \{root\} \\
\text{truth values} \quad & \beta ::= \alpha \subseteq \alpha' \mid \beta \wedge \beta' \mid \neg\beta \mid \forall q.\beta
\end{aligned}
$$

A formula with free variables in $\mathbb{Q} \subseteq \mathscr{Q}$ can then be interpreted over an unordered tree t and a assignment σ of $\mathbb{Q}$ to set of nodes of t. This defines a satisfaction relation $t, \sigma \models \beta$.

Theorem 13 ($\text{AUT}^{\#}$ *Expressiveness*). *A language of unordered data trees is definable by an $\text{AUT}^{\#}$ if and only if it is definable by a closed PMSO formula with the appropriate alphabet.*

Proof sketch. Let A be an $\text{AUT}^{\#}$. We translate rules $\psi \rightarrow q$ by $\psi \subseteq q$. We then impose that all sets q are minimal while satisfying all the PMSO formulæ for the rules. Finally, we impose that $\vee_{q \in \mathbb{Q}_{\text{fin}}} \{root\} \subseteq q$, and quantify existentially over all predicates. Conversely, we can show that the non-alternating Presburger tree automata from Boneva and Talbot [3] can be expressed by $\text{AUT}^{\#}$'s. Their definition is close to that of $\text{AUT}^{\#}$'s except that the semantics is defined in a non-deterministic manner, and not alternating. Nevertheless, our formalism subsumes in a natural manner the subclass of their Presburger tree automata that are horizontally deterministic Since they capture PMSO [3], $\text{AUT}^{\#}$'s do too (except that string patterns are not supported by their version of PMSO and Presburger automata, but they can be eliminated in a preprocessing step). Alternatively, a direct proof of this result can be obtained as usual when relating MSO to standard tree automata over ranked trees [17, 4]. $\qquad\square$

Proposition 14 (AUT$^\#$ *Complexity*). *Given vertically deterministic* AUT$^\#$ A, B *and a tree* $t \in \mathbb{T}$, *deciding whether* $t \in \mathscr{L}(A)$ *is* PTIME, $\mathscr{L}(A) = \varnothing$ *is* PSPACE-*hard,* $\mathscr{L}(A) = \mathbb{T}$ *is* PSPACE-*hard,* $\mathscr{L}(A) \cap \mathscr{L}(B) = \varnothing$ *is* PSPACE-*hard.*

Proof. These complexity results follow from known results on Presburger logic [14, 13]. $\qquad\square$

5   AUTs with Horizontal Rewriting

We next introduce alternating automata with horizontal rewriting (AUT$^{\twoheadrightarrow}$s) by instantiating AUTs with "horizontal" automata whose transitions are guarded by filters. AUT$^{\twoheadrightarrow}$s have the same expressiveness as AUT$^\#$s but differ in computational properties and succinctness. As we shall see in the next section, AUT$^{\twoheadrightarrow}$ make it indeed easier to formulate restrictions leading to more efficient static analysis.

Let $\mathbb{F}$ be the set of filters ϕ for words in $\mathbb{A}^* \times \wp(\Pi)$ from the previous section, i.e., $\phi ::= \pi \mid q \mid \phi \wedge \phi \mid \neg \phi$ where $q \in \Pi$ and $\pi \in \mathbb{E}$, where $\mathbb{E} \subseteq \mathbb{E}_{\mathrm{reg}}$.

Definition 15. A *horizontal automaton* is a triple $\langle \mathbb{A}, \mathbb{P}, \delta \rangle$ where $\mathbb{P}$ is a finite set of *horizontal states* and $\delta \subseteq \mathbb{P} \times \mathbb{F} \times \mathbb{P}$ is the *horizontal transition relation*.

