On the Use of Non-deterministic Automata for Presburger Arithmetic

Antoine Durand-Gasselin and Peter Habermehl

LIAFA, CNRS & University Paris Diderot (Paris 7), Case 7014, 75205 Paris 13, France
{adg,haberm}@liafa.jussieu.fr

Abstract. A well-known decision procedure for Presburger arithmetic uses deterministic finite-state automata. While the complexity of the decision procedure for Presburger arithmetic based on quantifier elimination is known (roughly, there is a double-exponential non-deterministic time lower bound and a triple exponential deterministic time upper bound), the exact complexity of the automata-based procedure was unknown. We show in this paper that it is triple-exponential as well by analysing the structure of the non-deterministic automata obtained during the construction. Furthermore, we analyse the sizes of deterministic and non-deterministic automata built for several subclasses of Presburger arithmetic such as disjunctions and conjunctions of atomic formulas. To retain a canonical representation which is one of the strengths of the use of automata we use residual finite-state automata, a subclass of non-deterministic automata.

1 Introduction

Presburger arithmetic (PA) is the first-order theory over integers with addition. It has many applications for example in system verification, constraint data bases, etc. PA was shown decidable [13] using the method of quantifier elimination. The complexity of the decision problem has been well studied. In [9] a double exponential non-deterministic time lower bound was given and Oppen [8] showed that Cooper's [5] quantifier elimination algorithm has triple exponential worst-case complexity in deterministic time. The bound was improved [14] considering the number of quantifier eliminations.

Another decision procedure for PA is based on the use of deterministic finite-state word automata (DFA). The idea comes from Büchi [4]. Integer vectors are represented as words over a suitable alphabet and an automaton for a given formula accepts exactly the words which correspond to vectors making the formula true. An automaton representing exactly the solutions of a formula can be constructed recursively from a formula by using automata constructions corresponding to the logical connectives and quantifiers. Eliminating an existential quantifier is done by projecting away the track corresponding to the given variable and then determinising and minimising the resulting automaton. An advantage of the approach is that a canonical representation — the minimal automaton — is obtained for a formula. This allows simple equivalence checking of two formulas which is often needed in verification applications. The automata based approach has been studied for example by [3,16] who gave direct constructions of automata for atomic linear constraints. It is implemented for example in the tools LASH

[17] and FAST [1,18]. A rigorous analysis of the size of the minimal DFA for a Presburger formula is given in [10]. The tight upper bound obtained is triple-exponential in the size of the formula. The result is shown not by analysing the automata based decision procedure outlined above but by eliminating the quantifiers, thus obtaining a quantifier-free formula and analysing the size of the automaton constructed from it. The exact size of the intermediate automata in the automata based decision procedure was left open. More precisely, there might have been an exponential blow-up coming from the determinisation, leading to automata with a quadruple exponential size.

In this paper we investigate the use of *non-deterministic automata* (NFA) for PA which has not yet been extensively studied. To retain the property of canonicity we study a subclass of NFA called residual finite-state automata (RFSA) [6] which admit a canonical minimal automaton. States of RFSA accept residuals of the language accepted by the automaton. Our two main contributions are: (1) Though Klaedtke [10] has shown using results of [9] that the lower bound for the size of the NFA for a Presburger formula is the same as the triple-exponential lower bound for the size of the DFA, it is interesting to investigate the potential gain of the use of NFA. We show first that for atomic formulas no gain at all can be achieved. Then, we show that for several subclasses of Presburger formulas (for example for disjunctions of inequations) a substantial gain in the size of the automata is obtained. For this we characterise exactly the number of states of the minimal DFA and compare them with the minimal RFSA. The results obtained are also of interest for constructing tools for Presburger arithmetic, since we show for some subclasses that the automata obtained by the usual product construction are minimal. (2) We show that the size of the automata obtained during the automata based decision procedure is triple-exponential solving an open problem from [10]. This is done by carefully inspecting the structure of the NFA obtained after projection. We show that the determinisation of this automaton does not lead to an exponential blow-up. From that follows a triple-exponential time upper bound for the automata based decision procedure.

2 Preliminaries

Presburger Arithmetic is the first-order theory on atomic formulas of the form $\sum_{i=1}^{n} a_i x_i \sim c$, where a_i and c are integer constants, x_i integer variables, and $\sim$ is an operator among $\{=, \neq, <, >\leq, \geq, \equiv_m\}$ (with $\equiv_m$ the congruence modulo m, for any $m \geq 2$).

We will restrict ourselves here to the case where $\sim \in \{=, >, \equiv_m\}$ and $gcd(a_1, \ldots, a_n) = 1$, and also consider the two trivial formulas $\top$ and $\bot$ (respectively the tautology and the unsatisfiable proposition) as atomic formulas. As boolean operators we use $\neg$, $\wedge$ and $\vee$.

A Presburger formula is defined as an object of this theory. The length of a Presburger formula is defined in the usual way (see [10]). We use the vectorial notation for the atomic formulas, i.e. $\mathbf{a}.\mathbf{x} \sim c$. Let φ be a Presburger formula with k free variables. $[\![\varphi]\!]$ is defined as the set of solutions of φ, that is the set of assignments (seen as a subset of $\mathbb{Z}^k$) of the free variables of φ validating φ (with the usual semantics of arithmetic). The set of solutions (a set of integer vectors) of any Presburger formula is called a Presburger set. Two Presburger formulas φ and φ' are called semantically equivalent iff $[\![\varphi]\!] = [\![\varphi']\!]$. Two n-tuples of Presburger formulas $(\varphi_1, \ldots, \varphi_n)$ and $(\varphi'_1, \ldots, \varphi'_n)$ are called semantically equivalent iff $[\![\varphi_i]\!] = [\![\varphi'_i]\!]$ for all $i \in \{1, \ldots, n\}$. For $1 \le k \le n$, we let $\mathbf{e}^n_k$ be the n-dimensional vector which has a 1 as its k-th component and 0 elsewhere.

Finite Word Automata. Let Σ be a finite alphabet and Σ^* the set of finite words over Σ. The empty word is denoted by ϵ and the length of a word u by $|u|$. A *non-deterministic finite automaton* (NFA) $\mathcal{A}$ is a 5-tuple $\langle \Sigma, Q, Q_0, F, \delta \rangle$, with Σ a finite alphabet, Q a finite set of states, $Q_0 \subseteq Q$ (resp. $F \subseteq Q$) the initial (resp. final) states and δ the transition function mapping $Q \times \Sigma$ to 2^Q. An NFA $\mathcal{A} = \langle \Sigma, Q, Q_0, F, \delta \rangle$ is said to be deterministic (DFA) if Q_0 is a singleton $\{q_0\}$ and if $\forall q \in Q,\ \forall x \in \Sigma,\ Card(\delta(q, x)) \le 1$. Therefore the transition function δ of a DFA can be seen as a partial application of $Q \times \Sigma$ to Q.

A path labelled by $u = u_1 \cdots u_n$ in an NFA $\mathcal{A} = \langle \Sigma, Q, Q_0, F, \delta \rangle$ from $q_0 \in Q$, is a sequence of states $q_0, q_1, \ldots, q_n$ such that $\forall i, 1 \le i \le n, q_i \in \delta(q_{i-1}, u_i)$. We define inductively $\widehat{\delta} : 2^Q \times \Sigma^* \to 2^Q$ as $\widehat{\delta}(Q', \epsilon) = Q'$ and $\widehat{\delta}(Q', au) = \widehat{\delta}(\bigcup_{q' \in Q'} \delta(q', a), u)$, that is the set of states that can be reached by a path labelled by u from a state of Q'. A path $q_0, \ldots, q_n$ in $\mathcal{A}$ is accepting if $q_n \in F$. A word u is accepted by $\mathcal{A}$ if there is a path $q_0, \ldots, q_n$ labelled by u in $\mathcal{A}$ such that $q_0 \in Q_0$ and $q_n \in F$. The language $L_{\mathcal{A}}$ is the set of words accepted by $\mathcal{A}$. The reverse of u, noted $\widetilde{u}$ is inductively defined as $\widetilde{\epsilon} = \epsilon$ and $\widetilde{x \cdot v} = \widetilde{v} \cdot x$, for $x \in \Sigma$ and $v \in \Sigma^*$. The reverse language $\widetilde{L}$ of $L \subseteq \Sigma^*$ is defined as $\widetilde{L} = \{\widetilde{u} \in \Sigma^* \mid u \in L\}$. The reverse automaton of an NFA $\mathcal{A} = \langle \Sigma, Q, Q_0, F, \delta \rangle$ is defined as $\widetilde{\mathcal{A}} = \langle \Sigma, Q, F, Q_0, \widetilde{\delta} \rangle$ with $q \in \widetilde{\delta}(q', x)$ if and only if $q' \in \delta(q, x)$. Clearly, $\widetilde{L_{\mathcal{A}}} = L_{\widetilde{\mathcal{A}}}$.

