Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 333 (2005) 225–263
Theoretical Computer Science
www.elsevier.com/locate/tcs

# Multitree automata that count☆

Denis Lugiez

*Lab. d'Informatique Fondamentale, UMR 6166, CNRS & Université de Provence,CMI 39 av. Joliot Curie, 13453 Marseille Cedex, France*

**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, etc. We define constrained multitree automata which accept sets of multitrees where the constraints are expressed in a first-order theory of multisets with counting formulas which is very expressive. We give constructions for union, intersection, determinization. Then, we give an algorithm to decide emptiness when the constraints belong to a subclass where counting is limited to distinct elements. We show that many classes of tree automata that have been defined for a wide variety of applications can be seen as instance of our general framework. Finally, we describe the quantifier elimination procedure used to decide the theory of constraints.
© 2004 Elsevier B.V. All rights reserved.

*Keywords:* Tree-automata; Presburger arithmetic; Formal languages; Logic

## 1. Introduction

Multitrees are unranked unordered trees and appear, explicitly or implicitly in a large variety of applications in computer science: process algebras like BPP,PA and PRS in the field of concurrency, XML schemes for typing or querying XML documents, knowledge representation (feature logics), automated theorem proving with associative-commutative operators, verification of cryptographic protocols, etc. These applications require to manipulate and combine sets of multitrees. For ordinary trees, regular tree languages and tree

☆ This article is an extended and revised version of a paper entitled *Multitree Automata with Counting and Equality Constraints* presented at FOSSACS 2003, Warszawa, Poland.

*E-mail address:* lugiez@cmi.univ-mrs.fr (D. Lugiez).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.10.023

automata have been used successfully for such purposes. However, despite of their good properties, regular languages suffer from a lack of expressive power, which leads computer scientists to propose extensions of these languages which are more expressive. The usual approach is to add *constraints* that restrict the applicability of a rule. For instance, a rule is applicable to a binary tree only if the left and the right sons are equal. This approach has led to significant progress in rewriting, logic and applications via a sequence of variations on this scheme, see [4,6,8].

For unranked trees, the first proposition [27] dealt with ordered trees where the constraints are regular expressions on the set of states. This proposition has been recently renewed for XML applications [23] and a new class of tree automata combining regular expressions and counting constraints has been introduced for XML applications too [9,24]. For unranked, unordered trees, a good notion of tree automata has not been defined in a clean way until recently although the definition is simple (replace regular expressions of the previous case by membership to a finitely generated commutative monoid) and underlies some theoretical works on the recognizability of forests of trees. A different scheme, see [25], was proposed in the field of knowledge representation (feature tree automata) but was not developed further on. Like in the tree case, such automata lack expressive power and it is desirable to enhance the power of the device without loosing too many of the good properties (closure under boolean combinations and decidability of emptiness of the language accepted by an automaton).

In this case, the problem is much more complex than in the tree case. First, since a multitree is a multiset of (multi)trees, there is no obvious way to define equality constraints like *the first son is equal to the second one*, since there is no longer a first nor a second son, and the number of sons is unbounded! Even for trees, equality constraints cannot be used too liberally: emptiness becomes undecidable when equality test can compare subterms at any position. The first contribution of this paper is to propose a notion of equality constraints which coincide with the usual notion for trees and capture natural constraints between multitrees. An orthogonal approach is to consider that another relevant notion in a multitree is the number of (distinct or not) sons that satisfy some property. This has been used in modal logics for instance and we have incorporated it in our constraints. The second contribution is to set decidability results on the theory of constraints which allows to shift the classical constructions on tree automata into this new framework. A generic emptiness proof is given for deterministic tree automata which improves previous results both on the decidability side and on the complexity side (the closest proposition [19] was much less expressive and relied on Dickson's lemma for deciding emptiness). The last contribution shows that almost all previous propositions for multitree automata fall into one particular subclass of our framework which has all desirable properties.

### *1.1. Content of the paper*

*A general framework for constraint multitrees*: We describe a very general framework for adding constraints in multitree automata and we show that the usual constructions can be carried out in this framework.

*Constraints for multitrees*: The constraints that we use are natural (combine equality and cardinality of multisets), expressive and are closely related to Skolem arithmetic (the theory of natural numbers with multiplication) extended by counting operations.

*Decidable subclasses*: The multitree automata are closed under the boolean operation and emptiness is decidable if the counting constraints are limited to counting the number of distinct subterms.

*A large expressive power*: Many previous propositions for handling multitrees in the unordered case fall in one of the decidable subclasses, thus providing a uniform presentation for all these devices.

### *1.2. Related works*

A lot of work has been done to extend tree automata with equality constraints, and is referred above, but the closest works cope with unranked structures and/or logic which have some relationship to such structures. Hedge automata, [23,27], used for XML applications have been recently extended to mix ordered and unordered structures in this framework, where the price to pay is a class of tree automata with less desirable properties [9,24]. Constraints for multitrees occur explicitly or not in several works according to two different presentations: either one uses Presburger formula (as we do in this work) or one uses regular languages of a finitely generated commutative monoid. For instance, this is done in Colcombet's work [7] for an application to typing in Process Algebras. Other works study the behaviour of tree automata under associative-commutative symbols, allowing equality of variables in rules or using the axioms of associativy-commutativy in the acceptance process or two-way automata [16,26,28,29], yielding various classes that can be closed or not, decidable or not.

Another trend of work is to extend a logic (modal logics, temporal logics, $\mu$-calculus, etc.) with counting capabilities. Since automata are often used for decidability problems in these logics, some combinations of counting constraints and word or tree automata have been proposed in this case. For instance [18] gives a class of tree automata that is used to decide the satisfiability problem in the *graded mu-calculus* where one use graded quantifiers *there exists at least n elements* or *all but n elements*. A recent work by Klaedtke and Ruess [17] defines extensions of WS1S and WS2S with cardinality constraints for which they prove undecidability and decidability results. In our work, we use multisets (and not sets) which is more complex, but our constraints consider flat multisets only, which is simpler: both approaches are uncomparable, although there are some similarities in the techniques that could be investigated.

### *1.3. A roadmap for the reader*

The basic notions on multitrees and Presburger arithmetic that are used throughout the paper are given in Section 2. Section 3 describes the constraints on multisets that we use. The main topic of the paper appears in Section 4 which presents multitree automata. Then, we establish results on closure under union and intersection and we show how determinization can be achieved (Section 5). The most difficult result of the paper is given in Section 6: it is the decidability of emptiness for constrained multitree automata. Some extensions are given in Section 7 and the expressivity of the class of multitree automata is investigated in Section 8 where we show that it embeds many previous classes of tree automata. Appendix A

gives some proofs that have not been given previously and Appendix B gives the decidability proofs for the theory of constraints.

## 2. Multitrees and related notions

### 2.1. Definition

A multitree is a finite labeled tree on a finite signature where some symbols are free symbols with a fixed arity, and where the other symbols are variadic, hence may have any number of arguments. Moreover, we shall assume that the order of the arguments is irrelevant. A typical example of an operation denoted by a symbol of the later kind is the parallel composition of processes in concurrency theory. For simplicity, we shall consider only one variadic symbol $\oplus$, but all our results are valid when the signature contains a finite number of such symbols.

More precisely, given a set of free symbols (each one with a fixed arity) and a variadic symbol $\oplus$, the set of multitrees $\mathcal{MT}$ is given by the grammar:

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

For instance, given two constants $a, b$, a binary symbol $f$, the multitree $f(a \oplus g(b), a \oplus a)$ belongs to $\mathcal{T}$ when the multitree $a \oplus a \oplus f(a, a)$ is in $\mathcal{S}$.

A less theoretical example is the bibliographic entry

$$\begin{array}{l} book < name < Knuth > \ \&year < 1970 > \\ \qquad\qquad\qquad\qquad \&title < The\_Art\_of\_Computer\_Programming >> \end{array}$$

which is a multitree of $\mathcal{T}$ build on a signature containing the constants *Knuth*, 1970, $The\_Art\_of\_Computer\_Programming$, the unary symbols *name*, *year*, *title*, *book* for the names of the fields of the reference, and the interleaving operation for composing fields (denoted by & instead of $\oplus$). The assumption that the arguments of & are unordered is justified if we consider this expression as a type or a query to some bibliographic database since it is unlikely that all documents have the same ordering for their fields.

*The issue*: In this paper, we deal with the problem of representing and combining (usually infinite) sets of multitrees by extending the notions of regular tree languages and tree automata which have been extremely successful in many areas of computer science.

### 2.2. Presburger arithmetic

Presburger arithmetic is the main tool to define automata for multitrees and constraints for these automata.

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. This theory is super-exponential: any non-deterministic decision procedure requires at least $O(2^{2^{cn}})$ steps to decide a formula of size $n$ [13]. More accurate complexity results are given

in [3]. Diophantine equations, inequations are examples of Presburger arithmetic formula (with a lower complexity since they are in NP). The notation $\models\varphi(\alpha_1,\ldots,\alpha_n)$ means that the tuple of non-negative integers $\alpha_1,\ldots,\alpha_n$ satisfies the formula $\varphi(x_1,\ldots,x_n)$.

Let $\vec{x} = (x_1,\ldots,x_n)$, $\vec{y} = (y_1,\ldots,y_n)$ be tuples of integer variables. We define $\vec{x} \leqslant \vec{y}$ by $\bigwedge_{i=1}^{i=n}(x_i \leqslant y_i)$. Given a formula $\varphi$ of Presburger arithmetic in the free variables $x_1,\ldots,x_n$, we denote by $Min\{(x_1,\ldots,x_n) \mid \models\varphi(x_1,\ldots,x_n)\}$ the set of tuples of integers that satisfy the formula

$$\varphi(x_1,\ldots,x_n) \wedge (\forall y_1,\ldots,y_n\ \varphi(y_1,\ldots,y_n) \Rightarrow \neg((y_1,\ldots,y_n) \leqslant (x_1,\ldots,x_n))).$$

This set is effectively computable and is always finite by Dickson's lemma ([10]).

### 2.3. Multisets

A multiset on a set of elements $\mathcal{S} = \{e_1, e_2, \ldots\}$ is a mapping from $\mathcal{S}$ to $\mathbb{N}$, and the image of an element $e_i$ is called its multiplicity. The set $\mathcal{S}$ is infinite, [1] but we shall consider *finite multisets* only, i.e. the multiplicity of all elements but a finite number is zero. Intuitively, a multiset is a set where elements can be repeated and we shall use a set-like notation to denote multisets. For example $\{e_1, e_1, e_2\}$ denotes the multiset where $e_1$ has multiplicity 2, $e_2$ multiplicity 1 and other elements have multiplicity 0. We may also denote this multiset $\{2.e_1, e_2\}$. The empty multiset is denoted by $\emptyset$. The number of elements of a multiset $\mathcal{M}$ is the sum of the multiplicities, it is denoted by $\#(\mathcal{M})$. The number of distinct elements of a multiset $\mathcal{M}$ is the number of elements which have a non-zero multiplicity, it is denoted by $\#_D(\mathcal{M})$. For instance $\#(\{2.e_1, e_2\}) = 3$ and $\#_D(\{2.e_1, e_2\}) = 2$. The union of two multisets $\mathcal{M}_1$ and $\mathcal{M}_2$ is the multiset such that the multiplity of an element is the sum of its multiplicity in $\mathcal{M}_1$ and of its multiplicity in $\mathcal{M}_2$.

## 3. Logical theories for multisets

In this section, we define several first-order theories on finite multisets that are used later on in the definition of multitree automata.

### 3.1. The first-order theory of inclusion

*The syntax of formula*: Let $X, Y, \ldots$, be 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. The predicate is the inclusion predicate $\subseteq$. [2] Formulas are given according to the grammar:

$$\phi ::= (T \subseteq T) \mid \neg\phi \mid \phi \wedge \phi \mid \exists X\ \phi.$$

[1] The case of a finite set $\mathcal{S}$ is not interesting: multisets are subsets of $\mathbb{N}^n$.

[2] An equivalent choice is to take multiset equality as the basic predicate.

*Semantics*: Let $\mathbb{M}$ be the set of finite multisets built on a denumerable set $\mathcal{S}$ of distinct elements $e_1, e_2, \ldots$. An *interpretation* $I$ associates to each variable $X$ a multiset $I(X) \in \mathbb{M}$. The function $\oplus$ is interpreted as the union of multisets, inclusion is multiset inclusion. The interpretation is extended to formulas as usual, and we define satisfiability, models,…in the standard way. We write $\models \phi(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ if $\mathcal{X}_1 = I(X_1), \ldots, \mathcal{X}_n = I(X_n)$ is a model of $\phi(X_1, \ldots, X_n)$. This theory is denoted by $FO_{\mathcal{M}}$.

**Example 1.** The multisets $\mathcal{X}, \mathcal{Y}$ satisfying $X \subseteq Y \wedge Y \subseteq X$ are the multisets that are equal.

From now on, we use the notation $X = Y$ as a shorthand for the previous formula. In the following, we usually use capital letters $X, Y, \ldots$ for multiset variables, calligraphic fonts $\mathcal{X}, \mathcal{Y}, \ldots$ for multisets, $x, y, \ldots$ for integer variables associated to multisets variables $X, Y, \ldots$, greek letters $\alpha, \beta, \ldots$ for integers instantiating $x, y, \ldots$, capital letters $N, M, \ldots$ for integer variables used as exponents or free variable integers and $n, m, \ldots$ for integers instantiating these variables.

We now extend $FO_{\mathcal{M}}$ by cardinality constraints.

### 3.2. Cardinality constraints

In a multiset either one counts the number of distinct elements or the number of elements with their multiplicities. These notions coincide for sets, but lead to different logical theories for multisets.

#### 3.2.1. Counting the number of distincts elements: the logic $FO^{\#D}_{\mathcal{M}}$

We enrich the signature with a unary symbol $\#_D$ and $\#_D(X)$ is interpreted by the number of distinct elements of $X$. The set of terms is the same as the set of terms of $FO_{\mathcal{M}}$ but formulas are defined according to the grammar:

$$\phi ::= (T \subseteq T) \mid \psi(\#_D(X_1), \ldots, \#_D(X_n), M_1, \ldots, M_l) \mid \neg\phi \mid \phi \wedge \phi \mid \exists X\ \phi \mid \exists M\ \phi,$$

where $X_1, \ldots, X_n, X$ denote multiset variables, $M_1, \ldots, M_l, M$ denote integer variables, $\psi$ is a Presburger arithmetic formula in $n + l$ variables. Interpretations $I$ are extended to associate to each integer variable $N$ some natural number $I(N) \in \mathbb{N}$. The resulting first-order theory is called $FO^{\#D}_{\mathcal{M}}$.

#### 3.2.2. Counting the number of elements: the logic $FO^{\#}_{\mathcal{M}}$

The first-order theory $FO^{\#}_{\mathcal{M}}$ is defined as $FO^{\#D}_{\mathcal{M}}$, except that $\#_D$ is replaced by $\#$ where $\#(X)$ is interpreted by the number of elements of $X$.

### 3.3. Expressivity of $FO_{\mathcal{M}}$, $FO^{\#D}_{\mathcal{M}}$, $FO^{\#}_{\mathcal{M}}$

Many natural properties can be expressed in these logics. We give examples that we use later on. Simpler descriptions can be given with a more generous use of $\#_D$ or $\#$ but we want to distinguish the expressivities of these logics.

