

# Tree Automata for Non-Linear Arithmetic

Naoki Kobayashi<sup>1</sup> and Hitoshi Ohsaki<sup>2</sup>

<sup>1</sup> Tohoku University, Japan  
koba@ecei.tohoku.ac.jp

<sup>2</sup> National Institute of Advanced Industrial Science and Technology, Japan  
ohsaki@ni.aist.go.jp

**Abstract.** Tree automata modulo associativity and commutativity axioms, called *AC tree automata*, accept trees by iterating the transition modulo equational reasoning. The class of languages accepted by *monotone AC tree automata* is known to include the solution set of the inequality  $x \times y \geq z$ , which implies that the class properly includes the AC closure of regular tree languages. In the paper, we characterize more precisely the expressiveness of monotone AC tree automata, based on the observation that, in addition to polynomials, a class of exponential constraints (called *monotone exponential Diophantine inequalities*) can be expressed by monotone AC tree automata with a minimal signature. Moreover, we show that a class of arithmetic logic consisting of monotone exponential Diophantine inequalities is definable by monotone AC tree automata. The results presented in the paper are obtained by applying our novel tree automata technique, called *linearly bounded projection*.

## 1 Introduction

When reasoning about system properties of complex software automatically, analysis tools are often required to deal with arithmetic constraints together with system transitions. Safety-critical software used in automobiles, airplanes, and spacecrafts are examples whose internal transitions are triggered according to whether a certain arithmetic condition is satisfied. Hybrid automata ([8]) is a typical framework that provides a formalism for modeling such hybrid systems. However, interesting decision problems about hybrid automata are undecidable in most cases.

Equational tree automata were proposed in [17] as an extension of tree automata, in which equational reasoning is allowed at each transition step. The idea of the transition *modulo* equivalence looks simple, but the flexibility introduced in the framework turns out to define an enriched classification of formal tree languages. It is known that tree automata are closely related to context-free grammars: the leaf language of trees accepted by a regular tree automaton is a context-free language, and conversely, the set of derivation trees generated by a context-free grammar is a tree language accepted by a regular tree automaton [20]. But therefore, tree encoding by regular tree automata is no longer powerful enough to express linear arithmetic. On the other hand, tree automata

with associativity and commutativity (AC) axioms are expressive enough to encode Presburger formulas. In fact, there are several papers discussing, based on equational tree automata or a related framework, how to embed linear arithmetic feature in XML type checking [4] and query processing [2].

Tree automata with AC axioms are called AC tree automata (AC-TA for short). Depending on which types of transition rules are equipped, we distinguish two classes of AC-TA ([17]): *regular* AC tree automata and *monotone* AC tree automata. The former is allowed to have transition rules of the form  $f(\alpha_1, \dots, \alpha_n) \rightarrow \beta$  or  $\alpha \rightarrow \beta$ . The latter class may additionally have rules of the form  $f(\alpha_1, \dots, \alpha_n) \rightarrow f(\beta_1, \dots, \beta_n)$ . One can easily observe that monotone AC tree automata are a super-class of regular AC tree automata, though it is not obvious that the language hierarchy of the two classes is strict. Indeed, in the absence of AC axioms, the additional type of transition rules does not increase the expressive power of tree automata.

In [15], however, Ohsaki *et al.* showed that tree languages accepted by monotone AC tree automata are not closed under complement. This implies that the class of monotone AC tree automata is a proper super-class of regular AC tree automata, as tree languages accepted by regular AC tree automata are closed under all Boolean operations. They also showed that there exists a monotone AC tree automaton whose accepted language  $L$  satisfies  $t \in L$  iff  $|t|_a \times |t|_b \geq |t|_c \wedge |t|_d = 1 \wedge |t|_e = 2$ , where  $|t|_a$  denotes the number of occurrences of a constant  $a$  in a tree  $t$ . This example reveals that monotone AC tree automata are a candidate framework for representing non-linear arithmetic constraints.

The goal of this paper is to investigate the expressive power of monotone AC tree automata. Although the closure properties of this class have been studied extensively [15, 17], little is known about the expressiveness. So we devote our attentions to the problem of defining the class of arithmetic reducible to monotone AC tree automata. Arithmetic constraints of particular interest in the paper are of the form of  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$ , such that  $E(\mathbf{x}_n)$  is an arithmetic expression formed of non-negative integers, first-order variables, addition, multiplication, and exponentiation. The right-hand side  $L(\mathbf{x}_n)$  is a linear polynomial with integer coefficients. We call the above constraint a *monotone exponential Diophantine inequality*. And a formula consisting of monotone exponential Diophantine inequalities is called a *monotone exponential Diophantine formula*. The syntax of the formulas is given later.

The paper is organized as follows. The remainder of the section reviews some background and related work. The next section introduces equational tree automata. The previous results and related properties are also presented. In Section 3, we demonstrate two monotone AC tree automata for multiplication and exponentiation. That is, we define an AC-TA  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$  such that  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$  accepts a tree  $t$  if and only if  $|t|_a \times |t|_b \geq |t|_c$ . Similarly, we define  $\mathcal{A}_{\mathcal{E}}^{\text{exp}}$  such that  $\mathcal{A}_{\mathcal{E}}^{\text{exp}}$  accepts a tree  $t$  if and only if  $|t|_a^{|t|_b} \geq |t|_c$ . Unlike the monotone AC tree automaton shown in the previous paper [15],  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$  does not require extra symbols to interpret  $|t|_a \times |t|_b \geq |t|_c$ . In Section 4, we discuss a decidable sub-class of exponential Diophantine formulas. In particular, we show through tree automata techniques

that the satisfiability of monotone Diophantine formulas is decidable, in contrast to the undecidability of Hilbert’s 10th problem for non-negative solutions. In Section 5, we show that every monotone exponential Diophantine inequality is *monotone AC tree automata definable*. This means that one can construct a monotone AC tree automaton over a minimal signature whose accepted language  $M$  satisfies  $t \in M$  iff  $E(|t|_{a_1}, \dots, |t|_{a_n}) \geq L(|t|_{a_1}, \dots, |t|_{a_n})$ . Our proof is based on a special projection, called a *linearly bounded projection*. This result is *not* an immediate consequence of the previous observation that  $x \times y \geq z$  and  $x^y \geq z$  are monotone AC-TA definable. For instance,  $x^3 \geq z$  is equivalent to  $\exists y(x^2 \geq y \wedge x \times y \geq z)$ ; however, we do not know whether the class of monotone AC tree automata is closed under projection in general. Finally, in Section 6, we conclude the paper by summarizing the results and remaining questions.

## Related Work

Given a signature  $F$  that contains only an AC symbol  $\otimes$  and constants, one can regard Petri nets as ground AC rewriting systems over the signature  $F$ . A configuration of a net  $\mathcal{N}$  is a tree  $t$  over  $F$ , such that  $|t|_a$  is the number of *tokens* on a *place*  $a$  of  $\mathcal{N}$ . In the setting, transition of  $\mathcal{N}$  with the configuration  $t$  is performed by ground AC rewrite rules. For instance, the transition consuming “a token on the place  $a$  and another token on  $b$ ” and producing “two tokens on  $a$  and a token on  $c$ ” is represented by the rewrite rule  $a \otimes b \rightarrow a \otimes a \otimes c$ . For an initial configuration  $t_0$  of the net  $\mathcal{N}$ , the set of all reachable trees from  $t_0$  is called a *reachability set*.

Monotone AC tree automata over the above signature  $F$  are a sub-class of Petri nets. So far we do not know whether Petri nets are properly more expressive than monotone AC tree automata, but it is worth investigating a reasonable sub-class of Petri nets for automated reasoning purposes. In fact, the membership problem for monotone AC tree automata is PSPACE-complete, while the upper bound complexity of its Petri net counterpart (the reachability problem) is not known. In [7] Hack showed that for a given polynomial  $P(\mathbf{x}_n)$  with non-negative integer coefficients, there effectively exists a Petri net such that the projection of its reachability set onto the first  $n + 1$  places is  $\{(x_1, \dots, x_n, y) \mid P(\mathbf{x}_n) \geq y\}$ . This observation is, however, not directly applied to the case of monotone AC tree automata, because the transition rules of monotone AC tree automata are more restricted than those of Petri nets.

## 2 Preliminaries

We assume the reader is familiar with term rewriting [1] and tree automata [3]. An *equational theory* is a pair  $\mathcal{E} = (F, E)$  of a signature  $F$  (a finite set of function symbols, each with an associated unique *arity*) and a finite set  $E$  of orientation-sensitive axioms over function symbols in  $F$  possibly with some variables. The binary relation  $\rightarrow_{\mathcal{E}}$  induced by  $\mathcal{E}$  is the rewrite relation, i.e.  $s \rightarrow_{\mathcal{E}} t$  if there exist an axiom  $l \approx r$  in  $E$ , a context  $C[\ ]$  and a substitution  $\sigma$  such that  $s = C[l\sigma]$  and  $t = C[r\sigma]$ . The equivalence closure and the reflexive-transitive of  $\rightarrow_{\mathcal{E}}$  are denoted

$=_{\mathcal{E}}$  and  $\rightarrow_{\mathcal{E}}^*$ , respectively. For a binary function symbol  $f \in F$ , the associativity axiom is written as  $f(f(x, y), z) \approx f(x, f(y, z))$ , and the commutativity axiom is  $f(x, y) \approx f(y, x)$ . The associative and commutative theory (AC-theory) is an equational theory whose axioms are the associativity and commutativity for some of the binary function symbols.

A (*monotone*) *equational tree automaton* (ETA) is a 4-tuple  $(\mathcal{E}, Q, Q_{\text{fin}}, \Delta)$ , that consists of an equational theory  $\mathcal{E}$ , a finite set  $Q$  of states disjoint from symbols in  $F$ , a subset  $Q_{\text{fin}}$  of  $Q$ , and a finite set  $\Delta$  of transition rules whose shapes are in the following forms:

$$\begin{array}{ccc} \text{(REGULAR)} & \text{(EPSILON)} & \text{(MONOTONE)} \\ f(\alpha_1, \dots, \alpha_n) \rightarrow \beta_1 & \alpha_1 \rightarrow \beta_1 & f(\alpha_1, \dots, \alpha_n) \rightarrow f(\beta_1, \dots, \beta_n) \end{array}$$

such that  $f \in F$ ,  $\text{arity}(f) = n$  and  $\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n \in Q$ . State symbols occurring at different positions,  $\alpha_i, \alpha_j$  or  $\beta_i, \beta_j$  ( $i \neq j$ ), in transition rules can be the same. In the paper we write  $\mathcal{A}_{\mathcal{E}}$  to denote an ETA  $\mathcal{A}$  equipped with an equational theory  $\mathcal{E}$ .

The move relation  $\rightarrow_{\mathcal{A}_{\mathcal{E}}}$  is the equational rewrite relation of  $\mathcal{A}_{\mathcal{E}}$ , that is,  $s \rightarrow_{\mathcal{A}_{\mathcal{E}}} t$  iff  $s =_{\mathcal{E}} C[l]$  and  $t =_{\mathcal{E}} C[r]$  for a transition rule  $l \rightarrow r$  in  $\Delta$  and a context  $C[\ ]$ . A tree  $t$  is *accepted* by  $\mathcal{A}_{\mathcal{E}}$  if  $t \in \mathcal{T}_F$  and  $t \rightarrow_{\mathcal{A}_{\mathcal{E}}}^* \alpha$  for some  $\alpha \in Q_{\text{fin}}$ , and the set of trees accepted by  $\mathcal{A}_{\mathcal{E}}$  is denoted  $\mathcal{L}(\mathcal{A}_{\mathcal{E}})$  in the paper. A tree language accepted by  $\mathcal{A}_{\mathcal{E}}$  is called  $\mathcal{E}$ -*monotone*. In particular, if  $\mathcal{E}$  is the AC-theory, the accepted tree language is called AC-monotone.