We define the notion of residual languages (or simply *residuals*) both for automata and languages. Let L be a language over an alphabet Σ and w a word of Σ^*. The residual language or left quotient of L by w, is $w^{-1}L = \{u \mid w \cdot u \in L\}$. Given an automaton $\mathcal{A}$, a residual (left-residual) of a state q, called $L_{\mathcal{A},q}$ or $\mathrm{post}_{\mathcal{A},q}$, is defined as $L_{\mathcal{A},q} = \{u \mid F \cap \widehat{\delta}(\{q\}, u) \neq \emptyset\}$. A right-residual of a state q, is defined as $\mathrm{pre}_{\mathcal{A},q} = \{u \mid q \in \widehat{\delta}(Q_0, u)\}$.

Obviously, residuals of states of a deterministic automaton are always residuals of the language that the automaton accepts (provided that all states are reachable).

The *size* of an automaton is defined the usual way, i.e. the number of states plus the number of transitions. An automaton is said to be minimal in a class of automata if any other automaton in it accepting the same language has at least as many states. It is well known that minimal DFA are canonical (unique up to isomorphism) whereas NFA do not have a minimal canonical form. Thus, we use RFSA [6], a subclass of NFA having a minimal canonical form. The number of states of the minimal RFSA lies between the number of states of the minimal NFA and the number of states of the minimal DFA.

Let $\mathcal{A}$ be an NFA, $\mathcal{A} = \langle \Sigma, Q, Q_0, F, \delta \rangle$. $\mathcal{A}$ is a *residual finite state automaton* (RFSA) if for all state $q \in Q$, $L_{\mathcal{A},q}$ is a residual language of $L_{\mathcal{A}}$. Formally, $\forall q \in Q, \exists u \in \Sigma^*$, such that $L_{\mathcal{A},q} = u^{-1}L_{\mathcal{A}}$. Therefore any DFA where all states are reachable, is an RFSA, but the converse is not true, and for some languages (like $(a+b)^*a(a+b)^n$, see [6]), the smallest RFSA is exponentially smaller than the minimal DFA.

Let L be a regular language over Σ, and u a word of Σ^*. The residual language $u^{-1}L$ is said to be *prime*, if it is not equal to the union of the residual languages of L it strictly contains. Let $R_u = \{v \in \Sigma^* | v^{-1}L \subsetneq u^{-1}L\}$, $u^{-1}L$ is a prime residual of L if $\bigcup_{v \in R_u} v^{-1}L \subsetneq u^{-1}L$. Intuitively, a residual language of L is prime if it cannot be obtained as a union of other residual languages of L. Each regular language L is accepted by a minimal canonical RFSA [6] defined below. Each of its states is a prime residual of L.

Definition 1. *Let L be a regular language over Σ. Its canonical RFSA $(\Sigma, Q, Q_0, F, \delta)$ is defined as follows. Q is the set of prime residuals of L, i.e. $Q = \{u^{-1}L \mid u^{-1}L$ is prime$\}$. Initial states are prime residual languages contained in L, i.e. $Q_0 = \{u^{-1}L \in Q \mid u^{-1}L \subseteq L\}$ and final states are prime residual languages containing the empty word, i.e. $F = \{u^{-1}L \in Q \mid \epsilon \in u^{-1}L\}$. The transition function is defined by $\delta(u^{-1}L, a) = \{v^{-1}L \in Q \mid v^{-1}L \subseteq (ua)^{-1}L\}$, for $u^{-1}L \in Q$ and $a \in \Sigma$.*

Encoding of Presburger sets as languages. We represent integer vectors as finite words, so we can map Presburger sets to languages. We use a vectorial least significant bit first coding. For $k > 0$ we define $\Sigma_k = \{0,1\}^k$. Moreover we use the separate sign alphabet $S_k = \{+,-\}^k$ (indicating if the corresponding integer is positive or negative). Words of $\Sigma_k^* S_k$ represent k-dimensional integer vectors. A word $w_0 \dots w_n s \in \Sigma_k^* S_k$ represents the integer vector we denote $\langle w_0 \dots w_n s\rangle$, each component of which is computed as follows (where $\pi_i(x)$ representing the value of the i-th component of x). If $s_i = +$, then $\pi_i(\langle w_0 \dots w_n s\rangle) = \sum_{j=0}^{n} 2^j.\pi_i(w_j)$ and if $s_i = -$, then $\pi_i(\langle w_0 \dots w_n s\rangle) = -2^{n+1} + \sum_{j=0}^{n} 2^j.\pi_i(w_j)$. In particular, $\langle +\rangle = 0$ and $\langle -\rangle = -1$. We also define the notation $\langle . \rangle_+$ over Σ_k^* as $\langle w\rangle_+ = \langle w+^k\rangle$.

Remark 1. Let $w', w \in \Sigma_k^*$, $s \in S_k$. We have $\langle w'ws\rangle = \langle w'\rangle_+ + 2^{|w'|}\langle ws\rangle$.

This representation is clearly surjective and provides an infinite number of representations for each vector. Indeed for any word $w_0 \dots w_n s \in \Sigma_k^* S_k$, any $w_0 \dots w_n (s')^* s$ (with $s'_i = 0$ if $s_i = +$ or $s'_i = 1$ if $s_i = -$) represents the same vector.

Given a Presburger formula $\varphi(\mathbf{x})$ (with $\mathbf{x}$ the vector of free variables of φ, and k its dimension), we say it defines the language $L_\varphi = \{w \in \Sigma_k^* S_k \mid \langle w\rangle \in [\![\varphi]\!]\}$. Such languages are called Presburger-definable and meet the following saturation property: if a representation of a vector is in the language then any other representation of that vector is also in the language. Our coding meets the following important property [12].

Property 1. Any residual of a saturated Presburger-definable language is either a saturated Presburger-definable language, or the empty word language.

Thus we not only have a residual closure property on Presburger-definable sets, but we also have an effective way to characterise them. In an automaton accepting all solutions of a Presburger formula $\varphi(\mathbf{x})$ with k free variables, a word $w \in \Sigma_k^*$ leads from the initial state to a state accepting exactly all solutions of $\varphi(2^{|w|}\mathbf{x} + \langle w\rangle_+)$.

Automata representing atomic Presburger formula. We study here automata that accept Presburger-definable languages. Notice that all final states of such automata are equivalent (there is only one residual that contains the empty word). All other states have a residual that is Presburger definable, i.e. definable by a Presburger formula. Thus, in the following, automata have as states a unique final state F and elements of $\mathcal{F}$, the set of Presburger formulas. Each state $\varphi \in \mathcal{F}$ characterises its residual, for example a state with empty residual will be represented by $\perp$.

We recall here the construction of the DFA for atomic Presburger constraints. The construction for equations and inequations with a least significant bit first coding was given by Boudet and Comon [3]; as they worked with natural numbers, they had not to handle the sign letters. Wolper and Boigelot [16] worked with integers but used a most significant bit first coding. They gave a backwards construction of the automaton yielding an NFA for inequations which they then had to determinise. Since we work with a least significant bit first coding we are spared this determinisation procedure. We also give the construction of the automaton for modular constraints directly, rather than using Klaedtke's [10] which is also based on a most significant bit first coding.

- The automaton $A_{\mathbf{a}.\mathbf{x}=c}$ for the formula $\mathbf{a}.\mathbf{x} = c$ is given by the following forward construction starting from the initial state $\mathbf{a}.\mathbf{x} = c$:
  - if $b \in \Sigma_k$, $\delta(\mathbf{a}.\mathbf{x} = \gamma, b) = \begin{cases} \mathbf{a}.\mathbf{x} = \gamma' \text{ with } \gamma' = \frac{\gamma - \mathbf{a}.\mathbf{b}}{2} \text{ when } 2 \mid \gamma - \mathbf{a}.\mathbf{b} \\ \perp \text{ otherwise} \end{cases}$
  - if $s \in S_k$, $\delta(\mathbf{a}.\mathbf{x} = \gamma, s) = F$ when $\mathbf{a}.\langle s \rangle = \gamma$ and $\delta(\mathbf{a}.\mathbf{x} = \gamma, s) = \perp$ otherwise
  - for $\alpha \in S_k \cup \Sigma_k$, $\delta(\perp, \alpha) = \delta(F, \alpha) = \perp$
- The automaton $A_{\mathbf{a}.\mathbf{x}>c}$ for the formula $\mathbf{a}.\mathbf{x} > c$ is given by the following forward construction starting from the initial state $\mathbf{a}.\mathbf{x} > c$:
  - if $b \in \Sigma_k$, $\delta(\mathbf{a}.\mathbf{x} > \gamma, b) = \mathbf{a}.\mathbf{x} > \gamma'$ with $\gamma' = \left\lfloor \frac{\gamma - \mathbf{a}.\mathbf{b}}{2} \right\rfloor$.
  - if $s \in S_k$, $\delta(\mathbf{a}.\mathbf{x} > \gamma, s) = F$ when $\mathbf{a}.\langle s \rangle > \gamma$ and $\delta(\mathbf{a}.\mathbf{x} > \gamma, s) = \perp$ otherwise
  - for $\alpha \in S_k \cup \Sigma_k$, $\delta(\perp, \alpha) = \delta(F, \alpha) = \perp$
- The automaton $A_{\mathbf{a}.\mathbf{x} \equiv_{2^m(2n+1)} c}$ for the formula $\mathbf{a}.\mathbf{x} \equiv_{2^m(2n+1)} c$ (with some $m, n \geq 0$) is given by the following forward construction starting from $\mathbf{a}.\mathbf{x} \equiv_{2^m(2n+1)} c$:
  - if $b \in \Sigma_k$ and $p \geq 1$,
    $\delta(\mathbf{a}.\mathbf{x} \equiv_{2^p(2n+1)} \gamma, b) = \begin{cases} \mathbf{a}.\mathbf{x} \equiv_{2^{p-1}(2n+1)} \gamma' \text{ with } \gamma' = \frac{\gamma - \mathbf{a}.\mathbf{b}}{2} \text{ when } 2 \mid \gamma - \mathbf{a}.\mathbf{b} \\ \perp \text{ otherwise} \end{cases}$
  - if $b \in \Sigma_k$ and $p = 0$,
    $\delta(\mathbf{a}.\mathbf{x} \equiv_{2n+1} \gamma, b) = \mathbf{a}.\mathbf{x} \equiv_{2n+1} \gamma'$ with $\gamma' = \begin{cases} \frac{\gamma - \mathbf{a}.\mathbf{b}}{2} \text{ when } 2 \mid \gamma - \mathbf{a}.\mathbf{b} \\ \frac{\gamma + 2n + 1 - \mathbf{a}.\mathbf{b}}{2} \text{ when } 2 \nmid \gamma - \mathbf{a}.\mathbf{b} \end{cases}$
  - if $s \in S_k$, $\delta(\mathbf{a}.\mathbf{x} \equiv_{2^p(2n+1)} \gamma, s) = \begin{cases} F \text{ when } \mathbf{a}.\langle s \rangle \equiv_{2^p(2n+1)} \gamma \\ \perp \text{ otherwise} \end{cases}$
  - for $\alpha \in S_k \cup \Sigma_k$, $\delta(\perp, \alpha) = \delta(F, \alpha) = \perp$