We will write $p\phi \to p'$ instead of $(p, \phi, p') \in \delta$. Any horizontal automaton $H = \langle \mathbb{A}, \mathbb{P}, \delta \rangle$ defines a descriptor class $\mathbb{H}_H = \langle \mathbb{P}^2, \models, |\cdot|, c \rangle$ for multisets over $\mathbb{A}^* \times \wp(\Pi)$. Its descriptors are pairs of horizontal states $(p, p') \in \mathbb{P}^2$, where p serves as an *initial* and p' as a *final* horizontal state of the descriptor. The *horizontal rewriting relation* of H is the binary relation $\twoheadrightarrow$ on $\mathbb{P} \times \mathbb{M}(\mathbb{A}^* \times \wp(\mathbb{Q}))$ given by:

$$(p, M + \{\!| d : Q |\!\}) \twoheadrightarrow (p', M) \qquad \text{if} \qquad \exists \phi : p\phi \to p' \quad \text{and} \quad (d, Q) \models \phi \, .$$

A multiset M over $\mathbb{A}^* \times \wp(\Pi)$ satisfies a descriptor (p, p') if p' can be reached from p while consuming M: $M \models (p, p') \iff (p, M) \twoheadrightarrow^* (p', \{\!| \, |\!\})$. The size of a descriptor p, p' is $|(p, p')| = 2$ while the cost of the class is the overall size of the horizontal automaton $c = \sum_{(p, \phi, p') \in \delta} |\phi|$.

Definition 16 (AUT$^{\twoheadrightarrow}$). The class AUT$^{\twoheadrightarrow}$ of *alternating bottom-up automaton for unordered unranked trees with horizontal sub-automata* is defined as the union of all classes AUT($\mathbb{H}_H$) such that H is a horizontal automaton with alphabet $\mathbb{A}$.

For vertically deterministic automata, filters can be applied only to pairs (d, Q) such that $\#Q \leq 1$. Therefore we will be interested in restricted problems for filters, in which the state set Q of all models are either empty or singletons. We will call the restricted problems of filter *singleton-membership*, *singleton-satisfiability*, *singleton-validity*, etc. It should be noticed that the singleton-restricted problems are usually much easier than the general case. For instance, if $\mathbb{E} = \varnothing$ then singleton-satisfiablity and singleton validity of filters is in PTIME. This also remains true, if only suffixes can be tested by patterns, i.e., if $\mathbb{E} = \{* \text{``}d\text{''} \mid d \in \mathbb{A}^*\}$.

Proposition 17 (AUT$^{\twoheadrightarrow}$ *Complexities*). *Given two vertically deterministic* AUT$^{\twoheadrightarrow}$ A, B, *a tree* $t \in \mathbb{T}$, *then if singleton-satisfiability of* ϕ *is decidable in time* $O(f(|\phi|))$, *whether* $\mathscr{L}(A) = \varnothing$ *can be tested in time* $O(|A|^2 \cdot f(|A|))$, *whether* $t \in \mathscr{L}(A)$ *is* NP-*complete, whether* $\mathscr{L}(A) = \mathbb{T}$ *is* PSPACE-*hard, and provided that singleton-satisfiability of a filter* ϕ *is testable in polynomial time, deciding whether* $\mathscr{L}(A) \cap \mathscr{L}(B) = \varnothing$ *is* CONP-*complete.*

Proof sketch. The hardness results follow from known lower bounds, see for instance [8, 7]. Emptiness follows from a vertical accessibility algorithm where each phase performs a horizontal accessibility algorithm. A proof for membership is just a vertical run, with assorted horizontal runs, checkable in PTIME, hence the upper bound. For $\mathscr{L}(A) \cap \mathscr{L}(B) = \varnothing$, the polynomial check of [8, 7] can be used in our case by replacing the infinite alphabet $\mathbb{A}$ by the pair of rules a labeled datavalue would use in the horizontal automata of A and B. To make sure we do not combine two mutually disjunctive rules, we need to make sure in polynomial time that their conjunction is singleton-satisfiable. $\square$

6   AUTs with Confluent Horizontal Rewriting

In this section we move towards more tractable classes: we define a subclass of vertically deterministic AUT$^{\rightarrow}$s for which the horizontal automata must be confluent. Intuitively, that means that, during the horizontal evaluation, one can choose any available transition in a "don't care" manner, since all possible choices will yield the same result at the end.

