Counting over Unranked Trees

internship L3 .

Lehrstuhl für Informatik 7, RWTH Aachen, Germany

benoit.groz@dptinfo.ens-cachan.fr

November 25, 2006

Introduction

Context of the Internship This report is relative to the internship focussing on tree automata I made with W.Thomas and W.Karianto in Aachen. Prof. Thomas's interests are generalized automata, automata over infinite objects, links between automata and logics, verification and infinite games. The chair in which this internship took place: "Logic and Theory of Discrete Systems", consists in a little less than twenty members, mostly young researchers(post-docs and Phd students). Amusing is the occasionnal stay in this chair of several former student of the ENS Cachan during my stage.This chair regularly organizes little talks in which members and visitors present thei current researches, which was especially interesting.

Subject of the Internship Unranked trees can be seen as a generalisation of the "first-order terms" trees, in which the number of sons for the vertices is still finite, but unbounded. Such trees appear in many fields like logic and XML-data processing. In this document, we will study various models of automata over unranked trees where counting conditions enter the definition of successful run. Counting conditions are especially relevant for XML-document queries. They have been studied over the past few years in various directions such as (and mainly) Parikh automata and Presburger formulas.

We extend the idea of Presburger counting proposed by Seidl, Schwentick and Muscholl [2], by adding some global constraint over the tree. The aim of this work is to introduce several types of automata with Presburger or Parikh counting conditions, applied either locally, or over the whole tree. We compare their expressive power, and show algorithmic results about the non-emptiness problem.

1 Definitions, Technical Preliminaries

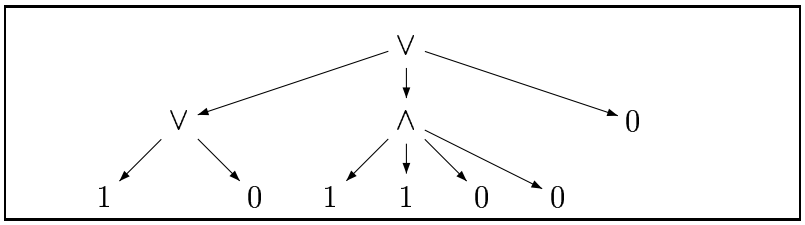
Let us define first the trees and automata we are going to use.

1.1 Standard Definitions

Definition 1 *An* unranked tree t *over the alphabet* Σ*(or* unranked Σ-valued tree*) is a mapping* $t : dom_t \to \Sigma$ *with* dom_t *(the* tree domain *of t) a non-empty, prefix-closed subset of* $(\mathbf{N})^*$ *that has to verify the following property: if* $xi \in dom_t$*, then* $\forall j$ *with* $1 \leq j < i$, $xj \in dom_t$*.*

The set of all regular word languages over the alphabet Σ will be denoted *Reg(*Σ*).*

Exemple 1 *We can, for instance, use unranked trees as models for generalized boolean expressions, i.e. expressions where the arities of* $\vee$ *and* $\wedge$ *are unbounded.*
The tree



(with $\Sigma = \{1, 0, \vee, \wedge\}$*, and* $dom_t = \{\epsilon, 1, 11, 12, 2, 21, 22, 23, 24, 3\}$*) is a tree representation of the expression (in infix-notation)* $(1 \vee 0) \vee (1 \wedge 1 \wedge 0 \wedge 0) \vee 0$

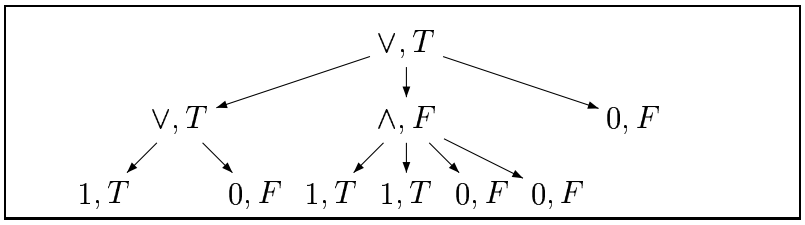
Definition 2 *A* non-deterministic bottom-up tree automaton *(*$\uparrow NTA$*) over unranked* Σ*-valued trees* $\mathscr{A} = (Q, \Sigma, \Delta, F)$ *consists of*
a finite set Q *of states,*
a finite input alphabet Σ
a finite set of transitions $\Delta \subseteq Reg(Q) \times \Sigma \times Q$
and the set of final states $F \subseteq Q$*.*

A run of $\mathscr{A}$ *over a* $\Sigma -$ *valued tree t is a Q-valued tree* ρ *with the same domain than t, such that* $\forall x \in dom_t$ *with an arity n,*
$\exists L, \rho(x1), ...\rho(xn) \in L$ *and* $(L, t(x), \rho(x)) \in \Delta$
$\mathscr{A}$ *accepts t if it has an accepting run over t, that is: with* $\rho(\epsilon) \in F$

Exemple 2 *The following* $\uparrow NTA$ $\mathscr{A}$*, again over* $\Sigma = \{1, 0, \vee, \wedge\}$ *accepts the trees that are models of valid boolean expressions.* $\mathscr{A} = (Q, \Sigma, \Delta, \mathscr{F})$ *where*