The correctness of this construction derives from Property 1. For all states φ and all $b \in \Sigma_k$ we have $\delta(\varphi(\mathbf{x})) = \varphi(2\mathbf{x} + b)$ showing that the transitions labelled by Σ_k are correct. The transitions labelled by S_k are correct by definition, as the last letter is always from S_k. The definitions are clearly exhaustive. Furthermore, as $gcd(\mathbf{a}) = 1$, all states are non-equivalent, indeed $[\![\mathbf{a}.\mathbf{x} \sim \gamma]\!] \neq [\![\mathbf{a}.\mathbf{x} \sim \gamma']\!]$ when $\gamma \neq \gamma'$. Thus our construction provides us with the *minimal DFA*. In the following we detail the number of states of the automata constructed. Those results were given by Klaedtke [10] on automata for most significant bit first coding. They are easily adapted for equations and inequations. For a vector $\mathbf{a}$ we define $||\mathbf{a}||_+ = \sum_{\{i \mid a_i \geq 0\}} a_i$ and $||\mathbf{a}||_- = \sum_{\{i \mid a_i \leq 0\}} |a_i|$.

Theorem 1. *Let $gcd(\mathbf{a}) = 1$. For the case of an equation $\mathbf{a}.\mathbf{x} = c$, the states $\{\mathbf{a}.\mathbf{x} = \gamma \mid -||a||_+ < \gamma < ||a||_-\}$ are reachable and form a maximal strongly connected component (SCC) and all other reachable states are in $\{\mathbf{a}.\mathbf{x} = \gamma \mid \gamma = c \vee \min(c, -||a||_+) < \gamma < \max(c, ||a||_-)\}$. In the case of an inequation $\mathbf{a}.\mathbf{x} > c$, the states $\{\mathbf{a}.\mathbf{x} > \gamma \mid -||\mathbf{a}||_+ \leq \gamma < ||\mathbf{a}||_-\}$ are reachable and form a maximal SCC and all other reachable states are in $\{\mathbf{a}.\mathbf{x} > \gamma \mid \gamma = c \vee \min(c, -||a||_+) \leq \gamma < \max(c, ||a||_-)\}$. For modulo constraints*

$\mathbf{a}.\mathbf{x} \equiv_{2^n(2p+1)} c$ *the states in* $\{\mathbf{a}.\mathbf{x} \equiv_{2p+1} \gamma \mid \gamma \in [0, 2p]\}$ *are reachable and form a maximal SCC and all other reachable states are in* $\{\mathbf{a}.\mathbf{x} \equiv_{2^m(2p+1)} \gamma \mid (m = n \wedge \gamma = c) \vee (m < n \wedge \gamma \in [0, 2^m(2p + 1) - 1])\}$.

The SCCs only depend on **a** and not on the constant c. Before proving the theorem we give in Figure 1 the automaton for a simple inequation. The reachable maximal SCC is formed by the states $\{-1, 0, 1, 2\}$. The transitory state 3 exists only because $3 > ||\mathbf{a}||_-$.

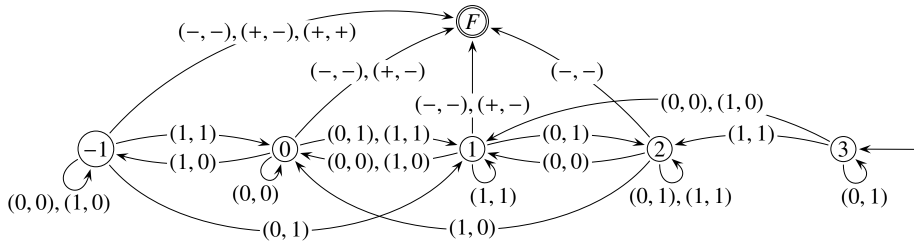


Fig. 1. An automaton for $x - 3y > 3$

Proof. Klaedtke [10] studied the number of reachable states for equations and inequations. Automata for equations are by construction backwards deterministic, thus we built the reverse automaton of Klaedtke's and all states in $\{\mathbf{a}.\mathbf{x} = \gamma \mid -||a||_+ < \gamma < ||a||_-\}$ are reachable and form a maximal SCC. We get the same reachable states forming a maximal SCC for an automaton for an inequation $\mathbf{a}.\mathbf{x} > \gamma$, as it just contains more transitions than the automaton for $\mathbf{a}.\mathbf{x} = \gamma$. For modulo constraints, for our coding, if the modulus is even, it is halved by the next transition, so all states are reached only for odd modulus. Thus, the automaton is possibly smaller compared to the automaton constructed for the most significant bit first coding which is as big as the modulus. □

3 Gain of Non-determinism

We study here the potential gain of using NFA, in particular RFSA, for subclasses of Presburger arithmetic. We start with the following negative result for atomic constraints.

Theorem 2. *For equations, inequations or modulo constraints with an odd modulus, no NFA is smaller than the minimal DFA.*

The proof of this theorem consists essentially in considering a class of languages for which the minimal DFA is already a minimal NFA. We use the following result [11] for biRFSA languages (i.e. languages accepted by biRFSAs). An automaton $\mathcal{A}$ is a biRFSA if $\mathcal{A}$ and $\widetilde{\mathcal{A}}$ are both RFSA.

Proposition 1. *The canonical RFSA of a biRFSA language is a minimal NFA.*

We will show that the languages we consider are biRFSA and that their canonical RFSA is not smaller than the minimal DFA. Therefore it is the minimal NFA as well.

Lemma 1. *Given a language L with residuals $L_0 = L, L_1, \ldots, L_n$, such that: (1) Given a residual, residuals languages included in it form a strictly increasing chain w.r.t. inclusion, i.e. $\forall L_i, L_j, L_k . L_j \subseteq L_i \wedge L_k \subseteq L_i \implies L_j \subseteq L_k \vee L_k \subseteq L_j$, (2) Two residuals are either disjoint or one is included in the other, i.e. $\forall L_i, L_j . L_i \cap L_j \neq \emptyset \implies L_i \subseteq L_j \vee L_j \subseteq L_i$. Then any NFA accepting this language is at least as big (in term of number of states) as the minimal DFA.*

Proof. Let L be a regular language satisfying the lemma's hypotheses. The minimal DFA has as many states as the canonical RFSA as all the residuals are prime. This is because for $L_0, \ldots, L_m$ residuals of L, if $L_0 = \bigcup_{i\geq 1} L_i$ then $\forall i . L_i \in L_0$ thus the L_i form an increasing chain, thus $\exists i \geq 1 . L_0 = L_i$.

Let $\mathcal{A} = \langle \Sigma, (q_i)_{i\leq m}, Q_0, F, \delta \rangle$ be the canonical RFSA of L. We show that L is biRFSA by observing that the reversed automaton of the canonical RFSA is a RFSA. With the two hypotheses we have on the residuals, it is easy to find for each residual L_i (suppose of the state q_i) a word w_i that is only in L_i and the residuals containing L_i.

