SDA : Serialized Digit Automata
(LaBRI technical report RR-1397-06)

Jérôme Leroux[1]

LaBRI (CNRS UMR 5800), MVTSI Team, Talence, France
leroux@labri.fr

Abstract. Since 1969 [Cob69], the Presburger arithmetic [Pre29] is known to be solvable by automata recognizing decompositions of integer vectors, corresponding to words of digit vectors. An effective and efficient implementation of such a solver is not an easy task: boolean operations over the languages do not correspond to boolean operations over the set of integers, quantifications provide languages that do not have the correct form, digit vectors must be serialized in order to avoid an exponential size of the alphabet, and so on. In this article, we present a new automata based symbolic representation, called Serialized Digit Automata (SDA) for which all these problems can be solved in a simple way.

1 Introduction

Presburger arithmetic [Pre29] (the linear first order logic of the integers) is a decidable logic used in a large range of applications. Different techniques [GBD02] and tools have been developed for manipulating *the Presburger-definable sets* (the sets of integer vectors satisfying a Presburger formula): by working directly on the Presburger-formulas (implemented in Omega [Ome]), by using semi-linear sets [GS66] (implemented in Brain [RV02]), or by using automata recognizing 2-complement decompositions of integer vectors, corresponding to words of digit vectors, least or most significant digit vector first [WB00] (implemented in Fast [BFLP03], Lash [Las] and Csl-Alv [BB03]).

Presburger-formulas and semi-linear sets lack canonicity: there does not exist a natural way to canonically represent a set. As a direct consequence, a set that possesses a simple representation could unfortunately be represented in an unduly complicated way. Moreover, deciding if a given vector of integers is in a given set, is at least *NP-hard*. On the other hand, a minimization procedure for automata provides a canonical representation for automata based symbolic representations. That means, the automaton that represents a given set only depends on the set and not on the way we have computed it. For this reason, automata are well adapted for applications that require a lot of boolean manipulations like model-checking systems manipulating integer variables.

However, using automata for manipulating Presburger-definable sets is not an easy task since the following problems naturally appear:

– Boolean operations over the Presburger-definable sets do not correspond exactly to the equivalent operations over the regular languages. This *boolean stability issue* is due to the non-uniqueness of the decomposition of an integer vector into a word of digit vectors. This issue can be solved by considering *saturated languages* [KMS02]: a language is saturated if for any integer vector either all the decompositions of these vector are in the language, or all the decompositions are in the complement.
– Even if the existential quantification of Presburger-definable sets corresponds to a simple transformation over the languages, the language obtained from a saturated one after such a transformation is not saturated. This *projection stability issue* can be solved thanks to a *saturation* that must be performed after each projection. In particular complex operations involving both boolean operations and projections cannot be computed on the fly, as a unique operation. For instance, the universal quantification requires four operations: (1) a complementation (2) an existential quantification (3) a saturation and (4) a second complementation.
– The number of digit vectors is exponential in the dimension of the represented vectors. That means the alphabet of an automaton that recognizes a language of words of digit vectors has a cardinal size exponential in the dimension. A *serialization* [WB00] easily overcome this *dimension issue*: a vector of digits is just serialized into a word of digits.

Naturally, implementing an efficient saturation algorithm for serialized languages is one of the most important part of a library manipulating Presburger-definable sets by automata that is not a "usual" automata operation. In the case of non-negative integer vectors, a simple and nice solution is implemented in the tool Mona [Mon]. However, in the general case of (signed) integer vectors, the problem is much more complex. Even if such an algorithm is already implemented in the tool Lash, there is no paper that present the algorithm and the complexity of the problem is unknown.

Our results:

– We introduce a new class of languages called *unambiguous*: a language is unambiguous if for any integer vector either only a finite number of decompositions are not in the language, or only a finite number of decompositions are in the language. Since an integer vector owns an infinite number of decompositions, an unambiguous language naturally *represents* the set of integer vectors that only have a finite number of decompositions not in the language. As the class of saturated languages, this class is stable by boolean operations and it also solves the boolean stability issue.
– We prove that the existential quantification of a saturated language is an unambiguous language. That means complex operations involving quantification and boolean operations can be implemented on the fly and the projection stability issue is partially solved. We just say partially because the class of unambiguous languages is not stable by existential quantification. Hence

more complex operations involving alternation of universal and existential quantifications is not possible without a saturation algorithm.

