Counting and Equality Constraints for Multitree Automata

Denis Lugiez

Lab. d'Informatique Fondamentale, UMR 6166
CNRS & Université de Provence.
CMI 39 av. Joliot Curie, 13453 Marseille Cedex, France.
lugiez@cmi.univ-mrs.fr

Abstract. Multitree are unranked, unordered trees and occur in many Computer Science applications like rewriting and logic, knowledge representation, XML queries, typing for concurrent systems, cryptographic protocols.... We define constrained multitree automata which accept sets of multitrees where the constraints are expressed in a first-order theory of multisets with counting formulae which is very expressive and decidable. The resulting class of multitree automata is closed under boolean combination, has a decidable emptiness problem and we show that this class strictly embeds all previous classes of similar devices which have been defined for a whole variety of applications.

Introduction

Tree automata and regular tree languages have been used successfully in many areas of Computer Science like type systems, rewriting, program analysis, protocol verification... However the expressive power of regular languages is often too weak and many extensions of regular languages have been proposed for solving some specific problems. One trend is to add constraints to the transition rules such that a rule is used only when the constraint is satisfied. The most natural extension is to add equality constraints which state that some subterms at some positions must be equal or different. The resulting class has good closure properties but the emptiness problem (decide if the language $\mathcal{L}(\mathcal{A})$ accepted by an automaton $\mathcal{A}$ is empty or not?) is undecidable. Therefore equality constraints have been restricted to get classes with a decidable emptiness problem: automata with equality/disequality constraints between brothers [BT92], reduction automata [CCC$^+$94], and automata with a bounded number of equality test along an accepting run [CJ94]. These automata are used in automated theorem proving and rewriting. Another direction for extending the expressivity of regular languages is to use axioms, especially associativity and associativity-commutativity axioms. Hedge automata [PQ68, Mur01] which are used in query languages for XML have rules where the left-hand side is a regular expression on the set of states. Automata where the left-hand side is a Presburger formula or a rational expression of vectors of integers have be proposed for applications in type system used in verification of infinite-state systems [Col02], inductive theorem proving [LM94] or knowledge representation [NP93] for which feature tree automata are introduced. In [Ohs01], one adds equality steps involving axioms during the acceptance process.

The next step is to combine both constraints and axioms which gives two possible reasons for getting into trouble. Since the most frequently used axiom is associativity-commutativity and the most frequently needed constraints are equality constraints, it is natural to look for tree automata combining these features. The underlying data structure is now the multitree structure where some operators have unbounded arity and the subterms are unordered. Therefore the first point is to define a theory of constraints which extends equality to multitrees and possibly adds some new constraints. The second point is to define a class of multitree automata which uses the constraints and still has the good properties required by applications (basically closure under boolean combination and decision of emptiness). We present a new class of multitree automata with a simple and natural definition, where the constraints are formulae of the first-order theory of equality for multisets enriched by Presburger constraints on the cardinality of multisets. For instance, one may constraint a rule $f(q, q, q) \to q'$ by a formula saying that the first subterm reaching q has twice as elements as the second subterm reaching q and that the first subterm is the union of the second subterm and of the third one. We show that these constraints are decidable (section 2), then we prove that the multitree automata class is closed under boolean combinations (section 4), and has an elementary emptiness problem in section 5. Then we prove that this class contains all known classes of tree automata which use AC axioms or/and equality constraints, are closed under boolean combinations and have a decidable emptiness problem (in section 6). Missing proofs will appear in the long version of the paper.

In this paper we focus on the definition of the new class and its basic properties. Since this class contains many classes previously known which have been extensively used in Computer Science, we can hint at many applications already known in knowledge representation (feature tree automata and feature logic), typing for infinite state systems (rational tree automata), inductive theorem proving (automata with Presburger constraint), logic and rewriting(automata with equality constraint between brothers)... Applications to cryptographic protocols in the spirit of [GLV02] which uses two-way automata with associativity-commutativity is also relevant.

1 Notations

Terms and Multitrees. Multisets on a (finite or infinite) set of elements $e_1, e_2, \ldots$ are sets where elements can be repeated. The empty multiset is denoted by $\emptyset$. The multiset composed of $e_{i_1}, \ldots, e_{i_p}$ (where one may have $e_{i_l} = e_{i_j}$) is denoted by $\{e_{i_1}, \ldots, e_{i_p}\}$. The number of repetition of an element is its multiplicity. The number of elements of a multiset $\#_E(M)$ is the number of elements counted with their multiplicities and $\#_D(M)$ denotes the number of distinct elements. For instance $\#_D(\{e_1, e_1, e_2\}) = 2$ and $\#_E(\{e_1, e_1, e_2\}) = 3$.

We consider terms on a finite set of free function symbols F and a finite set of binary function symbols $\oplus_1, \oplus_2, \ldots$ which are supposed to be associative-commutative (AC in short). For simplicity we use only one $\oplus$ operator, but our results are extended easily to the case of several AC symbols. The set of terms is $T_{\mathcal{F}}$ for $\mathcal{F} = F \cup \{\oplus\}$. Terms can be flattened using the rules $(x \oplus y) \oplus z) \rightarrow$ and writting a term $((\ldots(t_1 \oplus t_2) \oplus \ldots) \oplus t_n)$ as $t_1 \oplus t_2 \ldots \oplus t_n$. *Multitrees* corresponds to flattened terms but where the $\oplus$ operator is considered as a multiset constructor and are described by the grammar:

$$\begin{array}{lll}
\mathcal{MT} ::= \mathcal{S} \mid \mathcal{T} & & \\
\mathcal{S} \quad ::= \mathcal{T}_1 \oplus \ldots \oplus \mathcal{T}_n & n \geq 1 & (\textit{sum–like multitrees}) \\
\mathcal{T} \quad ::= f(\mathcal{MT}_1, \ldots, \mathcal{MT}_n) & arity(f) = n & (\textit{term–like multitrees})
\end{array}$$

For instance $f(a \oplus g(b), a \oplus a) \in \mathcal{T}$, $a \oplus a \oplus f(a, a) \in \mathcal{S}$. In the following we may say terms as well as multitrees for elements of $\mathcal{MT}$. A multitree $t_1 \oplus \ldots \oplus t_n$ is often denoted by $\Sigma_{i=1,\ldots,n} t_i$. Multitrees are equal up to permutation of arguments of $\oplus$. For instance, $f(a \oplus g(b), a \oplus b) = f(g(b) \oplus a, b \oplus a)$.

This paper deals with automata recognizing sets of multitrees.