The resulting expressive power lies strictly between CMSO and PMSO. For instance, one can test $\#q = \#q'$, which is not in CMSO, but cannot test $\#q \leqslant \#q'$, even though this can be tested in PMSO. Despite its high expressive power, this model has some good static analysis properties.

A horizontal automaton $H = \langle \mathbb{A}, \mathbb{P}, \delta \rangle$ is called *confluent* if the *failure-extended horizontal rewriting relation* $\rightarrowtail$ is confluent, where $\rightarrowtail$ is defined as the smallest relation such that

$$\twoheadrightarrow \; \subseteq \; \rightarrowtail \quad \text{and} \quad (p, M) \rightarrowtail \bot \quad \text{if} \quad M \neq \{\!\!|\;\;|\!\!\} \quad \text{and} \quad \not\exists p', M' : (p, M) \twoheadrightarrow (p', M') \, .$$

Its p_0-*confluent descriptor class* $\mathbb{H}^{\Diamond}_{H, p_0}$, for $p_0 \in \mathbb{P}$, is the subclass of $\mathbb{H}_H$ where the descriptors are limited to $\{p_0\} \times \mathbb{P}$. Indeed, having several initial states would be "cheating" the confluence.

Definition 18 (AUT$^{\Diamond}$). An AUT$^{\Diamond}$ is a vertically deterministic member of any class AUT($\mathbb{H}^{\Diamond}_{H, p_0}$), where $H = \langle \mathbb{A}, \mathbb{P}, \delta \rangle$ is a confluent horizontal automaton, and $p_0 \in \mathbb{P}$.

Proposition 19 (AUT$^{\Diamond}$ *Closure Properties*). *AUT$^{\Diamond}$s are neither closed under union nor complement.*

Proof. $\#``a" = \#``b"$ and $\#``a" = \#``c"$ are recognizable by a confluent automaton, $\#``a" = \#``b" \vee \#``a" = \#``c"$ is not: the class is not closed under union. $\#``a" = \#``b"$ is recognizable by a confluent automaton, $\#``a" \neq \#``b"$ is not: the class is not closed under complement. $\square$

Proposition 20 (AUT$^{\Diamond}$ *Membership*). *If singleton-membership of filters is in* PTIME, *then one can decide for any AUT$^{\Diamond}$ A and tree t whether $t \in \mathscr{L}(A)$ in polynomial time.*

Proof. Since the horizontal automaton is confluent, the greedy strategy of reading a data value whenever we can always gives the proper result. We make such a test for each node of the input tree in a bottom-up manner. $\square$

Proposition 21 (AUT$^{\Diamond}$ *Emptiness*). *If singleton-satisfiability of filters is in* PTIME, *then it is decidable in polynomial time for an AUT$^{\Diamond}$ A, whether $\mathscr{L}(A) = \varnothing$.*

Proof. This is a particular case of Prop 17. $\square$

Proposition 22 ($\textsc{aut}^\Diamond$ *Universality*). *If the singleton-validity of filters is in* PTIME*, then it is decidable in polynomial time for any* $\textsc{aut}^\Diamond$ *A whether* $\mathscr{L}(A) = \mathbb{T}$.

Proof. We check that all vertical (resp. horizontal) accessible states are accepting and can read any possible arity (resp. labeled data value). □

Proposition 23 ($\textsc{aut}^\Diamond$ *Disjointness*). *If the singleton-satisfiability of the conjunction of two filters is in* PTIME*, then deciding for two* $\textsc{aut}^\Diamond$ A_1, A_2, *whether* $L(A_1) \cap L(A_2) = \varnothing$ *is* CONP*-complete.*