– We introduce a new automata based symbolic representation called Serialized Digit Automata (SDA) that allows the saturation computation of an unambiguous language by a symbolic algorithm with a step by step linear time complexity that terminates in a quadratic number of steps. The word symbolic means that the algorithm performs digit by digit some computations rather than digit vectors by digit vectors in order to have a benefit of a serialization.

2 Notations

Usual notations: Throughout this paper, *intersection, union, difference and symmetric difference* of two sets A and B are written $A \cap B$, $A \cup B$, $A \backslash B$, and $A \Delta B = (A \backslash B) \cup (B \backslash A)$. We denote by $\mathbb{Z}$ and $\mathbb{N}$ respectively the set of integers and non-negative integers. The class of subsets (respectively finite sets) of a set A is denoted by $\mathcal{P}(A)$ (respectively $\mathcal{P}_f(A)$). The *cardinality* of a finite set X is written $|X| \in \mathbb{N}$. Given a finite subset $K \subseteq \mathbb{N}$, we denote by $\max(K) = -\infty$ if K is empty and $\max(K)$ is the greatest element for $\leq$ otherwise. Let $m \in \mathbb{N} \backslash \{0\}$. Given an integer $i \in \{1, \dots, m\}$ and a vector $x \in X^m$ where X is a non-empty set, we denote by $x[i]$ the the i-th component of x. A non empty finite Σ is called an *alphabet*. The sets Σ^*, Σ^+, the *empty word* ϵ, the length $|\sigma|$, the element $\sigma[i] \in \Sigma$ where $\sigma \in \Sigma^+$ and $i \in \{1, \dots, |\sigma|\}$, the *concatenation* $\sigma_1.\sigma_2$ or $\mathcal{L}_1.\mathcal{L}_2$ where $\sigma_1, \sigma_2 \in \Sigma^*$ and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$, the *residues* $\sigma^{-1}.\mathcal{L}$ are $\mathcal{L}.\sigma^{-1}$ where $\mathcal{L} \subseteq \Sigma^*$ and $\sigma \in \Sigma^*$ are defined as usual. A *graph* G labelled by an alphabet Σ, is a tuple $G = (Q, \Sigma, \delta)$ where Q is a non empty set of *states* and $\delta : Q \times \Sigma \to Q$ is the *transition function*. An *initialized graph* is a couple (G, q_0) where G is a graph and $q_0 \in Q$ is the *initial state*. A *path* in a graph G is a tuple (q, σ, q') also denoted by $q \xrightarrow{\sigma} q'$ such that $q' = \delta(q, \sigma)$. If $q = q'$ and $\sigma \neq \epsilon$, such a path is also called a *loop*. Given a language $\mathcal{L} \subseteq \Sigma^*$, we denote by $q \xrightarrow{\mathcal{L}} q'$ if there exists a path $q \xrightarrow{\sigma} q'$ with $\sigma \in \mathcal{L}$. We also denote by $q \to q'$ if $q \xrightarrow{\Sigma^*} q'$ and in this case q' is said *reachable* from q. An initialized graph (G, q_0) is said *reachable* if any state is reachable from the initial state q_0. An automaton $\mathcal{A}$ over the alphabet Σ is a tuple $\mathcal{A} = (G, q_0, F)$ such that (G, q_0) is an initialized graph and $F \subseteq Q$ is the *final set of states*. The language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ recognized by an automaton $\mathcal{A}$ is defined as usual and such a language is said *regular*.

Unusual notations: In the sequel m is an integer $m \in \mathbb{N} \backslash \{0\}$ called the *dimension*. We define $x[i]$ for any $i \in \mathbb{Z}$ and for any $x \in X^m$ as the unique element in X such that there exists $j \in \{1, \dots, m\} \cap (i + m.\mathbb{Z})$ such that $x[i] = x[j]$. Symmetrically, we define $\sigma[i] \in \Sigma$ for any $(\sigma, i) \in \mathbb{Z} \times \Sigma^+$. We denote by $\theta_m : \{1, \dots, m\} \to \{1, \dots, m\}$ the *circular permutation function* defined by $\theta_m(i) = i - 1$ if $i \geq 2$ and $\theta_m(1) = m$.