**Proposition 1 ([15, 17]).** *The class of monotone AC-TA is effectively closed under union and intersection, but is not closed under complement. Moreover, the membership and emptiness problems are decidable, but the inclusion problem is undecidable.*  $\square$

If transition rules in  $\Delta$  of  $\mathcal{A}_{\mathcal{E}}$  are all in the REGULAR form,  $\mathcal{A}_{\mathcal{E}}$  is called a *regular* ETA. A regular ETA with the AC-theory  $\mathcal{E}$  is a regular AC-TA. A tree language accepted by the regular ETA (resp. the regular AC-TA) is called  $\mathcal{E}$ -regular (AC-regular). If  $\mathcal{E}$  is the free theory ( $E = \emptyset$ ), a regular ETA is called, as is followed by customary, a *regular tree automaton*, and a tree language accepted by the regular tree automaton is called *regular*. The above definition of “regularity” (regular tree automata and regular tree languages) is identical to the standard notion, e.g. found in [3].

*Leaves* of a tree  $t$ , denoted  $\text{leaf}(t)$ , are the sequence in left-to-right order of constants occurring in the tree:  $\text{leaf}(f(t_1, \dots, t_n)) = \text{leaf}(t_1) \cdots \text{leaf}(t_n)$  if  $n \geq 1$ ;  $\text{leaf}(a) = a$ , otherwise. The *leaf language* associated to a tree language  $L$  is the set of leaves obtained from  $L$ . The *commutative image* of a tree language  $L$  is the commutative closure of a leaf language of  $L$ . Parikh mapping  $\Psi_F$  ([18]) is the mapping from tree languages to the commutative image of the languages: Given a tree language  $L$  whose signature  $F$  contains the set of constants  $F_0 = \{ \mathbf{a}_1, \dots, \mathbf{a}_k \}$ , then  $\Psi_F(L) = \{ (|t|_{\mathbf{a}_1}, \dots, |t|_{\mathbf{a}_k}) \mid \exists t \in L \}$ . A subset of vectors in  $\mathbb{N}^k$  is *linear* if the set is  $\{ v \mid \exists x_1, \dots, x_n \in \mathbb{N} : v = c + x_1 v_1 + \cdots + x_n v_n \}$  for some vectors  $c, v_1, \dots, v_n$  in  $\mathbb{N}^k$ . We call the vector expression  $c + x_1 v_1 + \cdots + x_n v_n$

a *non-negative vector addition system*<sup>3</sup> (NNVAS). The finite union of linear sets is a *semi-linear* set. By definition, every linear set is non-empty. However, the empty set is semi-linear, being the union of zero linear sets. A finite subset of vectors is also semi-linear, while the finite subset is not linear if it contains more than one element.

The Parikh image  $\Psi_F(L)$  of a regular tree language  $L$  is effectively semi-linear, meaning that: Given a regular tree automaton  $\mathcal{A}$ , one can construct a finite sequence of NNVAS's  $V_1, \dots, V_n$  ( $n \geq 0$ ) such that the union of linear sets generated by  $V_1, \dots, V_n$  coincides with  $\Psi_F(\mathcal{L}(\mathcal{A}))$ .

We introduce the two special operations for subsets of vectors. The  $i$ -th *projection*  $\text{pr}_i$  of a subset  $W$  of  $\mathbb{N}^k$  ( $1 \leq i \leq k$ ) is the mapping from  $\mathbb{N}^k$  to  $\mathbb{N}^{k-1}$  such that  $\text{pr}_i(W) = \{ (v(1), \dots, v(i-1), v(i+1), \dots, v(k)) \mid \exists v \in W \}$ . Similarly, the  $i$ -th *cylindrification*  $\text{cy}_i$  of  $W$  is the mapping from  $\mathbb{N}^k$  to  $\mathbb{N}^{k+1}$  such that  $\text{cy}_i(W) = \{ (v(1), \dots, v(i-1), x, v(i), \dots, v(k)) \mid \exists v \in W, x \in \mathbb{N} \}$ . Here we denote  $v(i)$  for the  $i$ -th element of a vector  $v$ .

**Proposition 2 ([6]).** *The class of semi-linear sets is effectively closed under Boolean operations, projection, and cylindrification. Moreover, the emptiness (and thus, the membership and inclusion problems also) are decidable.  $\square$*

Using the property, one can show that the class of AC-regular tree languages is closed under Boolean operations. Similarly, Dal Zilio and Lugies for multitree automata [4, 13] and Verma and Goubault-Larrecq for two-way AC-tree automata [21] showed that their equationally extended tree automata enjoy the closure properties of Boolean operations.

As the benefit from the closure properties of Boolean operations and the positive decidability results, several fundamental decision problems are translated to language problems in the class of regular AC-TA. Satisfiability of monadic second-order logic with Presburger arithmetic (Presburger MSO) [19] is one of the examples.

What about then the class of monotone AC-TA?

### 3 Monotone AC Tree Automata for Multiplication and Exponentiation

Hereafter in the following sections, we consider a special class of monotone AC-TA over a *flat signature*. A signature  $F$  is flat if  $F$  consists of one AC symbol  $\mathbf{f}$  and constants only. We write  $F_0$  for the set of all constants in  $F$ . So  $F = \{\mathbf{f}\} \cup F_0$ .

Ohsaki *et al.* showed in [15] that there exists a monotone AC-TA whose language  $L$  over  $F$  satisfies  $\Psi_F(L) = \{ (x, y, z, 1, 2) \in \mathbb{N}^5 \mid x \times y \geq z \wedge x + y + z > 0 \}$ . The class of regular AC-TA or the related automata does not satisfy this property due to the expressiveness limited by linear arithmetic. This example thus reveals that there exists the strict hierarchy between monotone AC-TA and regular AC-TA.

<sup>3</sup> Vector addition systems (e.g. [12]) are equipped with vectors  $v_1, \dots, v_n$  from  $\mathbb{Z}^k$ .

---


$$\begin{aligned}
F &: f \ a \ b \ c \\
E &: f(f(x, y), z) \approx f(x, f(y, z)) \quad f(x, y) \approx f(y, x) \\
\Delta_1 &: a \rightarrow \lambda \quad b \rightarrow \lambda \quad f(\lambda, \lambda) \rightarrow \lambda \\
\Delta_2 &: a \rightarrow \alpha \quad b \rightarrow \beta \quad c \rightarrow \gamma \quad \beta \rightarrow \beta_1 \quad \beta_1 \rightarrow \beta_2 \quad \beta_2 \rightarrow \delta \quad f(\alpha, \delta) \rightarrow \delta \\
&\quad f(\alpha, \beta_1) \rightarrow f(\alpha_1, \beta_1) \quad f(\alpha_1, \gamma) \rightarrow \alpha_2 \quad f(\alpha_2, \beta_2) \rightarrow f(\alpha, \beta_2) \quad f(\beta_2, \beta) \rightarrow \beta_1 \\
Q &: \alpha \ \alpha_1 \ \alpha_2 \ \beta \ \beta_1 \ \beta_2 \ \gamma \ \delta \ \lambda \\
Q_{\text{fin}} &: \lambda, \delta
\end{aligned}$$

**Fig. 1.**  $\mathcal{A}^{\text{mult}} = (\mathcal{E}, Q, Q_{\text{fin}}, \Delta_1 \cup \Delta_2)$  and  $\mathcal{E} = (F, E)$

---

In this section, we show that there exist monotone AC tree automata  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$  and  $\mathcal{A}_{\mathcal{E}}^{\text{exp}}$  such that

$$\begin{aligned}
\Psi_F(\mathcal{L}(\mathcal{A}_{\mathcal{E}}^{\text{mult}})) &= \{ (x, y, z) \in \mathbb{N}^3 \mid x \times y \geq z \wedge x + y + z > 0 \} \\
\Psi_F(\mathcal{L}(\mathcal{A}_{\mathcal{E}}^{\text{exp}})) &= \{ (x, y, z) \in \mathbb{N}^3 \mid x^y \geq z \wedge x + y > 0 \}
\end{aligned}$$

Note that the condition  $x + y + z > 0$  is necessary since our tree encoding does not allow to have a tree whose Parikh image is  $(0, 0, 0)$ .

In the previous paper ([15]), it was unknown whether there is an automaton with a minimal signature (i.e., an automaton without any extra constants). As far as we know, it was also unknown whether there exists a monotone AC tree automaton for exponentiation even when extra symbols are allowed.

### 3.1 AC Tree Automaton for “ $x \times y \geq z$ ”

**Lemma 1.** *Let  $F = \{f, a, b, c\}$  with one AC-symbol  $f$  and three constants  $a, b, c$ . There exists a monotone AC-TA  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$  over  $F$  such that*

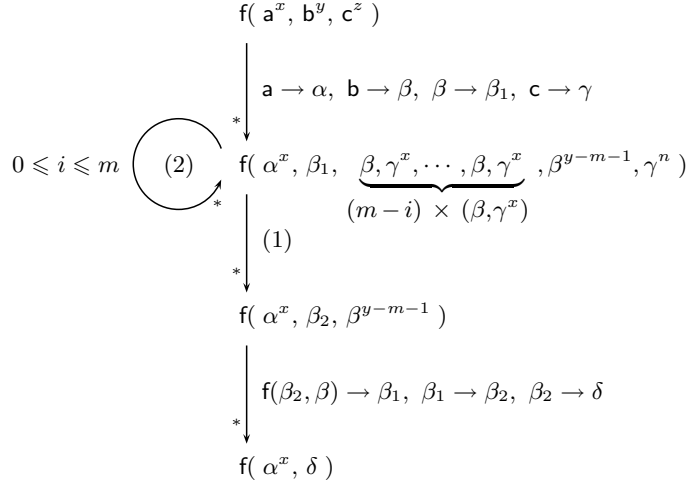
$$\Psi_F(\mathcal{L}(\mathcal{A}_{\mathcal{E}}^{\text{mult}})) = \{ (x, y, z) \in \mathbb{N}^3 \mid x \times y \geq z \wedge x + y + z > 0 \}.$$

□

Our example is exhibited in Fig. 1. Based on case analysis of the number of occurrences of  $c$ , the AC-TA  $\mathcal{A}^{\text{mult}} = (\mathcal{E}, Q, Q_{\text{fin}}, \Delta_1 \cup \Delta_2)$  is defined as the union of the two automata  $\mathcal{A}_1^{\text{mult}} = (\mathcal{E}, \{\lambda\}, \{\lambda\}, \Delta_1)$  and  $\mathcal{A}_2^{\text{mult}} = (\mathcal{E}, Q - \{\lambda\}, Q_{\text{fin}} - \{\lambda\}, \Delta_2)$ .