- $\Sigma = \{0, 1, \vee, \wedge\}$
- $Q = \{T, F\}$
- $\Delta = \{(\epsilon, 1, T), (\epsilon, 0, F),$
  $(Q^*TQ^*, \vee, T), (F^*, \vee, F), (Q^*FQ^*, \wedge, F), (T^*, \vee, T)\}$
- $\mathscr{F} = \{T\}$

Exemple 3 *The run of this automaton over the above described tree is the following:*



We then define deterministic bottom-up tree automata, in which for a given tree, the run is uniquely determined. Somehow, it is the translation for unranked tree of the idea that only one transition can be applied at a time, but since the number of sons of a vertice is unbounded, we use automata to describe the transitions.

Definition 3 *A* deterministic bottom-up tree automaton *($\uparrow$ DTA) is given by a tuple*
$\mathscr{A} = (Q, \Sigma, \Delta, (D_a)_{a\in\Sigma}, F)$ *with Q, Σ, and F as for $\uparrow$ NTA, and deterministic finite automata D_a with output defining the transitions of $\mathscr{A}$. Each of the D_a (with $a \in \Sigma$) is of the form $D_a = (S_a, Q, s_a^{in}, \delta_a, \lambda_a)$ with a finite set S_a of states, input alphabet Q, initial states s_a^{in}, transition function $\delta_a : S_a \times Q \to S_a$, and output function $\lambda_a : S_a \to Q$. As usual, we define $\delta_a^* : S_a \times Q^* \to S_a$ by $\delta_a^*(s, \varepsilon) = s$ and $\delta_a^*(s, uq) = \delta_a(\delta_a^*(s, u), q)$.*

Let us now define the Parikh automata, that enable us to introduce some counting on symbols for words.

Definition 4 *Let Σ be an alphabet and $n \geq 1$. Further, let D be a finite, nonempty subset of $\mathbf{N^n}$. The* extended Parikh mapping $\Phi : (\Sigma \times D)^* \to D$ *is defined by*
$\Phi(\epsilon) = 0^n$.
$\Phi(a, \overline{d}) = \overline{d}$ *for all* $(a, \overline{d}) \in (\Sigma \times D)$.
$\Phi(uv) = \Phi(u) + \Phi(v)$ *for each* $u, v \in (\Sigma \times D)^*$

("+" being the arithmetical addition over $\mathbf{N^n}$*) and of course we can define the* extended Parikh image *of L by:* $\Phi(L) = \{\Phi(w) | w \in L\}$

We also define the $\Sigma -$ *projection* Ψ *by*

$\Psi(\epsilon) = \epsilon$

$\Psi(a, \overline{d}) = a$ *for all* $(a, \overline{d}) \in (\Sigma \times D)$.

$\Psi(uv) = \Psi(u)\Psi(v)$ *for each* $u, v \in (\Sigma \times D)^*$

Definition 5 *A set is said to be a* semi-linear set *if it is the finite union of some linear set, which are the sets of the following form:*
$\{v_0 + k_1.v_1 + k_2.v_2 + ... + k_n.v_n \mid k_1, ...k_n \in \mathbf{N}\}$ *for some* $v_0, ...v_n \in \mathbf{N^n}$.

The restriction *of L with respect to C(*$C \subseteq \mathbf{N}^n$*) is*
$L_{|C} = \{\Psi(w) | w \in L$ *and* $\Phi(w) \in C\}$

Let us now define Parikh automata over words.

Definition 6 *A* Parikh automaton *of dimension* $n \geq 1$ *over an alphabet* Σ *is a system (*$\mathscr{A}$*,C) where* $\mathscr{A} = (Q, \Sigma \times D, \delta, q_0, F)$ *is a finite automaton over* $\Sigma \times D$*, for some finite nonempty set* $D \subseteq \mathbf{N^n}$ *and* $C \subseteq \mathbf{N^n}$ *is a semi-linear set (called the* constraint set*). It is deterministic if the transition function* δ *satisfies* $\sum_{\overline{d} \in D} |\delta(q, (a, \overline{d}))| \leq 1$

The language recognized by ($\mathscr{A}$,C) is defined as $L(\mathscr{A}, C) = L(\mathscr{A})_{|C}$

Exemple 4 *For instance, the language* $L = \{a^n b^m c^n a^m | n, m \in \mathbf{N}^*\}$ *over* $\Sigma = \{a, b, c\}$ *can be recognized by the Parikh automaton* $(A) = (Q, \Sigma \times D, \Delta, q_0, F)$ *where*

- $Q = \{q_0, q_1, q_2, q_3\}$
- $D = \{(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)\}$
- $\Delta = \{(q_0, (a, (1,0,0,0)), q_0), (q_0, (b, (0,1,0,0)), q_1),$
  $(q_1, (b, (0,1,0,0)), q_1), (q_1, (c, (0,0,1,0)), q_2), (q_2, (c, (0,0,1,0)), q_2),$
  $(q_2, (a, (0,0,0,1)), q_3), (q_3, (a, (0,0,0,1)), q_3)\}$
- $F = \{q_3\}$
- *and with the constraint set* $C = \{k_1.(1,0,1,0) + k_2.(0,1,0,1) | k_1, k_2 \in \mathbf{N}\}$

The figure 1 corresponds to this automaton.