3 Serialized digit automata

In this section, we present a method for decomposing an integer vector into a word of digits based on the usual 2-complement decomposition and the serialization method introduced in [Ler04].

Recall that the 2-complement decomposition of an integer $x \in \mathbb{Z}$ in a *basis of decomposition* $r \in \mathbb{N}\backslash\{0,1\}$ provides a couple $(\sigma, s) \in \Sigma_r^* \times S_r$ where $\Sigma_r = \{0, \dots, r-1\}$ is the set of r-*digits*, and $S_r = \{0, r-1\}$ is the set of r-signs, satisfying $x = r^{|\sigma|}.\frac{s}{1-r} + \sum_{i=1}^{|\sigma|} r^{i-1}.\sigma[i]$. Naturally, given a *dimension* $m \in \mathbb{N}\backslash\{0\}$, a vector $x \in \mathbb{Z}^m$ can be decomposed into a tuple (σ, s) where σ is a word over the alphabet Σ_r^m. In order to overcome the exponential size of Σ_r^m, we slightly extend the serialization method given in [Ler04] by introducing the unique sequence $(\gamma_{r,m,\sigma})_{\sigma\in\Sigma_r^*}$ of functions $\gamma_{r,m,\sigma} : \mathbb{Z}^m \to \mathbb{Z}^m$ such that $\gamma_{r,m,\sigma_1.\sigma_2} = \gamma_{r,m,\sigma_1} \circ \gamma_{r,m,\sigma_2}$ for any $(\sigma_1, \sigma_2) \in \Sigma_r^*$ and such that for any $b \in \Sigma_r$, we have:

$$\begin{array}{cccc} \gamma_{r,m,b} : & \mathbb{Z}^m & \to & \mathbb{Z}^m \\ & (x[1], \dots x[m]) & \to & (r.x[m] + b, x[1], \dots, x[m-1]) \end{array}$$

Remark that $\gamma_{r,m,\sigma}(x) = r.x + \sigma$ for any $(\sigma, x) \in \Sigma_r^m \times \mathbb{Z}^m$. This equality explains the word "serialized" used in this article. In fact, intuitively a vector of digits $(b_1, \dots, b_m)$ is serialized into a word $\sigma = b_1 \dots b_m$.

A (r,m)-*decomposition* (σ, s) of an integer vector $x \in \mathbb{Z}^m$ is a couple $(\sigma, s) \in \Sigma_r^* \times S_r^m$ such that $x = \rho_{r,m}(\sigma, s)$ where $\rho_{r,m} : \Sigma_r^* \times S_r^m \to \mathbb{Z}^m$ is the function defined by $\rho_{r,m}(\sigma, s) = \gamma_{r,m,\sigma}(\frac{s}{1-r})$. A vector of $s \in S_r^m$ is called a (r,m)-*sign vector*. Any vector $x \in \mathbb{Z}^m$ owns at least one (r,m)-decomposition, and more precisely an infinite number thanks to lemma 3.1. In fact, from $\gamma_{r,m,\sigma}(x) = r.x + \sigma$ for any $(\sigma, x) \in \Sigma_r^m \times \mathbb{Z}^m$, we deduce that there exists at least one (r,m)-decomposition (σ, s) of x where $\sigma \in (\Sigma_r^m)^*$. Remarks also that our *serialization method* does not assume that the length $|\sigma|$ of a (r,m)-decomposition (σ, s) is a multiple of m as in [WB00]. This property is crucial for obtaining algorithms that perform some computations digit by digit rather than digit vector by digit vector (see section 6).