If  $|t|_c = 0$ , obviously  $t \xrightarrow{*}_{\Delta_1/\mathcal{E}} \lambda$ , and thus  $\mathcal{A}_1^{\text{mult}}$  accepts  $t$ . Otherwise  $|t|_c > 0$ , we observe for the recursive computation that  $|t|_a \times |t|_b \geq |t|_c$  iff  $|t|_c = m \times |t|_a + n$  for some  $m, n$  such that  $0 \leq m < |t|_b$  and  $0 < n \leq |t|_a$ . That means, a tree  $t = f(a^x, b^y, c^z)$  with  $x \times y \geq z$  is considered to be the tree with the following leaves:

$$f(a^x, b, \underbrace{b, c^x, \dots, b, c^x}_{m \times (b, c^x)}, b^{y-m-1}, c^n).$$



**Fig. 2.** A derivation of  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z)$

Here  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z)$  represents that it has  $x$  occurrences of the constant  $\mathbf{a}$ ,  $y$  occurrences of  $\mathbf{b}$ , and  $z$  occurrences of  $\mathbf{c}$ , respectively.

Regarding the derivation from  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z)$ , we have the two properties

- (1)  $f(\alpha^x, \beta_1, \gamma^n) \xrightarrow{*}_{\Delta_2/\mathcal{E}} f(\alpha^x, \beta_2)$  if  $n \leq x$ ,
- (2)  $f(\alpha^x, \beta_1, \gamma^n, \beta) \xrightarrow{*}_{\Delta_2/\mathcal{E}} f(\alpha^x, \beta_1)$  if  $n \leq x$ .

The derivation in the property (1) is obtained from  $f(\alpha^x, \beta_1, \gamma^n)$  by applying  $f(\alpha, \beta_1) \rightarrow f(\alpha_1, \beta_1)$  and  $f(\alpha_1, \gamma) \rightarrow \alpha_2$  repeatedly  $n$  times, and then by applying  $\beta_1 \rightarrow \beta_2$  and  $f(\alpha_2, \beta_2) \rightarrow f(\alpha, \beta_2)$ . Furthermore, when  $n \leq x$ , we have

$$f(\alpha^x, \beta_1, \gamma^n, \beta) \xrightarrow{*}_{\Delta_2/\mathcal{E}} f(\alpha^x, \beta_2, \beta) \xrightarrow{\Delta_2/\mathcal{E}} f(\alpha^x, \beta_1)$$

that is the derivation in the property (2). According to this second property,  $t = f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z)$  reaches  $f(\alpha^x, \beta_1, \beta^{y-m-1}, \gamma^n)$  by  $\Delta_2/\mathcal{E}$ . Thus, by the first property,  $t$  reaches  $f(\alpha^x, \beta_2, \beta^{y-m-1})$ . That means, we have the derivation

$$t \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} f(\alpha^x, \beta^y, \gamma^z) \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} f(\alpha^x, \beta_1, \beta^{y-m-1}, \gamma^n) \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} f(\alpha^x, \beta_2, \beta^{y-m-1}).$$

Since  $f(\beta_2, \beta) \xrightarrow{*}_{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} \beta_2$  and  $\beta_2 \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} \delta$ , the derivation from  $t$  reaches the final state  $\delta$ :

$$f(\alpha^x, \beta_2, \beta^{y-m-1}) \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} f(\alpha^x, \beta_2) \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} f(\alpha^x, \delta) \xrightarrow{*}_{\mathcal{A}_{2\mathcal{E}}^{\text{mult}}} \delta.$$

Therefore, every tree  $t$  over  $F = \{f, \mathbf{a}, \mathbf{b}, \mathbf{c}\}$  is accepted if  $|t|_{\mathbf{a}} \times |t|_{\mathbf{b}} \geq |t|_{\mathbf{c}}$ . The derivation from  $t$  to  $f(\alpha^x, \delta)$  by  $\mathcal{A}_{2\mathcal{E}}^{\text{mult}}$  is illustrated in Fig. 2.

See Appendix A.1 for the completeness proof of  $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$ .

### 3.2 AC Tree Automaton for Exponentiation

Let us first illustrate the AC tree automaton  $\mathcal{A}^{\text{qr}} = (\mathcal{E}, Q, \{\lambda, \delta\}, \Delta_1 \cup \Delta_2)$  that accepts  $\{t \mid 2^{|t|_a} \geq |t|_b\}$ .

$$\begin{aligned} \Delta_1 : & \mathbf{a} \rightarrow \lambda \quad \mathbf{f}(\lambda, \lambda) \rightarrow \lambda \\ \Delta_2 : & \mathbf{a} \rightarrow \alpha \quad \mathbf{b} \rightarrow \beta \quad \mathbf{b} \rightarrow \delta \quad \alpha \rightarrow \gamma_1 \quad \gamma_1 \rightarrow \gamma_3 \\ & \left. \begin{array}{l} \mathbf{f}(\beta, \gamma_1) \rightarrow \mathbf{f}(\beta_1, \gamma_2) \\ \mathbf{f}(\beta, \gamma_2) \rightarrow \gamma_1 \end{array} \right\} (1) \quad \left. \begin{array}{l} \mathbf{f}(\beta_1, \gamma_3) \rightarrow \mathbf{f}(\beta, \gamma_3) \\ \mathbf{f}(\alpha, \gamma_3) \rightarrow \gamma_1 \end{array} \right\} (2) \quad \left. \begin{array}{l} \mathbf{f}(\beta, \gamma_3) \rightarrow \delta \\ \mathbf{f}(\alpha, \delta) \rightarrow \delta \end{array} \right\} (3) \end{aligned}$$

Regarding the above transition rules, we note that for every  $t \in \mathcal{T}_F$ , if  $|t|_b = 0$ ,  $t$  is accepted by  $\mathcal{A}_{\mathcal{E}}^{\text{qr}}$  using the transition rules in  $\Delta_1$ ; otherwise,  $t$  is accepted by the rules in  $\Delta_2$ . If  $|t|_a = 0$  and  $|t|_b = 1$ , it is obvious. If  $|t|_a > 0$ , the derivation from  $t$  to  $\delta$  by  $\Delta_2$  is illustrated in Fig. 3.

The derivation in Fig. 3. is separated to the three phases: First,  $\mathbf{a}, \mathbf{b}$  in  $t$  are replaced by  $\alpha, \beta$ , and one of  $\alpha$ 's is replaced by  $\gamma_1$ . This is for initializing the computation. Next, we apply the computation for “reducing by half” the number of occurrences of  $\beta$ . At most the half of  $\beta$ 's are replaced by  $\beta_1$ , and the same number of  $\beta$ 's are removed. This transition is performed by applying the rules in (1). In the third phase,  $\beta_1$ 's are replaced back to  $\beta$ 's by the rule  $\mathbf{f}(\beta_1, \gamma_3) \rightarrow \mathbf{f}(\beta, \gamma_3)$  in (2). Therefore, we have the derivation

$$\mathbf{f}(\alpha^m, \gamma_1, \beta^n) \xrightarrow{*}_{\mathcal{A}_{\mathcal{E}}^{\text{qr}}} \mathbf{f}(\alpha^m, \gamma_3, \beta^{n-k}) \text{ such that } 0 \leq k \leq \frac{1}{2}n.$$

If  $n - k = 1$ , the transition rule  $\mathbf{f}(\beta, \gamma_3) \rightarrow \delta$  in (3) is applied to  $\mathbf{f}(\alpha^m, \gamma_3, \beta^{n-k})$ , and the remaining  $\alpha$ 's are eliminated by using the rule  $\mathbf{f}(\alpha, \delta) \rightarrow \delta$ . Otherwise,  $\mathbf{f}(\alpha, \gamma_3) \rightarrow \gamma_1$  is applied for iterative computation, and it restarts the computation from the reduction-by-half phase.

Generalizing the above observation, we obtain the next lemma.

**Lemma 2.** *Let  $F = \{\mathbf{f}, \mathbf{a}, \mathbf{b}, \mathbf{c}\}$  with one AC-symbol  $\mathbf{f}$  and three constants  $\mathbf{a}, \mathbf{b}, \mathbf{c}$ . There exists a monotone AC-TA  $\mathcal{A}_{\mathcal{E}}^{\text{xp}}$  over  $F$  such that*

$$\Psi_F(\mathcal{L}(\mathcal{A}_{\mathcal{E}}^{\text{xp}})) = \{(x, y, z) \in \mathbb{N}^3 \mid x^y \geq z \wedge x + y > 0\}$$

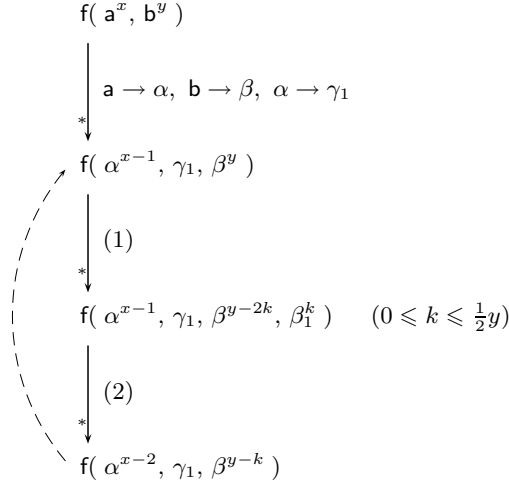
*Proof.* Appendix A.2. □

## 4 Monotone Exponential Diophantine Formulas

An arithmetic constraint over a finite set of variables  $x_1, \dots, x_n$  ( $n \geq 0$ ) is an *exponential Diophantine formula* if the formula is in  $D$  of the following syntax:

$$\begin{aligned} D ::= & A \mid \neg(D) \mid D \vee D \mid D \wedge D \\ A ::= & \exists x_i (D) \mid \sum_{i \in I} a_i x_i \geq b \mid x_i \times x_j \geq x_k \mid x_i^{x_j} \geq x_k \end{aligned}$$

such that  $a_i, b \in \mathbb{Z}$  for all  $i \in I$ . If a formula  $\psi$  does not contain an atomic formula  $x_i^{x_j} \geq x_k$ ,  $\psi$  is simply called a *Diophantine formula*.



**Fig. 3.** Three phases in the derivation from  $f(a^x, b^y)$

---

A formula  $\psi$  is *satisfiable* if there exists an assignment  $\theta$  to free-variables in  $\psi$  such that  $\psi\theta$  is true. Similarly,  $\psi$  is *valid* if  $\psi\theta$  is true for any assignment  $\theta$ . For instance, the formula  $x \geq 1$  is satisfiable but not valid, because  $(x \geq 1)\{x \mapsto 0\}$  is not true. In the paper we write  $\llbracket \psi \rrbracket_{\mathbb{N}}$  for the set of all solution vectors of  $\psi$ , that is,  $v \in \llbracket \psi \rrbracket_{\mathbb{N}}$  iff  $v \in \mathbb{N}^n$  and  $\psi\{x_1 \mapsto v(1), \dots, x_n \mapsto v(n)\}$  is true. Note that  $\psi$  is satisfiable iff  $\llbracket \psi \rrbracket_{\mathbb{N}} \neq \emptyset$ ;  $\psi$  is valid iff  $\llbracket \psi \rrbracket_{\mathbb{N}} = \mathbb{N}^n$ .

A formula  $\psi$  is *logically equivalent* to  $\phi$ , denoted  $\psi \Leftrightarrow \phi$ , if for every assignment  $\theta$  to free-variables in formulas,  $\psi\theta$  is true iff  $\phi\theta$  is true. Obviously,  $\Leftrightarrow$  is an equivalence relation. See the reference, e.g. [9], for more logical background.

We define a new class of arithmetic constraints.