The language of Example 4 is recognized by a Parikh automaton but not definable by a finite automaton with an extra semi-linear constraint.

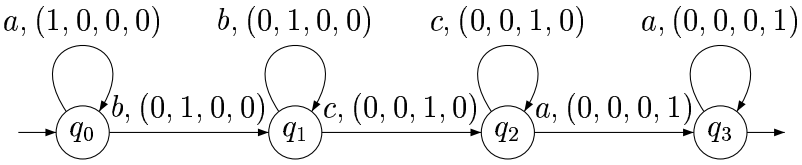


Figure 1: The automaton $\mathcal{A}$.

1.2 Automata over unranked trees that count

We have defined the tree automata over unranked trees and a way to do some counting over words. Let us see now how we can define some counting over unranked trees, which is what in xml-queries one may find useful: for instance, one may whish to select in a database the customers who buy more "classical" CDs than jazz. The customers can be conveniently represented as the root of trees, their sons being the CD bought, and each CD having a son describing the type of its data.

Definition 7 *Presburger arithmetics is the first-order theory(with equality) of the natural numbers with addition. In other words, Presburger formula are the formula built with* $=, +, \vee, \wedge, \exists, \forall$*, the negation, and variables, whose domain is implicitly* $\mathbf{N}$*. Then, we can use natural integers and inequality signs as abbreviations, since: '*$z = 0$*' is equivalent to '*$\exists z_1, z_1 = z_1 + z$*', and '*$x \leq y$*' , to '*$\exists z, x + z = y$*'...*

By a result of Ginsburg and Spanier, we know that Presburger formulas define precisely the semi-linear sets $C \subseteq \mathbf{N}^k$.

Definition 8 *A tree* automaton with Presburger transitions on letters *is as a tuple* $\mathscr{A} = (Q, \Sigma, \Delta, F, \Phi)$*. with*
Φ *: a finite set of Presburger formulas over* Σ*(that means formulas with one variable for each letter of* Σ*:*
$\forall w \in \Sigma = \{a_1, ..., a_n\},\ w \models \phi(x_1, ..., x_n)$ *if* $\phi(|w|_{a_1}, ..., |w|_{a_n})$ *is valid).*
Δ *: the finite set of transitions, with* $\Delta \subseteq Reg(Q) \times \Phi \times \Sigma \times Q$

A *run* of such an automaton over a tree t is a function $\rho : dom_t \to Q$ such that $\forall x \in dom_t$, there is some regular language L and some Presburger formula $\phi \in \Phi$ such that
$\rho(x1)\rho(x2)...\rho(xn) \in L$ and $(L, \phi, t(x), \rho(x)) \in \Delta$ and $t(x1)t(x2)...t(xn) \models \phi$

That way, we have one Presburger formula over the letters for each transition.

Definition 9 *For an* automaton with Presburger transitions on states, *the idea is similar, but the constraint is set upon the states: This time, the condition involving ϕ for the run is replaced by : $\rho(x1)\rho(x2)...\rho(xn) \models \phi$*

We can also define those counting modes for deterministic bottom-up automata ($\uparrow$ *D tree automata*) in the following way: we take usual $\uparrow$ *D tree automata* over unranked trees, and add the constraints in the same way; by considering it as a non-deterministic automaton; we ask for an accepting run that $\forall x \in dom_t$, we have $t(x1)t(x2)...t(xn) \models \phi$ (resp. $\rho(x1)\rho(x2)...\rho(xn) \models \phi$)...

This is the model described by Seidl, Muscholl and Schwentick [2]. We can extend it in the following ways:

Definition 10 *A* tree automaton with global counting *is an automaton as described above, but we add a formula ψ over the whole tree:*
$\mathscr{A} = (Q, \Sigma, \Delta, F, \Phi, \psi)$.

and in a run ψ has to be satisfied by the tree t,
i.e. $\psi(|t|_{q1}, |t|_{q2}, ..., |t|_{qn})$ holds (resp. $\psi(|t|_{a1}, |t|_{a2}, ..., |t|_{an})$ if we count over symbols)

Exemple 5 *For instance, the language*
$L = \{$ *tree t over* $\Sigma = \{a, b\}|$ *for all vertices x in the tree t,*
x has more sons labeled with 'a' than with 'b' ,
and the total number of 'a' labels in t is twice the number of 'b' labels$\}$

is accepted by the following automaton with global counting on letters:
$\mathscr{A} = (Q, \Sigma, \Delta, F, \Phi, \psi)$ *where*

- $Q = \{q\}$
- $\Phi = \{\phi(n_a, n_b) = "n_a > n_b"\}$
- $\Delta = \{q^*, \phi, x, q\}$
- $F = \{q\}$
- $\psi(n_a, n_b) = "n_a = 2.n_b"$