Proof. The CONP-hardness is inherited from the same problem on horizontal automata: given two confluent horizontal automata H_1 and H_2 on the same set of states $\mathbb{Q}$, and two descriptors (p_1, p'_1) of H_1 and (p_2, p'_2) of H_2, decide whether there exist a multiset M on singletons such that $M \models (p_1, p'_1)$ and $M \models (p_2, p'_2)$. This result is a reduction of 3-coloring a graph: we encode successful colorings as a Parikh language in the intersection of $L(p_1, p'_1) \cap L(p_2, p'_2)$. The problem is already in CONP for $\textsc{aut}^{\rightarrow}$ by Proposition 17, so it is also in CONP for the more restricted class $\textsc{aut}^\Diamond$. □

Proposition 24 ($\textsc{aut}^\Diamond$ *Inclusion*). *If the singleton-satisfiability of filters is in* NP*, then deciding for* $\textsc{aut}^\Diamond$ A_1 *and* A_2 *whether* $L(A_1) \subseteq L(A_2)$ *is* CONP*-complete.*

Proof. Again, CONP-hardness is inherited from the analogous problem on horizontal automata: given two $\textsc{aut}^\Diamond$s H_1 and H_2 on the same set of states $\mathbb{Q}$, and two descriptors (p_1, p'_1) of H_1 and (p_2, p'_2) of H_2, decide whether there exist a multiset M on singletons such that $M \models (p_1, p'_1)$ but $M \not\models (p_2, p'_2)$. This result is a reduction of 3-coloring a graph: we encode successful colorings as a Parikh language recognized by $L(p_1, p'_1) \backslash L(p_2, p'_2)$, with H_1 and H_2 confluent horizontal automata.

The CONP check on horizontal automata is proper to the confluent restriction, as the problem is PSPACE-hard in the general case. Since membership is polynomial we just have to ensure that there exists a counter-example of polynomial size in case inclusion does not hold. Let H_1 be a confluent automaton, p_0 an "initial" state, $p_1, \ldots, p_n$ "final" states. A *minimal acceptor* M of p_i is a multiset such that there is a state p_j ($j \neq 0$) such that $M \models (p_i, p_j)$, but for every $M' \subseteq M$ there is no final state p such that $M' \models (p_i, p)$. Since we consider confluent automata, these minimal acceptors describe a greedy strategy for accepting a multiset: we can partition M into minimal acceptors M_0 from p_0 to a p_{i_1}, then M_1 from p_{i_1} to a p_{i_2} … until M is entirely read. Since H_1 is confluent, this method works if and only if M goes from p_0 to a final state p. From this, we now consider a second confluent automaton H_2, with its initial state p'_0 and its "final" states $p'_1, \ldots, p'_m$. We try to read M in H_2 the same way: M_0 first, then M_1… If at any step M_k we do not end up in a final state in H_2, then $M_0 + \cdots + M_k$ is a counter-example for the inclusion. If $M_0 + \cdots + M_k$ ends up in a final state of H_2 then we get a pair of final states (p_{i_k}, p'_{j_k}). By a pumping argument, we can get rid of loops and need a less-than-quadratic number of minimal acceptors to reach a counter-example. Each minimal acceptor cannot be bigger than the number of states in H_1. Since there is a counter-example of polynomial size, and the membership problem is polynomial, we have a polynomial check.

We use this in the vertical automaton: We consider two $\textsc{aut}^\Diamond$ A_1, A_2. To put their labeling on the same automaton, we make sure that H_1 and H_2 now test arities labeled on pairs of $\mathbb{Q}_1 \times \mathbb{Q}_2$. We can nondeterministically guess a set of accessible pairs by using Prop 23. From there, we nondeterministically guess a counter-example of arity labeled on these accessible pairs that leads to a final state in A_1 but not in A_2. □

7   AUTs with Ordered Horizontal Rewriting

We now introduce a subclass whose expressive power id even more restricted than that of $\text{AUT}^{\diamond}$. In this class, the filters are required to be disjoint – or made to be so beforehand at quadratic cost – and are linearly ordered, the order itself being a parameter of the class. Each rule has its own horizontal automaton, restricted to reading the arity following this order. Compared to the confluent case, $\#a = \#b$ for instance is no longer expressible, but $\#a \equiv \#b \mod k$ still is.