A formula  $\psi$  is *monotone* if none of the negative sub-formulas  $\neg(\phi)$  of  $\psi$  contains an atomic formula of the form  $x_i \times x_j \geq x_k$  or  $x_i^{x_j} \geq x_k$ .

For instance,  $x^2 \geq y$  is logically equivalent to the formula  $\exists z(x \times z \geq y \wedge x \geq z)$ . However, there is no monotone (exponential) Diophantine formula equivalent to  $x^2 = y$ , because every formula equivalent to  $x^2 = y$  has a negative sub-formula containing a non-linear constraint, e.g.  $(x^2 \geq y) \wedge \exists z(\neg(x^2 \geq z) \wedge (z = y + 1))$ .

Next we define arithmetic expressions over variables  $x_1, \dots, x_n$  ( $n \geq 0$ ).

$$E(\mathbf{x}_n) ::= a \mid x_i \mid E(\mathbf{x}_n) + E(\mathbf{x}_n) \mid E(\mathbf{x}_n) \times E(\mathbf{x}_n) \mid E(\mathbf{x}_n)^{E(\mathbf{x}_n)}$$

where  $a \in \mathbb{N}$ . Elements in  $E(\mathbf{x}_n)$  are called *exponentials*. We write  $P(\mathbf{x}_n)$  instead of  $E(\mathbf{x}_n)$  for a *polynomial* with non-negative integer coefficients. A *linear polynomial* (with arbitrary integer coefficients) is denoted  $L(\mathbf{x}_n)$ .

Note that for every  $E(\mathbf{x}_n)$  and  $L(\mathbf{x}_n)$ , one can find a monotone exponential Diophantine formula equivalent to  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$ , because:

$$\begin{aligned}
E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y &\Leftrightarrow \exists z_1, \exists z_2 (E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1 + z_2 \geq y) \\
E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y &\Leftrightarrow \exists z_1, \exists z_2 (E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1 \times z_2 \geq y) \\
E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y &\Leftrightarrow \exists z_1, \exists z_2 (E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1^{z_2} \geq y)
\end{aligned}$$

Given a polynomial  $P(\mathbf{x}_n)$  with *arbitrary* integer coefficients, the question  $\llbracket P(\mathbf{x}_n) = 0 \rrbracket_{\mathbb{N}} \stackrel{?}{=} \emptyset$  (HILBERT'S 10TH PROBLEM for non-negative solutions) is undecidable [14]. Therefore, the satisfiability question for (exponential) Diophantine formulas is undecidable.

A formula  $\psi$  is *linear Diophantine*, on the other hand, if every atomic formula of  $\psi$  is a linear constraint  $\sum_{i \in I} a_i x_i \geq b$  such that  $a_i, b \in \mathbb{Z}$  for all  $i \in I$ . In the literature a linear Diophantine formula is often called a *Presburger formula*. The solution set of a given Presburger formula is effectively semi-linear. And thus, the satisfiability of Presburger formulas is decidable [5].

Every linear Diophantine formula is monotone, but a monotone Diophantine formula may not be linear. Moreover,  $x^2 = y$  is not a monotone formula, while the formula is Diophantine. So we have the following relation among the classes of arithmetic constraints:

$$\text{Presburger} \subsetneq \text{monotone Diophantine} \subsetneq \text{Diophantine}$$

The following decidability result follows immediately from the results of the previous section and the closure properties of monotone AC tree automata.

**Theorem 1 (Satisfiability).** *Satisfiability of monotone exponential Diophantine formulas is decidable.*

*Proof.* We observe that  $\exists x(\psi) \vee \phi \Leftrightarrow \exists x(\psi \vee \phi)$  and  $\exists x(\psi) \wedge \phi \Leftrightarrow \exists x(\psi \wedge \phi)$  if  $x$  does not freely occur in  $\phi$ . This implies that given a monotone exponential Diophantine formula  $\delta$ , one can move existential quantifiers outside of  $\delta$ , so that  $\delta$  is logically equivalent to a formula  $\exists \mathbf{x}(\psi)$  for some  $\psi$  in  $S$ .

$$S ::= x \times y \geq z \mid x^y \geq z \mid C \mid S \wedge S \mid S \vee S$$

Here  $C$  ranges over the set of Presburger formulas. By Lemmas 1 and 2 and Proposition 1, one can construct  $\mathcal{A}_{\mathcal{E}}$  such that  $\Psi_F(\mathcal{L}(\mathcal{A}_{\mathcal{E}})) = \llbracket \psi \rrbracket_{\mathbb{N}} - \mathbf{0}$ . Obviously,  $\exists \mathbf{x}(\psi)$  is satisfiable if and only if  $\mathcal{L}(\mathcal{A}_{\mathcal{E}}) \neq \emptyset$  or  $\mathbf{0}$  is a solution of  $\psi$ . By Proposition 2,  $\mathcal{L}(\mathcal{A}_{\mathcal{E}}) \neq \emptyset$  is decidable. Moreover, the question if  $\mathbf{0}$  is a solution of  $\psi$  is decidable. Therefore, the satisfiability of  $\exists \mathbf{x}(\psi)$  is so.  $\square$

## 5 Linearly Bounded Projection and Definable Formulas

In this section, we focus on the main question, that is, the expressiveness of monotone AC tree automata relative to the first-order theory of arithmetic.

According to the examples in Section 3 (Lemmas 1 and 2), the class of tree languages accepted by monotone AC tree automata includes the solution sets of  $x \times y \geq z$  and  $x^y \geq z$ . We generalize this result, so that the class covers more complex arithmetic constraints, such as  $(x^2 + y)^x \geq z$ .

A formula  $\psi$  is *monotone AC-TA definable* if there effectively exists a monotone AC-TA  $\mathcal{A}_\mathcal{E}$  over a flat signature  $F$  such that the Parikh image of the accepted language is the set of all solution vectors except zero, i.e.  $\Psi_F(\mathcal{L}(\mathcal{A}_\mathcal{E})) = \llbracket \phi \rrbracket_{\mathbb{N}} - \mathbf{0}$ . In the following of the section, we attempt to characterize the expressive power of monotone AC tree automata in comparison with the class of arithmetic formulas. In particular, we show that every formula in the following sub-class of monotone exponential Diophantine formulas is monotone AC-TA definable.

$$M ::= E(\mathbf{x}_n) \geq L(\mathbf{x}_n) \mid C \mid M \vee M \mid M \wedge M$$

As appeared previously,  $C$  ranges over the set of Presburger formulas.

*Remark 1.* Our discussion in the sequel does not lose generality, though  $\mathbf{0}$  is excluded in the above definition. Indeed, there are several ways to deal with the zero vector in tree automata. For instance, one can introduce the special constant  $\hat{\mathbf{0}}$  such that the solution set  $\llbracket \phi \rrbracket_{\mathbb{N}}$  contains  $\mathbf{0}$  iff  $\mathcal{A}_\mathcal{E}$  accepts  $\hat{\mathbf{0}}$ . Note that the notion of definability is invariant in the treatment of  $\mathbf{0}$ .

We note that the class of languages of *regular* AC-TA over a flat signature (which is a *commutative context-free grammar* in formal languages) coincides with the class of solution sets of Presburger formulas, called Presburger sets, [5]. For Petri nets, the set  $\{(x_1, \dots, x_n, y) \mid P(\mathbf{x}_n) \geq y\}$  is a Petri net language if  $P(\mathbf{x}_n)$  is a polynomial with non-negative integer coefficients [7], while the transition rules of monotone AC tree automata are more restricted.

According to the above observation about regular AC-TA together with the fact that the class of monotone AC-TA subsumes regular AC-TA and is closed under union and intersection (Proposition 1), we have the following lemma.

**Lemma 3.** *Every Presburger formula is monotone AC-TA definable. Moreover, if formulas  $\psi_1$  and  $\psi_2$  are monotone AC-TA definable, so are  $\psi_1 \wedge \psi_2$  and  $\psi_1 \vee \psi_2$ .*  $\square$

As noted in the proof of Theorem 1, every monotone exponential Diophantine formula is logically equivalent to some formula  $\exists \mathbf{x}(\psi)$  such that  $\psi$  is a monotone AC-TA definable formula. Therefore, the definability of monotone exponential Diophantine formulas could follow immediately if the class of monotone AC-TA were closed under projection. However, it is unknown so far whether the class of monotone AC-TA is closed under *arbitrary* projection.

Our key observation is that the language class of monotone AC-TA is closed under a special projection, called *linearly bounded projection*, discussed below. It turns out that the linearly bounded projection suffices to show that the above sub-class  $M$  of monotone exponential Diophantine formulas is monotone AC-TA definable.

**Lemma 4 (LINEARLY BOUNDED PROJECTION: SPECIAL CASE).** *Given a monotone AC-TA  $\mathcal{A}_\mathcal{E}$  over a flat signature  $F$  containing constants  $\mathbf{a}$  and  $\mathbf{b}$ , one can construct a monotone AC-TA  $\mathcal{B}_\mathcal{E}$  over  $F - \{\mathbf{a}\}$  such that*

$$\mathcal{L}(\mathcal{B}_\mathcal{E}) = \{u \mid \exists t \in \mathcal{L}(\mathcal{A}_\mathcal{E}): |t|_{\mathbf{a}} \leq |t|_{\mathbf{b}} \text{ and } |t|_c = |u|_c \text{ for all } c \in F_0 - \{\mathbf{a}\}\}.$$

*Namely, if a formula  $\psi$  is monotone AC-TA definable, so is  $\exists x(x \leq y \wedge \psi)$ .*

*Proof (Outline).* We suppose, without loss of generality, that  $\mathcal{A} = (\mathcal{E}, Q_1, \{\gamma\}, \Delta_1)$  such that  $\Delta_1$  contains no transition rule for  $\mathbf{a}, \mathbf{b}$  except  $\mathbf{a} \rightarrow \alpha$  and  $\mathbf{b} \rightarrow \beta$ . The set  $Q_2$  of state symbols of  $\mathcal{B}_{\mathcal{E}}$  are the union of  $Q_1$  and the set  $\{M \in \text{mul}(Q_1) \mid |M| \leq 2\}$  of multisets of  $Q_1$  whose size is at most 2. (Here,  $\text{mul}(S)$  stands for the set of multisets of a set  $S$ .) Moreover, to distinguish multisets from sets, we write  $\{\cdot\}$  to denote a multiset, and  $\uplus$  for the multiset union. The set  $\Delta_2$  of transition rules of  $\mathcal{B}_{\mathcal{E}}$  consists of rules from  $(\Delta_1 - \{\mathbf{a} \rightarrow \alpha, \mathbf{b} \rightarrow \beta\}) \cup \{\mathbf{b} \rightarrow \{\alpha, \beta\}, \mathbf{b} \rightarrow \{\beta\}\}$  and the rules defined below:

$$\begin{aligned}
M \uplus \{p\} &\rightarrow M \uplus \{q\} && \text{if } p \rightarrow_{\mathcal{A}_{\mathcal{E}}}^* q \\
\{p, q\} &\rightarrow \{r\} && \text{if } f(p, q) \rightarrow_{\mathcal{A}_{\mathcal{E}}}^* r \\
f(M \uplus \{p\}, q) &\rightarrow M \uplus \{r\} \\
f(M \uplus \{p\}, \{q\}) &\rightarrow M \uplus \{r\} \\
f(M \uplus \{p\}, N \uplus \{q\}) &\rightarrow f(M \uplus \{r\}, N) \quad (N \neq \emptyset) \\
\{p, q\} &\rightarrow \{r, s\} && \text{if } f(p, q) \rightarrow_{\mathcal{A}_{\mathcal{E}}}^* f(r, s) \\
f(M \uplus \{p\}, q) &\rightarrow f(M \uplus \{r\}, s) \\
f(M \uplus \{p\}, N \uplus \{q\}) &\rightarrow f(M \uplus \{r\}, N \uplus \{s\})
\end{aligned}$$

such that  $p, q, r, s \in Q_1$ .