Presburger Arithmetic. Let $\mathbb{N}$ be the set of natural numbers and let $+$ denote addition of natural numbers. Then the first-order theory of equality on this structure is called Presburger arithmetic and is decidable [1]. Diophantine equations, inequations are example of Presburger arithmetic formula (with a lower complexity since they are in NP). The models of Presburger arithmetic formulae are the *semilinear sets.*

2 First-Order Theory of Multisets with Cardinality Constraints

We define $FO_{\#}(\mathcal{M})$ the first-order theory of multisets with cardinality constraints.

The syntax of formula. Let $\mathcal{X} = \{X, Y, \ldots\}$ be a set of multiset variables, the set of terms is defined by the grammar:

$$T ::= X \mid T \oplus T$$

where the $\oplus$ operator is a binary associative-commutative symbol. We use also two unary symbols $\#_D$ and $\#_E$. The predicate is the equality predicate $=$. We also assume that $N_1, N_2, \ldots$ is a denumerable set of integer variables. Formula are given according to the grammar:

$$\phi ::= \ (T = T) \mid \psi(\#(X_1), \ldots, \#(X_n), N_1, \ldots, N_p) \mid \neg\phi \mid \phi \wedge \phi \mid \exists X\ \phi \mid \exists N\ \phi$$

where ψ is a Presburger arithmetic formula, $\#$ denotes $\#_D$ or $\#_E$.

[1] it is ATIME(double-expo,poly)-complete [Ber77]

Semantics. Let $\mathcal{M}$ be the set of finite multisets built on a denumerable set of distinct elements $e_1, e_2, \ldots$. An interpretation I associates to each variable X a multiset $I(X) \in \mathcal{M}$, and to each integer variable N some natural number $I(N) \in \mathbb{N}$. The function $\oplus$ is interpreted as the union of multisets, equality is equality of multisets, and $\#_D(X)$ (resp. $\#_E$) is interpreted as the number of distinct elements (resp. number of elements) of X. The interpretation is extended to formula as in first-order logic, and similarly we define satisfiability, models,... The values of the e_i's are not relevant for the satisfiability of a formula and satisfiability is preserved by one-to-one mapping of the e_i's.

Expressivity. This logic can express many natural properties.

- any Presburger formula related to the number of elements of multisets (counting formula). For instance X has as many elements as Y: $\#_E(X) = \#_E(Y)$.
- X is empty: $\#_E(X) = 0$, X is a singleton: $\#_E(X) = 1$
- Y is a subset of X: $\exists Z : X = Y \oplus Z$
- the intersection $X \cap Y$ is empty: $\forall Z : Z \subseteq X \wedge Z \subseteq Y \Rightarrow \mathit{empty}(Z)$
- X is a multiset containing only copies of some element: $\#_D(X) = 1$
- X is a multiset s.t. the multiplicity n_e of each element e satisfies the Presburger formula $\psi(n_e)$:

  $$\forall X_e\ Y\ X = X_e \oplus Y \wedge \#_D(X_e) = 1 \wedge X_e \cap Y = \emptyset \Rightarrow \psi(\#_E(X_e))$$

  This formula is called $\mathit{Mult}(\psi, X)$. This is extended for a tuple $X_1, \ldots, X_n$ and a variable ψ with n free variables, yielding a formula $\mathit{Mult}(\psi, X_1, \ldots, X_n)$.
- N is the maximal multiplicity of an element of X:

  $$\begin{aligned}&\forall X_e, Y\ X = X_e \oplus Y \wedge \#_D(X_e) = 1 \wedge X_e \cap Y = \emptyset \Rightarrow \#_E(X_e) \leq N\\ &\wedge \exists X_e, Y\ X = X_e \oplus Y \wedge \#_D(X_e) = 1 \wedge X_e \cap Y = \emptyset \wedge \#_E(X_e) = N\end{aligned}$$

  We shall abbreviate this formula into $N = \#_M(X)$.
- Y is the set of distinct elements of X:

  $$\begin{aligned}Y \subseteq X \wedge \forall X_e, X'\ (X = X_e \oplus X' \wedge X_e \neq \emptyset \Rightarrow Y \cap X_e \neq \emptyset)\\ \wedge \forall Y_e, Y'\ (Y = Y_e \oplus Y' \Rightarrow Y_e \cap Y' = \emptyset)\end{aligned}$$

  then $\#_D(X) = \#_E(Y)$ which shows that $\#_D$ is definable within the logic.

The extension of $FO_{\#}(\mathcal{M})$ with variables $x, y, \ldots$ for elements and the membership predicate $x \in X$ is achieved by introducing a multiset variable X_x for each variable x together with the condition that X_x is a singleton, and replacing $x \in X$ by $X_x \subseteq X$.

Theorem 2.1. *The first-order theory of multisets with cardinality constraints is decidable.*

When there is no $\#(X)$ occurrence, the result can be obtained by encoding the multiset theory in Skolem arithmetic (private communication from Achim Blumensath). The extended version of the paper gives an alternative proof based on semilinear sets which provides an explicit representation of the model of a formula. When no multiset variable occur free, we get:

Proposition 2.1. *The model of a formula $\phi(N_1, \ldots, N_p)$ of $FO_\#(\mathcal{M})$ is a semi-linear set constructible in elementary time.*

3 Multitree Automata with Constraints

Equality of multitrees up to permutation of elements of $\oplus$ defines an equivalence relation. Let $e_1, e_2, \ldots$ be a enumeration of the equivalence classes corresponding to the elements of $\mathcal{T}$. For simplicity we identify an element of $\mathcal{T}$ and its equivalence class. We interpret each multitree t in $\mathcal{MT} = \mathcal{S} \cup \mathcal{T}$ as a multiset $[\![t]\!]$ of e_i's as follows: - if $t \in \mathcal{T}$ then t is in some e_i and we set $[\![t]\!] = \{e_i\}$
- if $t \in \mathcal{S}$ then $t = e_{i_1} \oplus \ldots \oplus e_{i_p}$ and $[\![t]\!] = \{e_{i_1}, \ldots, e_{i_p}\}$

For instance $[\![f(a \oplus b)]\!] = \{f(a \oplus b)\}$ and $[\![a \oplus b]\!] = \{a, b\}$.

Definition 3.1. *A multitree automaton is composed of a finite set of states $\mathcal{Q} = \{q_1, \ldots, q_m\}$, a set of final states $\mathcal{Q}_{Final} \subseteq \mathcal{Q}$ and a set of rules R of the form: (type 1) $\phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \rightarrow q$ for f of arity n*
(type 2) $\phi(X_{q_1}, \ldots, X_{q_m}) \Rightarrow q$
where in each case, ϕ denotes a formula of $FO_\#(\mathcal{M})$.