Lemma 3.1. *Two (r,m)-decompositions (σ_1, s_1) and (σ_2, s_2) are associated to the same integer vector if and only if there exist $k_1, k_2 \in \mathbb{N}$ such that:*

$$\begin{aligned} &(\sigma_1.s_1[1] \dots s_1[k_1], s_1[k_1+1] \dots s_1[k_1+m]) \\ =&(\sigma_2.s_2[1] \dots s_2[k_2], s_2[k_2+1] \dots s_2[k_2+m]) \end{aligned}$$

Proof. Let us first show that $\gamma_{r,m,s[1]}(\rho_{r,m}(\epsilon, s[2] \dots s[m+1]))$ is equal to $\rho_{r,m}(\epsilon, s)$ for any (r,m)-sign vector $s \in S_r^m$. First remark that $\gamma_{r,m,s[1]}(\rho_{r,m}(\epsilon, s[2] \dots s[m+1]))$ is equal to $(r.\frac{s[m+1]}{1-r} + s[1], \frac{s[2]}{1-r}, \dots, \frac{s[m]}{1-r})$. As $s \in S_r^m$, we have $s[m+1] = s[1]$. Hence $r.\frac{s[m+1]}{1-r} + s[1] = \frac{s[1]}{1-r}$ and we have proved that $\gamma_{r,m,s[1]}(\rho_{r,m}(\epsilon, s[2] \dots s[m+1])) = \rho_{r,m}(\epsilon, s)$.

Consider two (r,m)-decompositions (σ_1, s_1) and (σ_2, s_2) such that there exist $k_1, k_2 \in \mathbb{N}$ satisfying the equality given in the lemma. The previous paragraph and an immediate induction show that (σ_1, s_1) and (σ_2, s_2) are associated to the same vector. For the converse, consider two (r,m)-decompositions (σ_1, s_1) and (σ_2, s_2) associated to the same vector. Consider k_1 and k_2 such that $|\sigma_1| + k_1 = |\sigma_2| + k_2$, and such that m divides this integer. From the first paragraph, we deduce that $(\sigma_1.s_1[1] \dots s_1[k_1], s_1[k_1+1] \dots s_1[k_1+m])$ and $(\sigma_2.s_2[1] \dots s_2[k_2], s_2[k_2+1] \dots s_2[k_2+m])$ are associated to the same vector. By uniqueness of the 2-complement decomposition with a fixed length, we deduce the lemma. $\square$

A (r,m)-language $\mathcal{L}$ is a subset of $\Sigma_r^* \times S_r^m$. The set $\rho_{r,m}(\mathcal{L})$ is called the set *associated* to the (r,m)-language $\mathcal{L}$.

Definition 3.2. *A* serialized (r,m)-digit automaton (SDA) $\mathcal{A}$ *is a tuple* $\mathcal{A} = (G, q_0, m, F)$ *such that* (G, q_0) *is an initialized graph labelled by* Σ_r *with a set* Q *of states, and* $F : Q \to \mathcal{P}(S_r^m)$ *is a final function.*

The (r,m)-language $\mathcal{L}_{r,m}(\mathcal{A})$ *recognized* by a (r,m)-SDA $\mathcal{A}$ is defined as the set of (r,m)-decompositions $(\sigma, s) \in \Sigma_r^* \times S_r^m$ such that $s \in F(\delta(q_0, \sigma))$. Naturally, a (r,m)-language $\mathcal{L}$ can be recognized by a (r,m)-SDA if and only if $\{\sigma \in \Sigma_r^*;\ (\sigma, s) \in \mathcal{L}\}$ is regular for any (r,m)-sign vector $s \in S_r^m$. Such a (r,m)-language is said *regular*.

The set $\rho_{r,m}(\mathcal{L}_{r,m}(\mathcal{A}))$ of $\mathbb{Z}^m$ is also called the set *associated* to a (r,m)-SDA $\mathcal{A}$. Thanks to [BHMV94], these sets can be easily characterized as the sets definable in FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$ where $V_r : \mathbb{Z} \to \mathbb{Z}$ is the *r-valuation function* defined by $V_r(0) = 0$ and $V_r(x)$ is the greatest power of r that divides $x \in \mathbb{Z} \backslash \{0\}$.

In the following, in order to provide fine complexity results, the final function F is *serialized*: a *serialization* $\mathcal{A}_F$ of a final function $F : Q \to \mathcal{P}(S_r^m)$ is a sequence $\mathcal{A}_F = (\mathcal{A}_F(q))_{q \in Q}$ of automata over the alphabet S_r such that $\mathcal{L}(\mathcal{A}_F(q)) = F(q)$ for any $q \in Q$.