Definition 11 *For a* tree automaton with Presburger Frontier-check, *the conditions are exactly the same as in Def.10, except that the formula ψ must be satisfied by the frontier instead of the whole tree.*

1.3 Another model: Automata with Parikh counting

In the direct application of "Presburger counting", the arithmetical constraint is applied to the occurrence numbers of letters in a word. A more powerful counting procedure is realized by the "Parikh automata" of Klaedtke and Ruess[5]. Such an automaton processes a word like a finite automaton, and-for some k- updates in each transition a vector from $\mathbf{N}^k$ by adding a vector from $\mathbf{N}^k$. Thus the update depends on the current state of the automaton. Presburger counting is a special case of this, using $k = |\Sigma|$ and update steps independent of the state(namely, adding (0,...,0,1,0,...,0), with the "1" in i-th position when the i-th letter of Σ is processed).

Definition 12 *An* automaton with Parikh transitions on states *can be defined as a tuple* $\mathscr{A} = (Q, \Sigma, \Delta, F, \Phi)$. *with*

Φ *: a finite set of Parikh automata over* $Q \times D$ *for some* D.

Δ *: the finite set of transitions, with* $\Delta \subseteq \Phi \times \Sigma \times Q$

A *run* of such an automaton over a tree t is a function $\rho : dom_t \to Q$ such that $\forall x \in dom_t$, there is some Parikh automaton $\mathscr{A} \in \Phi$ over $Q \times D$ for some D such that $(\mathscr{A}, t(x), \rho(x)) \in \Delta$ and there are some $d_1, d_2, ..., d_n \in D$ such that
$\mathscr{A}$ *accepts* $(\rho(x1), d_1)(\rho(x2), d_2)...(\rho(xn), d_n)$

That way, we have one Parikh automaton over the states for each transition.

Definition 13 *For an* automaton with Parikh transitions on letters, *the idea is similar, but the constraint is set upon the letters: This time,*

- *the Parikh automata are over* Σ.
- *the condition for the run is replaced by :*
  $\mathscr{A}$ *accepts* $(t(x1), d_1)(t(x2), d_2)...(t(xn), d_n)$

For $\uparrow$ *Deterministic automata with Parikh counting*, we use the usual definition of $\uparrow$ *Deterministic automata over unranked trees* , with Parikh automata instead of the usual automata in the transitions.

2 Comparing Expressiveness

2.1 Comparison between Parikh automata and Presburger counting

It is known that there are word languages accepted by Parikh automata but not definable by finite automata and Presburger constraints on the occurrence numbers of the letters (see Example 4 above). We show that this difference is irrelevant when comparing Presburger counting and Parikh automaton counting in the transitions of tree automata.

First, it is quite obvious that Presburger automata over unranked trees can be simulated by automata over unranked trees with Parikh transitions.

Proposition 1 *Automata over unranked trees with Parikh transitions can be simulated by Tree Automata with Presburger over unranked trees with Presburger transitions.*

Proof: The main idea is to simulate the vectors of the Parikh automaton into the states of the automaton with Presburger constraints: if, in the Parikh automaton, we have a transition leading to the state q, using the vector d_i, we say that this transition will lead to the state (q, d_i) in the automaton with Presburger constraint.

In this proof, $\overline{\mathscr{A}}$, $\overline{\mathscr{B}}$... will denote tree automata, while $\mathscr{A}$,$\mathscr{A}'$... will denote word automata used in the transitions. Let $\overline{\mathscr{A}}$ be an automaton over unranked trees with Parikh transitions: $\overline{\mathscr{A}} = (Q, \Sigma, \overline{\Delta}, F, \Phi)$.
We define a Presburger automaton $\overline{\mathscr{B}}$ over unranked trees as follows:

1. $\overline{\mathscr{B}} = (Q_B, \Sigma, \overline{\Delta}_B, F_B, \Psi)$
2. $\forall(\mathscr{A}, \phi, a, q) \in \overline{\Delta}$, *with* $\mathscr{A} = (S_{\mathscr{A}}, Q \times D, \delta_{\mathscr{A}}, i, F)$, $(\mathscr{A}, \phi)$ being a Parikh automaton over $Q \times D$,
   we define $\mathscr{A}'$ by $\mathscr{A}' = (S_{\mathscr{A}} \times D, Q \times D, \delta_{\mathscr{A}'}, i', F')$ with
   (a) $i' = i \times \{0\}$
   (b) $F' = F \times D$
   (c) for all $(\alpha \xrightarrow{(q_i,d_j)} \beta) \in \delta_{\mathscr{A}}$, we add, for all $d_k \in D$, the following rule: $((\alpha, d_k) \xrightarrow{(q_i,d_j)} (\beta, d_j))$ to $\delta_{\mathscr{A}'}$.

   ¿From the formula ϕ of the Parikh automaton $\mathscr{A}$, we define a formula $\psi(x_1, x_2, ..., x_n) = \phi(d_{11}x_1 + d_{21}x_2 + ... + d_{n1}x_n, ..., d_{1m}x_1 + d_{2m}x_2 + ... + d_{nm}x_n)$, where $D = \{d_1, d_2, ..., d_n\}$ and $d_i = (d_{i1}, d_{i2}, ..., d_{im})$.