Let  $\mathcal{B}_{\mathcal{E}}$  be the automaton equipped with the above  $Q_2$  and  $\Delta_2$  such that the final states are  $\gamma$  and  $\{\gamma\}$ . For instance, the transition move  $C[\mathbf{a}, \mathbf{b}] \rightarrow_{\mathcal{A}_{\mathcal{E}}} C[\alpha, \mathbf{b}] \rightarrow_{\mathcal{A}_{\mathcal{E}}} C[\alpha, \beta]$  is simulated in  $\mathcal{B}_{\mathcal{E}}$  by the single transition  $D[\mathbf{b}] \rightarrow_{\mathcal{B}_{\mathcal{E}}} D[\{\alpha, \beta\}]$ . It is not difficult to show that for all  $t \in \mathcal{T}_F$  and  $u \in \mathcal{T}_{F - \{\mathbf{a}\}}$ , if

- $|t|_{\mathbf{a}} \leq |t|_{\mathbf{b}}$
- $|u|_c = |t|_c$  for all  $c \in F_0 - \{\mathbf{a}\}$ ,

then  $\mathcal{A}_{\mathcal{E}}$  accepts  $t$  iff there exists a derivation  $u \rightarrow_{\mathcal{B}_{\mathcal{E}}}^* \gamma$  or  $u \rightarrow_{\mathcal{B}_{\mathcal{E}}}^* \{\gamma\}$ .  $\square$

This lemma can be generalized as follows.

**Lemma 5 (LINEARLY BOUNDED PROJECTION).** *If  $L(\mathbf{x}_n)$  is a linear polynomial with non-negative integer coefficients, every monotone AC-TA  $\mathcal{A}_{\mathcal{E}}$  over the flat signature  $F = \{\mathbf{f}\} \cup \{\mathbf{a}\} \cup \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$  can be transformed to  $\mathcal{B}_{\mathcal{E}}$  over  $F - \{\mathbf{a}\}$  such that*

$$\mathcal{L}(\mathcal{B}_{\mathcal{E}}) = \{u \mid \exists t \in \mathcal{L}(\mathcal{A}_{\mathcal{E}}) : |t|_{\mathbf{a}} \leq L(|t|_{\mathbf{b}_1}, \dots, |t|_{\mathbf{b}_n}) \text{ and } |t|_{\mathbf{b}_i} = |u|_{\mathbf{b}_i} \text{ for all } i\}.$$

*Proof.* Appendix A.3.  $\square$

The following theorem is an immediate consequence.

**Theorem 2.** *Let  $L(\mathbf{y}_n)$  be a linear polynomial (with integer coefficients) such that free-variables  $\mathbf{y}_n$  in  $L$  do not contain  $x$ . Then, if  $\psi$  is monotone AC-TA definable, so is  $\exists x (x \leq L(\mathbf{y}_n) \wedge \psi)$ .*  $\square$

Let us demonstrate the linearly bounded projection as a tool for translating exponential constraints by monotone AC tree automata.

*Example 1.* Consider the super-exponential inequality  $x^{x^x} \geq y$ . This constraint is equivalent to  $(x = 1 \wedge 1 \geq y) \vee \exists z (x \geq 2 \wedge x^z \geq y \wedge x^x \geq z)$ . Let  $\psi$  be the formula  $x \geq 2 \wedge x^z \geq y \wedge x^x \geq z$ . Then  $\psi$  is monotone AC-TA definable (Lemma 2). Note that the definability of  $x^x \geq y$  follows from Theorem 2, meaning that,  $x^x \geq y$  is logically equivalent to  $\exists w (w \leq x \wedge x^w \geq y \wedge w \geq x)$  and the sub-formula  $x^w \geq y \wedge w \geq x$  is monotone AC-TA definable.

Observe that  $x \geq 2$  implies  $x^z \geq z$ . Thus we have

$$\begin{aligned} \exists z (\psi) &\Leftrightarrow \exists z (z \leq y \wedge \psi) \vee \exists z (z \geq y \wedge \psi) \\ &\Leftrightarrow \exists z (z \leq y \wedge \psi) \vee \exists z (z \geq y \wedge x \geq 2 \wedge x^x \geq z) \\ &\Leftrightarrow \exists z (z \leq y \wedge \psi) \vee (x \geq 2 \wedge x^x \geq y) \end{aligned}$$

The left sub-formula  $\exists z (z \leq y \wedge \psi)$  is monotone AC-TA definable (Theorem 2), and the right sub-formula is so according to the above observation. Therefore,  $x^{x^x} \geq y$  is monotone AC-TA definable.  $\square$

It is not difficult to generalize the technique we have used in the example, so that we state the next lemma. See Appendix A.4 for a proof.

**Lemma 6.** *Every monotone exponential Diophantine inequality  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$  is monotone AC-TA definable.*  $\square$

Accordingly, we have the main result of the paper.

**Theorem 3 (Definability of exponential Diophantine formulas).** *Every arithmetic constraint in  $M$  of the following syntax is monotone AC-TA definable:*

$$M ::= E(\mathbf{x}_n) \geq L(\mathbf{x}_n) \mid C \mid M \vee M \mid M \wedge M$$

where  $C$  denotes a Presburger formula.  $\square$

*Remark 2.* We have the following inclusion of formulas:

$$\begin{array}{ccc} \text{quantifier-free monotone} & \subsetneq & M \subsetneq \text{monotone} \\ \text{exponential Diophantine} & & \text{exponential Diophantine} \end{array}$$

For the left (strict) inclusion, we consider  $x^2 + y^x \geq z$  in  $M$ . This constraint requires  $\exists$ -quantifiers when it is expressed as an exponential Diophantine formula. For the right strict inclusion, consider  $\exists z (z^3 \geq y \wedge x \geq z + 1)$ , which is equivalent to  $(x - 1)^3 \geq y \wedge x \geq 1$ . It does not belong to  $M$ , since the co-efficient of  $x^2$  is negative.

## 6 Concluding Remarks

In the paper we have discussed the expressiveness of monotone AC-TA. First we demonstrated in Lemmas 1 and 2 that the non-linear inequalities  $x \times y \geq z$  and  $x^y \geq z$  over natural numbers are interpretable by monotone AC-TA. These examples also refine the previous results in [15]. Next we proposed the class of monotone exponential Diophantine formulas, which is a decidable fragment of exponential Diophantine formulas (Theorem 1). Using the transformation by linearly

bounded projection, we have shown that every monotone exponential Diophantine inequality  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$  is monotone AC-TA definable (Lemma 6), and therefore, a sub-class of monotone exponential Diophantine formulas is monotone AC-TA definable (Theorem 3). As an example of the main result, we have presented that the super-exponential  $x^{x^x} \geq y$  is monotone AC-TA definable.

The definability results in terms of monotone AC-TA obtained in the paper are based on our original technique, LINEARLY BOUNDED PROJECTION. Due to the unsolved question about the closedness under projection of AC-monotone tree languages, this special projection plays an essential role overall in the paper.

As a remark of this projection, one can apply it for showing that  $x \times y \geq z$  is monotone AC-TA definable, using the previous result ([15]) that  $x \times y \geq z \wedge u = 1 \wedge v = 2$  is monotone AC-TA definable, because  $x \times y \geq z$  is logically equivalent to  $\exists u, v (u \leq 1 \wedge v \leq 2 \wedge x \times y \geq z \wedge u = 1 \wedge v = 2)$ . This is an alternative proof for Lemma 1.

There are two interesting questions remaining open:

1. *Closedness under projection:* Exponential monotone Diophantine formulas are monotone AC-TA definable if the class of monotone AC-TA is closed under projection. Moreover, under the same condition, one can show that the language classes of Petri nets and monotone AC-TA are identical, which seems to be negatively observed, without a clear proof, in [11].

As we have seen in the paper, using linearly bounded projection, one can eliminate a certain type of  $\exists$ -quantifiers in the formulas. Moreover, it may be possible to transform every monotone exponential Diophantine formula into an equivalent formula formed of linearly bounded monotone AC-TA definable subformulas. If so, regardless of the answer to the above question 1, one can have the positive answer to the definability question about monotone exponential Diophantine formulas—whether every monotone exponential Diophantine formula is monotone AC-TA definable. Furthermore, the validity of monotone exponential Diophantine formulas is an interesting question, though the decidability of the universality problem of monotone AC-TA is not known.

2. *Complete characterization of monotone AC-TA:* The definability theorem for exponential Diophantine formulas (Theorem 3) appeals that this class of formulas is related to monotone AC-TA. However, we do not know exactly which class of the arithmetic can be the counterpart of monotone AC-TA.

One can observe that the class of monotone AC-TA is a proper sub-class of monotone A-TA. *Monotone A-TA* are the class of ETA whose axioms are associativity only [16]. This hierarchy stems from the observation that every monotone AC-TA can be simulated by a monotone A-TA which additionally has transition rules  $f(\alpha, \beta) \rightarrow f(\beta, \alpha)$  for all states  $\alpha, \beta$  instead of the commutativity axioms of  $f$ . Tree languages accepted by monotone A-TA are closely related to context-sensitive languages, which is a sub-class of primitive recursive sets. Thus, monotone AC-TA may relate to the first-order theory of arithmetic with a certain type of, e.g. monotonic, primitive recursions.

**Acknowledgments.** The authors would like to thank Jun-ichi Abo for participating in our discussions with enthusiasm. The completeness proof of Lemma 1 is partly owing to him. We also thank for many fruitful comments from four anonymous reviewers.