The minimal (for the number of states) SDA in basis r recognizing $\rho_{r,1}^{-1}(\mathbb{Z})$, $\rho_{r,1}^{-1}(\mathbb{N})$, $\rho_{r,3}^{-1}(+)$ and $\rho_{r,2}^{-1}(V_r)$ are given in figures 1, 2 and 3. Remark that any state q is labelled by $\rho_{r,m} \circ \mathcal{L}_{r,m}(\mathcal{A}_q)$ where $\mathcal{A}_q$ is the SDA obtained from $\mathcal{A}$ by replacing the initial state q_0 by a state $q \in Q$; the set $F(q) \neq \emptyset$ is recognized by an automaton over the alphabet S_r and a dot-edge from q to the initial state of this automaton is also drawn.

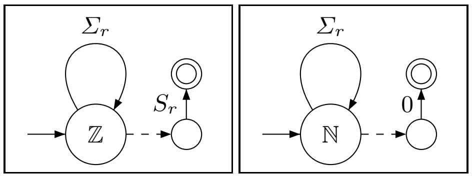

Fig. 1. On the left the minimal SDA recognizing $\rho_{r,1}^{-1}(\mathbb{Z})$. On the right the minimal SDA recognizing $\rho_{r,1}^{-1}(\mathbb{N})$

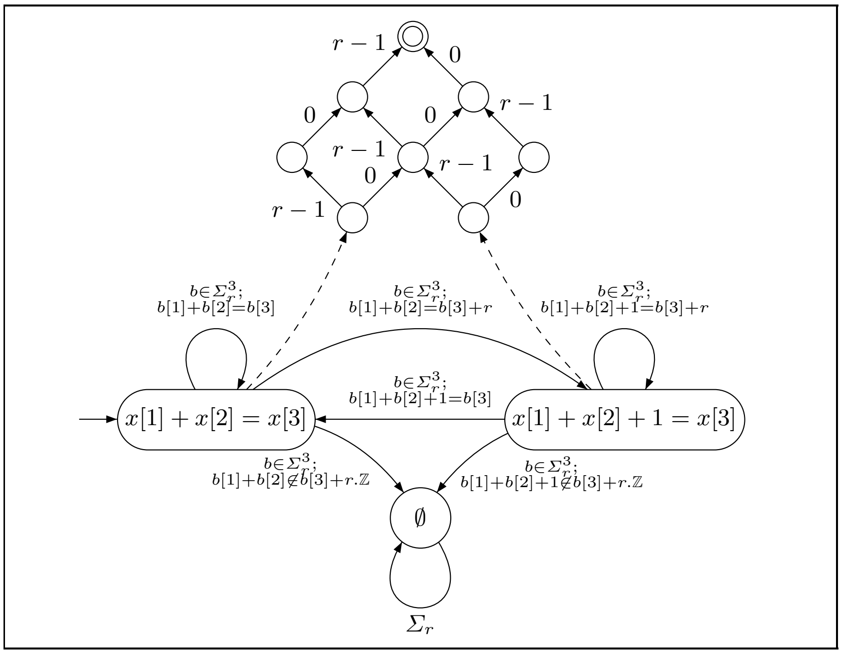

Fig. 2. The minimal SDA recognizing $\rho_{r,3}^{-1}(\{x \in \mathbb{Z}^3;\ x[1] + x[2] = x[3]\})$