Then we define the formula ψ' over the states of $\overline{\mathscr{B}}$ such that $\psi'((q_i, d_j)_{i,j}) = \psi(|q_1, d_1| + |q_2, d_1| + ... + |q_n, d_1|, |q_1, d_2| + |q_2, d_2| + ..., ...)$
And we add, for all d_k, $(\mathscr{L}(\mathscr{A}'), \psi', a, (q, d_k))$ to Δ_B. Or quite so, because in fact, there are other transitions in $\overline{\Delta}$ to be considered, with -a priori- different sets D of vectors. So that $\mathscr{A}'$ *and* ψ have to be "extended" to those additional vectors. They should not be relevant in the run of $\mathscr{A}'$, so we can impose the values of the vectors to be 0 over those dimensions corresponding to the former"counting vectors", except that in the formula giving Δ_B, "all the d_k" really means all of them, so there the other components have to span over the whole sets of vectors, in order to enable the next transition to "choose/guess" the input vectors it will need.

3. We define Q_B as $Q \times$ (*the cartesian product of the D that appear in the Parik automata of* $\overline{\Delta}$)

4. $F_B = F \times$ (*the cartesian product of the D that appear in the Parikh automata of* $\overline{\Delta}$)

Lemma 1 *The language accepted by the tree automaton* $\overline{\mathscr{A}}$ *with Parikh transitions is the same as the language accepted by the tree automaton* $\overline{\mathscr{B}}$ *with Presburger transitions.*

Proof:
a) Let $t \in \mathscr{L}(\overline{\mathscr{A}})$, let us prove that $t \in \mathscr{L}(\overline{\mathscr{B}})$.
For all transition $(q_1, ...q_n, a, q)$ in the run of $\overline{\mathscr{A}}$, there is some $(\mathscr{A}, \phi, q) \in \Delta$ such that $\exists (d_1, ...d_n), (q_1, d_1), ...(q_n, d_n) \in L(\mathscr{A})_{|\phi}$.
Then, there are some states $\alpha_1, \alpha_2, ...\alpha_{n+1}$ such that in $\mathscr{A}$ we have:
$\alpha_1 \xrightarrow{q_1,d_1} \alpha_2 \xrightarrow{q_2,d_2} ... \xrightarrow{q_n,d_n} \alpha_{n+1}$, with $\alpha_n \in F$, *and* $\alpha_1 \in i$
Therefore, in $\mathscr{A}'$; $(\alpha_1, 0) \in i'$ and we can have a run
$(\alpha_1, 0) \xrightarrow{q_1,d_1} (\alpha_2, d_1) \xrightarrow{q_2,d_2} ... \xrightarrow{q_n,d_n} (\alpha_{n+1}, d_n)$,
and $(\alpha_n, d_n) \in F$ so this run is an accepting one. Then, the Presburger constraint is satisfied by the states since the Parikh constraints was satisfied by the vectors.
Therefore, we can have, for any d we wish, a transition
$((q_1, d_1), (q_2, d_2)...(q_n, d_n), a, (q, d))$ in $\overline{\Delta_B}$.
By induction on the size of the tree, this proves that t is accepted by $\overline{\mathscr{B}}$.
b) Let $t \in \mathscr{L}(\overline{\mathscr{B}})$, let us prove that $t \in \mathscr{L}(\overline{\mathscr{A}})$
For all transition $((q_1, d_1), (q_2, d_2)...(q_n, d_n), a, (q, d))$ in the run of $\overline{\mathscr{B}}$,
there is some word automaton$\mathscr{A}'$ and some Presburger formula ϕ such that
$(\mathscr{L}(\mathscr{A}'), \phi, a, (q, d)) \in \overline{\Delta_B}$ and $\mathscr{A}' accepts (q_1, d_1), (q_2, d_2)...(q_n, d_n)$,
and $(q_1, d_1), (q_2, d_2)...(q_n, d_n) \models \phi$

So, there are some $\alpha_1, ...\alpha_n$ such that in $\mathscr{A}'$:
$(\alpha_1, 0) \xrightarrow{q_1,d_1} (\alpha_2, d_1) \xrightarrow{q_2,d_2} ... \xrightarrow{q_n,d_n} (\alpha_{n+1}, d_n)$,
Therefore, in $\mathscr{A}$ (the automaton used to define $\mathscr{A}'$),
$\alpha_1 \xrightarrow{q_1,d_1} \alpha_2 \xrightarrow{q_2,d_2} ... \xrightarrow{q_n,d_n} \alpha_{n+1}$,
Furthermore, the Parikh constraint is satisfied since ϕ is satisfied in the run of $\mathscr{L}(\overline{\mathscr{B}})$.
Hence $t \in \mathscr{L}(\overline{\mathscr{A}})$, by induction over the size of the tree. Q.E.D

¿From this lemma, the claim is self-evident.

Therefore, both counting modes are equivalent over trees.

2.2 Local vs Frontier vs Global counting

Let us now study the expressivity of the different models proposed for Presburger counting.