The filters in a given horizontal automaton being disjoint, we let $\Sigma = \{\phi_1, \ldots, \phi_n\}$ be the chosen finite alphabet of filters, and view an arity $\{\!\!\{ d_1 : Q_1, \ldots, d_n : Q_m \}\!\!\}$ as $\{\!\!\{ \varphi_1, \ldots, \varphi_m \}\!\!\}$, where φ_i is the unique $\phi \in \Sigma$ such that $(d_i, Q_i) \models \phi$; this is undefined if there is no such ϕ. Thus we see arities as Parikh images of words on the finite alphabet Σ, and horizontal automata as – deterministic – finite state automata on Σ.

Deterministic finite automata (DFA) are defined as usual as tuples $\kappa = \langle \Sigma, P, p_{\text{ini}}, P_{\text{fin}}, \delta \rangle$, where $\delta : P \times \Sigma \to P$. We write a transition simply $p\phi \to p'$. The word language of a DFA κ is written $\mathscr{L}(\kappa)$, and its *Parikh language* is the Parikh image of its word language, i.e. $\mathscr{P}(\kappa) = \{ \{\!\!\{ \varphi_1, \ldots, \varphi_m \}\!\!\} \mid \varphi_1 \ldots \varphi_m \in \mathscr{L}(\kappa) \}$. Given a linear order $\prec$ on Σ (or on $\mathbb{F} \supseteq \Sigma$) such that $\phi_1 \prec \cdots \prec \phi_n$, the $\prec$-ordered language of κ is $\mathscr{L}_{\prec}(\kappa) = \mathscr{L}(\kappa) \cap \phi_1^* \ldots \phi_n^*$, and we also let $\mathscr{P}_{\prec}(\kappa)$ be the Parikh image of $\mathscr{L}_{\prec}(\kappa)$.

We define the corresponding horizontal descriptor class $\langle K_{\Sigma, \prec}, \models, |\cdot|, 0 \rangle$, with $K_{\Sigma, \prec}$ being the set of DFA on a set $\Sigma \subseteq \mathbb{F}$ of mutually disjoint filters, $|\kappa|$ being the usual size for DFA, and the following satisfaction relation:

$$\{\!\!\{ d_1 : Q_1, \ldots, d_m : Q_m \}\!\!\} \models \kappa \quad \text{iff} \quad \forall i = 1..m, \exists \varphi_i \in \Sigma : d_i : Q_i \models \varphi_i \wedge \{\!\!\{ \varphi_1, \ldots, \varphi_m \}\!\!\} \in \mathscr{P}_{\prec}(\kappa) .$$

Definition 25 ($\text{AUT}^{\Sigma, \prec}$). An $\text{AUT}^{\Sigma, \prec}$ is a is vertically deterministic member of $\text{AUT}(K_{\Sigma, \prec})$.

Proposition 26 (*Reordering*). *For any* $\text{AUT}^{\Sigma, \prec}$ *A, and any total filter order* $\prec'$, *one can construct an* $\text{AUT}^{\Sigma, \prec'}$ *A' equivalent to A in time* $O(2^{|A| \cdot \#\Sigma})$.

In practice, a direct algorithm considering the "decision trees" underlying the horizontal automata, testing each ϕ separately in the order $\prec$, and reordering the decisions according to $\prec'$, generally avoids explosive size increase. Its asymptotic bound is currently worse, though: $O(2^{2|A| \cdot \#\Sigma})$.

Proposition 27 (CMso-*Equivalence*). *For any total order* $\prec$, $\text{AUT}^{\Sigma, \prec}$ *has exactly the same expressive power as* CMso.