- $X$ is empty: $\forall Y\ X \subseteq Y$, that we denote by $X = \emptyset$,
- $Z$ is the intersection of $X$ and $Y$

$$Z \subseteq X \wedge Z \subseteq Y \wedge \forall U(U \subseteq X \wedge U \subseteq Y \Rightarrow U \subseteq Z),$$

- a multiset $X$ is a set, i.e. each element of $X$ occurs once

$$\forall Y, Z\ (X = Y \oplus Z \Rightarrow Y \cap Z = \emptyset)$$

that we denote by $set(X)$.
- a multiset $X$ is the set of the distinct elements of the multiset $Y$

$$\begin{array}{l}(X = \emptyset \wedge Y = \emptyset)\\ \quad \vee (Y \neq \emptyset \wedge X \subseteq Y \wedge set(X) \wedge \forall Z, T\\ \quad ((Y = Z \oplus T \wedge Z \neq \emptyset) \Rightarrow X \cap Z \neq \emptyset)),\end{array}$$

that we denote by $X = setof(Y)$.
- $X$ is a singleton: $X \neq \emptyset \wedge \forall Y, Z(X = Y \oplus Z \Rightarrow (Y = \emptyset \vee Z = \emptyset))$ that we denote by $Sing(X)$, and $X$ contains only copies of the same element: $\forall Y(Y = setof(X) \Rightarrow Sing(Y))$ that we denote by $One(X)$.
- We can simulate Presburger arithmetic: the idea is to identify a multiset containing only $n$ copies of the same element $e$ and the natural number $n$, and to add constraints to ensure that all variables denote multisets containing copies of the same element. Let $\varphi(x_1, \ldots, x_n)$ be the Presburger arithmetic formula:

$$(\forall|\exists)x_{n+1} \ldots (\forall|\exists)x_{n+m}\psi(x_1, \ldots, x_n, \ldots, x_{n+m}).$$

Then $\phi_P(X_1, \ldots, X_n)$ defined by

$$\begin{array}{l}(\forall|\exists)X_{n+1} \ldots (\forall|\exists)X_{n+m} \wedge \Psi(X_1, \ldots, X_n, \ldots, X_{m+n})\\ \qquad \wedge \bigwedge_i (One(X_i) \vee X_i = \emptyset)\\ \qquad Sing(X_0) \wedge \bigwedge_{i,j}(X_i \neq \emptyset \wedge X_j \neq \emptyset \Rightarrow X_i \cap X_j \neq \emptyset)\end{array}$$

($\Psi$ is as $\psi$ where $\oplus$ replaces $+$ and $X_i$ replaces $x_i$, the number 1 is replaced by $X_0$) is a $FO_{\mathcal{M}}$ formula such that $\models\phi_P(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ iff $\mathcal{X}_1 = \{\alpha_1 e\}, \ldots, \mathcal{X}_n = \{\alpha_n e\}$ and $\models\varphi(\alpha_1, \ldots, \alpha_n)$.
- The multiset $X$ is such that the multiplicity $n_i$ of each element $e_i$ satisfies some Presburger formula $\psi$.

$$\forall X_e\ Y\ X = X_e \oplus Y \wedge One(X_e) \wedge X_e \cap Y = \emptyset \Rightarrow \Psi(X_e),$$

where $\Psi$ is the $FO_{\mathcal{M}}$ formula corresponding to $\psi$ defined as above. This formula is denoted by $Mult(X, \psi)$. In Section B.3.1, we generalize this formula to a tuple $\vec{X} = (X_1, \ldots, X_n)$ and a formula $\psi$ with $n$ free variables, yielding a formula $Mult(\vec{X}, \psi)$.
- We can simulate an extension of the logic with variables $x, y, \ldots$ for elements (the $e_i$'s) and the membership predicate $x \in X$ as follows: instead of $x$ we use $X_x$ and $x \in X$ is replaced by $Sing(X_x) \wedge X_x \subseteq X$.

- Any formula stating some relationship between the cardinality of elements of multisets is expressible by a Presburger arithmetic formula. For instance, $X$ has more distinct elements than $Y$: $\#_D(X) > \#_D(Y)$ (similarly for more elements using $\#$ instead of $\#_D$).
- $\#_D$ is definable from $\#$: $\#_D(X) = \#(Y) \wedge Y = setof(X)$.

The reader should notice that all these formulas except the last two ones belong to $FO_{\mathcal{M}}$ and do not require the cardinality operators. The relationship between these logics can be summarized as follows:

**Proposition 1.** *$FO_{\mathcal{M}}$ is less expressive than $FO^{\#D}_{\mathcal{M}}$ which is less expressive than $FO^{\#}_{\mathcal{M}}$.*

*A is less expressive than B means that each formula of A can be defined as a formula of B. The first statement is obvious and the second one relies on the encoding of $\#_D$ by $\#$.*

### *3.4. Decidability results*

We define $\exists FO^{\#}_{\mathcal{M}}$ the extended existential theory of $FO^{\#}_{\mathcal{M}}$ by the following grammar:

$$\phi_\exists ::= \psi(\vec{X}) \mid \phi(\vec{X}) \mid \phi_\exists \wedge \phi_\exists \mid \phi_\exists \vee \phi_\exists \mid \exists X\ \phi_\exists,$$

where $\psi(\vec{X})$ is any formula of $FO^{\#D}_{\mathcal{M}}$ in the free variables $\vec{X}$, $\phi(\vec{X})$ is any *quantifier-free* formula of $FO^{\#}_{\mathcal{M}}$ in the free variables $\vec{X}$. For instance $\exists X_2\ (X_1 = setof(X_2) \wedge \phi(X_2))$ with $\phi(X_2)$ an quantifier-free formula of $FO^{\#}_{\mathcal{M}}$ is a formula of $\exists FO^{\#}_{\mathcal{M}}$.

**Proposition 2.** *The following properties hold*:

- *$FO^{\#D}_{\mathcal{M}}$ is decidable.*
- *The set $\{\#(\vec{\mathcal{X}}) \mid \models \phi(\vec{\mathcal{X}})\}$ where $\phi$ is a formula of $\exists FO^{\#}_{\mathcal{M}}$ is the model of an effectively constructible Presburger arithmetic formula.*
- *The set $\{\#_D(\vec{\mathcal{X}}) \mid \models \phi(\vec{\mathcal{X}})\}$ where $\phi$ is a formula of $\exists FO^{\#}_{\mathcal{M}}$ is the model of an effectively constructible Presburger arithmetic formula.*

The second statement yields the decidability of the existential and of the universal fragment of $FO^{\#}_{\mathcal{M}}$. The status of the complete logic is still an open problem. [3]

**Proof** (Sketch). For the first decidability results, we relate $FO^{\#D}_{\mathcal{M}}$ to Skolem arithmetic (this connection has been suggested by A. Blumensath, private communication). To each $e_i$ we associate $p_i$ the $i$th prime number and we encode a multiset $\{n_1 e_{i_1}, \ldots, n_p e_{i_p}\}$ by the number $p_{i_1}^{n_1} * \ldots * p_{i_p}^{n_p}$. Then the $\oplus$ operation corresponds to the multiplication of integers and $\#_D(X)$ corresponds to the number $\#_D(x)$ of distinct prime numbers in the factorization

[3] Contrary to the original claim of [20].

of $x$. Therefore, $FO^{\#D}_{\mathcal{M}}$ is equivalent to the extension of Skolem arithmetic $(\mathbb{N}, *, =, \#_D)$ which is decidable [11].

The proof of the second statement is more difficult and is given in Appendix B.

The third statement is a direct consequence of the second one: if $\phi(\vec{X})$ is a formula of $\exists FO^{\#}_{\mathcal{M}}$, then $\exists \vec{X}\ (\bigwedge_{i=1}^{i=n} Y_i = setof(X_i) \wedge \phi(\vec{X}))$ is also a formula of $\exists FO^{\#}_{\mathcal{M}}$. Since the formula implies that $\#(\vec{Y}) = \#_D(\vec{X})$ the result follows from the second statement. □

We immediately get that a lower bound for testing the decidability of formulas of size $n$ is $O(2^{2^{2^{cn}}})$ from the result of Rackoff and Ferrante [12] for Skolem arithmetic. Usually, the formulas arising in constraints belong to fragments that have a lower complexity.

According to the results on Presburger arithmetic given in Section 2.2, we also get that the sets $Min\{\#(\vec{\mathcal{X}}) \mid \vDash \phi(\vec{\mathcal{X}})\}$ and $Min\{\#_D(\vec{\mathcal{X}}) \mid \vDash \phi(\vec{\mathcal{X}})\}$ are computable when $\phi$ is a formula of $\exists FO^{\#}_{\mathcal{M}}$.

## 4. Multitree automata with constraints

This section introduces the extension of multitree automata which constitutes the core of this paper. Firstly, we show how to link multitrees and multisets.

### 4.1. How to interpret a multitree as a multiset

The permutative equivalence $\equiv_P$ on multitrees is the smallest relation such that: (i) $f(t_1, \ldots, t_n) \equiv_P f(s_1, \ldots, s_n)$ iff $s_i \equiv_P t_i$ for $i = 1, \ldots, n$ and (ii) $t_1 \oplus \ldots \oplus t_n \equiv_P s_1 \oplus \ldots \oplus s_n$ iff $n = m$ and there is a substitution $\sigma$ of $\{1, \ldots, n\}$ such that $s_i \equiv_P t_{\sigma(i)}$. Let $\mathcal{S} = \{e_1, e_2, \ldots\}$ be the set 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 $e_i$ then $[\![t]\!] = \{e_i\}$,
- if $t = t_1 \oplus \ldots \oplus t_p \in \mathcal{S}$ with $t_i \in \mathcal{T}$ for $i = 1, \ldots, p$ then $[\![t]\!] = \{e_{j_1}, \ldots, e_{j_p}\}$ with $e_{j_i}$ the equivalence class of $t_i$, $i = 1, \ldots, p$.

For instance $[\![f(a \oplus b)]\!] = \{f(a \oplus b)\}$ and $[\![a \oplus b]\!] = \{a, b\}$. By definition $[\![s]\!] = [\![t]\!]$ if $s \equiv_P t$.

### 4.2. Definitions

**Definition 1.** A multitree automaton is composed of a finite set of states $\mathcal{Q} = \{q_1, \ldots, q_m\}$, a set of final states $\mathcal{Q}_{\text{Final}} \subseteq \mathcal{Q}$ and a set of rules $R$ of the form:

(type 1) $\phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q$ for $f$ of arity $n$

(type 2) $\phi(X_{q_1}, \ldots, X_{q_m}) \Rightarrow q$,

where in each case, $\phi$ denotes a formula of $FO^{\#}_{\mathcal{M}}$ in the free variables $X_1, \ldots, X_n$ or $X_{q_1}, \ldots, X_{q_p}$.

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

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

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

By definition, if $t \to q$ then $s \to q$ for any $s \equiv_P t$. A multitree is *accepted* iff $t \to_{\mathcal{A}} q$ with $q \in \mathcal{Q}_{\text{Final}}$. The language $\mathcal{L}(\mathcal{A})$ accepted by $\mathcal{A}$ is the set of multitrees accepted by $\mathcal{A}$.

**Example 2.** 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 $R = \{True \Rightarrow a \to q_a, True \Rightarrow b \to q_b, \#(X_{q_a}) < \#(X_{q_b}) \Rightarrow q_S\}$.

Then $b \oplus a \oplus b \to q_S$ since $\begin{cases} b \oplus a \oplus b \equiv_P a \oplus b \oplus b, \\ a \to q_a,\ b \to q_b,\ [\![a]\!] = \{a\} \text{ and } [\![b \oplus b]\!] = \{b, b\}, \\ \models \#([\![a]\!]) < \#([\![b \oplus b]\!]). \end{cases}$

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: $X_1 \neq X_2 \Rightarrow f(q_S, q_S) \to q_S$. □

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

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

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.** 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 $\#(X_{q_a}) < \#(X_{q_b}) \Rightarrow q_S$ by $\#(X_{q_a}) < \#(X_{q_b}) \Rightarrow q_S^{\mathcal{S}}$.

## 5. Properties of multitree automata with constraints

In this section, we shall see that the classical constructions can be lifted to multitree automata in a fairly natural way.

### 5.1. 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 $True \Rightarrow f(\ldots, q_S, \ldots) \to q_S$ and the rules $True \Rightarrow q_S$.

### 5.2. Compositional properties: product, union, intersection

Given $\mathcal{A} = (\mathcal{Q} = \{q_1, \ldots, q_n\}, \mathcal{Q}_{\text{Final}}, R)$, $\mathcal{A}' = (\mathcal{Q}'\{q'_1, \ldots, q'_{n'}\}, \mathcal{Q}'_{\text{Final}}, R')$ 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) $\phi(X_1, \ldots, X_n) \wedge \phi'(X_1, \ldots, X_n) \Rightarrow f((q_1, q'_1), \ldots, (q_n, q'_n)) \to (q, q')$

$$\text{iff} \begin{cases} \phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q \in R, \\ \phi'(X_1, \ldots, X_n) \Rightarrow f(q'_1, \ldots, q'_n) \to q' \in R'. \end{cases}$$

(type 2) $\phi(\bigoplus_{j\in\{1,\ldots,n'\}} X_{(q_1,q'_j)}, \ldots, \bigoplus_{j\in\{1,\ldots,n'\}} X_{(q_n,q'_j)}) \Rightarrow (q, q')$
$\wedge\ \phi'(\bigoplus_{i\in\{1,\ldots,n\}} X_{(q_i,q'_1)}, \ldots, \bigoplus_{i\in\{1,\ldots,n\}} X_{(q_i,q'_{n'})})$

$$\text{iff} \begin{cases} \phi(X_{q_1}, \ldots, X_{q_n}) \Rightarrow q \in R, \\ \phi'(X_{q'_1}, \ldots, X_{q'_{n'}}) \Rightarrow q' \in R'. \end{cases}$$

**Proposition 3.** *The construction of $\mathcal{A} \times \mathcal{A}'$ is done in time* $O(|\mathcal{A}||\mathcal{A}'|)$ *and $t \to_{\mathcal{A}\times\mathcal{A}'} (q, q')$ iff $t \to_{\mathcal{A}} q$ and $t \to_{\mathcal{A}'} q'$.*

**Proof.** The proof is by structural induction on $t$ and is similar to the classical proof for tree automata. □

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

### 5.3. Determinization

An automaton is *deterministic* iff

(i) for any pair of distinct type 1 rules $\phi(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q$, $\phi'(X_1, \ldots, X_n) \Rightarrow f(q'_1, \ldots, q'_n) \to q'$, either $q_i \neq q'_i$ for some $i \in 1, \ldots, n$ or the formula $\phi(X_1, \ldots, X_n) \wedge \phi'(X_1, \ldots, X_n)$ is unsatisfiable,

(ii) for any pair $\phi(X_1, \ldots, X_m) \Rightarrow q$, $\phi'(X_1, \ldots, X_m) \Rightarrow q'$ of distinct type 2 rules the formula $\phi(X_1, \ldots, X_m) \wedge \phi'(X_1, \ldots, X_m)$ is unsatisfiable.

By definition, a deterministic automaton is *non-ambiguous*, i.e. for each $t$ there is at most one $q$ such that $t \to q$.

Now, we show how to build a deterministic automaton $\mathcal{A}_D$ equivalent to a non-deterministic automaton $\mathcal{A} = (\mathcal{Q}_\mathcal{A}, \mathcal{Q}_{\text{Final}}, R)$.

The construction is an adaptation of the classical subset construction: the set of states of the deterministic automaton $\mathcal{A}_D$ is $\mathcal{Q}_D = 2^{\mathcal{Q}_\mathcal{A}}$ and the final states are the states containing

a final state of $\mathcal{A}$. The contruction of the new rules is more complex and proceeds in two steps. Firstly, we compute new conditions that are pairwise exclusive, secondly, we define the new rules (that use the new conditions).

*5.3.1. Type 1 rules*

*Determinization of constraints*: Assume that $f$ is a fixed symbol of arity $n$. Let $\phi_i(X_1, \ldots, X_n)$ for $i = 1, \ldots, m$ be the constraints of type 1 rules for the symbol $f$.

For each $I \subseteq \{1, \ldots, m\}$, let $\psi_I(X_1, \ldots, X_n)$ be defined by

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

By construction $\psi_I(X_1, \ldots, X_n) \wedge \psi_J(X_1, \ldots, X_n)$ is unsatisfiable if $I \neq J$ and $\phi_i(X_1, \ldots, X_n) \Leftrightarrow \bigvee_{i \in I} \psi_I(X_1, \ldots, X_n)$.

Now, we replace each rule $\phi_i(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q$ of $\mathcal{A}$ by the rules $\psi_I(X_1, \ldots, X_n) \Rightarrow f(q_1, \ldots, q_n) \to q$ for $i \in I$. This replacement is performed for each symbol $f$ of the signature.