Proposition 2 *Automata counting in transitions are strictly less expressive than automata counting in transition with an additional Presburger counting over the frontier.*

Proof: a) The inclusion claim is obvious.
b) To separate the two models, consider the language L of all the trees that have as many 'a's as 'b's in the leaves. It is easy to see that this language is recognized by an automata that counts over the frontier.

Let us assume that it is recognized by an automaton $\mathscr{A}$ that counts only in the transitions. Then we obtain a contradiction as follows. let us consider an infinite set of trees t_i such that the number of a-leaves of t_i is the number of its b-leaves plus i. Since $\mathscr{A}$ has only a finite number of states, there are two integers i_1 and i_2 such that the (possible) states at the root in a run of A over t_{i_1} are the same that those in a run over t_{i_2}. Hence, since $\mathscr{A}$ must accept the tree formed by some root-vertex whose two sons are t_{i_1} and t_{-i_1}, it has to accept the tree formed by some root whose two sons are t_{i_2} and t_{-i_1}, which proves that $\mathscr{A}$ can't recognize L. Q.E.D

Proposition 3 *Automata counting over frontier are strictly less expressive than automata counting over the whole tree.*

Proof: a) Automata counting over frontier can be simulated by automata counting over the whole tree as follows. We create new states q_i' that are "copies" of the q_i. For all transition (L,ϕ,a,q_i) $\in \Delta$,

if $\epsilon \in L$, then we replace the transition(s) (L,ϕ,a,q_i) by ($L - \epsilon$,a, q_i) and we add the transition (ϵ, ϕ,a, q_i') to Δ. Then we replace in all the

Presburger formula of the transitions q_i by $(q_i + q_i')$(ϕ is thereby generalised to 2n variables) .

Finally, if we had a frontier-check formula $\phi(|q_1|, |q_2|, ..., |q_n|)$, we replace it by the global-constraint formula $\phi(|q_1'|, |q_2'|, ..., |q_n'|)$ The new automaton thus defined recognizes the same language as the initial one.

b) To separate the two models, consider the language of all trees-let us even restrain to all unary trees- over $\Sigma = \{a, b\}$ that have as many "a"s than "b"s in the internal labels. This language is clearly not recognizable by frontier-check, while recognizable by global counting.

We therefore have (also) proved that counting in transitions is strictly less expressive than "counting over transitions + over whole tree".

In a nutshell: for the expressiveness of counting over different domains in trees we have:

$transitions < transition + frontier < transition + wholetree$

The next proposition is interesting, though:

Proposition 4 *When considering unranked proper trees(that is; with no unary internal vertex), countings over the frontier and over the whole tree have the same expressive power.*

Proof: a)The first part of the above proof still holds. b)However, we can't distinguish between the models. On the contrary, we can code the internal vertices on the frontier with the following injection: given some internal vertex, we move "one step right" to its rightmost son, then move "left" unto this son's leftmost heir: we put in this leaf the information over the state of the node.

More formally; let $\mathscr{A} = (Q, \Sigma, \Delta, F, \Phi, \psi)$ be an automaton with Presburger counting over the whole tree. We define an automaton
$\mathscr{A}' = (Q', \Sigma, \Delta', F', \Phi', \psi')$ with Presburger counting over the frontier as follows:

- $Q' = Q \times Q$
- Even if it means spliting transitions in two, we can suppose that all the languages in the transitions which are not of the form $\epsilon, \phi, a, q)$ don't contain ϵ, and therefore that their words are at least two letters long.

  For each such L with $(L, \phi, a, q) \in \Delta$, $\forall q_i \in Q$, let L_i be the regular language over $Q \times Q$ such that $L_i \in (Q \times q_i) \times (Q \times 0)^* \times (Q \times q)$ and the projection of L_i over its first component is L. (0 is a kind of "sink state: it is won't be taken into account at hte time of the counting)

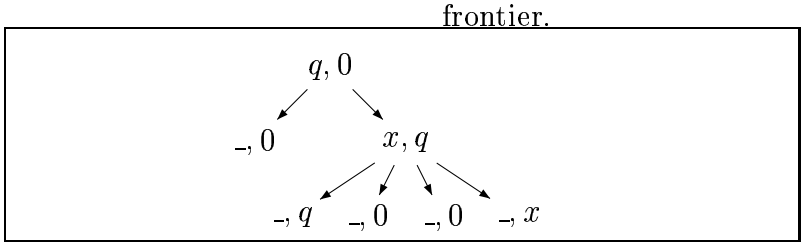
Then, for all i, we take $(L_i, \phi', a, (q, q_i))$ for Δ', where ϕ' is the formula that counts only over the first components, as ϕ would do)

We then add to Δ', for each $\epsilon, \phi, a, q) \in \Delta$, for all i,the transition $(\epsilon, \phi', a, (q, q_i))$.

- $\Psi'((n_i, n_j)) = \Phi((n_1, n_1) + (n_1, n_2) + ...(n_1, n_n) \ + \ (n_1, n_1) + (n_2, n_1) + ...(n_n, n_1), ..., (n_n, n_1) + ... + (n_n, n_n))$
- $F' = F \times \{0\}$

It is easy to see that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$

The figure next represents the encoding of the internal vertices on the frontier.