Proof sketch. It is obvious that $K_{\Sigma, \prec}$ can encode counting constraints, as they can encode $\#a \leqslant k$ and $\#a = k \mod n$, and are closed under Boolean operations. Conversely, $K_{\Sigma, \prec}$ can be seen as a succession of components dealing with ϕ_i^*-factors – as for reordering, – each of which can be put into Chrobak normal form [5], and can hence be expressed as a disjunction of modulos.   □

Proposition 28 ($\text{AUT}^{\Sigma, \prec}$ *is Easy*). *Given an order* $\prec$ *on filters, the membership, emptiness, universality, disjointness, equivalence, and inclusion decision problems for vertically deterministic* $\text{AUT}^{\Sigma, \prec}$ *are all polynomial, provided that the corresponding singleton problems for filters are.*

Proof. This follows from the same results for DFA.   □

	AUT$^{\#}$	AUT$^{\twoheadrightarrow}$	AUT$^{\diamondsuit}$	AUT$^{\Sigma,\prec}$
Characterisation:	PMSO	PMSO	CMSO $< \cdot <$ PMSO	CMSO
$t \in \mathscr{L}(A)$?	in PTIME	NP-complete	in PTIME	in PTIME
$\mathscr{L}(A) = \varnothing$?	PSPACE-hard	in PTIME	in PTIME	in PTIME
$\mathscr{L}(A) \cap \mathscr{L}(B) = \varnothing$?	PSPACE-hard	CONP-complete	CONP-complete	in PTIME
$\mathscr{L}(A) = \mathbb{T}$?	PSPACE-hard	PSPACE-hard	in PTIME	in PTIME
$\mathscr{L}(A) = \mathscr{L}(B)$?	PSPACE-hard	PSPACE-hard	in CONP	in PTIME
$\mathscr{L}(A) \subseteq \mathscr{L}(B)$?	PSPACE-hard	PSPACE-hard	CONP-complete	in PTIME

Table 1: Overview of the complexity results for vertically deterministic AUTs with various assumptions on patterns or filters.

8 Conclusion and Future Work

We have introduced a very general setting for bottom-up automata on unranked unordered trees on infinite alphabets, which captures the usual notions of alternation and determinism with respect to the *vertical* – bottom-up – structure of automata, and is parametrized by the modality of *horizontal* evaluation. We have shown that this model, with Presbuger formulæ or Parikh-like automata, captures PMSO, with complexity trade-offs between membership and emptiness. Searching for classes suitable both for querying and static analysis, we then examined two notions of *horizontal* determinism: confluence and fixed-orderedness, the latter yielding the same expressive power as CMSO, and the former a strict intermediate between CMSO and PMSO. Our complexity results are summarized in Table 1 [p201].

To extend this work, we intend to explore more powerful variants where filters support data joins, and to generalize the approach to tree transducers, with applications to static verification of scripts, some subclasses of which can be seen as transducers on filesystem trees.

References

[1] Véronique Benzaken, Giuseppe Castagna, Kim Nguyen & Jérôme Siméon (2013): *Static and dynamic semantics of NoSQL languages*. In Roberto Giacobazzi, Radhia Cousot, Roberto Giacobazzi & Radhia Cousot, editors: *POPL*, ACM, pp. 101–114, doi:10.1145/2429069.2429083.

[2] Kevin S. Beyer, Vuk Ercegovac, Rainer Gemulla, Andrey Balmin, Mohamed Y. Eltabakh, Carl C. Kanne, Fatma Özcan & Eugene J. Shekita (2011): *Jaql: A Scripting Language for Large Scale Semistructured Data Analysis*. PVLDB 4(12), pp. 1272–1283. Available at http://dblp.uni-trier.de/rec/bibtex/journals/pvldb/BeyerEGBEKOS11.

[3] Iovka Boneva & Jean-Marc Talbot (2005): *Automata and Logics for Unranked and Unordered Trees*. In: *20th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3467, Springer Verlag, pp. 500–515, doi:10.1007/978-3-540-32033-3_36.