This yields an automaton equivalent to $\mathcal{A}$, which has the same set of states and final states but has $O(2^{|\mathcal{A}|})$ rules in the worst case.

*The subset construction*: The type 1 rules of the deterministic automaton $\mathcal{A}_D$ are:

$$\psi_I(X_1, \ldots, X_n) \Rightarrow f(Q_1, \ldots, Q_n) \to Q \quad \text{s.t.}$$

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

*5.4. Type 2 rules*

The computation of type 2 rules is more involved.

*Determinization of constraints*: Since the states of the deterministic automaton are subsets of $\mathcal{Q}_\mathcal{A}$ the set of states of $\mathcal{A}$, a constraint is a formula on the variables $X_\emptyset, \ldots, X_J, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}$ where the variable $X_J$ is associated to the state $Q_J = \{q_j \mid j \in J\}$ for $J \subseteq \{1, \ldots, |\mathcal{Q}_\mathcal{A}|\}$.

Let $\phi_1(X_1, \ldots, X_{|\mathcal{Q}_\mathcal{A}|}) \Rightarrow q_1, \ldots, \phi_p(X_1, \ldots, X_{|\mathcal{Q}_\mathcal{A}|}) \Rightarrow q_p$ be the type 2 rules of $\mathcal{A}$. For each $k = 1, \ldots, p$, we define $\psi_k(X_\emptyset, \ldots, X_J, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$ by

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

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

The last point is to eliminate the remaining ambiguities (since the same multitree can satisfy several $\psi_k$ formulas) as we did for type 1 constraints, 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.** *$|\mathcal{A}_D| = \mathrm{O}(2^{2|\mathcal{A}|})$ and $t\to_{\mathcal{A}_D} Q$ iff $Q=\{q \mid t\to_\mathcal{A} q\}$.*

The proof is given in appendix.

### 5.5. 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.

### 5.6. Membership

Given an automaton $\mathcal{A}$, its size $|\mathcal{A}|$ is the number of symbols of its presentation, $\mathrm{O}(C(t))$ 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 multitrees modulo associativity-cmmutativity which can be solved in polynomial time in $|t|$, see [2].

**Proposition 5.** *$t\in\mathcal{L}(\mathcal{A})$ is decidable in time $\mathrm{O}(C(t).|t|.|\mathcal{A}|)$ for a deterministic automaton.*

**Proof.** Since the automaton is deterministic, a node in the multitree is labeled by at most one state. When each son of a node is labeled by a state, for each rule of $\mathcal{A}$ we test in $C(t)$ if the rules is applicable. This is done for each node of $t$, requiring $\mathrm{O}(C(t).(|t|.|\mathcal{A}|))$ time. □

For a non-deterministic automaton, the problem is in NP (if we guess the correct labelling, we just have to check if the constraints are satisfied).

## 6. Decision of emptiness

In this section we assume that $\mathcal{A}$ is a deterministic automaton such that the constraints are formulas of $FO^{\#D}_{\mathcal{M}}$. This class is preserved by all the constructions of the previous section, including determinization. In Section 7, we give some extensions of this class ensuring closure under determinization and boolean operations for which emptiness is still decidable.

The algorithm for deciding emptiness of $\mathcal{L}(\mathcal{A})$ is similar to the algorithm for 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 $X_1 \neq X_2 \Rightarrow f(q,q) \to q'$, we cannot 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, when two multitrees reach $q$, we also have that (at least) 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 the decidability results on $FO^{\#}_{\mathcal{M}}$ prove that they are effectively computable.

For simplicity, we assume that for each $q \in \mathcal{Q}_{\mathcal{T}}$, the corresponding type 1 rules have the form $\phi(X_{q_1}, \ldots, X_{q_n}) \Rightarrow f(q_1, \ldots, q_n) \to q$ for the same $f$. If it is not the case, we split $q$ into $q_f, q_g, \ldots$ and modify the rules accordingly. This preserves determinism and does not change the constraints. In the following, we use the word *element* to denote a equivalence class of $\equiv_P$.

### 6.1. Formulas for states

Let $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_{\text{Final}}, R)$ 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 formulas which ensure that a state $q$ can be reached by some element $X$, when we have already computed $Z_1$ a set of elements of $\mathcal{T}$ reaching $q_1$,..., $Z_p$ a set of elements 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 $setof(X)$ for the multiset equal to the set of distinct elements of $X$ (this can be defined in $FO_{\mathcal{M}}$, see Section 3).

*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 elements reaching $q_1$,...,$Z_p$ a set of elements reaching $q_p$ and it is defined by

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

This formula is a formula of $FO^{\#D}_{\mathcal{M}}$.

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

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

where $Sing(X)$ is the $FO^{\#D}_{\mathcal{M}}$ formula that states that $X$ is a singleton. Note that the definition is consistent since $\psi_q$ is already defined for $q \in \mathcal{Q}_{\mathcal{S}}$. Let $\phi_i(X_{q^i_1}, \ldots, X_{q^i_n}) \Rightarrow f(q^i_1, \ldots, q^i_n) \to q$ for $i = 1, \ldots, l$ be the type 1 rules for $q$ (remember that when $q$ is fixed, then the symbol $f$ is fixed). The formula $\psi_q(Z_1, \ldots, Z_p, X_1, \ldots, X_n)$ for $q \in \mathcal{Q}_{\mathcal{T}}$ is

defined by

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

Again this formula is a formula of $FO^{\#D}_{\mathcal{M}}$.

### 6.2. Bounds for states

Now, we compute a minimal bound on the number of distincts elements in the $Z_i$'s that are needed to ensure the existence of an element reaching a state $q$. For a state $q \in \mathcal{Q}_{\mathcal{S}}$ we compute

$$\mathcal{M}_q = Min\{\#_D(\vec{Z}) \mid \exists X\ \psi_q(Z_1,\ldots,Z_p,X)\}).$$

Then $\mathcal{M}_q$ is a finite set of tuples $(d_1,\ldots,d_p)$ and we set $D_q$ to be the maximal values of $d_i$'s for all tuples in the set ($+\infty$ if the formula is unsatisfiable).

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

$$\mathcal{M}_q = Min\{\#_D(\vec{Z}) \mid \exists X_1,\ldots,X_n\ \psi_q(Z_1,\ldots,Z_p,X_1,\ldots,X_n)\}.$$

Then $\mathcal{M}_q$ is a finite set of tuples $(d_1,\ldots,d_p)$ and we set $D_q$ to be the maximal values of $d_i$'s for all tuples in the set ($+\infty$ if the formula is unsatisfiable). This value bounds the minimal number of distinct elements of $\mathcal{T}$ that must reach $q_1$,...,$q_p$ to allow the construction of a element reaching $q$ (if there exists one). Let $D$ be the maximum of the finite $D_q$'s.

Now we compute a bound on the multiplicities of elements of $Z_i$ used in $X$ or $X_1,\ldots,X_n$, with the additional constraints that (i) $Z_1,\ldots,Z_p$ have less than $D+1$ distinct elements and (ii) we can construct $D+1$ elements reaching $q$.

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

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

This is a formula of $FO^{\#D}_{\mathcal{M}}$, and we can compute

$$\mathcal{M}_q^k = Min\{\#(\vec{X}) \mid \exists Z_1,\ldots,Z_p\ \rho_q^k(Z_1,\ldots,Z_p,X_1,\ldots,X_k) \wedge \bigwedge_{i=1}^{i=p} \#_D(Z_i) \leqslant D\}.$$

$\mathcal{M}_q^k$ is a finite set of tuples $(m_1,\ldots,m_k)$. We set $M_q^k$ to be the maximal value of the $m_i$'s for all tuples in the set.

Since the multiplicity of an element of a multiset is less than the cardinality of the multiset, this gives an upper 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 add the constraint that the number of elements of each $Z_i$ is bounded by $D$.

*Case of a state* $q \in \mathcal{Q}_{\mathcal{T}}$: 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 tuples 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

$$\rho_q^k\left(Z_1, \ldots, Z_p, \underbrace{X_1^1, \ldots, X_n^1}_{first\ tuple}, \ldots, \underbrace{X_1^k, \ldots, X_n^k}_{kth\ tuple}\right),$$

which states that we can compute $k$ distinct elements $f(X_1^1, \ldots, X_n^1)$,..., $f(X_1^k, \ldots, X_n^k)$ reaching $q$ from $Z_1, \ldots, Z_p$, by

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

This is a formula of $FO_{\mathcal{M}}^{\#D}$, and we can compute

$$\mathcal{M}_q^k = Min\{\#(\vec{X}) \mid \exists Z_1, \ldots, Z_p \quad \rho_q^k(Z_1, \ldots, Z_p, X_1^1, \ldots, X_n^1, \ldots, X_1^k, \ldots, X_n^k) \wedge \textstyle\bigwedge_{i=1}^{i=p} \#(Z_i) \leqslant D\}.$$

$\mathcal{M}_q^k$ is a finite set of tuples $(m_1^1, \ldots, m_n^k)$. We set $M_q^k$ to be the maximal value of the $m_i$'s for all tuples in the set. As discussed before, this yields a bound on the multiplicities of elements of $Z_i$'s occurring in the $X_j^l$'s.

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

**Remark 1.** If the constraints $\phi_i$'s belong to $\exists FO_{\mathcal{M}}^{\#}$, then the formulas $\psi_k$, $\rho_q^k$ and the formulas used in the computation of $\mathcal{M}_q^k$ are also in $\exists FO_{\mathcal{M}}^{\#}$.

### 6.3. The algorithm

Let $D$ and $M$ be defined as above 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 approximates 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 $M$. The notation $\#_M(t)$ denotes the maximal multiplicities of elements in $t$. The notation $|\mathcal{L}|_P$ denotes the number of distinct equivalence classes of $\equiv_P$ in the set $\mathcal{L}$.

[4] One step does not mean one application of rule, but *label the root of a multitree by some state when all the sons are labeled 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** $\mathcal{L}_i^{m+1} = \mathcal{L}_i^m \cup \{t \mid t \to q_i$ for $t \equiv_P f(t_1, \ldots, t_n)$ with $t_1, \ldots, t_n \in \mathcal{L}_j^m$
      or $t \equiv_P \bigoplus_j t_j$ with $t_j \in \mathcal{L}_j^m$, $\#_D(t_j) \leqslant D$
      and $\#_M(t_j) \leqslant M\}$
    **if** $|\mathcal{L}_i^{m+1}|_P \geqslant D + 1$ **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 $\mathcal{L}_i = \mathcal{L}_i^m$

**Proposition 6.** *The algorithm terminates and $q_i$ is reachable iff $\mathcal{L}_i \neq \emptyset$.*

The proof is given in appendix.

**Theorem 1.** *Let $\mathcal{A}$ be an multitree automaton with constraints in $FO_{\mathcal{M}}^{\#D}$, then the problem $\mathcal{L}(\mathcal{A}) = \emptyset$ is decidable.*

**Proof.** Since the constraints are in $FO_{\mathcal{M}}^{\#D}$, we can compute an equivalent deterministic automaton with constraints in $FO_{\mathcal{M}}^{\#D}$ and apply the reachability algorithm to test emptiness. □

## 7. Decidable extensions

Our results on the decidability of $FO_{\mathcal{M}}^{\#}$ are too weak to allow any constraints in this class. But looking look closely at (i) the decidability results of Appenidx B and (ii) the formulas involved in boolean operations, one can extend the current class to get classes which are closed under boolean operations, have an algorithm to decide emptiness, and use (in a restricted way) the counting operation # (and not only $\#_D$).

### *7.1. Quantifier-free $FO_{\mathcal{M}}^{\#}$ formulas in type 1 rules*

The boolean combination of such constraints always yield a formula in the same class. For type 1 rules, the new constraints constructed for determinization are also in this class. The formulas built in the decision of emptiness proof still yield formulas of $\exists FO_{\mathcal{M}}^{\#}$. Therefore the emptiness algorithm is still correct.

We cannot extend this to type 2 rules since the determinization process introduces existential and universal variables (via negation of the existential variables).

### 7.2. Counting elements in type 2 rules

The second extension requires that type 2 rules constraints have the form $\varphi(\#(X_{q_1}), \ldots, \#(X_{q_p}))$ where $\varphi$ is a Presburger arithmetic formula. Contrary to the previous case, the determinization and the product construction yields constraints that do not have the required form. We show that they can be replaced by equivalent constraints of the right class.

*Product*: Type 2 rules arising in products are

$$\begin{array}{ll} & \phi(\bigoplus_{j\in\{1,\ldots,n'\}} X_{(q_1,q'_j)}, \ldots, \bigoplus_{j\in\{1,\ldots,n'\}} X_{(q_n,q'_j)}) \quad \Rightarrow (q,q'),\\ \wedge & \phi'(\bigoplus_{i\in\{1,\ldots,n\}} X_{(q_i,q'_1)}, \ldots, \bigoplus_{i\in\{1,\ldots,n\}} X_{(q_i,q'_{n'})}).\end{array}$$

Since $\phi(\vec{X}) \equiv \varphi(\#(\vec{X}))$ and $\phi'(\vec{X}) \equiv \varphi'(\#(\vec{X}))$, this is equivalent to

$$\begin{array}{ll} & \varphi(\Sigma_{j\in\{1,\ldots,n'\}}\#(X_{(q_1,q'_j)}), \ldots, \Sigma_{j\in\{1,\ldots,n'\}}\#(X_{(q_n,q'_j)})) \Rightarrow (q,q')\\ \wedge & \varphi'(\Sigma_{i\in\{1,\ldots,n\}}\#(X_{(q_i,q'_1)}), \ldots, \Sigma_{i\in\{1,\ldots,n\}}\#(X_{(q_i,q'_{n'})}))\end{array}$$

which has the right form.

*Determinization*: The determinization process constructs formulas $\psi_k(X_\emptyset, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$ defined by

$$\bigwedge_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \left( \exists X_j^J \; X_J = \bigoplus_{j\in J} X_j^J \wedge \phi_k \times \left( \bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} X_1^J, \ldots, \bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} X_{|\mathcal{Q}_\mathcal{A}|}^J \right) \right),$$

where $\phi_k(Y_1, \ldots, Y_{|\mathcal{Q}_\mathcal{A}|})$ is some $\varphi_k(\#(Y_1), \ldots, \#(Y_{|\mathcal{Q}_\mathcal{A}|}))$. We claim that this is equivalent to $\psi'_k(X_\emptyset, \ldots, X_{\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}})$ defined by

$$\bigwedge_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} (\exists x_j^J \; \#(X_J) = \Sigma_{j\in J} x_j^J \wedge \varphi_k(\Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} x_1^J, \ldots, \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} x_{|\mathcal{Q}_\mathcal{A}|}^J)).$$

**Proof.** $\Rightarrow$ Direction.

Let $\mathcal{X}_J, \ldots,$ an assignment to the variables $X_J, \ldots,$ such that the first formula holds. By definition there exist $\mathcal{X}_j^J$ such that $\mathcal{X}_J = \bigoplus_{j\in J} \mathcal{X}_j^J$ and
$\models \phi_k(\bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \mathcal{X}_1^J, \ldots, \bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \mathcal{X}_{|\mathcal{Q}_\mathcal{A}|}^J)$

$\mathcal{X}_J = \bigoplus_{j\in J} \mathcal{X}_j^J$ implies $\#(\mathcal{X}_J) = \Sigma_{j\in J}\#(\mathcal{X}_j^J)$

Assigning $\alpha_j^J = \#(X_j^J)$ to $x_j^J$ and replacing $\phi_k$ by its definition yields
$\#(\mathcal{X}_J) = \Sigma_{j\in J}\alpha_j^J$ and $\models \varphi_k(\Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}\alpha_1^J, \ldots, \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}\alpha_{|\mathcal{Q}_\mathcal{A}|}^J)$

$\Leftarrow$ Direction.

Let $\mathcal{X}_J, \alpha_j^J, \ldots,$ be such that $\models \varphi_k(\Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}\alpha_1^J, \ldots, \Sigma_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}}\alpha_{|\mathcal{Q}_\mathcal{A}|}^J)$ and $\#(\mathcal{X}_J) = \Sigma_{j\in J}\alpha_j^J$.