Proposition 5 *Automata counting in transitions over symbols are strictly less expressive than those counting over states.*

Proof: a) We can simulate the symbols with the states in the following way: Given an automaton counting over symbols, we define a new automaton by:
$Q' = Q \times \Sigma$
For all transitions (L,a,q_i) $\in \Delta$, if $\epsilon \in L$, then we replace the transition(s) (L,a,q_i) by($L - \epsilon$,a, q_i) in Δ and we add the new transition (ϵ,a, (q_i,a)) to Δ'.

For all transitions $(L, a, q_i) \in \Delta$ (which no longer deals with leaves; only with internal vertices), we add all the transitions $(L', a, (q_i, a))$ to Δ', where L' is one of the language obtained from L by subtituting some (q_i, a_j) for q_i. There is only a finite number of such L'.

Then we replace the symbol-counting formula $\phi(a_1, a_2, ..., a_n)$ by the state-counting formula
$\phi'((q_1, a_1), (q_1, a_2), ...(q_1, a_m), (q_2, a_1), (q_2, a_2), ...(q_n, a_m))$
$= \phi((q_1, a_1) + (q_2, a_1) + ... + (q_n, a_1), (q_1, a_2) + (q_2, a_2)..., ...)$

The new automaton thus devised recognizes the same tree-language as the initial one.

b) These two countings aren't equivalent. Let us consider the tree language formed by all trees over $\Sigma = \{a\}$ such that you have more often two sons than three sons. Counting over symbols won't help very much! But if one has a

state q1 for the vertices having one son or more than four, q2 for those that have two sons, q3 if they have three sons, then one can just ask $|q3| \leq |q2|$. This justifies the claim.

3 Algorithmic results: Non emptiness

At last, we study the decision problems regarding our models of automata, namely: membership, emptiness and universality.

The **membership** problem:

Given a tree t and an automaton $\mathscr{A}$, is t in $L(\mathscr{A})$?

can be solved by an exhaustive search of the transitions that would give an accepting run.

The **emptiness** problem:

given an automaton $\mathscr{A}$, is $L(\mathscr{A})$ empty?

will be studied in this paragraph.

The **universality** problem:

given an automaton $\mathscr{A}$ over Σ, do all the trees over Σ belong to $L(\mathscr{A})$?

is not decidable for automata with Presburger constraints in the transitions [2]. So of course it isn't for the two other models either.

Proposition 6 *The* emptiness problem *is decidable for all the models described above: counting over the whole tree, over the frontier, or only in the transitions.*

We just need to prove it for counting over the whole tree, since the other models can be reduced to it.

Definition 14 *An* Extended Context-Free Grammar (ECFG) *is a tuple $G = (N, \Sigma, P, S)$ with*

- *N : nonterminals*
- *Σ : terminals, such that $N \cap \Sigma$ is empty*
- *$S \in \Sigma$ start symbol*

- *P set of rules of the form $N \to \alpha$ where α is a regular expression over $(N \cup \Sigma)$*

Proposition 7 *If G is an ECFG, then $L(G)$ is context-free.*

Proof: We construct a CFG G' from G as follows: for all $N \to \alpha$ *in* P, α (being a regular set) is context-free(this result can be proved by induction over α). So there is a CFG $G_\alpha = (N_\alpha, \Sigma_\alpha, P_\alpha, S_\alpha)$ that generates α. Now, we take for G' all the rules in P_α and $N \to S_\alpha$.
G' generates $L(G)$.

Definition 15 *A word $w = (q_0 q_1 ... q_n) \in (\Sigma \cup N)^*$ can be $\sim$ −*derived *into $w' \in (\Sigma \cup N)^*$ iff there is some $i \in [0..n]$ such that $q_i \to \gamma$ and some $x \in \gamma$ such that $w' = q_0 ... q_{i-1} x q_{i+1} ... q_n$*
We then write $w \sim\!\!\to w'$.
The derivation relation of G is $S \sim\!\!\to^* w$, where $\sim\!\!\to^*$ is the reflexive and transitive closure of $S \sim\!\!\to$.
$L(G) = \{w \in \Sigma^* | S \sim\!\!\longrightarrow^* w\}$

Let us prove now that the emptiness problem is decidable for automata counting over the whole tree.

Let f be the Parikh mapping over $Q \times Q'$, that is, given $w \in (Q \times Q')^*$, $f(w) = (|w|_{(q_1,q'_1)}, |w|_{(q_1,q'_2)}, ..., |w|_{(q_n,q'_n)})$.

For each regular language L and Presburger formula ϕ over a given alphabet, $f(L)$ is semi-linear. Therefore, by closure under intersection and union, $f(L) \cap L_\phi$ is also effectively semi-linear, which means they are definable by regular expressions. We will note $\alpha_{f(L)\cap L_\phi}$ one such expression, arbitrarily chosen.