Let us show that $\text{post}_{\widetilde{\mathcal{A}},q_i} = \widetilde{w_i}^{-1}\widetilde{L}$. By definition $\widetilde{w_i}^{-1}\widetilde{L} = \{v \mid \widetilde{w_i}.v \in \widetilde{L}\}$, then $v \in \text{post}_{\widetilde{\mathcal{A}},q_i} \Rightarrow \widetilde{v} \in \text{pre}_{\mathcal{A},q_i} \Rightarrow \widetilde{v}.w_i \in L \Rightarrow \widetilde{w_i}.v \in \widetilde{L}$. We have the first inclusion, $\text{post}_{\widetilde{\mathcal{A}},q_i} \subseteq \widetilde{w_i}^{-1}\widetilde{L}$ and now show the other inclusion, $\widetilde{w_i}.v \in \widetilde{L} \Rightarrow v \in \text{post}_{\widetilde{\mathcal{A}},q_i}$. If $\widetilde{w_i}.v \in \widetilde{L}$, then by definition, there is an accepting path labelled by $\widetilde{w_i}.v$ in $\widetilde{\mathcal{A}}$. We can extract from this path a path labelled by $\widetilde{w_i}$, which will reach a state, namely q. By definition of w_i, $\text{post}_{\mathcal{A},q} \supseteq \text{post}_{\mathcal{A},q_i}$, from which we deduce that $\text{pre}_{\mathcal{A},q} \subseteq \text{pre}_{\mathcal{A},q_i}$, from which we deduce that $\text{post}_{\widetilde{\mathcal{A}},q} \subseteq \text{post}_{\widetilde{\mathcal{A}},q_i}$. We have shown that the $\text{post}_{\widetilde{\mathcal{A}},q_i}$ are residuals of $\widetilde{L}$. Thus $\widetilde{\mathcal{A}}$ is a RFSA, so $\mathcal{A}$ is a biRFSA, and L a biRFSA language. There is no smaller NFA than the canonical RFSA which has as many states as the minimal DFA. □

Now, Theorem 2 follows by observing that the residuals of the corresponding languages verify the conditions of Lemma 1.

Boolean combinations of atomic constraints. We first consider general boolean combinations of atomic constraints. A boolean combination of formulas $\varphi_1, \ldots, \varphi_n$ is a formula generated by $\top, \bot, \varphi_1, \ldots, \varphi_n, \neg, \vee$ or $\wedge$. We denote by $C(\varphi_1, \ldots, \varphi_n)$ such a boolean combination. We define the notion of *simple* boolean combination as follows. The underlying propositional formula corresponding to $C(\varphi_1, \ldots, \varphi_n)$ is $C(b_1, \ldots, b_n)$ where $b_1, \ldots, b_n$ are propositional variables. We say that $C(b_1, \ldots, b_n)$ is simple, if the truth value of the formula depends on all propositional variables. Then, a boolean combination $C(\varphi_1, \ldots, \varphi_n)$ is simple, if its underlying propositional formula $C(b_1, \ldots, b_n)$ is simple. From any boolean combination $C(\varphi_1, \ldots, \varphi_n)$ we can always obtain a simple one by removing some atomic formulas if needed.

To build an automaton for a boolean combination of atomic formulas, we build a product automaton whose states are Presburger formulas (not tuples of formulas).

Definition 2. *Given a boolean combination of atomic formulas $C(\varphi_1(\mathbf{x}), \ldots, \varphi_n(\mathbf{x}))$, the product automaton $A_{C(\varphi_1(\mathbf{x}),\ldots,\varphi_n(\mathbf{x}))}$ is given by: Q is the set of Presburger formulas and the designated final state F, $q_0 = C(\varphi_1(\mathbf{x}), \ldots, \varphi_n(\mathbf{x}))$ and for all $b \in \Sigma_k$, $\delta(C(\psi_1(\mathbf{x}), \ldots, \psi_n(\mathbf{x})), b) = C(\psi'_1(\mathbf{x}), \ldots, \psi'_n(\mathbf{x}))$ each $\psi_i(\mathbf{x})$ being a state, possibly $\bot$, of $\mathcal{A}_{\varphi_i}$ (the automaton of φ_i), and $\psi'_i(\mathbf{x}) = \delta_{\varphi_i}(\psi_i(\mathbf{x}), b)$. If $s \in S^k$, then $\delta(C(\psi_1(\mathbf{x}), \ldots, \psi_n(\mathbf{x})), s) = F$, when $\langle s \rangle \in [\![C(\psi_1(\mathbf{x}), \ldots, \psi_n(\mathbf{x}))]\!]$ and $\delta(C(\psi_1(\mathbf{x}), \ldots, \psi_n(\mathbf{x})), s) = \bot$ otherwise.*

It is clear that this construction provides us with a deterministic finite automaton. To exhibit the gain of the canonical RFSA over the minimal DFA for the fragment of boolean combinations of inequations, we need to characterise precisely the number of states of the minimal DFA and the canonical RFSA.

Proposition 2. *If we consider a simple boolean combination of inequations $C(\mathbf{a}_1.\mathbf{x} > c_1, \ldots, \mathbf{a}_n.\mathbf{x} > c_n)$, such that $(\mathbf{a}_i)_{1\leq i\leq n}$ form a linearly independent family of vectors, then the product automaton is the minimal DFA.*

Proof. To prove this proposition we need first a simple lemma from linear algebra.

Lemma 2. *If $(\mathbf{a}_i)_{1\leq i\leq n}$ is a linearly independent family of vectors, then there is a family of integer vectors $(\mathbf{u}_1, \ldots, \mathbf{u}_n)$ such that $\mathbf{a}_i.\mathbf{u}_j = 0$ for all j with $i \neq j$ and $\mathbf{a}_i.\mathbf{u}_i \neq 0$.*

We show that any two states of the automaton for the formula $C(\mathbf{a}_1.\mathbf{x} > c_1, \ldots, \mathbf{a}_n.\mathbf{x} > c_n)$ are non-equivalent by exhibiting a word that is in one's residual and not in the other's, i.e. a representation of a solution of the formula characteristic of one state that is not a solution of the formula characteristic of the other. We consider two states $C(\mathbf{a}_1.\mathbf{x} > c_1, \ldots, \mathbf{a}_n.\mathbf{x} > c_n)$ and $C(\mathbf{a}_1.\mathbf{x} > c'_1, \ldots, \mathbf{a}_n.\mathbf{x} > c'_n)$. They are distinct if there is an i with $c_i \neq c'_i$ (we assume w.l.o.g that $c_i < c'_i$). We can find an integer vector $\mathbf{v}$ with $\mathbf{a}_i.\mathbf{v} = c'_i$. As the boolean combination is simple we can find $b_1, \ldots, b_n$ booleans such that (seeing C as a boolean function) $C(b_1, \ldots, b_{i-1}, true, b_{i+1}, \ldots, b_n) \neq C(b_1, \ldots, b_{i-1}, false, b_{i+1}, \ldots, b_n)$. For each $j \neq i$, we can find an integer λ_j with $\mathbf{a}_j.(\mathbf{v} + \lambda\mathbf{u}_j) > max(c_j, c'_j)$ if b_j is true or $\mathbf{a}_j.(\mathbf{v} + \lambda_j\mathbf{u}_j) < min(c_j, c'_j)$ if b_j is false (the u_j's are taken from the family defined in Lemma 2). By definition $\mathbf{a}_j.(\mathbf{v} + \sum_{k\neq i} \lambda_k\mathbf{u}_k) = \mathbf{a}_j.(\mathbf{v} + \lambda_j\mathbf{u}_j)$, thus $(\mathbf{v} + \sum_{k\neq i} \lambda_k\mathbf{u}_k)$ is a solution of only one of the two states: they are not equivalent. As no two states are equivalent, the automaton is minimal. □

We now analyse the number of states of the automaton for disjunctions of atomic inequations $\mathbf{a}.\mathbf{x} > c$. We have shown for one inequation that the set of states can be partitioned into two subsets: a set (depending on $\mathbf{a}$) which is a max. SCC and a possibly empty set of other states that reach this SCC. The product automaton has the same structure. Notice that we can canonically map the max. SCC of the automaton for an atomic inequation to an interval. As we use a forward construction of the automaton, not every state in the cartesian product of the basic automata is reached. We now determine precisely the shape of the maximal SCC in the product automaton.

Given a boolean combination of inequations $C(\mathbf{a}_1.\mathbf{x} > c_1, \ldots, \mathbf{a}_n.\mathbf{x} > c_n)$, we will characterise the set of states (aside from $\perp$ and F) of the product automaton we reach by the forward construction. It should be clear that the set of states will be a subset of $\{C(\mathbf{a}_1.\mathbf{x} > \gamma_1, \ldots, \mathbf{a}_n.\mathbf{x} > \gamma_n) \mid \gamma_i \in \mathbb{Z}\}$. We will therefore map states to elements of $\mathbb{Z}^n$. We define the polyhedron $\Pi = \{\mathbf{v} \in \mathbb{Q}^n \mid \exists\lambda \in]0, 1[^n, \mathbf{v} = (-\mathbf{a}_1.\lambda, \ldots, -\mathbf{a}_n.\lambda)\}$.