Since $\#(\mathcal{X}_J) = \Sigma_{j\in J}\alpha_j^J$, there exists a decomposition $\mathcal{X}_J = \bigoplus_{j\in J} \mathcal{X}_j^J$ such that $\alpha_j^J = \#(\mathcal{X}_j^J)$.

Since $\phi_k(Y_1, \ldots, Y_{|\mathcal{Q}_\mathcal{A}|}) \equiv \varphi_k(\#(Y_1), \ldots, \#(Y_{|\mathcal{Q}_\mathcal{A}|}))$ we have $\vDash \phi_k(\bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \mathcal{X}_1^J, \ldots, \bigoplus_{J\subseteq\{1,\ldots,|\mathcal{Q}_\mathcal{A}|\}} \mathcal{X}_{|\mathcal{Q}_\mathcal{A}|}^J)$ which proves the reverse implication. □

Emptiness works as previously since the formulas that we use to compute the bounds needed in the reachability algorithm belong to $\exists FO^{\#}_{\mathcal{M}}$.

**Remark 2.** One can slightly extend these classes by allowing type 2 rules which are conjunction of the allowed constraints and of a formula $\phi(\bigoplus_{q\in\mathcal{Q}} X_q)$ if $\phi(X)$ is a quantifier-free formula of $FO^{\#}_{\mathcal{M}}$ in the free variable $X$.

This improvement could be used to accept only multisets such that the multiplicities of elements satisfy some Presburger constraint.

## 8. Comparison with other classes of tree languages

In this section, we show that *CMTL* embeds many classes of tree automata that have been proposed for a whole variety of applications.

### *8.1. Language with equality/disequality constraints between brothers*

A tree automaton with equality/disequality constraint between brothers is a tree automaton $(\mathcal{Q}, \mathcal{Q}_F, R)$ where the rules have the form

$$\bigwedge_{(i,j)\in E} X_i = X_j \wedge \bigwedge_{(k,l)\in D} X_k \neq X_l \Rightarrow f(q_1, \ldots, q_n) \to q.$$

The transition relation is the smallest relation such that: if $\bigwedge_{(i,j)\in E} X_i = X_j \wedge \bigwedge_{(k,l)\in D} X_k \neq X_l \Rightarrow f(q_1, \ldots, q_n) \to q \in R$, if $t_1 \to q_1, \ldots, t_n \to q_n, t_i = t_j$ for $(i, j) \in E, t_k \neq t_l$ for $(k, l) \in D$ then $f(t_1, \ldots, t_n) \to q$.

These automata recognize sets of trees (not multitrees) since the signature contains only free symbols. They have been studied in [4] where it is proved that the class of languages accepted by these tree automata $L(AWEDC)$ is closed under boolean operations and has a decidable emptiness problem. They have been used to obtain or improve decision results of many problems (mainly in rewriting, constraint solving and logic). Since the rules of these automata are instances of type 1 rules when no associative-commutative symbol occur, these languages are particular instances of constrained multitree languages.

**Proposition 7.** $L(AWEDC) \subseteq CMTL$.

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 equality/disequality constraints have been studied, leading to reduction automata [6] or automata allowing only a bounded number of equality tests in a run [8].

Such constraints are inherently different from $FO_{\mathcal{M}}$ constraints and cannot be combined with them.

### 8.2. Closure of regular tree languages

The class of regular tree languages is not closed under associativity-commutativity (AC for short) but the class *CMTL* is closed under associativity-commutativity. Therefore a natural question is to ask if the closure of a regular-tree language under associativity-commutativity is necessarily in *CMTL*?

The reader is referred to [1] for details on flattening, AC equivalence, permutative equivalence,.... Let $\oplus$ be an AC symbol, $f, g, \ldots$ be free symbols. The flattening operation transforms the tree $s$ into the multitree $flat(s)$:

(1) Rewrite $s$ with the rule $x \oplus (y \oplus z) \to (x \oplus y) \oplus z$ until no rewriting can occur,
(2) $flat(s) = flat((..(s_1 \oplus s_2) \oplus \ldots) \oplus s_n) = flat(s_1) \oplus \ldots \oplus flat(s_n)$ (the root of $s_i$ is not $\oplus$).

For instance $flat((a \oplus g(b)) \oplus (a \oplus b)) = a \oplus g(b) \oplus a \oplus b$. The permutative equivalence $\equiv_P$ on flattened terms is the equivalence that states that the order of the arguments of $\oplus$ is irrelevant. Given a tree language $\mathcal{L}$, its closure $Cl(\mathcal{L})$ is the set of terms equivalent to a term of $\mathcal{L}$ modulo AC. The set $Cl_P(flat(\mathcal{L}))$ is the set of multitrees $\mathbf{s}$ such that $\mathbf{s} \equiv_P flat(s)$ with $s \in \mathcal{L}$ and the equality $flat(Cl(\mathcal{L})) = Cl_P(flat(\mathcal{L}))$ holds [1]. Given a finite (ordered) alphabet $q_1, \ldots, q_n$, we recall that the Parikh mapping of a word $w$ is the tuple $(\#(q_1), \ldots, \#(q_n))$ where $\#(q_i)$ is the number of occurrences of the letter $q_i$ in $w$. The set of Parikh mappings of the words of a language is the Parikh mapping of the language. The Parikh mapping of a context-free language is a semilinear set (see [14] for details), hence it is definable by a Presburger formula.

Let *REG* denote the class of regular tree languages.

**Proposition 8.** $flat(Cl(REG)) \subseteq CMTL$.

**Proof.** Let $\mathcal{L} \in REG$ and let $\mathcal{A}$ be a tree automaton such that $\mathcal{L} = \mathcal{L}(\mathcal{A})$. We can assume that $\mathcal{Q}$ the set of states of $\mathcal{A}$ is separated into two sets $\mathcal{Q}_{\mathcal{T}}$ and $\mathcal{Q}_{\mathcal{S}}$ such that the rules of $\mathcal{A}$ have the form $f(q_1, \ldots, q_n) \to q$ with $q \in \mathcal{Q}_{\mathcal{T}}$ or $q' \oplus q'' \to q$ with $q \in \mathcal{Q}_{\mathcal{S}}$.

*The multitree automaton* $\mathcal{B}$: Firstly, we define the multitree automaton $\mathcal{B}$ that must accept $flat(Cl(\mathcal{L}(\mathcal{A})))$. We keep the same set of states, final states and type 1 rules are the rules $f(q_1, \ldots, q_n) \to q$ of $\mathcal{A}$ for $f \neq \oplus$.

Type 2 rules require an auxiliary computation using a context-free word grammar on the alphabet $\mathcal{Q}_{\mathcal{T}}$, with non-terminals $X_q$ ($q \in \mathcal{Q}$) and rules $X_q \to X_{q'} X_{q''}$ if there is a rule $q \oplus q'' \to q$ in $\mathcal{A}$. Moreover we also have the rules $X_q \to q$ if $q \in \mathcal{Q}_{\mathcal{T}}$. $L_q$ is the word language generated by this grammar when $X_q$ is the axiom. Let $Pa(L_q)$ be the Parikh mapping of $L_q$ and let $Pa_q$ be the corresponding Presburger formula.

By definition, type 2 rules of $\mathcal{B}$ are $Pa_q(\#(X_{q_1}), \ldots, \#(X_{q_n})) \Rightarrow q$.

Proposition 8 is a consequence of the following lemma:

**Lemma 1.** $flat(Cl(\mathcal{L}(\mathcal{A}))) = \mathcal{L}(\mathcal{B})$

The proof is given in the appendix. □

### 8.3. Regular AC equational tree automata

A regular $E$ equational tree automaton is an ordinary tree automaton $\mathcal{A} = (\mathcal{Q}, \mathcal{Q}_F, R)$, but the transition relation is the rewrite relation defined by the rules $f(q_1, \ldots, q_n) \to q \in R$ and the rewriting is performed modulo the equational theory $E$. A term is accepted iff it rewrites to a final state. For instance given $\mathcal{A} = (\{q_a, q_b\}, \{q_b\}, a \to q_a, b \to q_b, f(q_b, q_a) \to q_b\})$ with $f$ an *AC* operator, the rewrite sequence
$f(f(a,b),a) \to f(f(q_a,q_b),a) \equiv_{AC} f(f(q_b,q_a),a) \to f(q_b,a) \to f(q_b,q_a) \to q_b$
shows that $f(f(a,b),a)$ is accepted by $\mathcal{A}$.

Regular $E$ equational tree automata have been introduced and studied by Ohsaki, see [26]. When the theory $E$ consists of linear axioms (which is the case for the AC axioms) the language accepted by a regular $E$ equational tree automaton $\mathcal{A}$ is the closure modulo $E$ of the language accepted by the tree automaton $\mathcal{A}$ (considered as a standard tree automaton). Therefore, if *REGAC* denotes the set of languages accepted by *AC*, and *flat* denotes the flattening operation on terms defined in the previous section, Proposition 8 yields that:

**Proposition 9.** $flat(REGAC)) \subseteq CMTL$.

In [26], the regular AC equational languages are generalized to AC equational tree languages. These languages are accepted by automata that allow also rules of the form $f(q_1, \ldots, q_n) \to f(q'_1, \ldots, q'_n)$. These automata are strictly more expressive than regular ones but emptiness remains decidable. It is not clear whether AC equational languages are included in *CMTL*.

### 8.4. Rational tree languages

In [7] Colcombet considers tree languages on a signature $\Sigma$ consisting of one constant 0, unary symbols $a(\_), b(\_), \ldots$, and an associative-commutative operator $\oplus$. Terms are flattened and considered modulo associativity-commutativity of $\oplus$ and have the form 0 or $a(t)$ or $t_1 \oplus \ldots \oplus t_n$. He defines a multiset automaton $\mathcal{A}$ by $\mathcal{A} = (\mathcal{Q}, (R_{a,q})_{a\in\Sigma, q\in\mathcal{Q}}, F)$ where $\mathcal{Q}$ is a finite set of states, $R_{a,q}$ and $F$ are rational languages of the commutative monoid generated by $\mathcal{Q}$, with the constant 0 as neutral element (We recall the class of rational languages of a finitely generated monoid $(\mathcal{Q}, \cdot, 0)$ is the smallest class of languages which is closed under $\cdot$, union and iteration $*$ and contains the emptyset and the languages consisting of one element). The transition relation is defined by
if $t_1 \to q_1, \ldots, t_n \to q_n$ then $t_1 \oplus \ldots \oplus t_n \to q_1 \oplus \ldots \oplus q_n$
if $w \in R_{a,q}, t \to w$ then $a(t) \to q$
A language is rational iff it is accepted by a multiset automaton. [5] For instance the language $\{t \mid t \equiv_P n.a(0) + n.b(0)\}$ is rational. These languages are closed under the boolean operations.

We show that a multiset automaton $\mathcal{A} = (\mathcal{Q}, (R_{a,q})_{a\in\Sigma, q\in\mathcal{Q}}, F)$ can be encoded into multitree automata. From [15], we know that $w$ belongs to a rational language $R$ on the

[5] Actually, Colcombet defines rational languages by another kind of tree automata and shows that it is equivalent to the definition with multiset automata.

commutative monoid generated by $\mathcal{Q}$ iff the Parikh mapping of $w$ satisfies some Presburger formula $\varphi_R$ (that can be effectively computed from $R$). The multitree automaton $\mathcal{B}$ is defined by

- the set of states is $\mathcal{Q} \cup \{q'_{a,q} \mid \in \Sigma, q \in \mathcal{Q}\} \cup \{q_F\}$,
- the set of final state is $\{q_F\} \cup \{q \in Q \mid q \in F\}$
- for each $R_{a,q}$ we introduce the rules
  - $\varphi_{a,q}(\#(X_{q_1}), \ldots, \#(X_{q_{|\mathcal{Q}|}})) \Rightarrow q_{R_{a,q}}$ where $\varphi_{a,q}$ is the Presburger formula associated to $R_{a,q}$,
  - $True \Rightarrow a(q_{R_{a,q}}) \to q$,
  - $True \Rightarrow a(0) \to q$, if $0 \in R_{a,q}$, $True \Rightarrow 0 \to q_F$ if $0 \in F$,
  - $\varphi_F(\#(X_{q_1}), \ldots, \#(X_{q_{|\mathcal{Q}|}})) \Rightarrow q_F$ where $\varphi_F$ is the Presburger formula associated to $F$.

Let $\to_{\mathcal{A}}$ (resp., $\to_{\mathcal{B}}$) denote the transition relation of $\mathcal{A}$ (resp., $\mathcal{B}$). We have that $q_1 \oplus \ldots \oplus q_n \in R_{a,q}$ iff the Parikh mapping of $q_1 \oplus \ldots \oplus q_n \in R_{a,q}$ satisfies $\varphi_{a,q}$.

By definition of transition relations:

- $t_1 \oplus \ldots \oplus t_n \to_{\mathcal{A}} w \in R_{a,q}$ iff $t_1 \oplus \ldots \oplus t_n \to_{\mathcal{B}} q_R$,
- $t \to_{\mathcal{A}} w \in R_{a,q}, a(t) \to_{\mathcal{A}} q$ iff $t \to_{\mathcal{B}} q_R, a(t) \to_{\mathcal{A}} q$,
- $t_1 \oplus \ldots \oplus t_n \to_{\mathcal{A}} w \in F$ iff $t_1 \oplus \ldots \oplus t_n \to_{\mathcal{B}} q_F$.

This yields that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$ which allows to state the next inclusion:

**Proposition 10.** $RATL \subseteq CMTL$.

### 8.5. *Multitree automata with arithmetic constraints*

Tree automata with arithmetic constraints [21] 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 cannot be reversed. The states of an automaton with arithmetical constraints $\mathcal{A}$ are divided in several sorts that we simplify into unprimed, primed, double primed states. Some unprimed on double primed states can be final. The rules of the automata are $f(r_1, \ldots, r_1) \to q$ for a free symbols $f$ ($r_i$ are states that can be unprimed or double primed), $\phi(N) : N.q \to q'$ and $\psi(\#(q'_1), \ldots, \#(q'_m)) \to q''$ where $\psi$ and $\phi$ are Presburger formula, and $\#(q)$ denotes the number of occurrences of $q$. The set of normalized multisets reaching a state $q''$ have the form $\{\underbrace{n^1_1.e^1_1, \ldots, n^1_{k_1}.e^1_{k_1}}_{\models\phi_1(n^1_i)}, \ldots, \underbrace{n^m_1.e^m_1, \ldots, n^m_{k_m}.e^m_{k_m}}_{\models\phi_m(n^m_i)}\}$ where $e^1_1 \to q_1, \ldots, e_{k_1} \to q_1, \ldots$, $e^m_1 \to q_m, \ldots, e^m_{k_m} \to q_m$ and $\models\psi(k_1, \ldots, k_m)$ for some Presburger formula $\psi$.

An automaton $\mathcal{B}$ accepting the same language has $True \vdash f(r_1, \ldots, r_1) \to q$ as type 1 rules and $\phi(X_{q_1}, \ldots, X_{q_m}) \to q''$ as type 3 rules where $\phi(X_{q_1}, \ldots, X_{q_m})$ is the formula:

$$X = X_{q_1} \oplus \ldots \oplus X_{q_m} \wedge \bigwedge_{i=1,\ldots,m} Mult(\phi_i, X_{q_i}) \wedge \psi(\#_D(X_{q_1}), \ldots, \#_D(X_{q_m}))$$

where $Mult(\phi, X)$ is the $FO^{\#D}_{\mathcal{M}}$ formula stating that the multiplicity of each element of $X$ satisfies $\phi$ (cf. Section 3). By definition of $\phi$, a multitree reaches $q''$ (resp., $q$) in $\mathcal{A}$ iff it is the normalization of a multitree reaching $q''$ (resp., $q$) in $\mathcal{B}$.

Therefore denoting by $L(TAC)$ the set of languages accepted by tree automata with arithmetic constraints, the following proposition holds (up to normalization):

**Proposition 11.** $L(TAC) \subseteq CMTL$.

Ref. [19] defines a class of multitree automata which is strictly included in *CMTL*. Type 1 constraints rules are only boolean combinations of equations where one side is a variable (e.g. $X_i = \bigoplus_{j\in\{1,\ldots,n\}} X_j$) 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 yields bad complexity bounds.