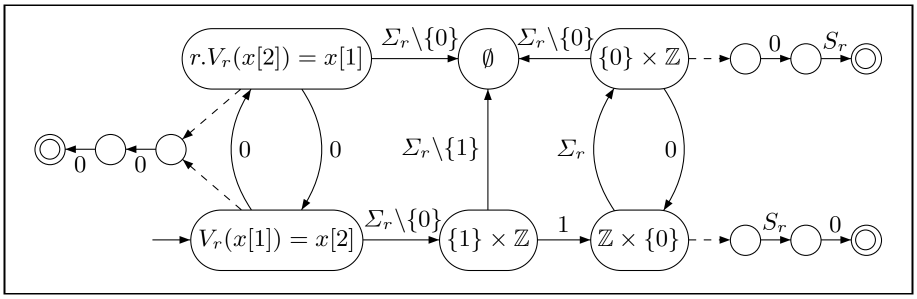

Fig. 3. The minimal SDA recognizing $\rho_{r,2}^{-1}(V_r)$

4 Saturated languages

Unfortunately, whereas FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$ is stable by boolean operations, boolean operations over the regular (r, m)-languages does not correspond exactly to the respective operations over the associated subsets of $\mathbb{Z}^m$. In fact there exist regular (r, m)-languages $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ and subsets $X, X_1, X_2 \subseteq \mathbb{Z}^m$ such that $X = \rho_{r,m}(\mathcal{L})$, $X_1 = \rho_{r,m}(\mathcal{L}_1)$, $X_2 = \rho_{r,m}(\mathcal{L}_2)$, and such that $\mathcal{L} = \mathcal{L}_1 \cap \mathcal{L}_2$ whereas $X \neq X_1 \cap X_2$. Such an example can be obtained even in the particular case $m = 1$ by considering $\mathcal{L} = \emptyset$, $\mathcal{L}_1 = \{(\epsilon, 0)\}$, $\mathcal{L}_2 = \{(0, 0)\}$, $X = \rho_{2,1}(\mathcal{L}) = \emptyset$, $X_1 = \rho_{2,1}(\mathcal{L}_1) = \{0\}$ and $X_2 = \rho_{2,1}(\mathcal{L}_2) = \{0\}$.

In this section, we introduce the class of *saturated languages* that solves the boolean stability issue, and we *characterize* the SDA that recognizes these languages.

Definition 4.1. *A (r, m)-language $\mathcal{L}$ is said saturated if for any $x \in \mathbb{Z}^m$ either (1) all the (r, m)-decompositions of x are in $\mathcal{L}$, or (2) all the (r, m)-decompositions of x are not in $\mathcal{L}$.*

Given two saturated (r, m)-languages $\mathcal{L}_1$ and $\mathcal{L}_2$, the (r, m)-language $\mathcal{L} = \mathcal{L}_1 \# \mathcal{L}_2$ is saturated and it satisfies $\rho_{r,m}(\mathcal{L}) = \rho_{r,m}(\mathcal{L}_1) \# \rho_{r,m}(\mathcal{L}_2)$ for any $\# \in \{\setminus, \cap, \cup, \Delta\}$.

From lemma 3.1, we deduce the following corollary 4.2 that characterizes the SDA that recognizes unambiguous languages (for a polynomial time characterization, see lemma 4.4).

Corollary 4.2. *A reachable (r, m)-SDA $\mathcal{A}$ recognizes a saturated language if and only if for any couple $(q, s) \in Q \times S_r^m$ either (1) we have $s[k+1] \dots s[k+m] \in F(\delta(q, s[1] \dots s[k]))$ for any $k \in \mathbb{N}$, or (2) we have $s[k+1] \dots s[k+m] \in F(\delta(q, s[1] \dots s[k]))$ for any $k \in \mathbb{N}$.*

The following proposition 4.3 proves that the class of sets associated to a saturated regular (r, m)-language and the class of subsets of $\mathbb{Z}^m$ definable in FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$ are equal. In fact, given a set X associated to a (r, m)-language $\mathcal{L}$, remark that $\mathcal{L}' = \rho_{r,m}^{-1} \circ \rho_{r,m}(\mathcal{L})$ is the unique saturated (r, m)-language such that X is associated to $\mathcal{L}'$, and thanks to the proposition 4.3, if $\mathcal{L}$ is regular then $\mathcal{L}'$ is also regular.

Proposition 4.3. *The language $\rho_{r,m}^{-1} \circ \rho_{r,m}(\mathcal{L})$ is (r, m)-regular for any regular (r, m)-language $\mathcal{L}$.*