Theorem 3. *Given a boolean combination of inequations, the set of integer points in $\mathcal{P} = \Pi +]-1, 0[^n$ form a maximal SCC in the corresponding product automaton.*

Proof. We show in two steps that every integer point in $\mathcal{P}$ is reachable from any state. First we compute a candidate word, then we show that this candidate allows us to reach that state. Suppose we want to reach a state $\gamma \in \Pi +]-1, 0[^n$ from the state $\mathbf{c}$.

What makes this reachability problem difficult is that we work with integers. Let us first define a similar reachability problem in $\mathbb{Q}^n$. Let $w = b_1 \dots b_p$ be a word of Σ_k. We define inductively the sequence $(\mathbf{r}_j)_{0 \le j \le p}$ as $\mathbf{r}_0 = \mathbf{c}$ and $\mathbf{r}_{j+1} = \frac{\mathbf{r}_j - (\mathbf{a}_1.b_{j+1}, \cdots, \mathbf{a}_n.b_{j+1})}{2}$. We can deduce that $\mathbf{r}_p = \frac{\mathbf{c}}{2^p} - \frac{(\mathbf{a}_1.\langle w \rangle_+, \cdots, \mathbf{a}_n.\langle w \rangle_+)}{2^p}$.

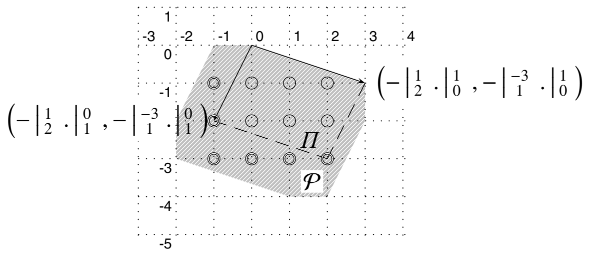
As $\gamma \in \Pi +]-1, 0[^n$, we can write $\gamma = \mathbf{u} + \mathbf{e}$ with $\mathbf{u} \in \Pi$ and $\mathbf{e} \in]-1, 0[^n$. By definition of Π, there exists $\lambda \in]0, 1[^n$ such that $\mathbf{u} = -(\mathbf{a}_1.\lambda, \dots, \mathbf{a}_n.\lambda)$. It should be clear that we can choose w such that $\frac{\langle w \rangle_+}{2^{|w|}}$ is arbitrarily close to any vector of $[0, 1]^n$, thus arbitrarily close to λ, and therefore we can choose w such that $\frac{(\mathbf{a}_1.\langle w \rangle_+, \dots, \mathbf{a}_n.\langle w \rangle_+)}{2^p}$ is arbitrarily close to $\mathbf{u}$. If we prefix w by $(0, \dots, 0)^*$ we do not change the value of $\frac{\langle w \rangle_+}{2^{|w|}}$ but we increase its length, and thus $\frac{\mathbf{c}}{2^{|w|}}$ gets arbitrarily small. Therefore we can choose w such that $\mathbf{r}_p$ is arbitrarily close to $\mathbf{u}$.

We now have a candidate w to reach the state γ: we want to show that the path labelled by w in the product automaton reaches γ. We define the sequence $(\mathbf{q}_j)_{0 \le j \le p}$ that represents the path of w in the product automaton: $\mathbf{q}_0 = \mathbf{c}$ and $\mathbf{q}_{j+1} = \left\lfloor \frac{\mathbf{q}_j - (\mathbf{a}_1.b_{j+1}, \dots, \mathbf{a}_n.b_{j+1})}{2} \right\rfloor$ (where $\lfloor \cdot \rfloor$ means applying the floor function componentwise), $\mathbf{q}_p$ is the state reached by w from $\mathbf{c}$. We can show inductively that $0 \le \mathbf{r}_j - \mathbf{q}_j < 1$, indeed $\mathbf{r}_{j+1} - \mathbf{q}_{j+1} = \frac{\mathbf{r}_j - \mathbf{q}_j}{2} + \left(\frac{\mathbf{q}_j - (\mathbf{a}_1.b_{j+1}, \dots, \mathbf{a}_n.b_{j+1})}{2} - \left\lfloor \frac{\mathbf{q}_j - (\mathbf{a}_1.b_{j+1}, \dots, \mathbf{a}_n.b_{j+1})}{2} \right\rfloor \right)$, so we can deduce that $\mathbf{q}_p = \lfloor \mathbf{r}_p \rfloor$.

By definition of $\lfloor . \rfloor$, there is $\mathbf{f} \in [0, 1[^n$ such that $\lfloor \mathbf{r}_p \rfloor = \mathbf{r_p} + \mathbf{f}$, thus $\mathbf{q}_p = \gamma + \mathbf{f} - (\mathbf{e} + (\mathbf{u} - \mathbf{r}_p))$. As $\mathbf{r}_p$ is arbitrarily close to $\mathbf{u}$, $\mathbf{e} + (\mathbf{u} - \mathbf{r}_p) \in]-1, 0[^n$. Since $\mathbf{q}_p$ and γ have integer components and $\mathbf{f} \in [0, 1[^n$, we have $\mathbf{q}_p = \gamma$.

Thus, any integer point in $\mathcal{P}$ is a state of the product automaton, $\mathcal{P}$ forms an SCC which is maximal, as from any state $\gamma \in \mathcal{P}$ one can only reach states inside $\mathcal{P}$. $\square$

Figure 2 gives an example of the states of the product automaton reached for a disjunction of two simple inequations together with its polyhedra Π and $\mathcal{P}$.



1
-3
-2
-1
0
1
2
3
4
0
-1
-3
-4
-5
$\left(-\left|\begin{smallmatrix}1\\2\end{smallmatrix}\right. \cdot \left|\begin{smallmatrix}1\\0\end{smallmatrix}\right., -\left|\begin{smallmatrix}-3\\1\end{smallmatrix}\right. \cdot \left|\begin{smallmatrix}1\\0\end{smallmatrix}\right.\right)$
$\left(-\left|\begin{smallmatrix}1\\2\end{smallmatrix}\right. \cdot \left|\begin{smallmatrix}0\\1\end{smallmatrix}\right., -\left|\begin{smallmatrix}-3\\1\end{smallmatrix}\right. \cdot \left|\begin{smallmatrix}0\\1\end{smallmatrix}\right.\right)$
Π
$\mathcal{P}$

Fig. 2. $\mathbb{Z}^2$-states for $x - 3y > c_1 \vee 2x + y > c_2$, the doubly circled states are the states of the canonical RFSA

We precisely characterised the set of states forming the maximal SCC constructed by our forward construction of the automaton for disjunctions of inequations. This automaton is always minimal when the inequations are linearly independent.

We study now what the gain of using non-determinism is. We analyse the number of states of RFSA for conjunctions and disjunctions of inequations. The case of disjunction of inequations appears to be the most simple for building non-deterministic automata. For example we can take the simple union of the automata for each inequation of the disjunction. However they are not necessarily RFSA.

Proposition 3. *Given a disjunction of inequations, let Q be the set of states of the minimal DFA seen as a subset of $\mathbb{Z}^n$, by canonically mapping $\bigvee_{i=1}^n \mathbf{a}_i.\mathbf{x} > \gamma_i$ to $(\gamma_1, \dots, \gamma_n)$. Then, the states of the canonical RFSA are in $\mathcal{S} = \{\gamma \in Q \mid \{\gamma\} + \{-1, 0\}^n \not\subseteq Q\}$.*

Proof. By definition, the residual of a state γ is not prime if and only if there exists a family of states $(\gamma^{(k)})$ $(\gamma \notin (\gamma^{(k)}))$, such that the residual of $\bigvee_{i=1}^{n} \mathbf{a}_i.\mathbf{x} > \gamma_i$ is equivalent to the union over k of the residuals of $\bigvee_{i=1}^{n} \mathbf{a}_i.\mathbf{x} > \gamma_i^{(k)}$. If $\gamma \notin \mathcal{S}$, the $(\gamma^{(k)} = \gamma - \mathbf{e}_k^n)$ are all states in Q, it is straightforward to notice that $\bigvee_i \mathbf{a}_i.\mathbf{x} > \gamma_i = \bigcup_i \bigvee_j \mathbf{a}_j.\mathbf{x} > \gamma_j^{(i)}$, thus any state not in $\mathcal{S}$ is not prime. □

We can then heuristically characterise the gain in the number of states of the RFSA in comparison with the minimal DFA. Instead of the whole polyhedron representing the max. SCC, the RFSA only has at most the states of its lower surface $\mathcal{S}$.

The RFSA is in general bigger than the non-deterministic automaton built as union of automata. Disjunctions of inequations having not many variables in common have small minimal RFSA compared to minimal DFA, for example the minimal DFA of a formula of the form $\bigvee_{i=1}^{n/2} x_{2i} + x_{2i+1} > 0$ has an exponential number of states in n, whereas the minimal RFSA has a linear number of states.

Actually even for a fixed number of variables we can have an exponential gain. For the language $\cup_{l \leq n} \Sigma_1^l.1.\Sigma_1^n.1.0^*.+$ (which only represents a finite, thus Presburger, set of integers) the minimal RFSA is exponentially smaller than the minimal DFA. It also shows that there are Presburger sets for which the most significant bit first coding is accepted by an exponentially smaller automaton compared to the least significant bit first coding.