## References

1. F. Baader and T. Nipkow: *Term Rewriting and All That*, Cambridge University Press, 1998.
2. I. Boneva, J.-M. Talbot and S. Tison: *Expressiveness of a Spatial Logic for Trees*, Proc. of 20th LICS, Chicago (USA), pp. 280–289, IEEE Computer Society, 2005.
3. H. Comon-Lundh, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi: *Tree Automata Techniques and Applications*, draft, 2005. Available at <http://www.grappa.univ-lille3.fr/tata>
4. S. Dal Zilio and D. Lugiez: *XML Schema, Tree Logic and Sheaves Automata*, *Applicable Algebra in Engineering, Communication and Computing* 17, pp. 337–377, Springer, 2006.
5. S. Ginsburg and E.H. Spanier: *Semigroups, Presburger Formulas, and Languages*, *Pacific Journal of Mathematics* 16, pp. 285–296, 1966.
6. S. Ginsburg: *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, 1966.
7. M.H.T. Hack: *Decidability Questions for Petri Nets*, Ph.D. thesis, Massachusetts Institute of Technology, USA, 1976.
8. T.A. Henzinger: *The Theory of Hybrid Automata*, Proc. of 11th LICS, New Brunswick (USA), pp. 278–292 IEEE Computer Society, 1996. Extended version available at <http://mtc.epfl.ch/~tah/Publications>
9. P.G. Hinman: *Fundamentals of Mathematical Logic*, A K Peters, 2005.
10. N. Kobayashi and H. Ohsaki: *Tree Automata for Non-Linear Arithmetic*, draft, February 2008. Available at <http://staff.aist.go.jp/hitoshi.ohsaki/>
11. M. Kudlek and V. Mitrana: *Normal Forms of Grammars, Finite Automata, Abstract Families, and Closure Properties of Multiset Languages*, Proc. of WMP2000, Curtea de Arges (Romania), LNCS 2235, pp. 135–146, Springer, 2001.
12. L.H. Landweber: *Properties of Vector Addition Systems*, Technical Report 258, University of Wisconsin-Madison, USA, 1975.
13. D. Lugiez: *Multitree Automata That Count*, *Theoretical Computer Science* 333, pp. 225–263, Elsevier, 2005.
14. Y. Matiyasevich: *Hilbert’s Tenth Problem*, MIT Press, 1993.
15. H. Ohsaki, J.-M. Talbot, S. Tison and Y. Roos: *Monotone AC-Tree Automata*, Proc. of 12th LPAR, Montego Bay (Jamaica), LNAI 3855, pp. 337–351, Springer, 2005.
16. H. Ohsaki, H. Seki and T. Takai: *Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism*, Proc. of 14th RTA, Valencia (Spain), LNCS 2706, pp. 483–498, Springer, 2003.
17. H. Ohsaki: *Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories*, Proc. of 15th CSL, Paris (France), LNCS 2142, pp. 539–553, Springer, 2001.
18. R. Parikh: *On Context-Free Languages*, *JACM* 13, pp. 570–581, ACM Press, 1966.
19. H. Seidl, T. Schwentick and A. Muscholl: *Numerical Document Queries*, Proc. of 22nd PODS, San Diego (USA), pp. 155–166, ACM Press, 2003.
20. J.W. Thatcher: *Characterizing Derivation Trees of Context-Free Grammars Through a Generalization of Automata Theory*, *Journal of Computer and System Sciences* 1, pp. 317–322, Elsevier, 1967.
21. K.N. Verma and J. Goubault-Larrecq: *Alternating Two-Way AC-Tree Automata*, *Information and Computation* 205, pp. 817–869, Elsevier, 2007.

## A Proofs

### A.1 Proof of completeness of $\mathcal{A}_{\mathcal{E}}^{\text{mult}}$

We define the mappings  $f, g, h$  for trees in  $T_{F \cup Q}$  :

$$f(t) = (|t|_a + |t|_\alpha + |t|_{\alpha_1} + |t|_{\alpha_2}) \times (|t|_b + |t|_\beta + |t|_{\beta_1} + |t|_{\beta_2} + |t|_\delta) \\ - (|t|_c + |t|_\gamma + |t|_{\alpha_2}) - (|t|_{\beta_2} + |t|_\delta) \times (|t|_a + |t|_\alpha)$$

$$g(t) = |t|_{\beta_1} + |t|_{\beta_2} + |t|_\delta$$

$$h(t) = (|t|_c + |t|_\gamma) - |t|_{\alpha_1}$$

Then we claim that

$$t \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} \delta \text{ implies } f(t) \geq 0, g(t) \leq 1, \text{ and } h(t) \geq 0.$$

By assumption, transition rules in  $\Delta_1$  can be ignored in the following discussion. We show the above invariant property by induction on the number of steps  $n$  of  $t \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} \delta$ . If  $n = 0$ , then  $t \equiv \delta$ . So  $f(t) = h(t) = 0$  and  $g(t) = 1$ . For induction step  $n > 0$ , we suppose  $t \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} t' \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} \delta$ . The proof proceeds by case analysis depending on the applied transition rule at the step in  $t \xrightarrow{\mathcal{A}_{\mathcal{E}}^{\text{mult}}} t'$ .

1. If the transition is performed by  $a \rightarrow \alpha$  (or  $b \rightarrow \beta$ ,  $c \rightarrow \gamma$ ), obviously  $f(t) \geq 0$ ,  $g(t) \leq 1$  and  $h(t) \geq 0$  by induction hypothesis.
2. In case of the transition by  $\beta \rightarrow \beta_1$  (or  $\beta_1 \rightarrow \beta_2$ ,  $\beta_2 \rightarrow \delta$ ), we have  $f(t) = f(t')$ ,  $g(t) + 1 = g(t')$  and  $h(t) = h(t')$ . Thus, the result follows immediately follow from the induction hypothesis.
3. In case of  $f(\alpha, \delta) \rightarrow \delta$ , we have  $f(t) - f(t') = (|t'|_b + |t'|_\beta + |t'|_{\beta_1} + |t'|_{\beta_2} + |t'|_\delta) - (|t'|_{\beta_2} + |t'|_\delta) = |t'|_b + |t'|_\beta + |t'|_{\beta_1} \geq 0$ . Thus,  $f(t) \geq f(t')$ ,  $g(t) = g(t')$  and  $h(t) = h(t')$ .
4. In case of  $f(\alpha_1, \gamma) \rightarrow \alpha_2$ , obviously  $f(t) = f(t')$  and  $g(t) = g(t')$ . Moreover, since  $h(t) - h(t') = -1 + 1 = 0$ , we have  $h(t) = h(t')$ .
5. In case of  $f(\beta, \beta_2) \rightarrow \beta_1$ , we have  $f(t) - f(t') = (|t'|_a + |t'|_\alpha + |t'|_{\alpha_1} + |t'|_{\alpha_2}) - (|t'|_a + |t'|_\alpha) = |t'|_{\alpha_1} + |t'|_{\alpha_2} \geq 0$ . Obviously,  $g(t) = g(t')$  and  $h(t) = h(t')$ .
6. If  $f(\alpha, \beta_1) \rightarrow f(\alpha_1, \beta_1)$  is applied, then  $f(t) - f(t') = -(|t'|_{\beta_2} + |t'|_\delta)$  and  $g(t) = g(t')$ . By the induction hypothesis  $g(t') \leq 1$ , we have  $|t'|_{\beta_1} = 1$ . This implies  $|t'|_{\beta_2} = |t'|_\delta = 0$ , and thus  $f(t) - f(t') = 0$ . On the other hand,  $h(t) = h(t') + 1 \geq 0$  follows from the induction hypothesis  $h(t') \geq 0$ .
7. If  $f(\alpha_2, \beta_2) \rightarrow f(\alpha, \beta_2)$  is applied,  $f(t) - f(t') = -1 - (|t|_{\beta_2} + |t|_\delta) \times (-1)$ . Similar to the previous case 2, since  $|t'|_{\beta_2} = 1$ , we have  $|t'|_\delta = 0$  by induction hypothesis. Thus,  $f(t) - f(t') = -1 + 1 = 0$ . Besides,  $g(t) = g(t')$  and  $h(t) = h(t')$ .

Hence, as a corollary of the above claim,  $t \in \mathcal{L}(\mathcal{A}_{\mathcal{E}}^{\text{mult}})$  implies  $|t|_a \times |t|_b \geq |t|_c$ .

### A.2 Proof of Lemma 2

The formula  $x^y \geq z \wedge x + y > 0$  is logically equivalent to  $(x = 0 \wedge y > 0 \wedge z = 0) \vee (x \geq 1 \wedge x^y \geq z)$ . It suffices to show that the sub-formula  $x \geq 1 \wedge x^y \geq z$  is

monotone AC-TA definable: Define  $\mathcal{A}^{\text{exp}} = (\mathcal{E}, Q, \{\lambda, \delta\}, \Delta_1 \cup \Delta_2)$ , based on the previous construction. The sets  $\Delta_1, \Delta_2$  consist of the following transition rules.

$$\begin{aligned} \Delta_1 : & \quad \mathbf{a} \rightarrow \lambda \quad \mathbf{b} \rightarrow \lambda \quad \mathbf{f}(\lambda, \lambda) \rightarrow \lambda \\ \Delta_2 : & \quad \mathbf{a} \rightarrow \alpha \quad \mathbf{b} \rightarrow \beta \quad \mathbf{c} \rightarrow \gamma \quad \mathbf{f}(\alpha, \gamma) \rightarrow \delta \\ & \quad \left. \begin{array}{l} \mathbf{f}(\alpha, \beta) \rightarrow \theta_1 \quad \mathbf{f}(\alpha, \theta_2) \rightarrow \mathbf{f}(\alpha_1, \theta_2) \\ \mathbf{f}(\gamma, \theta_1) \rightarrow \mathbf{f}(\gamma_1, \theta_2) \quad \mathbf{f}(\alpha_1, \gamma) \rightarrow \alpha_2 \end{array} \right\} (1) \end{aligned}$$

$$\left. \begin{array}{l} \theta_2 \rightarrow \theta_3 \\ \mathbf{f}(\alpha_2, \theta_3) \rightarrow \mathbf{f}(\alpha, \theta_3) \\ \theta_3 \rightarrow \theta_1 \end{array} \right\} (2) \quad \left. \begin{array}{l} \theta_3 \rightarrow \theta_4 \\ \mathbf{f}(\gamma_1, \theta_4) \rightarrow \mathbf{f}(\gamma, \theta_4) \\ \mathbf{f}(\beta, \theta_4) \rightarrow \theta_1 \end{array} \right\} (3) \quad \left. \begin{array}{l} \mathbf{f}(\gamma, \theta_4) \rightarrow \delta \\ \mathbf{f}(\alpha, \delta) \rightarrow \delta \\ \mathbf{f}(\beta, \delta) \rightarrow \delta \end{array} \right\} (4)$$

$\Delta_1$  is the set of rules for the case  $z = 0$ , and  $\Delta_2$  is for the case  $z > 0$ . The above AC-TA  $\mathcal{A}_{\mathcal{E}}^{\text{exp}}$  accepts a tree  $t$  iff  $|t|_a \geq 1$  and  $(|t|_a)^{|t|_b} \geq |t|_c$ .

*Soundness:* If  $x = 0 \wedge y > 0 \wedge z = 0$ , by using the rules of  $\Delta_1$ , we have the derivation  $\mathbf{f}(\mathbf{a}^0, \mathbf{b}^y, \mathbf{c}^z) \rightarrow^* \lambda$ . If  $x \geq 1 \wedge y = 0 \wedge z \geq 1$ , by using the rules of  $\Delta_2$ , we have

$$\mathbf{f}(\mathbf{a}, \mathbf{b}^y, \mathbf{c}) \rightarrow^* \mathbf{f}(\alpha, \beta^y, \gamma) \rightarrow^* \mathbf{f}(\beta^y, \delta) \rightarrow^* \delta.$$

Next, we consider is the case  $x \geq 1 \wedge y \geq 1 \wedge x^y \geq z$ . Observe that:

$$\text{If } (n-1)x < m \leq nx, \text{ then } \mathbf{f}(\alpha^{x-1}, \theta_1, \gamma^m) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \theta_4, \gamma^n) \dots (*)$$

This observation is obtained as follows. First, use the rules in (1) to replace  $m$  occurrences ( $m \leq x$ ) of  $\gamma$  with a single  $\gamma_1$ , and to replace also  $\alpha$  with  $\alpha_2$ . Then use the rules in (2) to restore  $\alpha$  and go back to the previous stage of (1). By repeating this transition, we have the following derivation:

$$\mathbf{f}(\alpha^{x-1}, \theta_1, \gamma^m) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \theta_3, \gamma_1^n).$$

We then use the rules in (3) to have the derivation:

$$\mathbf{f}(\alpha^{x-1}, \theta_3, \gamma_1^n) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \theta_4, \gamma^n).$$

Next, we show the following observation, by induction on  $y$ .

$$\text{If } x \geq 1 \wedge y \geq 1 \wedge x^y \geq z, \text{ then } \mathbf{f}(\alpha^{x-1}, \beta^{y-1}, \theta_1, \gamma^z) \rightarrow^* \delta \dots (**)$$

If  $y = 1$ , then  $(1-1)x \leq z \leq x$ . By using the property (\*),

$$\mathbf{f}(\alpha^{x-1}, \beta^0, \theta_1, \gamma^z) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \theta_4, \gamma) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \delta) \rightarrow^* \delta.$$

If  $y > 1$ , let  $z' = \lceil z/x \rceil$ . Then,  $(z'-1)x < z \leq z'x$ . By using the property (\*),

$$\mathbf{f}(\alpha^{x-1}, \beta^{y-1}, \theta_1, \gamma^z) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \beta^{y-1}, \theta_4, \gamma^{z'}) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \beta^{y-2}, \theta_1, \gamma^{z'}).$$

Since  $x^y \geq z$  implies  $x^y \geq z'x$ , we have  $x^{y-1} \geq z'$ . By the induction hypothesis,

$$\mathbf{f}(\alpha^{x-1}, \beta^{y-2}, \theta_1, \gamma^{z'}) \rightarrow^* \delta.$$

This proves the property (\*\*). Hence, if  $x \geq 1 \wedge y \geq 1 \wedge x^y \geq z$ , we have:

$$\mathbf{f}(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z) \rightarrow^* \mathbf{f}(\alpha^x, \beta^y, \gamma^z) \rightarrow^* \mathbf{f}(\alpha^{x-1}, \beta^{y-1}, \theta_1, \gamma^z) \rightarrow^* \delta.$$

This completes the proof of soundness.