### *8.6. Feature tree automata*

Feature tree automata have been introduced by Podelski and Niehren [25] to provide a notion of recognizable languages for feature trees. Given a set $\mathcal{S}$ of constructors $\{a, b, \ldots\}$, a set $\mathcal{F}$ of features $\{F, G, \ldots\}$ features trees are multitrees such that the nodes are labeled by constructors edges are labeled by features and for each node, for each feature $F$, there is at most one outgoing edge labeled by $F$. For instance $(a, \{F, (b, \emptyset), G, (a, \emptyset)\})$ is the feature tree with root labeled by $a$, one outgoing edge labeled by $F$, one outgoing edge labeled by $G$. The first edge joins the root to a terminal node labeled by $b$ and the second one joins the root to a terminal node labeled by $a$.

In [25], recognizable feature tree languages are characterized by the following constraints ($X$ denotes a multitree, $S$, $T$ are subsets of $\mathcal{S}$, $\mathcal{F}$ respectively):

$C(X)$ ::= #({edges labeled by $f \in T$ from the root of $X$ to an element labeled by $a \in S$}) belongs to a given semilinear set $\mathcal{L}$ of $\mathbb{N}$
| the root of X is labeled by a symbol of S
| $C(X) \wedge C(X)$
| $C(X) \vee C(X)$.

There are several ways to encode feature trees as multitrees. Given a feature tree, we relabel the node as follows: a node labeled by symbol $a$ with on-going edge labeled by $F$ is labeled by $(a, F)$. The root is labeled by $(a, \perp)$. The restriction on the labeling of outgoing edges ensures that in a multitree all roots of elements are distinct.

To show that recognizable feature tree languages are in *CMTL*, we express counting constraints in $FO^{\#D}_{\mathcal{M}}$. Since our constraints are closed under boolean operations, we simply do this for the first counting constraint. This constraint is equivalent to the $FO^{\#D}_{\mathcal{M}}$ formula

$$X = X_{(S,T)} \oplus X_{\neq(S,T)} \wedge \#_D(X_{(S,T)}) \in \mathcal{L},$$

where $X_{(S,T)}$ (resp., $X_{\neq(S,T)}$) denotes multitrees where the root of each element is labeled by $(a, F)$ with $a \in S, F \in \mathcal{F}$ (resp. $(a, F) \notin (S, T)$) which can be recognized by an automaton without constraints. We can use $\#_D$ instead of # because of the restriction on outgoing edges in feature trees. However, if we drop this restriction and use #, we still stay in a decidable class described in 7.

**Proposition 12.** $L(FTA) \subseteq CMTL$.

Feature trees are defined also for an infinite set of features and constructors. To get the inclusion in this case, we must extend multitrees in this framework and devise automata that can deal with finite or co-finite sets of function symbols which can be done in a fairly standard way.

## 9. Conclusion

We have presented a general framework for constrained multitree automata. The key of our approach is the use of a rich set of constraints which provide nice properties for the resulting class. This work suggest several possible developments.

The first one is to settle the question of the decidability of $FO^{\#}_{\mathcal{M}}$. A positive answer will immediately yields that emptiness is decidable for automata with constraints in $FO^{\#}_{\mathcal{M}}$, since emptiness proof only relies on the bounds for multiplicities and number of distinct factors (but we need the decidability of $FO^{\#}_{\mathcal{M}}$ to compute these bounds).

The constraints that we use for multisets do not impose any relation on the components of multisets. For instance, we cannot say that $X$ is a multiset such that all its elements are greater than the elements of another multiset $Y$. This constraints are useful for several applications and it is natural to ask whether the framework can be extended to embed such constraints. The first positive result in this direction is the decidability of $FO_{\mathcal{M}}$ augmented with a total ordering $>$ on the elements $e_i$'s. Again this is used by an encoding into Skolem arithmetic enriched by $>_P$, i.e. the usual natural order restricted to prime numbers, which is decidable [22]. Therefore all our constructions work with these constraints, but we cannot lift our algorithm for deciding emptiness to this case. The reason is that the satisfiability relation is no longer invariant under permutation of elements. Therefore some new algorithm is needed in this case.

A fruitful approach to tree automata is to consider them as Horn clauses, and several propositions have been done to extend the class of clauses (using push and pop clauses for instance). Combined with associative-commutative axioms, this defines devices similar to multitrees see [16,28] which may have good properties. It remains to see how far we could go in combining this approach and the use of constraints.

The complexity for some well-chosen subclasses is an also issue. The algorithms that we have presented for constraints and emptiness are not tractable. But some subclasses may have a good behaviour and be worthwhile for applications (that we have not considered in this paper).

## Appendix A.

### A.1. Correctness of the determinization algorithm

We give the proof of Proposition 4.

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

*Case* $t \equiv_P f(t_1, \ldots, t_n)$.

Since the constraints of rules are either identical or pairwise incompatible, the proof is similar to the correctness proof for the determinization of tree automata.

*Case* $t \equiv_P t_1 \oplus \ldots \oplus t_n$.

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

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

where the decomposition in the $T_J$'s 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[6] $T_J \equiv_P \bigoplus_{j \in J} T_j^J$ such that

$$\models \phi_i \left( \left[\!\!\left[ \bigoplus_{J \subseteq \{1,\ldots \mathcal{Q}_{\mathcal{A}}\}} T_1^J \right]\!\!\right], \ldots, \left[\!\!\left[ \bigoplus_{J \subseteq \{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}} T_{|\mathcal{Q}_{\mathcal{A}}|}^J \right]\!\!\right] \right)$$

  This proves that $t \equiv_P \bigoplus_{j=1}^{j=|\mathcal{Q}_{\mathcal{A}}|} \bigoplus_{J \subseteq \{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}} T_j^J \to_{\mathcal{A}} q_i$.
  For $i \notin I$, we cannot find any such decomposition by definition of $\neg\psi_i$ (otherwise $t \models \psi_i$ for $i \notin I$ and $t \not\to_{\mathcal{A}_D} Q_I$), which proves that $t \not\to_{\mathcal{A}} q_i$.
  Combining the two previous results, we get $Q_I = \{q_i \mid t \to_{\mathcal{A}} q_i\}$.
- Conversely, let $Q = \{q \mid t \to_{\mathcal{A}} q\} = Q_I$ for some $I$. According to our notation,

$$t \equiv_P \bigoplus_{J \subseteq \{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}} T_J \text{ where } T_J = \bigoplus_{t_j \to_{\mathcal{A}} J} t_j.$$

  By definition of $Q_I$, for $i \in I$, there is a decomposition[7] $T_J \equiv_P \bigoplus_{j \in J} t_j^J$ s.t.

$$\models \phi_i \left( \left[\!\!\left[ \bigoplus_{J \subseteq \{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}} t_1^J \right]\!\!\right], \ldots, \left[\!\!\left[ \bigoplus_{J \subseteq \{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}} t_{|\mathcal{Q}_{\mathcal{A}}|}^J \right]\!\!\right] \right).$$

  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 $t \to_{\mathcal{A}} q_i$), therefore $\not\models \psi_i([\![T_\emptyset]\!], \ldots, T[\![_{\{1,\ldots,|\mathcal{Q}_{\mathcal{A}}|\}}]\!])$.
  Combining the two properties we get that $t \to_{\mathcal{A}_D} Q_I$. □

### *A.2. Termination and correctness of the reachability algorithm*

We give the proof of Proposition 6.

**Proof.** *Termination*: by construction $\mathcal{L}_i^m$ is closed under the permutative equivalence $\equiv_P$. At each iteration of the repeat loop, either the state $q_i$ is marked and $\mathcal{L}_i^m = \mathcal{L}_i^{m+1} = \ldots$ or

[6] This decomposition depends on $i$ but we do not write this explicitly for simplicity

[7] Again, we do not mention explicitly that the decomposition depends on $i$

$\mathcal{L}_i^{m+1}$ contains one new element which is not equivalent to any element of $\mathcal{L}_i^m$, then $|\mathcal{L}_i^{m+1}|_P > |\mathcal{L}_i^m|_P$. Since $q_i$ is marked when $|\mathcal{L}_i^m|_P \geqslant D+1$, all states are marked after at most $(D+1)|\mathcal{Q}|$ iterations.

*Correctness*: The determinism of the algorithm ensures that $\mathcal{L}_i^m \cap \mathcal{L}_j^m = \emptyset$ if $i \neq j$. Let us define

$$L_i^0 = \emptyset$$
$$L_i^{m+1} = L_i^m \cup \{t \mid t \to q_i \ for \ t \equiv_P f(t_1,\ldots,t_n) \ and \ t_i \in L_i^m, i=1,\ldots,n \ or \ t \equiv_P t_1 \oplus \ldots \oplus t_l \ and \ t_i \in L_{j_i}^m, i=1,\ldots,l\}.$$

Then we prove that

**Lemma A.1.** $\forall m, i, \mathcal{L}_i^m \subseteq L_i^m \ and \ (\mathcal{L}_i^m = L_i^m \ or \ |\mathcal{L}_i^m|_P > D)$.

**Proof.** The inclusion $\mathcal{L}_i^m \subseteq L_i^m$ follows from the definition.

We prove the other property by induction on $m$.

*Assumption*: For all $m' \leqslant m$, for all $j = 1,\ldots,p$, $\mathcal{L}_j^{m+1} = L_j^{m+1}$ or $|\mathcal{L}_j^{m+1}|_P \geqslant D+1$.

*Goal*: prove that $\mathcal{L}_i^{m+1} = L_i^{m+1}$ or $|\mathcal{L}_i^{m+1}|_P \geqslant D+1$ for any $i = 1,\ldots,p$.

Let $i$ be fixed.

*Case* A.1: $q_i \in Q_{\mathcal{S}}$.

Two cases may occur.

*Subcase* A.1.1: there exists $t \in \mathcal{L}_i^{m+1}$ of the form

$$t \equiv_P \bigoplus_{l=1}^{l=k} n_l e_l \oplus t' \text{ such that } \begin{cases} e_l \in \mathcal{L}_j^m \text{ and } e_l \in \mathcal{T}, \\ t' \ contains \ no \ element \ of \ \mathcal{L}_j^m, \text{ and } |\mathcal{L}_j^m|_P \geqslant D+1. \\ k \leqslant D \end{cases}$$

Since $\mathcal{L}_j^m$ contains at least $D+1$ elements, and $k \leqslant D$, there are at least $C_{D+1}^D = D+1$ distinct possible subsets $\{e_{i_1},\ldots,e_{i_k}\}$ of elements of $\mathcal{L}_j^m$.

Therefore we have $C_{D+1}^D = D+1$ non-equivalent multitrees $\bigoplus_{l=1}^{l=k} n_l e_{i_l} \oplus t'$ which also belong to $\mathcal{L}_i^{m+1}$ since the permutation of the $e_i$'s does not change the satisfiability of constraints.

Therefore $|\mathcal{L}_i^{m+1}|_P \geqslant D+1$.

*Subcase* A.1.2: There is no such multitree in $\mathcal{L}_i^{m+1}$.

Therefore the elements of $\mathcal{L}_i^{m+1}$ have the form $\bigoplus_{l=1}^{l=k} n_l e_l$ (up to $\equiv_P$) where all $e_l \in \mathcal{L}_m^l$ such that $|\mathcal{L}_j^m|_P < D$.

By induction hypothesis $\mathcal{L}_m^l = L_m^l$ for $l = 1,\ldots,k$.

By definition of $M$, we can construct at least $D+1$ non-equivalent multitrees reaching $q_i$ from the elements of $\mathcal{L}_j^m$ by restricting the multitrees of $\mathcal{S}$ to the form $n_1.e_1 \oplus \ldots \oplus n_k e_k$ with $n_l \leqslant M$, $l = 1,\ldots,k$ or else we compute all multitrees reaching $q_i$ from $\mathcal{L}_j^m$ using multitrees satisfying this restriction.

By definition of $\mathcal{L}_i^{m+1}$, we get that either $\mathcal{L}_i^{m+1} = L_i^{m+1}$ or $|\mathcal{L}_i^{m+1}|_P > D$.

*Case* A.2: $q_i \in Q_{\mathcal{T}}$.

To a multitree $t \equiv_P f(t_1,\ldots,t_n)$ we associate the tuple of multitrees $\vec{t} = (t_1,\ldots,t_n)$ which we can write $\vec{t} = \bigoplus_l \vec{n_l}\ e_l$ where the $\vec{n_l}$ are tuples of natural numbers. The proof

proceeds as in the previous case except that multitrees $t, T_j, T_l$ are replaced by tuples of multitrees $\vec{t}, \vec{T}_j, \vec{T}_l$ (but the $e_l$'s are still multitrees of $\mathcal{T}$ Then).

Two cases may occur.

*Subcase* A.2.1: There exists $t = f(t_1, \ldots, t_n) \in \mathcal{L}_i^{m+1}$ such that

$$\vec{t} \equiv_P \bigoplus_{l=1}^{l=k} \vec{n_l}\, e_l \oplus t' \text{ and } \begin{cases} e_l \in \mathcal{L}_j^m \text{ and} \\ t' \textit{ contains no element of } \mathcal{L}_j^m, \text{ with } |\mathcal{L}_j^m| > D. \\ k \leqslant D \end{cases}$$

Since $\mathcal{L}_j^m$ contains at least $D+1$ non-equivalent elements, and $k \leqslant D$, there are at least $C_{D+1}^D = D+1$ distinct possible subsets $\{e_{i_1}, \ldots, e_{i_k}\}$ of elements of $\mathcal{L}_j^m$.

Therefore we have $C_{D+1}^D = D+1$ distinct tuples $\bigoplus_{l=1}^{l=k} n_l e_{i_l} \oplus t'$ which also belong to $\mathcal{L}_i^{m+1}$ since the permutation of the $e_i$'s does not change the satisfiability of constraints.

Therefore $|\mathcal{L}_i^{m+1}|_P \geqslant D+1$.

*Subcase* A.2.2: no such term exists in $\mathcal{L}_i^{m+1}$.

Therefore the terms of $\mathcal{L}_i^{m+1}$ have the form $t = f(t_1, \ldots, t_n)$ where

$\vec{t} \equiv_P \bigoplus_{l=1}^{l=k} \vec{n_l}\, e_l$ and $e_l \in \mathcal{L}_m^l$ such that $|\mathcal{L}_j^m|_P < D$ for $l = 1, \ldots, k$.

By induction hypothesis $\mathcal{L}_m^l = L_m^l$ for $l = 1, \ldots, k$.

By definition of $M$, we can construct at least $D+1$ non-equivalent multitrees reaching $q_i$ from the elements of $\mathcal{L}_j^m$ by restricting the multitrees of $\mathcal{S}$ to the form $n_1.e_1 \oplus \ldots \oplus n_k e_k$ with $n_l \leqslant M,\ l = 1, \ldots, k$ or else we compute all multitrees reaching $q_i$ from $\mathcal{L}_j^m$ using multitrees satisfying this restriction.

By definition of $\mathcal{L}_i^{m+1}$, we get that either $\mathcal{L}_i^{m+1} = L_i^{m+1}$ or $|\mathcal{L}_i^{m+1}|_P > D$. □

The correctness of the algorithm follows immediately: there is some $t$ such that $t \to q_i$ iff $t \in L_i^m$ for some $m$, and $L_i^m \neq \emptyset$ iff $\mathcal{L}_i \neq \emptyset$ by Lemma A.1. □

### A.3. *Closure of regular languages and multitree automata*

We give the proof of Lemma 1.

$flat(Cl(\mathcal{L}(\mathcal{A}))) \subseteq \mathcal{L}(\mathcal{B})$.

**Lemma A.2.** *If* $s \to_{\mathcal{A}} q$ *then* $flat(s) \to_{\mathcal{B}} q$.

The proof is a structural induction on $s$.