We now provide a characterisation of the canonical RFSA for a conjunction of inequations, which surprisingly can be as small as the canonical RFSA for disjunctions.

Proposition 4. *Given a conjunction of inequations $\bigwedge_i \mathbf{a}_i.\mathbf{x} > c_i$, let Q be the set of states of its minimal DFA, seen as a subset of $\mathbb{Z}^n$. Let $\mathcal{S}' = \{\gamma \in Q \mid \exists k, \gamma + \mathbf{e}_k^n \notin Q\}$ and $\mathcal{G} = \{\gamma \mid \bigwedge_{i=1}^{n} \mathbf{a}_i.\mathbf{x} = \gamma_i + 1$ has no integer solution$\}$. The set of states (seen as elements of $\mathbb{Z}^n$) of the canonical RFSA for this conjunction is a subset of $\mathcal{S}' \cup \mathcal{G}$. If $\mathbf{c}$, the initial state, is in $\mathcal{P}$ and the $\mathbf{a}_i$ are linearly independent, this is exactly the set of states of the canonical RFSA.*

Proof. Here again, the overapproximation is proved by looking at the definition of an RFSA. We remark that $\bigvee_k \bigwedge_i \mathbf{a}_i.\mathbf{x} > \gamma_i + \delta_k(i)$ ($\delta_k(i) = 1$ when $k = i$ and 0 otherwise) is equivalent to $\bigwedge_i \mathbf{a}_i.\mathbf{x} > \gamma_i \wedge \neg \bigvee_i \gamma_i + 1 \geq \mathbf{a}_i.\mathbf{x} > \gamma_i$. If $\bigwedge_{i=1}^{n} \mathbf{a}_i.\mathbf{x} = \gamma_i + 1$ has no integer solution, this is equivalent to $\bigwedge_i \mathbf{a}_i.\mathbf{x} > \gamma_i$, thus if a state is not in $\mathcal{S}' \cup \mathcal{G}$, it is not prime.

We now prove that we have characterised the states with prime residuals when the $\mathbf{a}_i$ are linearly independent and the initial state is in $\mathcal{P}$. The second hypothesis allows us to use the convexity of $\mathcal{P}$. We assume that the union of the residuals of the states of the family $(\gamma^{(k)})$ $(\gamma \notin (\gamma^{(k)}))$ is equal to the residual of γ. Then, we show that $\gamma \in \mathcal{S}' \cup \mathcal{G}$. As the $\mathbf{a}_i$ are linearly independent, for any j and κ, $\bigwedge_{i \neq j} \mathbf{a}_i.\mathbf{x} = \gamma_i + 1 \wedge \mathbf{a}_j.\mathbf{x} > \kappa$ characterises an affine half-space of dimension at least 1. Thus such a system has infinitely many solutions. So, for any k, only the residuals of the states $\bigwedge_{i \neq j} \mathbf{a}_i.\mathbf{x} > \gamma'_i + 1 \wedge \mathbf{a}_j.\mathbf{x} > \gamma'_j$ with $\gamma'_i \leq \gamma_i$ (for all $i \neq j$) contain these solutions. Only those states with $\gamma'_i \geq \gamma_i$ are included in the residual of γ. Thus $(\gamma^{(k)})$ necessarily contains, for all j a $(\gamma_1, \ldots, \gamma_{j-1}, \kappa_j, \gamma_{j+1}, \ldots)$ with some $\kappa_j > \gamma_j$. With the convexity of $\mathcal{P}$, we have $\gamma \notin \mathcal{S}'$. □

4 Complexity of the Automata Based Decision Procedure

The well-known decision procedure for Presburger arithmetic using automata is based on recursively constructing an automaton accepting solutions of a Presburger formula by using automata constructions for handling logical connectives and quantifiers. Automata for basic formulas can be easily constructed (see section 2). Each logical connective ($\wedge$,$\vee$, $\neg$) corresponds then naturally to an operation on automata. Furthermore to get an automaton for $\exists y.\varphi(y, \mathbf{x})$ given an automaton for $\varphi(y, \mathbf{x})$ one *projects away* [1] the component for y and obtains a *non-deterministic* automaton. Then, this automaton is determinised to be able to continue the recursive construction and minimised. Given a formula of size n, Klaedtke has shown [10] — by eliminating the quantifiers on the logical level and translating the obtained quantifier-free formula to an automaton — that the minimal DFA obtained *at the end* of this procedure and the minimal DFA for subformulas obtained *during* the procedure have triple-exponential size. However, the DFA obtained after determinising the NFA which is a result of a projection might be of quadruple exponential size, as an automaton of triple exponential size is determinised. In this section we show that *all* automata obtained during the construction are in fact of at most triple exponential size solving an open problem from [10]. We do this by carefully inspecting the structure of the NFA which is determinised. Notice that our upper bound is obtained using the least significant bit first coding of integer vectors which allows to reason conveniently about states of the automata corresponding to formulas.

The following theorem on quantifier elimination is from Klaedtke [10]. We use here a simplified version, where the only parameter is the length of the formula. In [10], other parameters (e.g. alternation depth) are used. Given a quantifier free Presburger formula ψ, let d_ψ be the number of different atomic modulo constraints of the form $\mathbf{a}.\mathbf{x} \equiv_m \beta$ and $max_{div}(\psi)$ the biggest value of m appearing in them. Let t_ψ be the number of *different* vectors $\mathbf{a}$ appearing in atomic formulas of the form $\mathbf{a}.\mathbf{x} \sim \gamma$ with $\sim \in \{=, \neq, <, >, \leq, \geq, \}$ in ψ, $max_{coef}(\psi)$ the biggest absolute value of their coefficients and $max_{const}(\psi)$ the biggest absolute value of the constants γ appearing in them. We use the abbreviations $exp2(x) = 2^{2^x}$ and $exp3(x) = 2^{2^{2^x}}$.

Theorem 4 ([10], Theorem 4.5). *For every Presburger formula φ of length n, there is a semantically equivalent quantifier free formula ψ such that: $t_\psi \leq exp2(cn \log n)$, $d_\psi \leq exp2(cn \log n)$, $max_{coef}(\psi) < exp2(cn)$, $max_{div}(\psi) < exp2(cn)$ and $max_{const}(\psi) < exp3(cn \log n)$, where c is a constant independent of n.*

We can suppose w.l.o.g. that ψ is a boolean combination of modulo constraints of the form $\mathbf{a}.\mathbf{x} \equiv_m \beta$ and of atomic formulas of the form $\mathbf{a}.\mathbf{x} > \gamma$ only. The following theorem gives a bound on the size of the minimal DFA accepting solutions of a Presburger formula. Klaedtke gives a corresponding theorem for the most significant bit first coding. His proof is simpler due to the fact that only one automaton has to be constructed for all inequations with the same coefficients.

Theorem 5. *The size of the minimal DFA accepting solutions of a Presburger formula φ of length n is at most $exp3(cn \log n)$ for some constant c.*

[1] Since the automaton should accept all encodings of the solutions, one has to sometimes add additional transitions with a sign letter going to the final state.

Proof. Let k be the number of free variables of φ. Let ψ be the quantifier free formula obtained from φ using Theorem 4. We have $t_\psi \leq exp2(c_1 n \log n)$, $d_\psi \leq exp2(c_1 n \log n)$, $max_{coef}(\psi) < exp2(c_1 n)$, $max_{div}(\psi) < exp2(c_1 n)$ and $max_{const}(\psi) < exp3(c_1 n \log n)$ for some constant c_1. If we build the product automaton for the quantifier free formula ψ equivalent to φ according to Definition 2, a naive analysis of its size gives a quadruply exponential automaton, since there are possibly $t_\psi^{2max_{const}(\psi)}$ distinct inequations in ψ. However a closer inspection reveals a triple exponential bound, due to the special structure of automata for inequations. Here, we give a slightly different construction of the automaton A_ψ accepting solutions of ψ which we will use in the rest of the section.

Let $\mathbf{a}_1, \ldots, \mathbf{a}_{t_\psi}$ be the different vectors appearing in the atomic inequations of ψ and $\psi_1, \ldots, \psi_{l_\psi}$ an enumeration of all atomic formulas of the form $\mathbf{a}_i.\mathbf{x} > \gamma_j$ for all $1 \leq i \leq t_\psi$ and γ_j with $|\gamma_j| \in [-||\mathbf{a}_i||_+ - 1, ||\mathbf{a}_i||_-]$. Clearly, $l_\psi \leq exp2(c_2 n \log n)$ for some constant c_2. Let $\phi_1, \ldots, \phi_{d_\psi}$ be an enumeration of all the modulo constraints appearing in ψ and $\mathcal{BC}$ be the set of boolean combinations of the form $C(\psi_1, \ldots, \psi_{l_\psi}, \phi_1, \ldots, \phi_{d_\psi})$. For each member of $\mathcal{BC}$ an automaton can be built with the product construction of Definition 2.