Proof: Given some automaton $\mathscr{A} = (Q, \Sigma, \Delta, i, F, \Phi, \psi)$. Let us define an ECFG $G = (N, \Sigma, P, S)$ as follows:

- N=Q
- $\Sigma = Q'$, *where Q' is a "copy" from Q : $Q' = \{q'_i | q_i \in Q\}$*
- for each $q \in F$, we put a rule $S \longrightarrow q$ in P.
- for all $q \in Q$, we add the following rule in P :

$$q \longrightarrow \bigcup_{(L,\phi,a,q)\in\Delta} q' \alpha_{f(L)\cap L_\phi} \qquad (1)$$

G being an ECFG, it follows that $L(G)$ is context-free. Then, Parikh's theorem states that $f(L(G))$ is effectively semi-linear.
We can then decide wether $f(L(G)) \cap L_\psi$(ψ *being checked over the* (q'_i)!) is empty or not.

Lemma 2 *$(f(L(G)) \cap L_\psi)$ is empty iff $\mathscr{L}(\mathscr{A})$ is.*

Proof: a)If $\mathscr{A}$ accepts some t in a run ρ, then we can build the following derivation in G: if we consider the run in a "top down" order, each time that a transition $(\rho(x1)\rho(x2)...\rho(xn), \phi, t(x), \rho(x))$ is encountered, there exists some permutation σ such that $\rho(x\sigma(1))\rho(x\sigma(2))...\rho(x\sigma(n)) \in \alpha_{f(L)\cap\phi}$ for the L corresponding to $\rho(x1)\rho(x2)...\rho(xn)$.
We can use the derivation: $\rho(x) \longrightarrow \rho(x)'\rho(x\sigma(1))\rho(x\sigma(2))...\rho(x\sigma(n))$ to simulate the states of the tree in prefix-order. As a matter of fact, it doesn't really give the states of the tree in prefix-order because of the permutation, but it would if σ was the identity. The last non terminal symbols will disappear when simulating the leaves, since transitions of the form $(\epsilon, \phi, t(x), \rho(x))$ will then be encountered, which leads to a simulation by rules of the form: $\rho(x) \longrightarrow \rho(x)'$.

That way, we have some $w \in L(G)$ that satisfies ψ(over the (q'_i))

b) Reciprocally, if there is some w in $f(L(G)) \cap L_\psi$, w being obtained by a derivation :
$S \longrightarrow q_{i_0} \longrightarrow q'_{i_0}xq_{i_1}y \longrightarrow q'_{i_0}xq'_{i_1}zy \longrightarrow ... \longrightarrow w$, then we can find a tree in $\mathscr{L}(\mathscr{A})$ from that derivation in the following way:
We first put q_{i_0} at the root of the tree.
Then we look for the next step of the derivation. since we have a $q_{i_0} \longrightarrow q'_{i_0}xq_{i_1}y$ rule in G, there exists some word u over Q, some letter a in the tree-alphabet, and some formula ϕ such that $(u, \phi, a, q_{i_0}) \in \Delta$ and $f(u) = f(xq_{i_1}y)$.
We replace q_{i_0} with $'a'$ at the root of the tree, and add some sons to a so that they form the word u. We go on running over the derivation from the left to the right, doing the same, that is;
whenever a rule $q_{i_k} \longrightarrow q'_{i_k}sq_{i(k+1)}t$ corresponding to some $(v, \phi, b, q_{i_k}) \in \Delta$ is applied, we replace one of the q_{i_k} (no matter which) with $'b'$, and give him for sons the states of v, in the same order as in v.
When this will have been completed, we will dipose of a tree that is accepted by $\mathscr{A}$, since the states that have been replaced by letters simulate the state in an accepting run over this tree. Q.E.D.

Perspectives and Acknowledgement So, this stage made me discover and define a new kind of tree automata. I would like to thank especially W.Thomas for his kindness and his good advices, as well as W. Warianto

for his great availability, his precise explanations, and his correcting all my definitions and proofs. Many thanks also to all the members of the chair that helped me tackle software and technical difficulties...and made this internship pleasant.

References

[1] J.Cristau, C.Löding, W.Thomas *Deterministic Automata on Unranked Trees*Proc.15th Symp on Foundations of computational Theory, FCT2005, Springer LNCS 3921 (2005).

[2] H.Seidl, A.Muscholl, T.Schwentick *Numerical Document Queries.*Proceedings PODS 2003.ACM Press(2003) 681-696

[3] W.Karianto *Parikh automata with pushdown stack.* Diploma thesis, RWTH Aachen,Germany(2004).

[4] F.Klaedtke *Automata-based decision procedures for weak arithmetics*Phd Thesis, Univ. Freiburg 2004.

[5] F.Klaedtke, H.Ruess *Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints.* Technical Report 177, Institute of Computer Science at Freiburg University, 2002.

[6] E.Jurvanen, A.Potthoff, W.Thomas *Tree Languages Recognizable by Regular Frontier Check.*Bericht Nr.9311, juni 1993, Christian-Albrechts-Universität Kiel.

[7] R.J.Parikh *On Context-Free Languages*, Journal of the ACM, 13(1966) pp.570-581.