Proof. Let us consider a regular (r, m)-language $\mathcal{L}$. For any $s \in S_r^m$, we denote by $\mathcal{L}_s$ the regular language $\mathcal{L}_s = \{\sigma \in \Sigma_r^*;\ (\sigma, s) \in S_r^m\}$. Let us consider the (r, m)-language $\mathcal{L}' = \bigcup_{s \in S_r^m} \bigcup_{k=-\infty}^{m} ((\mathcal{L}_s.(s[k+1] \dots s[m])^{-1}.(s[k+1] \dots s[k+m])^*) \times \{s[k+1] \dots s[k+m]\})$. Naturally, as the set of languages of the form $\mathcal{L}_s.(s[k+1] \dots s[m])^{-1}$ is finite (recall that $\mathcal{L}_s$ is a regular language), the previous infinite union is in fact a finite one. In particular $\mathcal{L}'$ is (r, m)-regular. Thanks to lemma 3.1, we deduce that $\mathcal{L}' = \rho_{r,m}^{-1} \circ \rho_{r,m}(\mathcal{L})$. □

Lemma 4.4. *A reachable (r, m)-SDA recognizes a saturated (r, m)-language if and only if $b^{-1}.F(q) = F(q').b^{-1}$ for any transition $q \xrightarrow{b} q'$.*

Proof. Let us first consider a reachable (r,m)-SDA $\mathcal{A}$ that recognizes a saturated (r,m)-language $\mathfrak{L}$ and let us prove that $b^{-1}.F(q) = F(q').b^{-1}$ for any transition $q \xrightarrow{b} q'$. Naturally as $F(q), F(q') \subseteq S_r^m$, if $b \notin S_r$ then $b^{-1}.F(q) = \emptyset = F(q').b^{-1}$. So, we can assume that $b \in S_r$. Let $w \in S_r^{m-1}$. Lemma 3.1 shows that $\rho_{r,m}(\sigma, b.w) = \rho_{r,m}(\sigma.b, w.b)$. As $\mathfrak{L}$ is saturated we deduce that $(\sigma, b.w) \in \mathfrak{L}$ if and only if $(\sigma.b, w.b) \in \mathfrak{L}$. Hence $b.w \in F(q)$ if and only if $w.b \in F(q')$. Therefore $b^{-1}.F(q) = F(q').b^{-1}$.

Conversely, let us consider a (r,m)-SDA $\mathcal{A}$ such that $b^{-1}.F(q) = F(q').b^{-1}$ for any transition $q \xrightarrow{b} q'$ and let us prove that the (r,m)-language $\mathfrak{L}$ recognized by $\mathcal{A}$ is saturated. Let us consider a vector $x \in \mathbb{Z}^m$ and assume that there exists a (r,m)-decomposition (σ, s) of x such that $(\sigma, s) \in \mathfrak{L}$. Hence $s \in F(\delta(q_0, \sigma))$. An immediate induction and lemma 3.1 show that all the (r,m)-decompositions of x are in $\mathfrak{L}$. Therefore $\mathfrak{L}$ is saturated. $\square$

5 Unambiguous languages

However, even if the intersection stability issue is solved for the class of (r,m)-saturated languages, this class is not stable by projecting away some *components* whereas FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$ is naturally stable by variable existential quantification. In this section, we recall the definition of projection function and we define the class of unambiguous languages that solves the boolean stability issue and partially the projection stability issue.

Let J be a subset of $m' \in \mathbb{N}\backslash\{0\}$ indices in $\{1, \dots, m\}$ that intuitively corresponds to the *components we kept* (the other one are projected away). The *projection function* $\Pi_{J,m} : \mathbb{Z}^m \to \mathbb{Z}^{m'}$ is defined by $\Pi_{J,m}(x) = (x[j_1], \dots, x[j_{m'}])$ where $j_1 < \dots < j_{m'}$ are the indices such that $J = \{j_1, \dots, j_{m'}\}$. Given a subset $X \subseteq \mathbb{Z}^m$ defined by a formula $\phi(x[1], \dots x[m])$ in FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$, remark that $\phi'(x[j_1], \dots x[j'_m]) := \exists_{k \notin J} x[k];\ \phi(x[1], \dots, x[m])$ defines the set $X' = \Pi_{J,m}(X)$.

Given a subset $X \subseteq \mathbb{Z}^m$ associated to a (r,m)-language $\mathfrak{L}$, we are interested in defining a projection function also denoted by $\Pi_{J,m}$ over $\Sigma_r^* \times S_r^m$ such that $\Pi_{J,m} \circ \rho_{r,m}(\sigma, s) = \rho_{r,m'} \circ \Pi_{J,m}(\sigma, s)$ for any (r,m)-decomposition (σ, s).