We describe now informally the automaton A_ψ we construct from ψ. It has first the form of a complete tree starting at the initial state. Its branching factor is the size of the alphabet Σ_k and its depth is $exp2(c_1 n \log n)$. Each of the states in the tree recognises the solutions of the formula $\psi(2^{|w|}\mathbf{x} + \langle w \rangle_+)$ where $w \in \Sigma_k^*$ with $|w| \leq exp2(c_1 n \log n)$ is the word leading to the state from the initial state. Then, at level $exp2(c_1 n \log n)$ there are separate automata accepting solutions of the corresponding formulas reached after reading the word leading to them. All these automata correspond to boolean combinations of $\mathcal{BC}$. Indeed, for any atomic formula $\zeta(\mathbf{x}) = \mathbf{a}.\mathbf{x} > \gamma$ of ψ and any word $w \in \Sigma_k^*$ with $|w| = exp2(c_1 n \log n)$ we have $\zeta(2^{|w|}\mathbf{x} + \langle w \rangle_+) \Leftrightarrow \mathbf{a}.\mathbf{x} > \gamma'$ for some $\gamma' \in [-||a||_+ - 1, ||a||_-]$. Therefore, for any atomic subformula $\zeta(\mathbf{x})$ of ψ, $\zeta(2^{|w|}\mathbf{x} + \langle w \rangle_+)$ is equivalent to a ψ_i, so $\psi(2^{|w|}\mathbf{x} + \langle w \rangle_+)$ is equivalent to a formula of $\mathcal{BC}$. Notice that in any member of $\mathcal{BC}$ *all* atomic formulas of a given form appear. That is not a restriction, since we can just expand each boolean combination to be of this form. Let $W = \{w \in \Sigma_k^* \mid |w| = exp2(c_1 n \log n)\}$. For any $w \in W$, let $C_w \in \mathcal{BC}$ be the boolean combination equivalent to $\psi(2^{|w|}\mathbf{x} + \langle w \rangle_+)$. For each C_w we can construct an automaton $A_{C_w} = (Q_w \cup \{F\}, q_{w,0}, \{F\}, \delta_w)$ according to Definition 2. Notice that the automata A_{C_w} only differ in the transitions going to the final state, since the atomic formulas composing them are all the same. The final state F is the same in each automaton.

We can now give the definition of the automaton for the formula ψ formally, i.e. $A_\psi = (Q, q_\epsilon, \{F\}, \delta)$ where $Q = Q_1 \cup Q_2 \cup \{F\}$ with $Q_1 = \{q_w \mid w \in \Sigma_k^* \wedge |w| < exp2(c_1 n \log n)\}$ and $Q_2 = \bigcup_{w \in W} Q_w$. Furthermore, $\delta(q_w, b) = \{q_{wb}\}$ for all $b \in \Sigma_k$ and $|w| < exp2(c_1 n \log n) - 1$, $\delta(q_w, b) = \{q_{wb,0}\}$ for all $b \in \Sigma_k$ and $|w| = exp2(c_1 n \log n) - 1$ and $\delta(q, b) = \delta_w(q, b)$ for all $b \in \Sigma_k$ and $q \in Q_2$. Clearly, the number of states (and also the size) of the automaton A_ψ is smaller than $exp3(cn \log n)$ for some constant c. □

We can now prove the main theorem of the section which shows that elimination of a variable does not lead to an exponential blow-up in the size of the automaton.

Theorem 6. *Let $\exists y.\varphi(y, \mathbf{x})$ be a Presburger formula of size n, A the minimal DFA accepting the solutions of $\varphi(y, \mathbf{x})$ and A' the automaton obtained by projecting A on* $\mathbf{x}$*. Then, the automaton A'' obtained by determinising A' with the standard on-the-fly subset construction is of size at most $exp3(cn \log n)$ for some constant c.*

Proof. Theorem 4 yields a quantifier free formula ψ semantically equivalent to $\varphi(y, \mathbf{x})$ with $t_\psi \leq exp2(c_1 n \log n)$, $d_\psi \leq exp2(c_1 n \log n)$, $max_{coef}(\psi) < exp2(c_1 n)$, $max_{div}(\psi) < exp2(c_1 n)$ and $max_{const}(\psi) < exp3(c_1 n \log n)$ for some constant c_1. For ψ, according to Theorem 5, there is an automaton $A_\psi = (Q, q_0, \{F\}, \delta)$ of triple-exponential size (not necessarily minimal) accepting the solutions of ψ. We use the same notation as in the proof of Theorem 5 for the parts of the automaton. A is the minimal automaton corresponding to A_ψ. Obviously, A_ψ might be bigger than A. We show that the size of the automaton A''_ψ obtained by determinising (using the standard on-the-fly subset construction) A'_ψ which is the automaton obtained from A_ψ by projecting away y is bounded by $exp3(cn \log n)$. Then the bound on A'' (whose states are sets of states of A') follows, as two different states in A'' correspond to two different states in A''_ψ.

Since we use for the determinisation of A'_ψ the standard subset construction, states of A''_ψ are sets of states of A'_ψ which are exactly the states of A_ψ.

We first introduce some notations. Let k be the number of free variables of $\varphi(y, \mathbf{x})$ and ψ. For any word $w \in \Sigma_k^*$ we denote by $w{\downarrow_1} \in \Sigma_{k-1}^*$ the word obtained from w by projecting away the first component and by $w{\downarrow_2} \in \{0,1\}^*$ the word obtained from w by projecting on the first component. For any $w \in \Sigma_{k-1}^*$ we define $w{\uparrow} = \{w' \in \Sigma_k^* \mid w'{\downarrow_1} = w\}$. For any $w \in \Sigma_{k-1}^*$ and $z \in [0, 2^{|w|} - 1]$ we define $w{\uparrow^z} = w'$, if $\langle w'{\downarrow_2}\rangle_+ = z$ and $w'{\downarrow_1} = w$.

Let $S = \{\widehat{\delta}(w{\uparrow}, \{q_0\}) \mid w \in \Sigma_{k-1}^*\}$. Our goal is to show that the size of S is bounded by a triple-exponential. This implies that the number of states of A''_ψ has the same bound. We split S into two sets $S_<$ and $S_\geq$ where $S_< = \{\widehat{\delta}(w{\uparrow}, \{q_0\}) \mid w \in \Sigma_{k-1}^* \wedge |w| < exp2(c_1 n \log n)\}$ and $S_\geq = \{\widehat{\delta}(w{\uparrow}, \{q_0\}) \mid w \in \Sigma_{k-1}^* \wedge |w| \geq exp2(c_1 n \log n)\}$. It is obvious that $|S_<| \leq exp3(c_2 n \log n)$ for some constant c_2. We now show a bound on $|S_\geq|$. We first enumerate all words $w \in \Sigma_{k-1}^*$ of size exactly $exp2(c_1 n \log n)$ as $w_1, \ldots, w_m$ where $m \leq exp3(c_3 n \log n)$ for some constant c_3. We have $S_\geq = \bigcup_{i=1}^m S_i$ where $S_i = \{\{\widehat{\delta}(w_i w{\uparrow}, \{q_0\}) \mid w \in \Sigma_{k-1}^*\}$. We will show that $|S_i| \leq exp3(c_4 n \log n)$ for some constant c_4 which implies that $S_\geq$ is bounded by a triple-exponential as well. We have $S_i = \{\widehat{\delta}(w{\uparrow}, \widehat{\delta}(w_i{\uparrow}, \{q_0\})) \mid w \in \Sigma_{k-1}^*\} = \bigcup_{z \in [0, (exp3(c_1 n \log n) - 1)]} \{\widehat{\delta}(w{\uparrow}, \widehat{\delta}(w_i{\uparrow^z}, \{q_0\})) \mid w \in \Sigma_{k-1}^*\}$. Let $S_i^0 = \{\widehat{\delta}(w{\uparrow}, \widehat{\delta}(w_i{\uparrow^0}, \{q_0\})) \mid w \in \Sigma_{k-1}^*\}$. We have $|S_i| = |S_i^0|$, as $\widehat{\delta}(w_i{\uparrow^0}, \{q_0\})) = C_{w_i\uparrow^0}$ and for all $0 < z \leq exp3(c_1 n \log n) - 1$ we have $\widehat{\delta}(w_i{\uparrow^z}, \{q_0\}) = C_{w_i\uparrow^z}$ and all automata corresponding to $C_{w_i\uparrow^z}$ for $0 \leq z \leq exp3(c_1 n \log n) - 1$ are the same except for the transitions leading to the final state. That means that there is a one-to-one correspondence between each state in sets of states of S_i^0 and each state in sets of states of the other S_i^z.