- *Case* $s = a$: straightforward.
- *Case* $s = f(s_1, \ldots, s_n)$: Assume $s \to_{\mathcal{A}} q$ with the rule $f(q_1, \ldots, q_n) \to q$.
  By induction hypothesis, $s_i \to_{\mathcal{A}} q_i$ implies $flat(s_i) \to_{\mathcal{B}} q_i$.
  Type 1 rules of $\mathcal{B}$ are rules of $\mathcal{A}$, $flat(s) = f(flat(s_1), \ldots, flat(s_n))$ therefore $flat(s) \to_{\mathcal{B}} q$.
- *Case* $s = s' \oplus s''$: Assume $s \to_{\mathcal{A}} q$ with the rule $q' \oplus q'' \to q$ of $\mathcal{A}$.
  By induction hypothesis $flat(s') \to_{\mathcal{B}} q'$ and $flat(s'') \to_{\mathcal{B}} q''$.

The rule $q' \oplus q'' \to q$ is a rule of $\mathcal{A}$, then the grammar defining $L_q$ contains the rule $X_q \to X_{q'} X_{q''}$.
Therefore $Pa(L_{q'}) + Pa(L_{q''}) \subseteq Pa(L_q)$
Therefore $flat(s') \oplus flat(s'') \to_{\mathcal{B}} q$. □

This proves that $flat(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{B})$. Since languages of *CMTL* are closed under $\equiv_P$ and since $Cl_P(flat(\mathcal{L}(\mathcal{A}))) = flat(Cl(\mathcal{L}(\mathcal{A})))$ the inclusion holds.
$\mathcal{L}(\mathcal{B}) \subseteq flat(Cl(\mathcal{L}(\mathcal{A})))$.

**Lemma A.3.** *If* $\mathbf{s} \to_{\mathcal{B}} q$ *then there exists some* $s \to_{\mathcal{A}} q$ *such that* $\mathbf{s} \equiv_P flat(s)$

Again the proof is by structural induction on **s**.

- *Case* $\mathbf{s} = a$: straightforward.
- *Case* $\mathbf{s} = f(\mathbf{s}_1, \ldots, \mathbf{s}_n)$:
  Assume $\mathbf{s} \to_{\mathcal{B}} q$ with the type 1 rule $f(q_1, \ldots, q_n) \to q$.
  By induction hypothesis, there is some $s_i$ such that $flat(s_i) \equiv_P \mathbf{s}_i$ and $s_i \to_{\mathcal{A}} q_i$ for each $i$.
  Type 1 rules of $\mathcal{B}$ are rules of $\mathcal{A}$.
  Therefore $s = f(s_1, \ldots, s_n) \to_{\mathcal{A}} q$ and $flat(s) \equiv_P \mathbf{s}$.
- *Case* $\mathbf{s} = \mathbf{s}_1 \oplus \ldots \oplus \mathbf{s}_n$:
  Assume $s \to_{\mathcal{B}} q$.
  By definition $\mathbf{s}_i \to_{\mathcal{B}} q_i \in \mathcal{Q}_{\mathcal{T}}$ for each $i$ and the Parikh mapping of the word $q_1 \cdot \ldots \cdot q_n$ is in $Pa(L_q)$.
  Therefore there is a permutation $\sigma$ of $\{1, \ldots, n\}$ such that $q_{\sigma(1)} \cdot \ldots \cdot q_{\sigma(n)}$ is in $L_q$.
  Therefore there is some rule $q' \oplus q'' \to q$, some $i$ such that $q_{\sigma(1)} \cdot \ldots \cdot q_{\sigma(i)} \in L_{q'}$ and $q_{\sigma(i+1)} \cdot \ldots \cdot q_{\sigma(n)} \in L_{q'}$.
  By induction hypothesis, there is some $s', s''$ such that $s' \to_{\mathcal{A}} q'$, $s'' \to_{\mathcal{A}} q''$ in $\mathcal{A}$ and $flat(s') = s_{\sigma(1)} \cdot \ldots \cdot s_{\sigma(i)}$, $flat(s'') = s_{\sigma(i+1)} \cdot \ldots \cdot s_{\sigma(n)}$.
  Therefore $s = s' \oplus s'' \to_{\mathcal{A}} q$ and $flat(s) = flat(s') \oplus flat(s'') \equiv_P \mathbf{s}$. □

The inclusion is a direct application of the lemma.

## Appendix B. Decidability results for multiset logics

All our decidability results are obtained by reducing formulas on multisets to Presburger arithmetic formula. The reduction extensively uses *semilinear sets* and their properties.

### *B.1. Presburger arithmetic and semilinear sets*

The reader may refer to [5,14] for missing proofs and more results on the connections between semilinear sets and language theory.

We usually use the vector notation to denote tuples, like $\vec{x}$ for $(x_1, \ldots, x_n)$. Given $\vec{b} \in \mathbb{N}^n$, $P = \{\vec{p}_1, \ldots, \vec{p}_m\}$ with $\vec{p}_i \in \mathbb{N}^n$ the *linear set* of base $\vec{b}$ and periods $P$ is $L(\vec{b}, P) = \{\vec{\alpha} \in \mathbb{N}^n \mid \vec{\alpha} = \vec{b} + \sum_{i=1}^{i=m} \lambda_i \vec{p}_i, \lambda_i \in \mathbb{N}\}$. The element $\vec{b}$ is called the basis and the $\vec{p}_i$'s are the periods (if $P = \emptyset$, then $L(\vec{b}, P) = \{\vec{b}\}$). A *semilinear set* is

a finite union of linear sets. Semilinear sets are closed under set operations and are the models of Presburger's arithmetic formulas. Moreover the correspondence is effective, i.e. the semilinear set associated to a formula is effectively computable and the definition of a semilinear set is an existential Presburger formula.

We now introduce *decomposition contraints* for vectors of integers. In the following definitions $\mathcal{L}, \mathcal{L}_1, \ldots, \mathcal{L}_p$ are some fixed languages of $\mathbb{N}^p$, but $\vec{x}, N_1, \ldots, N_p$ are free variables.

By definition, the formula denoted by

$$\vec{x} = \bigoplus_{i=1}^{i=p} \vec{x}\ (\mathcal{L}_i, N_i) \wedge \phi(\vec{x}, N_1, \ldots, N_p)$$

states that there exists $\vec{x}_1, \ldots, \vec{x}_p \in \mathbb{N}^n$ such that

(1) $\vec{x} = \vec{x}_1 + \ldots + \vec{x}_p$,

(2) for $i = 1, \ldots, p$, $\vec{x}_i = \sum_{j=1}^{N_i} \vec{x}_{i,j}$ with $\vec{x}_{i,j} \in \mathcal{L}_i$ for $j = 1, \ldots, N_i$,

(3) $\models \phi(\vec{x}, N_1, \ldots, N_p)$

with the convention that $N_i = 0$ implies $\vec{x}_i = \vec{0}$.

This is not a first-order formula since the number of elements in $\vec{x}_i$ depends on the free variables $N_i$'s. However, when the $\mathcal{L}_i$'s are semilinear sets and $\phi(\vec{x}, N_1, \ldots, N_p)$ is a Presburger formula, we have:

**Proposition B.1.** *The formula*

$$\vec{x} = \bigoplus_{i=1}^{i=p} \vec{x}\ (\mathcal{L}_i, N_i) \wedge \phi(\vec{x}, N_1, \ldots, N_p)$$

*in the free variables $\vec{x}, N_1, \ldots, N_p$ where the $\mathcal{L}_i$'s are semilinear sets of $\mathbb{N}^n$ and $\phi$ is a Presburger's arithmetic formula, is an effectively computable Presburger's arithmetic formula.*

**Proof.** We give the main lemmas that can be checked by simple algebraic computations.

**Lemma B.1.** *Let $\mathcal{L} = L(\vec{b}, P)$. For $N > 0$, $\vec{x} = \vec{x}\ (\mathcal{L}, N)$ is equivalent to $\exists \lambda_i\ \vec{x} = N\ \vec{b} + \sum_{\vec{p}_i \in P} \lambda_i\ \vec{p}_i$*

(*The case $N = 0$ is trivial and states that $\vec{x} = \vec{0}$.*)

**Lemma B.2.** *If $\mathcal{L} = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_p$, then $\vec{x} = \vec{x}\ (\mathcal{L}, N)$ is equivalent to*

$$\exists N_1, \ldots, N_p\ \vec{x} = \bigoplus_{i=1}^{i=p} \vec{x}\ (\mathcal{L}_i, N_i) \wedge \left( N = \sum_{i=1}^{i=p} N_i \right)$$

The result follows easily from these two lemmas. □

*B.2. Basis of $\mathbb{N}^n$*

We say that the non-empty semilinear sets $\mathcal{L}_1, \ldots, \mathcal{L}_p$ define a basis of $\mathbb{N}^n$ iff they define a partition of $\mathbb{N}^n$ i.e. $\mathcal{L}_i \cap \mathcal{L}_j = \emptyset$ if $i \neq j$, $\mathbb{N}^n = \mathcal{L}_1 \cup \ldots \cup \mathcal{L}_p$.

We say that the basis *respects cardinality* iff for each set $\mathcal{L}$ in the basis there exists $I \subseteq \{1, \ldots, n\}$ such that $(\alpha_1, \ldots, \alpha_n) \in \mathcal{L}$ implies that $\alpha_i = 0$ iff $i \in I$.

**Proposition B.2.** *Any basis can be partitioned into a basis that respects cardinality.*

**Proof.** Each set $\mathcal{L}$ of the basis can be partitioned into $\mathcal{L}_I = \{(\alpha_1, \ldots, \alpha_n) \in \mathcal{L} \mid \alpha_i = 0 \text{ iff } i \in I\}$. The $\mathcal{L}_I$ are semilinear sets since they can be defined by a Presburger arithmetic formula. The new basis is obtained by replacing $\mathcal{L}$ by the non-empty $\mathcal{L}_I$'s. □

**Proposition B.3.** *Let $\mathcal{L}_1, \ldots, \mathcal{L}_p$ be a basis, $\mathcal{L}'_1, \ldots, \mathcal{L}'_r$ be another basis, then the non-empty sets of $\mathcal{L}_i \cap \mathcal{L}'_j$, $i = 1, \ldots, p$, $j = 1, \ldots, r$ define a basis of $\mathbb{N}^n$. If one of the two initial bases respects cardinality, this is true also for the new basis.*

**Proof.** The intersection of two partitions is a partition.
The property $\vec{\alpha} \in \mathcal{L}$ implies $\alpha_i = 0$ iff $i \in I$ is also true in any subset of $\mathcal{L}$. □

To deal with the existential quantification in $FO^{\#}_{\mathcal{M}}$, we study how basis behave under projection. Let $\pi : \mathbb{N}^n \to \mathbb{N}^{n-1}$ be the projection defined by $\pi(\alpha_1, \ldots, \alpha_n) = (\alpha_1, \ldots, \alpha_{\pi-1}, \alpha_{\pi+1}, \ldots, \alpha_n)$, i.e. $\pi$ erases component $\pi$

**Proposition B.4.** *Let $\mathcal{L}_1, \ldots, \mathcal{L}_p$ be a basis of $\mathbb{N}^n$. Then there exists $L_1, \ldots, L_m$ a basis of $\mathbb{N}^n$ such that* (i) *for all pair $i, j$ either $\pi(L_i) = \pi(L_j)$ or $\pi(L_i) \cap \pi(L_j) = \emptyset$* (ii) *each $\mathcal{L}_i$ is the disjoint union of some $L_j$*'s.

**Proof.** For each $I \subseteq \{1, \ldots, p\}$, let $\mathcal{M}_I = \bigcap_{i \in I} \pi(\mathcal{L}_i) \cap \bigcap_{i \notin I} \overline{\pi(\mathcal{L}_i)}$.

By definition $\mathcal{M}_I \cap \mathcal{M}_J = \emptyset$ if $I \neq J$ hence $\pi^{-1}(\mathcal{M}_I) \cap \pi^{-1}(\mathcal{M}_J) = \emptyset$.

If $\vec{x} \in \mathcal{L}_i$ then $\pi(\vec{x}) \in \pi(\mathcal{L}_i)$ hence $\vec{x} \in \pi^{-1}(\mathcal{M}_I)$ for some $I$.

Therefore the sets $L_{i,I} = \mathcal{L}_i \cap_{i \in I} \pi^{-1}(\mathcal{M}_I)$ for $I \subseteq \{1, \ldots, p\}$, $i \in I$, define a partition of $\mathcal{L}_i$ that yields (ii) and that all these sets define a basis of $\mathbb{N}^n$.

To prove (i) we remark that $\pi(L_{i,I}) = \mathcal{M}_I$, therefore $\pi(L_{i,I}) = \pi(L_{j,J})$ iff $I = J$ (the sets are defined only for $i \in I$, $j \in J$. □

**Example B.1.** If the initial basis is $\mathcal{L}_1, \mathcal{L}_2$, then $\mathcal{M}_1 = \pi(\mathcal{L}_1) \backslash \pi(\mathcal{L}_2)$, $\mathcal{M}_2 = \pi(\mathcal{L}_1) \cap \pi(\mathcal{L}_2)$, $\mathcal{M}_3 = \pi(\mathcal{L}_2) \backslash \pi(\mathcal{L}_1)$ (we use number as subscript instead of sets). The new basis is $L_1 = \mathcal{L}_1 \cap \pi^{-1}(\mathcal{M}_1)$, $L_2 = \mathcal{L}_1 \cap \pi^{-1}(\mathcal{M}_2)$, $L_3 = \mathcal{L}_2 \cap \pi^{-1}(\mathcal{M}_3)$, $L_4 = \mathcal{L}_2 \cap \pi^{-1}(\mathcal{M}_2)$. In this example $\pi(L_2) = \pi(L_4)$ otherwise the projections are pairwise disjoint.

Since we perform only intersection and complementation operations, if $\mathcal{L}_1, \ldots, \mathcal{L}_p$ respects cardinality, then $L_1, \ldots, L_m$ has the same property. Furthermore the set consisting of the projections $\pi(L_i)$ is a basis of $\mathbb{N}^{n-1}$.

### B.3. The decidability proof for multiset logics

We turn now to multisets with elements in $\mathcal{S} = \{e_1, e_2, \ldots\}$ and to the logics $FO_{\mathcal{M}}$, $FO^{\#D}_{\mathcal{M}}$, $FO^{\#}_{\mathcal{M}}$. The idea of the decidability proof is to reduce formulas to simpler ones that we call *solved forms*.

In the following, we use tuples of multisets variables denoted by $\vec{X}$,..., tuples of multisets denoted by $\vec{\mathcal{X}}$.... Given a tuple $\vec{X} = (X_1, \ldots, X_n)$, the tuple $(\#(X_1), \ldots, \#(X_n))$ is denoted by $\#(\vec{X})$ (and the notation is extended to a tuple $\vec{\mathcal{X}}$, and to the $\#_D$ operator).

#### B.3.1. Solved forms

Let $\mathcal{L}$ be a semilinear set of $\mathbb{N}^n$. We define the formula $Mult(\vec{X}, \mathcal{L})$ which is satisfied by the tuples $\vec{\mathcal{X}}$ such that either $\vec{\mathcal{X}} = \vec{\emptyset}$ or $\vec{\mathcal{X}} = \Sigma_{j\in J}\ \vec{\lambda}_j\ e_j$ with $\vec{\lambda}_j \in \mathcal{L}$ for all $j \in J$ (it generalizes to tuples the formula $Mult(X, \psi)$ of Section 3). We recall that $One(Y)$ is the formula that states that the multiset $Y$ has the form $\{n.e\}$.

$\forall\ \vec{Y}, \vec{Z}$ [
$(\vec{X} = \vec{Y} \oplus \vec{Z} \wedge \bigvee_{i=1}^{i=n}(Y_i \neq \emptyset)$ /* $\vec{Y}$ is a non-empty subset of $\vec{X}$ */
$\wedge \bigwedge_{i=1}^{i=n}(One(Y_i) \vee Y_i = \emptyset)$ /* at most one element in $Y_i$ */
$\wedge \bigwedge_{i,j=1,\ldots,n} Y_i \cap Z_j = \emptyset$ /* that occurs only in $\vec{Y}$ */
$\wedge \bigwedge_{i,j=1,\ldots,n}((Y_i \neq \emptyset \wedge Y_j \neq \emptyset) \Rightarrow (Y_i \cap Y_j \neq \emptyset)))$
$\Rightarrow$
$\#(\vec{Y}) \in \mathcal{L}$ ] /* the multiplicity is in $\mathcal{L}$ */.