The transition relation $\rightarrow_\mathcal{A}$ is defined by $t \rightarrow_\mathcal{A} q$ iff

$$t = f(t_1, \ldots, t_n) \rightarrow_\mathcal{A} q \text{ if } \begin{array}{l} \phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \rightarrow q \in R \\ t_i \rightarrow_\mathcal{A} q_i \text{ for } i = 1, \ldots, n \\ \models \phi([\![t_1]\!], \ldots, [\![t_n]\!]) \end{array}$$

$$t = e_1 \oplus \ldots \oplus e_p \rightarrow_\mathcal{A} q \text{ if } \begin{array}{l} \phi(X_{q_1}, \ldots, X_{q_m}) \Rightarrow q \in R \\ t = t_1 \oplus \ldots \oplus t_m \text{ where, for } i = 1, \ldots, m, \\ t_i = e_{i,1} \oplus \ldots \oplus e_{i,n_i} \text{ and } e_{i,j} \rightarrow_\mathcal{A} q_i \text{ for } j = 1, \ldots, n_i \\ \models \phi([\![t_1]\!], \ldots, [\![t_m]\!]) \end{array}$$

A multitree is *accepted* iff $t \rightarrow_\mathcal{A} q$ with $q \in \mathcal{Q}_{Final}$. The language $\mathcal{L}(\mathcal{A})$ accepted by $\mathcal{A}$ is the set of multitrees accepted by $\mathcal{A}$.

Example 3.1. Given a signature consisting of two constants a, b, one binary symbol f, an automaton accepting only multisets with two constants a and b such that the number of b's is greater than the number of a's can be $\mathcal{A} = (\{q_a, q_b, q_S\}, \{q_S\}, R)$ with $R = \{True \Rightarrow a \rightarrow q_a, True \Rightarrow b \rightarrow q_b, \#_E(X_{q_a}) < \#_E(X_{q_b}) \Rightarrow q_S\ \}$.

Then $a \oplus b \oplus b \rightarrow q_S$ since $\begin{cases} a \rightarrow q_a,\ b \rightarrow q_b,\ [\![a]\!] = \{a\} \textit{ and } [\![b \oplus b]\!] = \{b, b\} \\ \models \#_E([\![a]\!]) < \#_E([\![b \oplus b]\!]) \end{cases}$

To accept also the multitrees s.t. that each subterm $f(t_1, t_2)$ satisfies $t_1 \neq t_2$ and $t_1, t_2 \in \mathcal{L}(\mathcal{A})$, we simply add the rule: $X_1 \neq X_2 \Rightarrow f(q_S, q_S) \rightarrow q_S$ □

Two automata are *equivalent* if they have the same language. The class of multitree languages accepted by multitree automata with constraints is denoted by $CMTL$. For simplicity, it is easier to consider automata such that

(i) $\mathcal{Q} = \mathcal{Q}_{\mathcal{T}} \cup \mathcal{Q}_{\mathcal{S}}$ with $\mathcal{Q}_{\mathcal{T}} \cap \mathcal{Q}_{\mathcal{S}} = \emptyset$,
(ii) for all type 1 rules $\phi(X_1, \dots, X_n) \Rightarrow f(q_1, \dots, q_n) \to q$, we have $q \in \mathcal{Q}_{\mathcal{T}}$
(iii) for all type 2 rules $\phi(X_{q_1}, \dots, X_{q_m}) \Rightarrow q$, we have $q \in \mathcal{Q}_{\mathcal{S}}$ and $\phi(X_{q_1}, \dots, X_{q_m}) \equiv \phi'(X_{q_1}, \dots, X_{q_{m'}})$ where $\{q_1, \dots, q_{m'}\} = \mathcal{Q}_{\mathcal{T}}$.

This can be achieved by splitting each state q into q_M and q_F, replacing type 1 rules $\ldots \to q$ by $\ldots \to q_F$ and type 2 rules $\ldots \Rightarrow q$ by $\ldots \Rightarrow q_M$ and any occurrence of q elsewhere by q_M and q_F.

Example 3.2. In the second automaton of the previous example, the state q_S can be reached by multitrees of $\mathcal{S}$ as well as multitrees of $\mathcal{T}$. Therefore we split it into $q_S^{\mathcal{S}}$ and $q_S^{\mathcal{T}}$ and replace the rule $X_1 \neq X_2 \Rightarrow f(q_S, q_S) \to q_S$ by the rules $X_1 \neq X_2 \Rightarrow f(\_, \_) \to q_S^{\mathcal{T}}$ where _ is any of $q_S^{\mathcal{T}}, q_S^{\mathcal{T}}$, and the rule $\#_E(X_{q_a}) < \#_E(X_{q_b}) \Rightarrow q_S$ by $\#_E(X_{q_a}) < \#_E(X_{q_b}) \Rightarrow q_S^{\mathcal{S}}$. □

4 Properties of Multitree Automata with Constraints

Membership. Given an automaton $\mathcal{A}$, its size $|\mathcal{A}|$ is the number of symbols of its presentation, $C(t)$ is a bound on the time for checking the satisfiability of constraints of $\mathcal{A}$ on the subterms of t. If the constraints are quantifier-free formulas, this amounts to solving equality of terms modulo AC which can be solved in polynomial time in $|t|$, see [BKN85].

Proposition 4.1. *$t \in \mathcal{L}(\mathcal{A})$ is decidable in time $O(|\mathcal{A}||t||C(t)|)$*

Proof. Consider all $|\mathcal{A}||t|$ possible labelling of nodes in t where the root is labelled by a final state, and check the applicability of rules. □

Completion. An automaton is *complete* if each multitree reaches at least one state. To get a complete automaton equivalent to a given automaton, we add a sink state q_S, the rules $True \Rightarrow f(\dots, q_S, \dots) \to q_S$ and the rules $True \Rightarrow q_S$.

Determinization. An automaton is *deterministic* iff for each multitree t, there exists at most one state q such that $t \to_{\mathcal{A}} q$. We show how to build a deterministic automaton $\mathcal{A}_D$ equivalent to a non-deterministic automaton $\mathcal{A} = (\mathcal{Q}_{\mathcal{A}}, \mathcal{Q}_{Final}, R)$. The deterministic automaton $\mathcal{A}_D$ has a set of states $\mathcal{Q}_D = 2^{\mathcal{Q}_{\mathcal{A}}}$, the final states are the states containing a final state of $\mathcal{A}$.