Now, we derive a bound of $|S_i^0|$. The word $w_i{\uparrow^0}$ leads to the state $C_{w_i\uparrow^0}$ in A_ψ which is the initial state of an automaton, call it A_0, recognising all solutions of $C_{w_i\uparrow^0}$. A_0 is obtained as a product of automata for atomic inequations and modulo constraints. Let $\psi_1, \ldots, \psi_{l_\psi}$ be the atomic formulas which are inequations and $\phi_1, \ldots, \phi_{d_\psi}$ the atomic formulas which are modulo constraints of the boolean combination $C_{w_i\uparrow^0}$. In the following it is convenient to consider states of A_0 to be $(l_\psi + d_\psi)$-tuples of states instead of considering them as formulas. That is, a state $C(\psi'_1, \ldots, \psi'_{l_\psi}, \phi'_1, \ldots, \phi'_{d_\psi})$ is considered to be the tuple $(\psi'_1, \ldots, \psi'_{l_\psi}, \phi'_1, \ldots, \phi'_{d_\psi})$. For $1 \leq i \leq l_\psi$ let $\psi_i(y, \mathbf{x}) = a_1^i y + \mathbf{a}^i.\mathbf{x} > \gamma_i$ and for $1 \leq i \leq d_\psi$ let $\phi_i(y, \mathbf{x}) = b_1^i y + \mathbf{b}^i.\mathbf{x} \equiv_{m_i} \beta_i$.

Let us fix a $w \in \Sigma_{k-1}^*$ and let $w' \in w{\uparrow}$. Let $y' = \langle w'{\downarrow_2}\rangle_+$ and $l = |w| = |w'|$. It is clear that $0 \leq y' < 2^l$. For each $1 \leq i \leq t_\psi$, the state reached in $A_{a_1^i y + \mathbf{a}^i.\mathbf{x} > \gamma_i}$ by w' is the state semantically equivalent to $a_1^i(2^l y + y') + \mathbf{a}^i.(2^l\mathbf{x} + \langle w\rangle_+) > \gamma_i$ which is equivalent to $a_1^i y + \mathbf{a}^i.\mathbf{x} > (\gamma_i - a_1^i y' - \mathbf{a}.\langle w\rangle_+)/2^l$.

Therefore, the first t_ψ components of the states reached in A_0 by words $w' \in w\!\uparrow$ are semantically equivalent to $(a_1^1 y + \mathbf{a}^1.\mathbf{x} > (\gamma_1 - a_1^1 y' - \mathbf{a}.\langle w\rangle_+)/2^l, \ldots, a_1^{t_\psi} y + \mathbf{a}^{t_\psi}.\mathbf{x} > (\gamma_i - a_1^{t_\psi} y' - \mathbf{a}^{t_\psi}.\langle w\rangle_+)/2^l)$. There are 2^l different values for y'. However there are at most $\sum_{i=1}^{t_\psi}(|a_1^i| + 1)$ semantically different corresponding t_ψ-tuples of formulas, since $0 \leq y' < 2^l$ and therefore the semantics of the i-th atomic formula changes at most a_1^i times in a monotone fashion for increasing y'. Therefore if we consider the first t_ψ components of states reached by words of $w\!\uparrow$ in A_0, we get only $\sum_{i=1}^{t_\psi}(|a_1^i| + 1)$ semantically different ones, since the automata for basic formulas are minimal.

Now we consider the set of words $V = \{w' \in \Sigma_k^* \mid w'\!\downarrow_1 = w\}$ which lead to the *same* first t_ψ components of states in A_0 and consider the other components (corresponding to the modulo constraints) they can reach. The words in V differ only in the component corresponding to y. Clearly, the set $\{y' \mid y' = \langle w'\!\downarrow_2\rangle_+$ and $w' \in V\}$ is an interval of the form $[p, q]$ where $0 \leq p \leq q < 2^l$.

A state (formula) reached in $A_{b_1^i y + \mathbf{b}^i.\mathbf{x} \equiv_{m_i} \beta_i}$ after reading a word w' of V with $y' = \langle w'\!\downarrow_2\rangle_+$ is semantically equivalent to $2^l(b_1^i y + \mathbf{b}^i.\mathbf{x}) \equiv_{m_i} \beta_i - b_1^i y' - \mathbf{b}^i.\langle w'\rangle_+$. It is clear that there are at most m_i semantically different formulas of this kind. Furthermore, we can order them starting from $y' = 0$ until $y' = m_i - 1$. Then it is clear that the set of states (formulas) reached by words of V (whose corresponding y' form intervals) must be an interval of states respecting this order. There are at most m_i^2 such intervals.

Finally, we can give an upper bound on the number of subsets of states of $S_i^0 = \{\widehat{\delta}(w\!\uparrow, \{q_{w_i,0}\} \mid w \in \Sigma_{k-1}^*\}$ which are subsets of states of A_0. Given any word $w \in \Sigma_{k-1}^*$ we know from the above that words of $w\!\uparrow$ lead to at most $s := \sum_{i=1}^{t_\psi}(|a_1^i| + 1) \leq exp2(c_5 n \log n)$ (for some constant c_5) different tuples of the first t_ψ components of A_0. Furthermore, we know that the number of subsets of states of A_{ϕ_i} which can be reached simultaneously by words of subsets V of $w\!\uparrow$ such that all $w' \in V$ lead to the same tuples of the first t_ψ components is at most m_i^2. Therefore overall, S_i^0 has at most $|A_0|^s \Pi_{i=1}^{d_\psi} m_i^2 \leq exp3(cn \log n)$ states for some constant c and $|A_0|$ being the number of states of A_0. From this follows in turn a triple-exponential bound on $|S_i|$, $|S_\geq|$, the number of states and size of A''_ψ and A''. □

The number of transitions of an automaton is bounded by $|Q||\Sigma|$ for a DFA and possibly $|Q|^2|\Sigma|$ for an NFA. As Σ is at most simply exponential w.r.t. the size of the formula, the sizes of the automata build have all a triple-exponential upper bound as well. Therefore the following is an easy consequence of Theorem 6.

Corollary 1. *The automata based decision procedure for Presburger arithmetic takes triple-exponential time in the size of the formula.*

5 Conclusion

We have investigated the use of non-deterministic automata for Presburger arithmetic (PA). We show that for some subclasses NFA might lead to a substantial gain in the size of the automata. We plan to continue our investigations into other subclasses like conjunctions and disjunctions of modulo constraints together with inequations and equations as well as general formulas in conjunctive or disjunctive normal form. The use of alternating automata might improve the sizes as well. But we then lose the canonicity property of RFSA. We also plan to investigate how the use of RFSA improves the performance of tools for PA. RFSA have also been used recently in learning [7,2]. It remains to be seen if these learning algorithms lead to improved performances for learning based verification of counter systems as studied for example in [15] using DFA.

References

1. Bardin, S., Leroux, J., Point, G.: FAST Extended Release. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 63–66. Springer, Heidelberg (2006)
2. Bollig, B., Habermehl, P., Kern, C., Leucker, M.: Angluin-Style Learning of NFA. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, CA, USA, July 2009, pp. 1004–1009. AAAI Press, Menlo Park (2009)
3. Boudet, A., Comon, H.: Diophantine Equations, Presburger Arithmetic and Finite Automata. In: Kirchner, H. (ed.) CAAP 1996. LNCS, vol. 1059, pp. 30–43. Springer, Heidelberg (1996)
4. Büchi, J.: Weak second-order Arithmetic and Finite Automata. Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 6, 66–92 (1960)
5. Cooper, D.C.: Theorem Proving in Arithmetic without Multiplication. In: Machine Intelligence, pp. 91–100 (1972)
6. Denis, F., Lemay, A., Terlutte, A.: Residual Finite State Automata. Fundamenta Informaticae 51(4), 339–368 (2002)
7. Denis, F., Lemay, A., Terlutte, A.: Learning Regular Languages using RFSAs. Theoretical Comput. Sci. 313(2), 267–294 (2004)
8. Oppen, D.C.: A $2^{2^{2^{pn}}}$ Upper Bound on the Complexity of Presburger Arithmetic. J. Comput. Syst. Sci. 16(3), 323–332 (1978)
9. Fischer, M.J., Rabin, M.O.: Super-exponential Complexity of Presburger Aithmetic. In: Symp. on Applied Mathematics. SIAM-AMS Proceedings, vol. VII, pp. 27–41 (1974)
10. Klaedtke, F.: Bounds on the Automata Size for Presburger Arithmetic. ACM Trans. Comput. Logic 9(2), 1–34 (2008)
11. Latteux, M., Lemay, A., Roos, Y., Terlutte, A.: Identification of biRFSA Languages. Theoretical Computer Science 356(1-2), 212–223 (2006)
12. Leroux, J.: Structural Presburger Digit Vector Automata. Theoretical Computer Science 409(3), 549–556 (2008)
13. Presburger, M.: Uber die Vollstandigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du Premier Congres des Mathematicienes des Pays Slaves, pp. 92–101 (1929)
14. Reddy, C.R., Loveland, D.W.: Presburger Arithmetic with Bounded Quantifier Alternation. In: ACM Symposium on Theory of Computing, pp. 320–325 (1978)
15. Vardhan, A., Viswanathan, M.: Learning to verify branching time properties. Formal Methods in System Design 31(1), 35–61 (2007)
16. Wolper, P., Boigelot, B.: An Automata-theoretic Approach to Presburger Arithmetic Constraints. In: Mycroft, A. (ed.) SAS 1995. LNCS, vol. 983, pp. 21–32. Springer, Heidelberg (1995)
17. LASH homepage, http://www.montefiore.ulg.ac.be/~boigelot/research/lash/
18. FAST homepage, http://www.lsv.ens-cachan.fr/Software/fast/