The sub-formula $\#(\vec{Y}) \in \mathcal{L}$ can be replaced by a more complex formula of $FO_{\mathcal{M}}$.

Let $\vec{X} = (X_1, \ldots, X_n)$, $\vec{Y} = (Y_1, \ldots, Y_n)$ then $Disjoint(\vec{X}, \vec{Y})$ is the formula $\bigwedge_{1\leqslant i,j\leqslant n} X_i \cap Y_j = \emptyset$ that states that the components of $\vec{X}$ and $\vec{Y}$ are pairwise disjoint.

Finally, to simplify the definition of solved forms we write $\vec{X} = \bigoplus_{\neq\ i=1}^{i=p}\ \vec{X}_i$ for $\vec{X} = \bigoplus_{i=1}^{i=p}\ \vec{X}_i \wedge \bigwedge_{\substack{1\leqslant i,j\leqslant p \\ i\neq j}}\ Disjoint(\vec{X}_i, \vec{X}_j)$

(i.e. the multisets occurring in the decomposition of $\vec{X}$ are pairwise disjoint.)

**Definition B.1.** A *solved form* is a formula

$$\exists\ \vec{X}_i \vec{X} = \bigoplus_{\neq\ i=1}^{i=p}\ \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p)),$$

where $\vec{X}$ is a tuple of multiset variables, $\mathcal{L}_1, \ldots, \mathcal{L}_p$ is a basis of $\mathbb{N}^n$ and $\phi$ is a Presburger arithmetic formula.

A *solution* of the solved form is a tuple $\vec{\mathcal{X}}$ that satisfies the solved form (according to the definition of $FO^{\#}_{\mathcal{M}}$). Two solved forms are *equivalent* iff they have the same set of solutions.
A solved form is *non-ambiguous* if the basis respects cardinality.

**Notation.** In the following, we shall drop the existential quantification $\exists\ \vec{X}_i$ which is always clear from the context.

**Proposition B.5.** *Let $\mathcal{L}_1, \ldots, \mathcal{L}_p$ be a basis of $\mathbb{N}^n$, then for any tuple of multisets $\vec{\mathcal{X}}$ there is a unique decomposition $\vec{\mathcal{X}}=\vec{\mathcal{X}}_1 \oplus \ldots \oplus \vec{\mathcal{X}}_p$ such that $Mult(\vec{\mathcal{X}}_i, \mathcal{L}_i)$ and $\models Disjoint(\vec{\mathcal{X}}_i, \vec{\mathcal{X}}_j)$ for $i \neq j$.*

**Proof.** *Case* B.1: $\vec{\mathcal{X}}=\vec{\emptyset}$: the unique possible decomposition is $\vec{\mathcal{X}}_1= \ldots =\vec{\mathcal{X}}_p=\vec{\emptyset}$.

*Case* B.2: $\vec{\mathcal{X}}= \bigoplus_{j\in J} \vec{\lambda}_j\ e_j$.
Since the decomposition must be disjoint, all occurrences of $e_j$ must belong to the same $\vec{\mathcal{X}}_i$.
Since the $\mathcal{L}_i$'s define a basis, $\vec{\lambda}_j$ belongs to a unique $\mathcal{L}_{i_j}$.
For each $i$, either no $\vec{\lambda}_j \in \mathcal{L}_i$ and $\vec{\mathcal{X}}_i= \emptyset$ or else $\vec{\mathcal{X}}_i= \bigoplus_{\vec{\lambda}_j \in \mathcal{L}_i} \vec{\lambda}_j\ e_j$. □

*B.3.2. Basic properties*
*Splitting*:

**Proposition B.6.** *Let $\mathcal{L}_i$ be a basis and let $\mathcal{L}_{i,1}, \ldots, \mathcal{L}_{i,k_i}$ be a partition of $\mathcal{L}_i$ for $i = 1, \ldots, p$. Then the solved form*

$$\vec{X}= \bigoplus_{\neq\ i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p))$$

*is equivalent to the solved form*

$$\begin{aligned}\vec{X} = & \bigoplus_{\neq\ i=1}^{i=p} \bigoplus_{\neq\ j=1}^{j=k_i} \vec{X}_{i,j} \wedge Mult(\vec{X}_{i,j}, \mathcal{L}_{i,j}) \\ & \wedge \phi(\textstyle\sum_{j=1}^{j=k_1} \#(\vec{X}_{i,j}), \ldots, \sum_{j=1}^{j=k_p} \#(\vec{X}_{p,j}), \sum_{j=1}^{j=k_1} \#_D(\vec{X}_{1,j}), \\ & \ldots, \textstyle\sum_{j=1}^{j=k_p} \#_D(\vec{X}_{p,j})).\end{aligned}$$

*Furthermore if the first solved form is non-ambiguous, then the second one is non-ambiguous.*

**Proof.** Use the fact that partitioning a partition yields a new partition and Proposition B.5. □

**Proposition B.7.** *A solved form is equivalent to a non-ambiguous solved form.*

**Proof.** Direct from Propositions B.2 and B.6. □

*B.3.3. Combination of non-ambiguous solved forms*

**Conjunction**: The conjunction of the non-ambiguous solved form

$$\vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p))$$

and of the non-ambiguous solved form

$$\vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \psi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p))$$

is the non-ambiguous solved form

$$\vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge (\phi \wedge \psi)(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p)).$$

**Proof.** Since the solved forms use the same basis, the decomposition of a solution $\vec{\mathcal{X}}$ into $\vec{\mathcal{X}}_1 \oplus \ldots \oplus \vec{\mathcal{X}}_p$ where $\vec{\mathcal{X}}_i = \Sigma_{j=1}^{j=N_i} \vec{\lambda}_{i,j}\, e_{i,j}$, $\vec{\lambda}_{i,j} \in \mathcal{L}_i$ is unique. Therefore the conjunction of the solved forms is true iff the conjunction of the Presburger formulas is true. □

If the two solved forms do not use the same basis, say $\mathcal{L}_1, \ldots, \mathcal{L}_p$ for the first one, and $\mathcal{L}'_1, \ldots, \mathcal{L}'_q$ for the second one, we can perform a splitting to get two non-ambiguous solved forms using the common basis $\mathcal{L}_i \cap \mathcal{L}'_j$, $i = 1, \ldots, p$, $j = 1, \ldots, q$ (discarding the empty intersections).

*Negation*: The negation of the non-ambiguous solved form

$$\vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p))$$

is the non-ambiguous solved form

$$\vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \neg\phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p)).$$

**Proof.** The decomposition of any multiset $\vec{\mathcal{X}} = \bigoplus_{\neq\, i=1}^{i=p} \vec{\mathcal{X}}_i$ for $\vec{\mathcal{X}}_i = \Sigma_{j=1}^{j=N_i} \vec{\lambda}_{i,j}\, e_{i,j}$, $\vec{\lambda}_{i,j} \in \mathcal{L}_i$ is unique by Proposition B.5. Therefore, the negation of the solved form is true iff the negation of the Presburger formula is true. □

*Elimination of existential quantification*: Let $\pi : \mathbb{N}^n \to \mathbb{N}^{n-1}$ be the projection defined by

$$\pi(\alpha_1, \ldots, \alpha_n) = (\alpha_1, \ldots, \alpha_{\pi-1}, \alpha_{\pi+1}, \ldots, \alpha_n)$$

(we identify the projection and the index of the component that is erased).

We consider a non-ambiguous solved form $S(\vec{X})$ such the Presburger part is $\phi(\#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p))$ i.e. contains no occurrence of $\#(\vec{X}_i)$. Let $\mathcal{L}_1, \ldots, \mathcal{L}_p$ be the basis of the solved form. By propositions B.4 and B.6, we can assume that there is a basis $L_1, \ldots, L_m$ of $\mathbb{N}^{n-1}$ such that for each $i = 1, \ldots, p$, $\pi(\mathcal{L}_i)$ is some $L_j$. We denote by $\pi^{-1}(j)$ the set of indices $i$ such that $L_j = \pi(\mathcal{L}_i)$.

Our goal is to show that the formula $\exists X_\pi S(\vec{X})$ is equivalent to a solved form. Elimination of the existentially quantified variable $X_\pi$ is closely related to projection, the main technical difficulty is to relate $\#_D(\vec{\mathcal{Y}})$ and $\#_D(\vec{\mathcal{X}})$ when $\vec{\mathcal{Y}} = \pi(\vec{\mathcal{X}})$.

Let $\mathcal{L}_i, L_j$ such that $\pi(\mathcal{L}_i) = L_j$. Let $\vec{M} = (M_1, \ldots, M_\pi, \ldots, M_n)$ and $\vec{N} = (N_1, \ldots, N_{\pi-1}, N_{\pi+1}, \ldots, N_n)$ be tuples of integer variables.
Let $I = \{k \in \{1, \ldots, n\} \mid (\alpha_1, \ldots, \alpha_n) \in \mathcal{L}_i \text{ iff } \alpha_k = 0\}$. We define

$$\vec{M} = \pi_{j,i}^{-1}(\vec{N}) \text{ by } \bigwedge_{i \neq \pi} M_i = N_i \wedge \begin{cases} M_\pi = N_l & \texttt{if } \exists l\ \pi, l \notin I, \\ M_\pi = 0 & \texttt{if } \pi \in I, \\ M_\pi = M_\pi & \text{otherwise.} \end{cases}$$

**Lemma B.3.** *Let $\vec{\mathcal{X}} \in \mathcal{L}_i$, $\vec{\mathcal{Y}} \in L_j$ be such that $\vec{\mathcal{Y}} = \pi(\vec{\mathcal{X}})$.*
*Then $\#_D(\vec{\mathcal{X}}) = \pi_{j,i}^{-1}(\#_D(\vec{\mathcal{Y}}))$.*
*Conversely, let $\vec{\mathcal{Y}} \in L_j$ such that $\vec{\alpha} = \pi_{j,i}^{-1}(\#_D(\vec{\mathcal{Y}}))$. Then there exists $\vec{\mathcal{X}} \in \mathcal{L}_i$ such that $\vec{\mathcal{Y}} = \pi(\vec{\mathcal{X}})$ and $\#(\vec{\mathcal{X}}) = \vec{\alpha}$.*

**Proof.** The first statement is a direct consequence of the definition.
To prove the second one, let $\vec{\mathcal{Y}} = \bigoplus_{k=1}^{k=m} \vec{\lambda}_k\ e_k$. By definition $\vec{\lambda}_k \in L_j$.

Since $\pi(\mathcal{L}_i) = L_j$, for each $k$ there exists $\vec{\mu}_k \in \mathcal{L}_i$ such that $\pi(\vec{\mu}_k) = \vec{\lambda}_k$.

Depending on the value $\alpha_\pi$, there are three possibilities:

- $\alpha_\pi = 0$. Therefore the component $\pi$ of any $\vec{\mu}_k \in \mathcal{L}_i$ is zero.
- $\alpha_\pi = \alpha_j$ for $j \notin I$. Therefore components $l, \pi$ of any $\vec{\mu}_k \in \mathcal{L}_i$ are non-zero.
- none of the above. Therefore all components of $\mathcal{L}_i$ but $\pi$ are zero.

In the first two cases $\vec{\mathcal{X}} = \bigoplus_{k=1}^{k=m} \vec{\mu}_k\ e_k$ is suitable, and in the last case any $\vec{\mathcal{X}} = \bigoplus_{k=1}^{k=\alpha_\pi} \vec{\mu}_k\ e_k$ with $\vec{\mu}_k \in \mathcal{L}_i$ is suitable. □

Remember that $i \in \pi^{-1}(j)$ denotes the $i$'s such that $\pi(\mathcal{L}_i) = L_j$. In the next proposition, we extend $\pi$ to tuples of variables.

**Proposition B.8.** *The non-ambiguous solved form*

$$\exists X_\pi \left( \vec{X} = \bigoplus_{\neq\, i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p)) \right)$$

*is equivalent to the non-ambiguous solved form*

$$\begin{aligned} \pi(\vec{X}) = \bigoplus_{\neq\, j=1}^{j=m} \vec{Y}_j \wedge Mult(\vec{Y}_j, L_j) \wedge \exists\, \vec{N}_{j,i} \bigwedge_{j=1}^{j=m} \#_D(\vec{Y}_j) = \Sigma_{i\in\pi^{-1}(j)}\, \vec{N}_{j,i} \\ \wedge \exists\, \vec{M}_{j,i} \bigwedge_{i\in\pi^{-1}(j)} \vec{M}_{j,i} = \pi_{j,i}^{-1}(\vec{N}_{j,i}) \\ \wedge \phi(\vec{M}_{\pi(1),1}, \ldots, \vec{M}_{\pi(p),p}). \end{aligned}$$

The proposition is not true if the # operation occurs in $\phi$.

**Proof.** $\Rightarrow$ Direction.

Let $\vec{\mathcal{X}}$ be a solution of the first formula. By definition there is a decomposition $\vec{\mathcal{X}} = \bigoplus_{\neq\, i=1}^{i=p} \vec{\mathcal{X}}_i$ such that $\models Mult(\vec{\mathcal{X}}_i, \mathcal{L}_i)$

By definition of $\pi$ and $L_i$ we get $\pi(\vec{\mathcal{X}}) = \bigoplus_{\neq\, i=1}^{i=p} \pi(\vec{\mathcal{X}}_i)$ and $Mult(\pi(\vec{\mathcal{X}}_i), L_j)$ where $\pi(\mathcal{L}_i) = L_j$.

Let $\vec{\mathcal{Y}}_{j,i} = \pi(\vec{\mathcal{X}}_i)$ and $\vec{\mathcal{Y}}_j = \bigoplus_{i\in pi^{-1}(j)} \vec{\mathcal{Y}}_{j,i}$. By definition $\models Mult(\vec{\mathcal{Y}}_j, L_j)$.

By Lemma B.3 $\models \#_D(\vec{\mathcal{X}}_i) = \pi_{i,j}^{-1} \#_D(\vec{\mathcal{Y}}_{j,i})$ and $\models \phi(\#_D(\vec{\mathcal{X}}_1), \ldots, \#(\vec{\mathcal{X}}_p))$ by definition.

$\Leftarrow$ Direction.

Let $\vec{\mathcal{Y}}$ be a solution of the second formula.

By definition there exists a decomposition of $\vec{\mathcal{Y}}$ into $\vec{\mathcal{Y}}_j$ such that $Mult(\vec{\mathcal{Y}}_j, L_j)$ and there exist $\vec{m}_{j,i}$, $\vec{n}_{j,i}$ such that

(i) $\#_D(\vec{\mathcal{Y}}_j) = \Sigma_{i\in\pi^{-1}(j)}\, \vec{n}_{j,i}$,

(ii) $\models \vec{m}_{j,i} = \pi_{i,j}^{-1}(\vec{n}_{j,i})$,

(iii) $\models \phi(M_{\pi(1),1}, \ldots, M_{\pi(p),p})$.

By (i) we can decompose $\vec{\mathcal{Y}}_j$ into $\bigoplus_{\neq} \vec{\mathcal{Y}}_{j,i}$ with $\#_D(\vec{\mathcal{Y}}_{j,i}) = \vec{n}_{j,i}$ (usually, several decompositions exist, we choose one).

By Lemma B.3, there exists $\vec{\mathcal{X}}_{j,i}$ such that $Mult(\vec{\mathcal{X}}_{j,i}, L_i)$, $\pi(\vec{\mathcal{X}}_{j,i}) = \vec{\mathcal{Y}}_{j,i}$, and $\#_D(\vec{\mathcal{X}}_{j,i}) = \vec{m}_{j,i}$.