Determinization of conditions. First we replace rules by rules such that constraints are pairwise incompatible. For a symbol f of arity n, let $\phi_i(X_1, \dots, X_n)$ for $i = 1, \dots, m$ be the conditions of corresponding type 1 rules. For each $I \subseteq \{1, \dots, m\}$, let $\psi_I(X_1, \dots, X_n)$ be defined by

$$\bigwedge_{i \in I} \phi_i(X_1, \dots, X_n) \wedge \bigwedge_{i \notin I} \neg\phi_i(X_1, \dots, X_n)$$

By construction $\psi_I(X_1, \dots, X_n) \wedge \psi_J(X_1, \dots, X_n)$ is unsatisfiable if $I \neq J$ and $\phi_i(X_1, \dots, X_n) \Leftrightarrow \bigvee_{i \in I} \psi_I(X_1, \dots, X_n)$. Then each rule $\phi_i(X_1, \dots, X_n) \Rightarrow f(q_1, \dots, q_n) \to q$ is replaced by the rules $\psi_I(X_1, \dots, X_n) \Rightarrow f(q_1, \dots, q_n) \to q$ for $i \in I$.

The subset construction: type 1 rules. Type 1 rules of $\mathcal{A}_D$ are

$$\psi_I(X_1, \ldots, X_n) : f(Q_1, \ldots, Q_n) \to Q$$

with $Q = \{q \mid \exists q_1 \in Q_1, \ldots, q_n \in Q_n,\ \psi_I(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q \in R\}$

The subset construction: type 2 rules. For $J \subseteq \{1, \ldots, |\mathcal{Q}_\mathcal{A}|\}$, let Q_J denote $\{q_j \mid j \in J\}$ and let X_J the variable associated to Q_J in type 2 rules of $\mathcal{A}_D$. Let the type 2 rules of $\mathcal{A}$ be $\phi_1(X_1, \ldots, X_{|\mathcal{Q}_\mathcal{A}|}) \Rightarrow q_1,\ \ldots,\ \phi_p(X_1, \ldots, X_{|\mathcal{Q}_\mathcal{A}|}) \Rightarrow q_p$ For each $k = 1, \ldots, p$, we define $\psi_k(X_\emptyset, \ldots, X_J, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$ by:

$$\bigwedge_{J\subseteq\{1,..,|\mathcal{Q}_\mathcal{A}|\}} (\exists X_j^J\ X_J = \Sigma_{j\in J} X_j^J \wedge \phi_k(\Sigma_{J\subseteq\{1,..,|\mathcal{Q}_\mathcal{A}|\}} X_1^J, .., \Sigma_{J\subseteq\{1,..,|\mathcal{Q}_\mathcal{A}|\}} X_{|\mathcal{Q}_\mathcal{A}|}^J)$$

The idea underlying the construction of ψ_k is the following one: X_J represents a sum of e_i's s.t. each e_i reaches exactly all the states q_j for $j \in J$. In a derivation of $\mathcal{A}$, each e_i reaches only one of the possible q_j which is represented by the decomposition of X_J into the sum $\Sigma_{j\in J} X_j^J$. Finally, we sum all terms that reach the same state $q_i \in Q_\mathcal{A}$ for $i = 1, \ldots, |\mathcal{Q}_\mathcal{A}|$, and we check whether the condition ϕ_k is satisfiable, which means that the state q_k can be reached.

The last point is to eliminate the remaining ambiguities (since the same term can satisfy several ψ_k formulas) yielding the following type 2 rules of $\mathcal{A}_D$:

$$\bigwedge_{i\in I} \psi_i(X_\emptyset, \ldots, X_{\{1,\ldots,\mathcal{Q}_\mathcal{A}\}}) \wedge \bigwedge_{i\notin I} \neg\psi_i(X_\emptyset, \ldots, X_{\{1,\ldots,\mathcal{Q}_\mathcal{A}\}}) \Rightarrow Q_I$$

Proposition 4.2. $|\mathcal{A}_D| = O(2^{2^{|\mathcal{A}|}})$ *and* $t \to_{\mathcal{A}_D} Q$ *iff* $Q = \{q \mid t \to_\mathcal{A} q\}$.

Proof. We show that $t \to_{\mathcal{A}_D} Q$ iff $Q = \{q \mid t \to_\mathcal{A} q\}$ by structural induction on t.

Case $t = f(t_1, \ldots, t_n)$. Since the conditions of rules are either identical or pairwise incompatible, the proof is similar to the correctness proof for the determinization of tree automata.

Case $t = t_1 \oplus \ldots \oplus t_n$. Assume that the property holds for the t_i's. We denote by $t \to_\mathcal{A} I$ the property that $I = \{i \mid t \to_\mathcal{A} q_i\}$ for any multitree t. We can write

$$t = \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \Sigma_{t_j \to_\mathcal{A} J} t_j = \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} T_J \textit{ with } T_J = \Sigma_{t_j \to_\mathcal{A} J} t_j$$

where the decomposition is unique by induction hypothesis.

– Assume that $t \to_{\mathcal{A}_D} Q_I$ using $\bigwedge_{i\in I} \psi_i(X_\emptyset, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}) \wedge \bigwedge_{i\notin I} \neg\psi_i(X_\emptyset, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}) \Rightarrow Q_I$.
  Let $i \in I$. By definition $\models \psi_i(T_\emptyset, \ldots, T_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$ Therefore, for all $J \subseteq \{1, \ldots, |\mathcal{Q}_\mathcal{A}|\}$ we find a decomposition[2] $T_J = \Sigma_{j\in J} T_j^J$ such that

$$\models \phi_i(\Sigma_{J\subseteq\{1,\ldots\mathcal{Q}_\mathcal{A}\}} T_1^J, \ldots, \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} T_{|\mathcal{Q}_\mathcal{A}|}^J)$$

[2] this decomposition depends on i but we don't write this explicitely for simplicity

This proves that $t = \Sigma_{j=1}^{j=|\mathcal{Q}_\mathcal{A}|}\Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}T_j^J \rightarrow_\mathcal{A} q_i$.

For $i \notin I$, we can't find any such decomposition by definition of $\neg\psi_i$ (otherwise $t \models \psi_i$ for $i \notin I$ and $t \not\rightarrow_{\mathcal{A}_D} Q_I$), which proves that $t \not\rightarrow_\mathcal{A} q_i$.
Combining the two previous results, we get $Q_I = \{q_i \mid t \rightarrow_\mathcal{A} q_i\}$.

– Conversely, let $Q = \{q \mid t \rightarrow_\mathcal{A} q\} = Q_I$ for some I. According to our notation,

$$t = \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}T_J \textit{ where } T_J = \Sigma_{t_j \rightarrow_\mathcal{A} J} t_j$$

By definition of Q_I, for $i \in I$, there is a decomposition[3] $T_J = \Sigma_{j\in J}t_j^J$ s.t.

$$\phi_i(\Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}t_1^J, \ldots, \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}t_{|\mathcal{Q}_\mathcal{A}|}^J)$$

This proves that $\models \psi_i(T_\emptyset, \ldots, T_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$.