*Completeness:* If  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z) \rightarrow^* \lambda$ , then it must be the case that  $x+y > 0 \wedge z = 0$ , and thus  $x^y \geq z$ . For  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z) \rightarrow_{\Delta_2}^* \delta$ , we define  $I(t)$  to be the conjunction of the following 6 properties.

1.  $|t|_{\theta_1} + |t|_{\theta_2} + |t|_{\theta_3} + |t|_{\theta_4} + |t|_{\delta} \leq 1$
2. If  $|t|_{\theta_1} + |t|_{\theta_2} + |t|_{\theta_3} + |t|_{\theta_4} + |t|_{\delta} = 0$ ,
 
$$(|t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})^{|t|_{\mathbf{b}} + |t|_{\beta}} \geq (|t|_{\mathbf{c}} + |t|_{\gamma} - |t|_{\alpha_1}) + (|t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})|t|_{\gamma_1}$$
3. If  $|t|_{\delta} = 1$ ,  
 $t$  contains  $\mathbf{a}, \mathbf{b}, \alpha, \beta, \delta$  only.
4. If  $|t|_{\theta_1} + |t|_{\theta_3} = 1$ ,
 
$$(1 + |t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})^{1 + |t|_{\mathbf{b}} + |t|_{\beta}} \geq (|t|_{\mathbf{c}} + |t|_{\gamma} - |t|_{\alpha_1}) + (1 + |t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})|t|_{\gamma_1}$$
5. If  $|t|_{\theta_2} = 1$ ,
 
$$(1 + |t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})^{1 + |t|_{\mathbf{b}} + |t|_{\beta}} \geq (|t|_{\mathbf{c}} + |t|_{\gamma} - |t|_{\alpha} - |t|_{\alpha_1}) + (1 + |t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})|t|_{\gamma_1}$$
6. If  $|t|_{\theta_4} = 1$ ,
 
$$(1 + |t|_{\mathbf{a}} + |t|_{\alpha} + |t|_{\alpha_1} + |t|_{\alpha_2})^{|t|_{\mathbf{b}} + |t|_{\beta}} \geq (|t|_{\mathbf{c}} + |t|_{\gamma} - |t|_{\alpha_1}) + |t|_{\gamma_1}$$

One can easily show that (i)  $I(\delta)$  holds and (ii) if  $t \rightarrow t'$  and  $I(t')$ , then  $I(t)$ . Hence,  $t \rightarrow^* \delta$  implies  $I(t)$ . That means, if  $f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z) \rightarrow^* \delta$ , then  $I(f(\mathbf{a}^x, \mathbf{b}^y, \mathbf{c}^z))$ . Therefore, we have  $x^y \geq z$  from the above property 2.

### A.3 Proof of Lemma 5

Let  $L(\mathbf{x}_k) = c_1x_1 + \dots + c_kx_k + d$  for some  $c_1, \dots, c_k, d$  in  $\mathbb{N}$ . If  $L(\mathbf{x}_k)$  is a single variable  $x_i$ , the transformation from  $\mathcal{A}_{\mathcal{E}}$  to  $\mathcal{B}_{\mathcal{E}}$  has been discussed in Lemma 4. Our proof in the following is obtained by generalizing that of Lemma 4. Let  $\mathcal{A} = (\mathcal{E}, Q_1, \{\gamma\}, \Delta_1)$ , then we assume that  $\Delta_1$  contains no transition rules for  $\mathbf{a}, \mathbf{b}_1, \dots, \mathbf{b}_k$  except  $\mathbf{a} \rightarrow \alpha$  and  $\mathbf{b}_i \rightarrow \beta_i$  ( $1 \leq i \leq k$ ). Define state symbols of  $\mathcal{B}_{\mathcal{E}}$ :

$$Q_2 = Q_1 \cup \{ \langle M, i \rangle \mid \exists M \in \text{mul}(Q_1 \cup \{\bar{\alpha}\}), i \leq d: |M| \leq \max\{c_1, \dots, c_k\} + d + 1 \}.$$

Transition rules for  $\mathbf{b}_i$  in  $\Delta_2$  are

$$\mathbf{b}_i \rightarrow \langle M, |M|_{\bar{\alpha}} \rangle$$

where  $M = \{ \beta_i, \alpha, \dots, \alpha, \bar{\alpha}, \dots, \bar{\alpha} \}$  such that  $|M|_{\alpha} \leq c_i$  and  $|M|_{\bar{\alpha}} \leq d$ . Here  $|M|_{\bar{\alpha}}$  represents the number of  $\bar{\alpha}$  in  $M$ . The other rules in  $\Delta_2$  are ones from  $\Delta_1 - \{ \mathbf{a} \rightarrow \alpha, \mathbf{b} \rightarrow \beta \}$  and the rules defined below:

$$\begin{aligned}
\langle M \uplus \{\bar{\alpha}\}, i \rangle &\rightarrow \langle M \uplus \{\alpha\}, i \rangle && \text{if } 1 \leq i \leq d \\
\langle M \uplus \{p\}, i \rangle &\rightarrow \langle M \uplus \{q\}, i \rangle && \text{if } p \rightarrow_{\mathcal{A}_E}^* q \\
\langle M \uplus \{p, q\}, i \rangle &\rightarrow \langle M \uplus \{r\}, i \rangle && \text{if } f(p, q) \rightarrow_{\mathcal{A}_E}^* r \\
f(\langle M \uplus \{p\}, i \rangle, q) &\rightarrow \langle M \uplus \{r\}, i \rangle \\
f(\langle M \uplus \{p\}, i \rangle, \langle \{q\}, j \rangle) &\rightarrow \langle M \uplus \{r\}, i + j \rangle \\
f(\langle M \uplus \{p\}, i \rangle, \langle N \uplus \{q\}, j \rangle) &\rightarrow f(\langle M \uplus \{r\}, i \rangle, \langle N, j \rangle) && (N \neq \emptyset) \\
\langle M \uplus \{p, q\}, i \rangle &\rightarrow \langle M \uplus \{r, s\}, i \rangle && \text{if } f(p, q) \rightarrow_{\mathcal{A}_E}^* f(r, s) \\
f(\langle M \uplus \{p\}, i \rangle, q) &\rightarrow f(\langle M \uplus \{r\}, i \rangle, s) \\
f(\langle M \uplus \{p\}, i \rangle, \langle N \uplus \{q\}, j \rangle) &\rightarrow f(\langle M \uplus \{r\}, i \rangle, \langle N \uplus \{s\}, j \rangle)
\end{aligned}$$

such that  $p, q, r, s \in Q_1$ . Suppose  $\{\gamma\} \cup \{\langle \{\gamma\}, i \rangle \mid 0 \leq i \leq d\}$  is the set of final states of  $\mathcal{B}_E$ .

If  $d = 0$ , the above transition rules are essentially the same as the rules defined on page 12 (Section 5). Thus, for all  $t \in \mathcal{T}_F$  and  $u \in \mathcal{T}_{F-\{a\}}$ , if

$$\begin{aligned}
- |t|_a &\leq c_1|t|_{b_1} + \dots + c_k|t|_{b_k} \\
- |t|_{b_i} &= |u|_{b_i} \text{ for all } 1 \leq i \leq k,
\end{aligned}$$

then  $\mathcal{A}_E$  accepts  $t$  iff  $\mathcal{B}_E$  accepts  $u$ .

If  $d > 0$ , we observe that  $u \rightarrow_{\mathcal{B}/E}^* \langle M, i \rangle$  iff there does not exist  $u'$  such that  $u \rightarrow_{\mathcal{B}/E}^* u' \rightarrow_{\mathcal{B}/E}^* \langle M, i \rangle$  and  $\bar{\alpha}$  occurs more than  $i$ -times in  $u'$ . In this setting,  $i$  is the total occurrences of  $\bar{\alpha}$  in a tree  $u'$ . For instance, if  $u' = f(\langle \bar{\alpha}, \alpha, \beta, 2 \rangle, \langle \bar{\alpha}, \bar{\alpha}, 2 \rangle)$ , then  $\bar{\alpha}$  occurs three times in  $u'$ . Using this observation, one can show that if

$$\begin{aligned}
- |t|_a &\leq c_1|t|_{b_1} + \dots + c_k|t|_{b_k} + d \\
- |t|_{b_i} &= |u|_{b_i} \text{ for all } 1 \leq i \leq k,
\end{aligned}$$

then  $\mathcal{A}_E$  accepts  $t$  iff there exists a derivation  $u \rightarrow_{\mathcal{B}_E}^* \gamma$  or  $u \rightarrow_{\mathcal{B}_E}^* \langle \{\gamma\}, i \rangle$  with  $i \leq d$ .

#### A.4 Proof of Lemma 6

We may write  $\exists x \leq L(\mathbf{y})(\psi)$  for  $\exists x(x \leq L(\mathbf{y}) \wedge \psi)$  in the following sections. First we show that the inequality  $E(\mathbf{x}_n) \geq y$  is monotone AC-TA definable. Recall that  $E(\mathbf{x}_n)$  is generated by the following syntax:

$$E(\mathbf{x}_n) ::= a \mid x_i \mid E(\mathbf{x}_n) + E(\mathbf{x}_n) \mid E(\mathbf{x}_n) \times E(\mathbf{x}_n) \mid E(\mathbf{x}_n)^{E(\mathbf{x}_n)}$$

Since the inequalities  $a \geq y$  and  $x_i \geq y$  are Presburger formulas, it suffices to show that

$$\text{if } E_1(\mathbf{x}_n) \geq y \text{ and } E_2(\mathbf{x}_n) \geq y \text{ are monotone AC-TA definable, then so are } E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y, E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y, \text{ and } E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y.$$

**Lemma 7.** *For every  $c$  in  $\mathbb{N}$ , formulas  $E(\mathbf{x}_n) \leq c$ ,  $E(\mathbf{x}_n) = c$ ,  $E(\mathbf{x}_n) \geq c$  are monotone AC-TA definable.*

*Proof.* We show that  $E(\mathbf{x}_n) \leq c$  can be expressed as a Presburger formula. Use induction on the structure of  $E(\mathbf{x}_n)$ . We distinguish the four cases:

- If  $E(\mathbf{x}_n) = L(\mathbf{x}_n)$ , it is trivial.
- If  $E(\mathbf{x}_n) = E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n)$ , then  $E(\mathbf{x}_n) \leq c$  is logically equivalent to:

$$\bigvee_{0 \leq i \leq c} (E_1(\mathbf{x}_n) \leq i \wedge E_2(\mathbf{x}_n) \leq c - i)$$

Thus, the result follows from the induction hypothesis.