[4] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison & Marc Tommasi (2007): *Tree Automata Techniques and Applications*. Available online since 1997: http://tata.gforge.inria.fr.

[5] Paweł Gawrychowski (2011): *Chrobak normal form revisited, with applications*. In: *Implementation and Application of Automata*, Springer, pp. 142–153, doi:10.1007/978-3-642-22256-6_14.

[6] John E Hopcroft, Rajeev Motwani & Jeffrey D Ullman (2001): *Introduction to automata theory, languages, and computation*. *ACM SIGACT News* 32(1), pp. 60–65, doi:10.1145/568438.568455.

[7] Eryk Kopczynski (2014): *Complexity of Problems of Commutative Grammars*. *DRAFT* 0(0), p. 0.

[8] Eryk Kopczynski & Anthony Widjaja To (2010): *Parikh Images of Grammars: Complexity and Applications*. In: *LICS*, IEEE Computer Society, pp. 80–89, doi:10.1109/LICS.2010.21.

[9] Martin Müller, Joachim Niehren & Ralf Treinen (1998): *The First-Order Theory of Ordering Constraints over Feature Trees*. In: *13th annual IEEE Symposium on Logic in Computer Sience*, IEEE Comp. Soc. Press, Indianapolis, Indiana, pp. 432–443, doi:10.1109/LICS.1998.705677.

[10] Joachim Niehren & Andreas Podelski (1993): *Feature Automata and Recognizable Sets of Feature Trees*. In Marie-Claude Gaudel & Jean-Pierre Jouannaud, editors: *TAPSOFT: Theory and Practice of Software Development: Joint International Conference CAAP/FASE/TOOLS.*, Lecture Notes in Computer Science 668, Springer Verlag, pp. 356–375, doi:10.1007/3-540-56610-4_76.

[11] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds & Clemente Izurieta (2009): *Comparison of JSON and XML Data Interchange Formats: A Case Study*. In Dunren Che & Dunren Che, editors: *CAINE*, ISCA, pp. 157–162. Available at `http://dblp.uni-trier.de/rec/bibtex/conf/caine/NurseitovPRI09`.

[12] Helmut Seidl, Thomas Schwentick & Anca Muscholl (2003): *Numerical document queries*. In: *Proceedings of the Symposium on Principles Of Database Systems*, pp. 155–166, doi:10.1145/773153.773169.

[13] Helmut Seidl, Thomas Schwentick & Anca Muscholl (2008): *Counting in trees*. In Jörg Flum, Erich Grädel & Thomas Wilke, editors: *Logic and Automata*, Texts in Logic and Games 2, Amsterdam University Press, pp. 575–612.

[14] Helmut Seidl, Thomas Schwentick, Anca Muscholl & Peter Habermehl (2004): *Counting in Trees for Free*. In Josep Díaz, Juhani Karhumäki, Arto Lepistö & Donald Sannella, editors: *ICALP*, Lecture Notes in Computer Science 3142, Springer, pp. 1136–1149, doi:10.1007/978-3-540-27836-8_94.

[15] Gert Smolka (1992): *Feature Constraint Logics for Unification Grammars*. *Journal of Logic Programming* 12, pp. 51–87, doi:10.1016/0743-1066(92)90039-6.

[16] Gert Smolka & Ralf Treinen (1994): *Records for Logic Programming*. *J. Log. Program.* 18(3), pp. 229–258, doi:10.1016/0743-1066(94)90044-2.

[17] J. W. Thatcher & J. B. Wright (1968): *Generalized finite automata with an application to a decision problem of second-order logic*. *Mathematical System Theory* 2, pp. 57–82.

[18] S. Dal Zilio & D. Lugiez (2003): *XML Schema, Tree Logic and Sheaves Automata*. In R. Nieuwenhuis, editor: *Proc. of RTA - Rewriting Techniques and Applications*, Lecture Notes in Computer Science 2706, Springer Verlag, pp. 246–263, doi:10.1007/3-540-44881-0_18.