For $i \notin I$ there is no such decomposition (otherwise $t \rightarrow_\mathcal{A} q_i$), therefore $\not\models \psi_i(T_\emptyset, \ldots, T_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$.
Combining the two properties we get that $t \rightarrow_{\mathcal{A}_D} Q_I$ □

Compositional properties: Product, union, intersection. Given $\mathcal{A} = (\mathcal{Q} = \{q_1, \ldots, q_n\}, \mathcal{Q}_{Final}, R)$, $\mathcal{A}' = (\mathcal{Q}'\{q_1', \ldots, q_{n'}'\}, \mathcal{Q}_{Final}', R')$ two automata, the set of states of the product $\mathcal{A} \times \mathcal{A}'$ is $\mathcal{Q}_\times = \mathcal{Q} \times \mathcal{Q}'$, the set of final states is empty, and the rules are given by:

(type 1) $\phi(X_1, \ldots, X_n) \wedge \phi'(X_1, \ldots, X_n) \Rightarrow f((q_1, q_1'), \ldots, (q_n, q_n')) \rightarrow (q, q')$
$$\text{iff} \begin{cases} \phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \rightarrow q \in R, \\ \phi'(X_1, \ldots, X_n) \Rightarrow f(q_1', \ldots, q_n') \rightarrow q' \in R' \end{cases}$$

(type 2) $\phi(\Sigma_{j\in\{1,\ldots,n'\}}X_{(q_1,q_j')}, \ldots, \Sigma_{j\in\{1,\ldots,n'\}}X_{(q_n,q_j')}) \Rightarrow (q, q')$
$\wedge\ \phi'(\Sigma_{i\in\{1,\ldots,n\}}X_{(q_i,q_1')}, \ldots, \Sigma_{i\in\{1,\ldots,n\}}X_{(q_i,q_{n'}')})$
$$\text{iff} \begin{cases} \phi(X_{q_1}, \ldots, X_{q_n}) \Rightarrow q \in R \\ \phi'(X_{q_1'}, \ldots, X_{q_{n'}'}) \Rightarrow q' \in R' \end{cases}$$

Proposition 4.3. *The construction of $\mathcal{A} \times \mathcal{A}'$ is done in time $O(|\mathcal{A}||\mathcal{A}'|)$ and $t \rightarrow_{\mathcal{A}\times\mathcal{A}'} (q, q')$ iff $t \rightarrow_\mathcal{A} q$ and $t \rightarrow_{\mathcal{A}'} q'$*

From this proposition we get closure under intersection and union (simply adjust the set of final states accordingly).

Complementation Complementation is straightforward for a complete deterministic automata: exchange final and non-final states. Since every automaton is equivalent to a complete deterministic one, we are done.

[3] again, we don't mention explicitely that the decomposition depends on i

5 Decision of Emptiness

Now we come to our most technical result. The principle of the algorithm for deciding emptiness of $\mathcal{L}(\mathcal{A})$ is the same as for all classes of (finite) tree automata: it is a marking algorithm which marks all reachable states until no new state can be marked. As usual, constraints make life more difficult: given a rule $X_1 \neq X_2 \Rightarrow f(q,q) \rightarrow q'$, we can't mark the state q as soon as we know that some multitree reach this state since the satisfiability of the constraint $X_1 \neq X_2$ requires that at least two different multitrees reach q. Actually this ensures that also two multitrees reach q', establishing an invariant property of the marking algorithm. Since we deal with multitrees, we shall use two bounds: D on the number of different multitrees reaching each state, and M on the maximal multiplicity of an element in a multitree. These bounds are computed from the constraints of the rules and they are effectively computable because $FO_{\#}(\mathcal{M})$ is decidable.

Remark 5.1. Emptiness can be decided directly for a non-deterministic automaton, but in this case the algorithm realizes an implicit determinization which complicates the construction without adding significant improvments. Therefore we shall give the algorithm for deciding the emptiness of the language accepted by a deterministic automaton.

5.1 Formulae for States

Let $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_{Final}, R)$ be a deterministic automaton. We assume that $\mathcal{Q}$ is the disjoint union of $\mathcal{Q}_{\mathcal{S}}$ and $\mathcal{Q}_{\mathcal{T}} = \{q_1, \ldots, q_p\}$ such that only multitrees of $\mathcal{T}$ can reach a state of $\mathcal{Q}_{\mathcal{T}}$ and only multitrees of $\mathcal{S}$ can reach a state of $\mathcal{Q}_{\mathcal{S}}$. We now write formulae which ensure that a state q can be reached by some multitree X, when we have already computed Z_1 a set of multitrees of $\mathcal{T}$ reaching q_1,..., Z_p a set of multitrees of $\mathcal{T}$ reaching q_p. Since the automaton is deterministic, we have $Z_i \cap Z_j = \emptyset$ if $i \neq j$. We use the notation $set(X)$ for the multiset equal to the set of distinct elements of X (this can be defined in $FO_{\#}(\mathcal{M})$, see section 2).

Case of a state $q \in \mathcal{Q}_{\mathcal{S}}$. Let $\phi_i(X_{q_1}, \ldots, X_{q_p}) \Rightarrow q$ for $i = 1, \ldots, l$ be the type 2 rules for q. The formula $\psi_q(Z_1, \ldots, Z_p, X)$ states that X reaches q when Z_1 is a set of multitrees reaching q_1,...,Z_p a set of multitrees reaching q_p and it is defined by:

$$\bigvee_{i=1}^{i=l} (\exists X^i_{q_1}, \ldots, X^i_{q_p}\ X = \Sigma_{j=1}^{j=p} X^i_{q_j} \wedge \bigwedge_{j=1}^{j=p} set(X^i_{q_j}) \subseteq Z_j \wedge \phi_i(X^i_{q_1}, \ldots, X^i_{q_p}))$$
$$/*\ there\ is\ some\ rule\ reaching\ q\ that\ can\ be\ fired\ for\ X\ */$$

Case of a state $q \in \mathcal{Q}_{\mathcal{T}}$. First, we define the formula $X \in L_q$ which expresses that the multitree X is in the language accepted by q by:

- $\#(X) = 1 \wedge X \subseteq Z_i$ if q is some $q_i \in \mathcal{Q}_{\mathcal{T}}$
- $\psi_q(Z_1, \ldots, Z_p, X)$ if $q \in \mathcal{Q}_{\mathcal{S}}$