Therefore $\vec{\mathcal{X}} = \bigoplus_{\neq\, i=1}^{i=p} \vec{\mathcal{X}}_i$ for $\vec{\mathcal{X}}_1 = \vec{\mathcal{X}}_{\pi(1),1}, \ldots, \vec{\mathcal{X}}_n = \vec{\mathcal{X}}_{\pi(n),n}$ satisfy the first solved form. □

*B.3.4. Solved forms and multiset logics*

A quantifier-free positive atomic formula of $FO^{\#}_{\mathcal{M}}$ on the free variables $X_1, \ldots, X_n$ has the form

$$\left(\bigoplus_{i\in I} X_i \subseteq \bigoplus_{j\in J} X_j\right) \wedge \phi(\#(X_1), \ldots, \#(X_n), \#_D(X_1), \ldots, \#_D(X_n)).$$

Let $\mathcal{L}$ be the semilinear set of $\mathbb{N}^n$ defining the solutions of $\Sigma_{i\in I} x_i \leqslant \Sigma_{j\in J} x_j$. From the basis $\mathcal{L}, \overline{\mathcal{L}}$ of $\mathbb{N}^p$ compute a basis $\mathcal{L}_i$ for $i = 1, \ldots, p$ that respects cardinality.

**Proposition B.9.** *The atomic proposition is equivalent to the non-ambiguous solved form*

$$\begin{aligned}\vec{X} = & \bigoplus^{i=p}_{\neq i=1} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i)\\ & \wedge \phi(\Sigma^{i=p}_{i=1}\#(X_{i,1}), \ldots, \Sigma^{i=p}_{i=1}\#(X_{i,n}), \Sigma^{i=p}_{i=1}\#_D(X_{i,1}), \ldots, \Sigma^{i=p}_{i=1}\#_D(X_{i,n}))\\ & \wedge \bigwedge^{j=n}_{j=1} \Sigma_{i \mid \mathcal{L}_i\subseteq\overline{\mathcal{L}}}\#_D(X_{i,j}) = 0,\end{aligned}$$

*where* $\vec{X}_i = (X_{i,1}, \ldots, X_{i,n})$.

**Proof.** $\Rightarrow$ Direction.

Let $\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_n)$ satisfy the atomic formula.

Then there is a unique decomposition $\vec{\mathcal{X}} = \oplus^{i=p}_{i=1} \vec{\mathcal{X}}_i$ such that $\vec{\mathcal{X}}_i = \Sigma_{j\in I_i} \vec{\lambda}_j\, e_j$ with $\vec{\lambda}_j \in \mathcal{L}_j$ or $\vec{\mathcal{X}}_i = \emptyset$.

Necessarily $\vec{\lambda}_j \in \mathcal{L}$ otherwise the atomic formula is false.

This implies that $\vec{\mathcal{X}}_i = \emptyset$ if $\mathcal{L}_i \subseteq \overline{\mathcal{L}}$.

For each $i = 1, \ldots, p$ let $\vec{\mathcal{X}}_i$ be $(\mathcal{X}_{i,1}, \ldots, \mathcal{X}_{i,n})$. Then $\Sigma_{i \mid \mathcal{L}_i\subseteq\overline{\mathcal{L}}}\#_D(\mathcal{X}_{i,j}) = 0$.

By definition $\#(\mathcal{X}_j) = \Sigma^{i=p}_{i=1}\#(\mathcal{X}_{i,j})$ and $\#_D(X_i) = \Sigma^{i=p}_{i=1}\#(\mathcal{X}_{i,j})$ which implies that the Presburger formula of the solved form is true.

Therefore $\vec{\mathcal{X}}$ is a solution of the solved form.

$\Leftarrow$ Direction.

Let $\vec{\mathcal{X}}$ be a solution of the solved form. By definition $\vec{\mathcal{X}} = \oplus^{i=p}_{i=1} \vec{\mathcal{X}}_i$ such that $\vec{\mathcal{X}}_i = \emptyset$ or $\vec{\mathcal{X}}_i = \Sigma_{j\in I_i} \vec{\lambda}_j\, e_j$ with $\vec{\lambda}_j \in \mathcal{L}_j$.

The arithmetic condition implies that $\vec{\mathcal{X}}_i = \emptyset$ if $\mathcal{L}_i \subseteq \overline{\mathcal{L}}$.

Let us define $\mathcal{X}_i = \bigoplus^{j=p}_{j=1} \mathcal{X}_{j,i}$.

By definition of $\mathcal{L}$, $\bigoplus_{i\in I} \mathcal{X}_i \subseteq \bigoplus_{j\in J} \mathcal{X}_j$.

By definition of a solution $\models \phi(\#(X_1), \ldots, \#(X_n), \#_D(X_1), \ldots, \#_D(X_n))$. $\square$

If the atomic formula is a formula of $FO^{\#D}_{\mathcal{M}}$, then the solved form contains no occurrence of # in the Presburger part.

Summing up all the results on negation, conjunction and elimination of existential quantification, we get:

**Proposition B.10.** *Any quantifier-free formula of $FO^{\#}_{\mathcal{M}}$ is equivalent to a non-ambiguous solved form. Any formula of $FO^{\#D}_{\mathcal{M}}$ is equivalent to a non-ambiguous solved form.*

**Proof.** Since atomic proposition are equivalent to solved forms and since solved form are preserved by conjunction and negation (hence disjunction) the first result holds.

Since existential variables can be eliminated for $FO^{\#D}_{\mathcal{M}}$ formula and since the constructions for partitioning basis, conjunction, negation, elimination of existential variables do not introduce any occurrence of the # operation, the second result holds. □

*B.3.5. Decidable fragments*

Given a semilinear set $\mathcal{L}$ that respects cardinality (for a set of indexes $I$), an element $\vec{x}\ (\mathcal{L}, N)$, the tuple $\#N = (N_1, \ldots, N_p)$ is defined by $N_i = 0$ if $i \in I$ and $N_i = N$ otherwise. In the next proposition, $\mathcal{L}_1, \ldots, \mathcal{L}_p$ are semilinear sets that respects cardinality.

**Proposition B.11.** *Let $(S)$ be the solved form*:

$$\vec{X} = \bigoplus_{\neq\ i=1}^{i=p} \vec{X}_i \wedge Mult(\vec{X}_i, \mathcal{L}_i) \wedge \phi(\#(\vec{X}_1), \ldots, \#(\vec{X}_p), \#_D(\vec{X}_1), \ldots, \#_D(\vec{X}_p)).$$

*Let $(P)$ be the Presburger formula*:

$$\vec{x} = \bigoplus_{i=1}^{i=p} \vec{x}\ (\mathcal{L}_i, N_i) \wedge \phi(\vec{x}\ (\mathcal{L}_1, N_1), \ldots, \vec{x}\ (\mathcal{L}_p, N_p), \#N_1, \ldots, \#N_p)$$

*Then the set $\{\#(\vec{\mathcal{X}}) \mid \vec{\mathcal{X}}\ solution\ of\ (S)\}$ is the set of solutions of $(P)$.*

**Proof.** We prove each inclusion:

$\subseteq$. Let $\vec{\mathcal{X}} = \vec{\mathcal{X}}_1 \oplus \ldots \oplus \vec{\mathcal{X}}_p$ where $\vec{\mathcal{X}}_i = \Sigma_{j=1}^{j=N_i} \vec{\lambda}_{i,j}\ e_{i,j}$ a solution of the solved form.

By definition (i) $\vec{x}_i = \Sigma_{j=1}^{j=N_i} \vec{\lambda}_{i,j} = \vec{x}\ (\mathcal{L}_i, N_i)$,

(ii) $\#(\vec{\mathcal{X}}_i) = \Sigma_{j=1}^{j=N_i} \vec{\lambda}_{i,j} = \vec{x}\ (\mathcal{L}_i, N_i)$,

(iii) $\#_D(\vec{\mathcal{X}}_i) = \#N_i$,

therefore $\models \phi(\vec{x}\ (\mathcal{L}_1, N_1), \ldots, \vec{x}\ (\mathcal{L}_p, N_p), \#N_1, \ldots, \#N_p)$ and the Presburger formula is true.

$\supseteq$ *inclusion*: By definition a solution $\vec{x}$ of the Presburger formula can be decomposed as $\bigoplus_{i=1}^{i=p} \vec{x}\ (\mathcal{L}_i, N_i)$ where $\vec{x}\ (\mathcal{L}_i, N_i) = \Sigma_{j \in J_i} \vec{\lambda}_{i,j}$ with $\vec{\lambda}_{i,j} \in \mathcal{L}_i$. By definition, $\models \phi(\Sigma_{j \in J_1} \vec{\lambda}_{1,j}, \ldots, \Sigma_{j \in J_p} \vec{\lambda}_{p,j}, \#N_1, \ldots, \#N_p)$.

Therefore, given $e_{i,j}$'s ($i = 1, \ldots, p$, $j \in J_i$ any family of pairwise distinct elements, $\vec{\mathcal{X}} = \vec{\mathcal{X}}_1 \oplus \ldots \oplus \vec{\mathcal{X}}_p$ where $\vec{\mathcal{X}}_i = \Sigma_{j \in J_i} \vec{\lambda}_{i,j}\, e_{i,j}$ is a solution of the solved form. □

A straightforward consequence is that $FO^{\#D}_{\mathcal{M}}$ is decidable (by proposition B.10). This result has been obtained by Fefermann and Vaught [11] using product of decidable structures. Our approach yields more precise decidability results that we use in the proof of emptiness of multitree automata. Since solved forms are stable under conjunction and negation, we get:

**Proposition B.12.** *The following properties hold*:

- *The set* $\{\#(\vec{\mathcal{X}}) \mid \models \phi(\vec{\mathcal{X}})\}$ *where* $\phi$ *is a formula of* $\exists FO_{\mathcal{M}}$ *is an effectively constructible semilinear set.*
- *The set* $\{\#_D(\vec{\mathcal{X}}) \mid \models \phi(\vec{\mathcal{X}})\}$ *where* $\phi$ *is a formula of* $\exists FO_{\mathcal{M}}$ *is an effectively constructible semilinear set.*

A direct consequence is the decidability of the existential fragment and of the universal fragment of $FO^{\#}_{\mathcal{M}}$. The connection between Skolem arithmetic and $FO^{\#}_{\mathcal{M}}$ explained in the proof of Proposition 2 yields a decidable fragment of Skolem arithmetic enriched by the function summing the exponents of the prime occurring in the prime decomposition of a natural number.

## References

[1] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, Cambridge, 1998.
[2] D. Benanav, D. Kapur, P. Narendran, Complexity of matching problems, in: Proc. 1st Internat. Conf. on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science, Vol. 202, Springer, Berlin, 1985, pp. 417–429.
[3] L. Berman, Precise bounds for Presburger arithmetic and the reals with addition: preliminary report, in: Proc. 18th Symp. on Foundation of Computer Science (FOCS), IEEE Computer Society, Silver Spring, MD, 1977, pp. 95–99.
[4] B. Bogaert, S. Tison, Equality and disequality constraints on direct subterms in tree automata, in: Proc. 9th Symp. on Theoretical Computer Science (STACS), Lecture Notes in Computer Science, Vol. 577, Springer, Berlin, 1992, pp. 161–172.
[5] V. Bruyere, G. Hansel, C. Michaux, R. Villemaire, Logic and $p$-recognizable sets of integers, Bull. Belgian Math. Soc. 1 (1994) 191–238.
[6] A.C. Caron, H. Comon, J.L. Coquidé, M. Dauchet, F. Jacquemard, Pumping, cleaning and symbolic constraints solving, in: Proc. 21st Internat. Colloq. on Automata, Languages, and Programming (ICALP), Lecture Notes in Computer Science, Vol. 820, Springer, Berlin, 1994, pp. 436–449.
[7] Th. Colcombet, Rewriting in the partial algebra of typed terms modulo AC, Electron. Notes Theoret. Comput. Sci. 68 (2002).
[8] H. Comon, F. Jacquemard, Ground reducibility and automata with disequality constraints, in: Proc. 11th Symp. on Theoretical Computer Science (STACS), Lecture Notes in Computer Science, Vol. 820, Springer, Berlin, 1994, pp. 151–162.
[9] S. DalZilio, D. Lugiez, XML schema, tree logic and sheaves automata, in: Proc. 14th Internat. Conf. on Rewriting Techniques and Applications (RTA), Lecture Notes in Computer Science, Vol. 2706, Springer, Berlin, 2003, pp. 246–253.
[10] L. Dickson, Finiteness of the odd perfect and primitive abundant numbers with $n$ distinct prime factors, Amer. J. Math. 35 (1913) 113–122.

[11] S. Feferman, R.L. Vaught, The first-order properties of products of algebraic systems, Fund. Math. (1959) 57–103.
[12] J. Ferrante, C.W. Rackoff, The computational Complexities of Logical Theories, Lecture Notes in Mathematics, Vol. 718, Springer, Berlin, 1978.
[13] M.J. Fischer, M.O. Rabin, Super-exponential complexity of Presburger arithmetic, in: Complexity of Computation, SIAM-AMS Proc., Vol. 7, 1974, pp. 27–41.
[14] S. Ginsburg, Mathematical Theory of Context-free Languages, Mc-Graw-Hill, New York, 1966.
[15] S. Ginsburg, E. Spanier, Semigroups, Presburger formulas and languages, Pacific J. Math. 16 (1966) 285–296.
[16] J. Goubault-Larrecq, K.N. Verma, Alternating two-way AC-tree automata, Technical Report 02-11, LSV, ENS Cachan, September 2002.
[17] F. Klaedtke, H. Rueß, Parikh automata and monadic second-order logics with linear cardinality constraints, Technical Report 177, Albert-Ludwigs-Universität Freiburg, July 2002.
[18] O. Kupferman, U. Sattler, M.Y. Vardi, The complexity of the graded mu-calculus, in: Proc. 18th Conf. on Automated Deduction (CADE), Lecture Notes in Computer Science, Vol. 2392, Springer, Berlin, 2002, pp. 423–437.
[19] D. Lugiez, A good class of tree automata, in: Proc. 25th Internat. Colloq. on Automata, Languages, and Programming (ICALP), Lecture Notes in Computer Science, Vol. 1443, Springer, Berlin, 1998, pp. 409–420.
[20] D. Lugiez, Counting and equality constraints for multitree automata, in: Proc. 6th Conf. on Foundations of Software Science and Computational Structures (FOSSACS),Lecture Notes in Computer Science, Vol. 2620, Springer, Berlin, 2003, pp. 328–342.
[21] D. Lugiez, J.L. Moysset, Tree automata help one to solve equational formulae in AC-theories, J. Symbolic Comput. 18 (4) (1994) 297–318.
[22] F. Maurin, The theory of integer multiplication with order restricted to primes is decidable, J. Symbolic Logic 62 (1) (1997) 123–130.
[23] M. Murata, Extended path expression for XML, in: Proc. 20th Symp. on Principles of Database Systems (PODS), ACM Press, New York, 2001, pp. 126–137.
[24] A. Muscholl, T. Schwentick, H. Seidl, Numerical document queries, in: Proc. 22nd ACM Symp. on Principles of Database Systems (PODS), ACM Press, New York, 2003, pp. 155–166.
[25] J. Niehren, A. Podelski, Feature automata and recognizable sets of feature trees, in: Proc. 4th Internat. Conf. on Theory and Practice of Software Development (TAPSOFT), Lecture Notes in Computer Science, Vol. 668, Springer, Berlin, 1993, pp. 356–375.
[26] H. Ohsaki, Beyond the regularity: equational tree automata for associative and commutative theories, in: Proc. 15th Internat. Workshop Computer Science Logic, (CSL), Lecture Notes in Computer Science, Vol. 2142, Springer, Berlin, 2001, pp. 539–553.
[27] C. Pair, A. Quéré, Définition et étude des bilangages réguliers, Inform. Control 13 (6) (1968) 565–593.
[28] K.N. Verma, Automates d'arbres bidirectionnels modulo théories équationnelles, Ph.D. Thesis, ENS-Cachan, September 2003, http://www.lsv.ens-cachan.fr/Publis/PAPERS/Verma-these.ps.
[29] K.N. Verma, Two-way equational tree automata for AC-like theories; decidability and closure properties, in: Proc. 14th Internat. Conf. on Rewriting Techniques and Applications, Lecture Notes in Computer Science, Vol. 2706, Springer, Berlin, 2003, pp. 180–196.