- If  $E(\mathbf{x}_n) = E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n)$ , then  $E(\mathbf{x}_n) \leq c$  is logically equivalent to:

$$E_1(\mathbf{x}_n) = 0 \vee E_2(\mathbf{x}_n) = 0 \vee \bigvee_{1 \leq i \leq c} \bigvee_{1 \leq j \leq c} (E_1(\mathbf{x}_n) \leq i \wedge E_2(\mathbf{x}_n) \leq j \wedge i \times j \leq c).$$

- If  $E(\mathbf{x}_n) = E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)}$ , then  $E(\mathbf{x}_n) \leq c$  is logically equivalent to:

$$(E_1(\mathbf{x}_n) = 0 \wedge E_2(\mathbf{x}_n) \geq 1) \vee (E_1(\mathbf{x}_n) = 1 \wedge c \geq 1) \vee \bigvee_{2 \leq i \leq c} \bigvee_{1 \leq j \leq c} (E_1(\mathbf{x}_n) \leq i \wedge E_2(\mathbf{x}_n) \leq j \wedge i^j \leq c).$$

Note that  $i^j$  is a natural number.

Hence,  $E(\mathbf{x}_n)$  is a Presburger formula.

Observe that  $E(\mathbf{x}_n) \geq c \Leftrightarrow (c = 0 \vee \neg(E(\mathbf{x}_n) \leq c - 1))$  and  $E(\mathbf{x}_n) = c \Leftrightarrow (E(\mathbf{x}_n) \leq c \wedge E(\mathbf{x}_n) \geq c)$ . That means, the constraints  $E(\mathbf{x}_n) \geq c$  and  $E(\mathbf{x}_n) = c$  are Presburger formulas. Therefore,  $E(\mathbf{x}_n) \leq c$ ,  $E(\mathbf{x}_n) = c$ ,  $E(\mathbf{x}_n) \geq c$  are monotone AC-TA definable.  $\square$

**Lemma 8.** *If  $E_1(\mathbf{x}_n) \geq y$  and  $E_2(\mathbf{x}_n) \geq y$  are monotone AC-TA definable, then so is  $E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y$ .*

*Proof.* Let  $\psi$  be the formula  $E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1 + z_2 \geq y$ . Then  $E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y$  can be transformed as follows.

$$\begin{aligned} & E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y \\ \Leftrightarrow & \exists z_1, z_2 (\psi) \\ \Leftrightarrow & \exists z_1 \leq y, z_2 \leq y (\psi) \vee \exists z_1, z_2 (z_1 \geq y \wedge \psi) \vee \exists z_1, z_2 (z_2 \geq y \wedge \psi) \end{aligned}$$

By assumption and Theorem 2, the first sub-formula  $\exists z_1 \leq y, z_2 \leq y (\psi)$  is monotone AC-TA definable. Moreover, since  $z_1 \geq y$  implies  $z_1 + z_2 \geq y$ , the second sub-formula can be transformed as follows.

$$\begin{aligned} & \exists z_1, z_2 (z_1 \geq y \wedge \psi) \\ \Leftrightarrow & \exists z_1, z_2 (z_1 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2) \\ \Leftrightarrow & E_1(\mathbf{x}_n) \geq y \wedge \exists z_2 (E_2(\mathbf{x}_n) \geq z_2) \\ \Leftrightarrow & E_1(\mathbf{x}_n) \geq y \end{aligned}$$

Note that  $\exists z_2 (E_2(\mathbf{x}_n) \geq z_2)$  is a valid formula, because  $E_2(\mathbf{x}_n) \geq 0$  is true for

any assignment. Thus, the second sub-formula is also monotone AC-TA definable. Similarly, the third sub-formula is so. Therefore,  $E_1(\mathbf{x}_n) + E_2(\mathbf{x}_n) \geq y$  is monotone AC-TA definable.  $\square$

**Lemma 9.** *If  $E_1(\mathbf{x}_n) \geq y$  and  $E_2(\mathbf{x}_n) \geq y$  are monotone AC-TA definable, then so is  $E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y$ .*

*Proof.* Similar to the previous proof, let  $\psi$  be the formula  $E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1 \times z_2 \geq y$ . Then  $E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y$  can be transformed as follows.

$$\begin{aligned} & E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y \\ \Leftrightarrow & \exists z_1, z_2 (\psi) \\ \Leftrightarrow & \exists z_1 \leq y, z_2 \leq y (\psi) \vee \exists z_1, z_2 (z_1 \geq y \wedge \psi) \vee \exists z_1, z_2 (z_2 \geq y \wedge \psi) \end{aligned}$$

By assumption and Theorem 2, the first sub-formula  $\exists z_1 \leq y, z_2 \leq y. \psi$  is monotone AC-TA definable. Moreover, since  $z_1 \geq y \wedge z_2 \geq 1$  implies  $z_1 \times z_2 \geq y$ , the second sub-formula can be transformed as follows.

$$\begin{aligned} & \exists z_1, z_2 (z_1 \geq y \wedge \psi) \\ \Leftrightarrow & \exists z_1 (z_2 = 0 \wedge z_1 \geq y \wedge \psi) \vee \exists z_1, z_2 (z_2 \geq 1 \wedge z_1 \geq y \wedge \psi) \\ \Leftrightarrow & \exists z_1 (z_1 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge 0 \geq y) \vee \\ & \exists z_1, z_2 (z_2 \geq 1 \wedge z_1 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2) \\ \Leftrightarrow & y = 0 \vee (E_1(\mathbf{x}_n) \geq y \wedge E_2(\mathbf{x}_n) \geq 1) \end{aligned}$$

Thus, the second sub-formula is also monotone AC-TA definable. Similarly, the third sub-formula is so. Therefore,  $E_1(\mathbf{x}_n) \times E_2(\mathbf{x}_n) \geq y$  is monotone AC-TA definable.  $\square$

**Lemma 10.** *If  $E_1(\mathbf{x}_n) \geq y$  and  $E_2(\mathbf{x}_n) \geq y$  are monotone AC-TA definable, then so is  $E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y$ .*

*Proof.* Similar to the previous proof, let  $\psi$  be the formula  $E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1^{z_2} \geq y$ . Then  $E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y$  can be transformed as follows.

$$\begin{aligned} & E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y \\ \Leftrightarrow & \exists z_1, z_2 (\psi) \\ \Leftrightarrow & \exists z_1 \leq y, z_2 \leq y (\psi) \vee \exists z_1, z_2 (z_1 \geq y \wedge \psi) \vee \exists z_1, z_2 (z_2 \geq y \wedge \psi) \end{aligned}$$

By assumption and Theorem 2, the first sub-formula is monotone AC-TA definable. The second sub-formula can be transformed as follows.

$$\begin{aligned} & \exists z_1, z_2 (z_1 \geq y \wedge \psi) \\ \Leftrightarrow & \exists z_1 (z_2 = 0 \wedge z_1 \geq y \wedge \psi) \vee \exists z_1, z_2 (z_2 \geq 1 \wedge z_1 \geq y \wedge \psi) \\ \Leftrightarrow & \exists z_1 (z_1 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge z_1 \geq 1 \wedge 1 \geq y) \vee \\ & \exists z_1, z_2 (z_2 \geq 1 \wedge z_1 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2) \\ \Leftrightarrow & (E_1(\mathbf{x}_n) \geq y \wedge E_1(\mathbf{x}_n) \geq 1 \wedge 1 \geq y) \vee \\ & (E_1(\mathbf{x}_n) \geq y \wedge E_2(\mathbf{x}_n) \geq 1) \end{aligned}$$

Note that  $z_1^0 \geq y$  iff  $z_1 \geq 1 \wedge 1 \geq y$  (as  $0^0$  is undefined). Thus, the second sub-formula is monotone AC-TA definable. Moreover, the third sub-formula can be transformed as follows.

$$\begin{aligned}
& \exists z_1, z_2 (z_2 \geq y \wedge \psi) \\
\Leftrightarrow & \exists z_2 (z_1 = 0 \wedge z_2 \geq y \wedge \psi) \vee \\
& \exists z_2 (z_1 = 1 \wedge z_2 \geq y \wedge \psi) \vee \\
& \exists z_1, z_2 (z_1 \geq 2 \wedge z_2 \geq y \wedge \psi) \\
\Leftrightarrow & \exists z_2 (z_2 \geq y \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_2 \geq 1 \wedge y = 0) \vee \\
& \exists z_2 (z_2 \geq y \wedge E_1(\mathbf{x}_n) \geq 1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge 1 \geq y) \vee \\
& \exists z_1, z_2 (z_1 \geq 2 \wedge z_2 \geq y \wedge E_1(\mathbf{x}_n) \geq z_1 \wedge E_2(\mathbf{x}_n) \geq z_2 \wedge z_1^{z_2} \geq y)
\end{aligned}$$

Note that  $0^{z_2} \geq y$  iff  $z_2 \geq 1 \wedge y = 0$ . Furthermore, since  $z_1 \geq 2 \wedge z_2 \geq y$  implies  $z_1^{z_2} \geq y$ , we have the equivalent formula:

$$\begin{aligned}
& (E_2(\mathbf{x}_n) \geq 1 \wedge z_2 \geq 1 \wedge y = 0) \vee \\
& (E_1(\mathbf{x}_n) \geq 1 \wedge E_2(\mathbf{x}_n) \geq y \wedge 1 \geq y) \vee \\
& (E_1(\mathbf{x}_n) \geq 2 \wedge E_2(\mathbf{x}_n) \geq y)
\end{aligned}$$

Thus, the third sub-formula is monotone AC-TA definable. Hence,  $E_1(\mathbf{x}_n)^{E_2(\mathbf{x}_n)} \geq y$  is so.  $\square$

Now we prove Lemma 6, that is:  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$  is monotone AC-TA definable. By induction on the structure of  $E(\mathbf{x}_n)$ , one can show that  $E(\mathbf{x}_n) \geq y$  is monotone AC-TA definable, using Lemmas 8–10. The formula  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$  is transformed as follows.

$$\begin{aligned}
& E(\mathbf{x}_n) \geq L(\mathbf{x}_n) \\
\Leftrightarrow & \exists y (E(\mathbf{x}_n) \geq y \wedge y = L(\mathbf{x}_n)) \\
\Leftrightarrow & \exists y \leq L(\mathbf{x}_n) (E(\mathbf{x}_n) \geq y \wedge y \geq L(\mathbf{x}_n))
\end{aligned}$$

Therefore, by LINEARLY BOUNDED PROJECTION (Theorem 2),  $E(\mathbf{x}_n) \geq L(\mathbf{x}_n)$  is monotone AC-TA definable.