Note that the definition is consistent since ψ_q is already defined for $q \in \mathcal{Q}_\mathcal{S}$. For simplicity we assume that $\phi_i(X_{q_1^i}, \dots, X_{q_n^i}) \Rightarrow f(q_1^i, \dots, q_n^i) \rightarrow q$ for $i = 1, \dots, l$ are the type 2 rules for q (this can be achieved easily modulo introducing new states). The formula $\psi_q((Z_1, \dots, Z_p, X_1, \dots, X_n)$ for $q \in \mathcal{Q}_\mathcal{T}$ is defined by:

$$\bigvee_{i=1}^{i=l} (\bigwedge_{j=1}^{j=n} X_j \in L(q_j^i) \wedge \phi_i(X_1, \dots, X_n))$$
$$/* \ there\ is\ some\ rule\ reaching\ q\ that\ can\ be\ fired\ for\ f(X_1, \dots, X_n) */$$

5.2 Minimal Solutions of Formulae in $FO_\#(\mathcal{M})$

Let $\phi(N_1, \dots, N_p)$ be a formula of $FO_\#(\mathcal{M})$ with no multiset free variables, a p-uple $(n_1, \dots, n_p) \in \mathbb{N}^p$ is *minimal* for ϕ iff (i) $\models \phi(n_1, \dots, n_p)$ and (ii) there is no $(n_1', \dots, n_p')$ s.t. $\models \phi(n_1', \dots, n_p') \wedge \bigwedge_{i=1}^{i=p} n_i' < n_i$. Conditions (i) and (ii) are expressible by a formula $Minimal_\phi(N_1, \dots, N_p)$ of $FO_\#(\mathcal{M})$. The set of p-uples minimal for ϕ is a semilinear set computable in elementary time (by proposition 2.1). The *minimum* of $\phi(N_1, \dots, N_p)$, denoted by $m = Min(\phi)$ is the unique m minimal for $\psi(M) \equiv \exists N_1, \dots, N_p \ Minimal_\phi(N_1, \dots, N_p) \wedge \bigwedge_{i=1}^{i=p} M \geq N_i$. When the formula is unsatisfiable we set $m = +\infty$. By definition m is the smallest natural number which is greater that any component of any minimal solution of ϕ (if there exists one) and m is computable in elementary time.

5.3 Bounds for States

We define a bound D on the number of distinct elements and a bound M on the multiplicities of elements in multitrees of $\mathcal{MT}$ that reach a state q. For a state $q \in \mathcal{Q}_\mathcal{S}$ we compute D_q as

$$Min(\exists Z_1, \dots, Z_p, X \ \psi_q(Z_1, \dots, Z_p, X) \wedge \bigwedge_{i=1}^{i=p} \#(Z_i) = M_i)$$

For a state $q \in \mathcal{Q}_\mathcal{T}$ associated to f of arity n we compute D_q as

$$Min(\exists Z_1, \dots, Z_p, X_1, \dots, X_n \ \psi_q(Z_1, \dots, Z_p, X_1, \dots, X_n) \wedge \bigwedge_{i=1}^{i=p} \#(Z_i) = M_i)$$

and we set D as the maximum of the finite D_q's. This value bounds the number of multitrees that must reach q_1,...,q_p to allow the construction of a multitree reaching q (if such a multitree exists). Now we compute bounds on the multiplicities of elements of Z_i used in X or $X_1, \dots, X_n$, with the additional constraints that (i) $Z_1, \dots, Z_p$ have less than $D+1$ elements and (ii) we can construct $D+1$ multitrees reaching q (instead of a single one). We recall that $\#_M(X)$ is the formula defining the maximal multiplicity of elements of X.

For a state $q \in \mathcal{Q}_\mathcal{S}$, for $k = 1, \dots, D+1$ the formula $\rho_q^k(Z_1, \dots, Z_p, X_1, \dots, X_k)$ states that we can compute k distinct multitrees reaching q from $Z_1, \dots, Z_p$. It is defined by

$$\bigwedge_{j=1}^{j=k} \psi_q(Z_1, \ldots, Z_p, X_j) \wedge \bigwedge_{\substack{1 \leq i,j \leq k \\ i \neq j}} X_i \neq X_j$$

and we compute M_q^k as

$$Min(\exists Z_1, .., Z_p, X_1, .., X_k \ \rho_q^k(Z_1, .., Z_p, X) \wedge \bigwedge_{i=1}^{i=p} \#(Z_i) \leq D \wedge \bigwedge_{i=1}^{i=k} \#_M(X_i) \leq M)$$

which gives the bound on the multiplicities of occurrences of elements of Z_i's occurring in the X_j's for $j = 1, \ldots, k$ when we have the additionnal constraint that the number of elements of each Z_i is bounded by D.

Now, we must perform a similar computation for states of $\mathcal{Q}_\mathcal{T}$. The notation $(X_1^i, \ldots, X_n^i) \neq (X_1^i, \ldots, X_n^i)$ denotes the formula $\bigvee_{l=1}^{l=n} X_l^i \neq X_l^j$ and states that the two n-uples of multisets are distinct. For a state $q \in \mathcal{Q}_\mathcal{T}$ associated to f of arity n, for $k = 1, \ldots, D+1$, we define $\rho_q^k(Z_1, \ldots, Z_p, \underbrace{X_1^1, \ldots, X_n^1}_{first\ n-uple}, \ldots, \underbrace{X_1^k, \ldots, X_n^k}_{kth\ n-uple})$ which states that we can compute k distinct multitrees $f(X_1^1, \ldots, X_n^1)$,..., $f(X_1^k, \ldots, X_n^k)$ reaching q from $Z_1, \ldots, Z_p$, by

$$\bigwedge_{j=1}^{j=k} \psi_q(Z_1, \ldots, Z_p, X_1^j, \ldots, X_n^j) \wedge \bigwedge_{\substack{1 \leq i,j \leq k \\ i \neq j}} (X_1^i, \ldots, X_n^i) \neq (X_1^j, \ldots, X_n^j)$$

and we compute M_q^k as

$$Min(\exists Z_1, .., Z_p, X_1^1, .., X_n^1, .., X_1^k, .., X_n^k \quad \rho_q^k(Z_1, .., Z_p, X_1^1, .., X_n^1, .., X_1^k, .., X_n^k)$$
$$\wedge \bigwedge_{i=1}^{i=p} \#(Z_i) \leq D \wedge \bigwedge_{i=1}^{i=p} \#_M(X_i) \leq M)$$

Let M be the maximum of the finite M_q^k for $q \in \mathcal{Q}_\mathcal{S} \cup \mathcal{Q}_\mathcal{T}$ and $k = 1, \ldots, D+1$.

5.4 The Algorithm

Let D and M be as defined previously and let $\mathcal{Q} = \{q_1, \ldots, q_p\}$ be the set of states q of $\mathcal{A}$. For each $q_i \in \mathcal{Q}$, the *Reachability* algorithm computes the set $\mathcal{L}_i^m$ which is (an approximation of) the set of multitrees that reach the state q_i in m steps at most [4], where the approximation amounts to bounding the number of multitrees in $(\mathcal{L}_i^m)$ by D and the multiplicity of elements in a sum by M.

[4] One step doesn't mean one application of rule, but *label the root of a multitree by some state when all the sons are labelled by a state*

The Reachability algorithm

```
/*Initialize*/
$m = 0$, $\mathcal{L}_i^m = \emptyset$, set $q_i$ unmarked for all $i = 1, \ldots, p$
/*Loop*/
repeat /*Compute the increasing sequence $(\mathcal{L}_i^m)$*/
  for all $i = 1, \ldots, p$ do
      if $q_i$ is marked then $\mathcal{L}_i^{m+1} = \mathcal{L}_i^m$
      else $\mathcal{L}_i^{m+1}$=$\mathcal{L}_i^m \cup$ $\{t \mid t \rightarrow q_i$ for $t = f(t_1, \ldots, t_n)$ with $t_i \in \mathcal{L}_j^m$
                                      or $t = \Sigma_j t_j$ with $t_j \in \mathcal{L}_j^m$, $\#_D(t) \leq D$
                                                  and $\#_M(t) \leq M$  $\}$
      if $|\mathcal{L}_i^{m+1}| \geq D + 1$ then mark $q_i$
until $\mathcal{L}_i^m = \mathcal{L}_i^{m+1}$ for all $i = 1, \ldots, p$.
for each $i = 1, \ldots, p$ do set $\mathcal{L}_i = L_i^m$
```

Proposition 5.1. *The algorithm terminates and q_i is reachable iff $\mathcal{L}_i \neq \emptyset$.*

Proof. (Idea). Termination is obvious. To prove correctness, we set $L_i^0 = \emptyset$ and

$$L_i^{m+1} = L_i^m \cup \{t \mid t \rightarrow q_i \; for \; t = f(t_1, \ldots, t_n) \; and \; t_i \in L_i^m, i = 1, \ldots, n \quad or \quad t = t_1 \oplus \ldots \oplus t_l \; and \; t_i \in L_{j_i}^m, i = 1, \ldots, l\}$$

Then we prove that: $\forall m, i, \mathcal{L}_i^m \subseteq L_i^m$ *and* $(\mathcal{L}_i^m = L_i^m$ *or* $|\mathcal{L}_i^m| > D)$ □

An immediate consequence of the last proposition is:

Proposition 5.2. $\mathcal{L}(\mathcal{A}) = \emptyset$ *is decidable.*

The determinization process, the computations of bounds and the reachability algorithm involve only a fixed number of exponential steps. Therefore the decision procedure for emptiness is elementary.

6 Comparison with Other Classes of Tree Languages

Language with Equality/Disequality Constraints between Brothers. Tree automata with equality/disequality constraint between brothers are the most significant extension of tree automata which retains the good properties of tree automata: closure under boolean properties and decision of emptiness. This class, denoted by $L(AWEDC)$, has been used to get or improve decision results of many problems (mainly in rewriting, constraint solving and logic). No AC symbols occur in the signature and the only constraints rules have the form $\bigwedge_{i,j} X_i = X_j \wedge \bigwedge_{k,l} X_k \neq X_l \Rightarrow f(q_1, \ldots, q_n) \rightarrow q$ (a variable X_m representing the m^{th} son of the term on which the rule is tested). Such rules are a subcase of type 1 rules when no AC symbol occur, therefore these languages are particular instances of constrained multitree languages.

Proposition 6.1. $L(AWEDC) \subseteq CMTL$

The class Reg of regular languages is a subclass of $L(AWEDC)$, therefore $Reg \subset CMTL$. Unrestricted equality constraints leads to classes with an undecidable emptiness problem, but special inequality/disequality constraints have been studied leading to reduction automata [CCC$^+$94] or automata allowing only a bounded number of equality tests in a run [CJ94]. Such constraints are different in nature to $FO_{\#}(\mathcal{M})$ constraints and can't be combined with them.

Closure of Regular Tree Languages. Regular tree languages are usually not closed under associativity or associativity-commutativity but $CMTL$ languages are closed under associativity- commutativity. Therefore the relevant question is whether the closure of a regular-tree language under AC is necessarily in $CMTL$? Let $Cl(Reg)$ denote the closure of regular languages under AC where we assume that terms are flattened i.e. transformed into multitrees, see [BN98]. The expressivity of type 2 rules allows to get the following inclusion:

Proposition 6.2. *$Cl(Reg) \subseteq CMTL$*

Tree Language with Rational Constraints. Tree automata with rational constraints, $TARC$ in short, have usual tree automata rules as type 1 rule and the constraints for type 2 rules are Presburger formula $\psi(\#(X_1), \ldots, \#(X_n))$. In [Col02], these constraints are rational expressions that the representation of $(\#(X_1), \ldots, \#(X_n))$ in some basis satisfies [5]. Therefore we get:

Proposition 6.3. *$L(TARC) \subseteq CMTL$*

Since equational tree automata defined in [Ohs01] coincide with $TARC$ (unpublished result) we get another inclusion for free.

Multitree Automata with Arithmetic Constraints. Tree automata with arithmetic constraints [LM94] work on normalized multitrees where all occurrences of the same element e are replaced by a pair (multiplicity of e, e). A normalized multitree can be denoted by $n_1.e_1 \oplus \ldots \oplus n_p.e_p$. This normalization process is costly and can't be reversed. The states of these automata are divided in several sorts that we simplify into unprimed, primed, double primed states. The relevant rules of the automata are $\phi(N) : N.q \rightarrow q'$ and $\psi(\#(q'_1), \ldots, \#(q'_m)) \rightarrow q''$ where ψ and ϕ are Presburger formula, and $\#(q)$ denotes the number of occurrences of q. Furthermore, there is no constrained rules similar to type 1 rule. Normalized multisets accepted by these automata have the form $\{\underbrace{n^1_1.e^1_1, \ldots, n^1_{k_1}.e^1_{k_p}}_{\models \phi_1(n^1_i)}, \ldots, \underbrace{n^m_1.e^m_1, \ldots, n^m_{k_m}.e^m_{k_m}}_{\models \phi_m(n^m_i)}\}$ where $\models \psi(k_1, \ldots, k_m)$ for some Presburger formula ψ. The expressivity of $FO_{\#}(\mathcal{M})$ allows to express that some (not normalized) multiset has the above form after normalization. The constraint is

$$X = X_{q'_1} \oplus \ldots \oplus X_{q'_m} \wedge \bigwedge_{i=1,\ldots,m} Mult(\phi_i, X_{q'_i}) \wedge \psi(\#_D(X_{q'_1}), \ldots, \#_D(X_{q'_m}))$$

[5] in this paper, all but operators but $\oplus$ have arity 0 or 1

where $Mult(\phi, X)$ is the $FO_{\#}(\mathcal{M})$ formula stating that the multiplicity of each element of X satisfies ϕ (cf section 2). Therefore denoting by $L(TAC)$ the set of languages accepted by tree automata with arithmetic constraints, we get:

Proposition 6.4. $L(TAC) \subseteq CMTL$

[Lug98] defines a class of multitree automata which is stricly included in $CMTL$. The constraints of type 1 rules are only boolean combinations of equations where one side is a variable (e.g. $X_i = \Sigma_j \in \{1, \ldots, n\} X_j$) and the rules for terms of $\mathcal{S}$ can be replaced by type 2 rules where the constraint is a Presburger arithmetic formula. For instance, it is impossible to express normalization in this class, therefore it is disjoint from TAC. But due to the high expressive power of $FO_{\#}(\mathcal{M})$, both classes are included in $CMTL$. Moreover the algorithm to decide emptiness of $\mathcal{L}(\mathcal{A})$ used Dickson's lemma which prevented from stating that the complexity of the problem was elementary.

Feature Tree Automata. Feature tree automata have been introduced by Podelski and Niehren [NP93] to provide a notion of recognizable languages for feature trees. Feature trees can be seen as multisets constructed from a finite set of multiset constructors $\{,\}_A \{,\}_B \ldots$ and free unary symbols $f_1, f_2, \ldots$ (feature constructors)[6]. Recognizable sets are the multisets satisfying boolean combination of counting constraints of the form $\phi(\#(f_i))$. This kind of constraints is a very special case of $FO_{\#}(\mathcal{M})$ formula. In some sense, there are *local* constraints, since they don't relate the number of occurrences of some feature f and the number of occurrences of some other feature g. If we denote by $L(FTA)$ the set of multitree languages accepted by feature tree automata, we get:

Proposition 6.5. $L(FTA) \subseteq CMTL$.

Conclusion

One possible extension of this work is to look also at the associativity axiom. It is straightforward to add rules like hedge automata rules in this framework without losing properties, but a more interesting idea is to combine regularity constraints (like in hedge automata) and $FO_{\#}(\mathcal{M})$ formula. However we have found out that the resulting class is not closed under complement and doesn't enjoy determinization even when we allows only Presburger formulae [DL02]. Another possible question is to look for extensions allowing a bounded number of equality test along the acceptance process, but there is probably no satisfactory result to hope for in this direction since the relevant classes of automata are not closed under all the boolean operations, even when no associativity-commutativity axiom is used. Another direction of research is to use two-way tree automata (one can go up and down in the multitree). Some work has been

[6] In the original presentation of [NP93], the A's are constructors labelling nodes and the features f's label edges

done in this direction [GLV02], but undecidability quickly shows up and the counting/equality constraints that we use are probably too expressive to work well in that extension. Designing efficient implementations of our algorithms is also an issue: the main point is to balance the expressivity of constraints and a reasonable algorithmic efficiency using good data structures for multisets.

References

[Ber77] L. Berman. Precise bounds for Presburger arithmetic and the reals with addition: Preliminary report. In *Proc. of Symp. on Foundation Of Computer Science*, pages 95–99, 1977.

[BKN85] Dan Benanav, Deepak Kapur, and Paliath Narendran. Complexity of matching problems. In *Proc. of 1st Int. Conf. on Rewriting techniques and Applications*, vol. 202 of *Lect. Notes in Comp. Sci.*, pages 417–429, 1985.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[BT92] B. Bogaert and S. Tison. Equality and disequality constraints on direct subterms in tree automata. In *Proc. of the 9th STACS*, vol. 577 of *Lect. Notes in Comp. Sci.*, pp. 161–172, 1992.

[CCC$^+$94] A.C. Caron, H. Comon, J.L. Coquidé, M. Dauchet, and F. Jacquemard. Pumping, cleaning and symbolic constraints solving. In *Proc. 21st ICALP, Jerusalem (Israel)*, pages 436–449, 1994.

[CJ94] H. Comon and F. Jacquemard. Ground reducibility and automata with disequality constraints. In Springer-Verlag, editor, *Proc. of 11th STACS*, vol. 820 of *Lect. Notes in Comp. Sci.*, pages 151–162, 1994.

[Col02] Th. Colcombet. Rewriting in partial algebra of typed terms modulo aci. presented at the Infinity workshop, August 2002.

[DL02] S. DalZilio and D. Lugiez. Multitrees automata, Presburger's constraints and tree logics. Tech. Report 4631, INRIA, 2002.

[GLV02] J. Goubault-Larrecq and K.N. Verma. Alternating two-way AC-tree automata. Technical report, LSV, ENS Cachan, 2002.

[LM94] D. Lugiez and J.L. Moysset. Tree automata help one to solve equational formulae in AC-theories. *Journal of Symbolic Computation*, 18(4):297–318, 1994.

[Lug98] D. Lugiez. A good class of tree automata. In K. Larsen, S. Skyum, and G. Winskel, editors, *Proc. of 15th ICALP*, vol. 1443 of *Lect. Notes in Comp. Sci.*, pages 409–420. Springer-Verlag, 1998.

[Mur01] Makoto Murata. Extended path expression for XML. In ACM, editor, *Proc. of the 20th Symp. on Principles of Database Systems (PODS)*, Santa Barbara, USA, 2001. ACM.

[NP93] Joachim Niehren and Andreas Podelski. Feature automata and recognizable sets of feature trees. In *Proc. TAPSOFT'93*, vol. 668 of *Lect. Notes in Comp. Sci.*, pages 356–375, 1993.

[Ohs01] Hitoshi Ohsaki. Beyond the regularity: Equational tree automata for associative and commutative theories. In *CSL 2001*, vol. 2142 of *Lect. Notes in Comp. Sci.*. Springer-Verlag, 2001.

[PQ68] C. Pair and A. Quéré. Définition et étude des bilangages réguliers. *Information and Control*, 13(6):565–593, 1968.