Such a function can be easily defined by first considering the unique sequence $(\pi_{I,m})_{I \subseteq \{1,\dots,m\}}$ of functions $\pi_{I,m} : \Sigma_r^* \to \Sigma_r^*$ satisfying $\pi_{I,m}(\epsilon) = \epsilon$, $\pi_{I,m}(b) = b$ if $1 \in I$ and $\pi_{I,m}(b) = \epsilon$ otherwise for any $b \in \Sigma_r$, and $\pi_{I,m}(\sigma_1.\sigma_2) = \pi_{I,m}(\sigma_1).\pi_{\theta_m^{|\sigma_1|}(I),m}(\sigma_2)$ for any $\sigma_1, \sigma_2 \in \Sigma_r^*$. The function $\Pi_{J,m}$ is naturally defined by $\Pi_{J,m}(\sigma, s) = (\pi_{J,m}(\sigma), \pi_{\theta_m^{|\sigma|}(J),m}(s))$ for any (r,m)-decomposition (σ, s).

The two projection functions $\Pi_{J,m}$ are quite similar since $\Pi_{J,m} \circ \rho_{r,m}(\sigma, s) = \rho_{r,m'} \circ \Pi_{J,m}(\sigma, s)$ for any (r,m)-decomposition (σ, s). Moreover, for any regular (r,m)-language $\mathfrak{L}$, we can easily prove that $\Pi_{J,m}(\mathfrak{L})$ is (r,m')-regular. In fact, a simple power-set construction provides from a (r,m)-SDA that recognizes $\mathfrak{L}$, a (r,m')-SDA that recognizes $\Pi_{J,m}(\mathfrak{L})$.

Unfortunately, there exist saturated regular (r,m)-languages $\mathfrak{L}$ such that $\Pi_{J,m}(\mathfrak{L})$ is not saturated. For instance, consider the saturated regular $(2,2)$-language $\mathfrak{L} = (0,1).(0,0)^* \times \{(0,0)\}$ and remark that $\mathfrak{L}' = \Pi_{\{2\},2}(\mathfrak{L})$ satisfies

$\mathcal{L}' = 0.0^* \times \{0\}$ which is not saturated because the $(2,1)$-decompositions $(\epsilon, 0)$ and $(0,0)$ of 0 are such that $(\epsilon, 0) \not\in \mathcal{L}'$ and $(0,0) \in \mathcal{L}'$. Naturally, given a set $X \subseteq \mathbb{Z}^m$ associated to a saturated regular (r,m)-language $\mathcal{L}$, the set $X' = \Pi_{J,m}(X)$ is associated to the saturated regular (r,m')-language $\mathcal{L}' = \rho_{r,m'}^{-1} \circ \rho_{r,m'} \circ \Pi_{J,m}(\mathcal{L})$. Hence, given a (r,m)-language $\mathcal{L}$, we do need to perform a kind of saturation operation $\rho_{r,m'}^{-1} \circ \rho_{r,m'}$ over a regular language after each projection. In particular, we cannot compute on the fly complex operations involving both projections and boolean operations (for instance universal quantification requires an other kind of saturation algorithm [Cou04]).

In order to solve this stability issue, we introduce a new class of (r,m)-languages, called *unambiguous.*

Definition 5.1. *A (r,m)-language $\mathcal{L}$ is said unambiguous if for any $x \in \mathbb{Z}^m$ either (1) only a finite set of (r,m)-decompositions of x are not in $\mathcal{L}$, or (2) only a finite set of (r,m)-decompositions of x are in $\mathcal{L}$.*

The *saturation* $R_{r,m}(\mathcal{L})$ of a (r,m)-unambiguous language $\mathcal{L}$ is the saturated (r,m)-language of (r,m)-decompositions (σ, s) such that there exist only a finite number of (r,m)-decompositions of $\rho_{r,m}(\sigma, s)$ not in $\mathcal{L}$. Naturally a saturated (r,m)-language is in particular unambiguous and $R_{r,m}(\mathcal{L}) = \mathcal{L}$. Moreover, given two unambiguous (r,m)-languages $\mathcal{L}_1$ and $\mathcal{L}_2$ and $\# \in \{\backslash, \cap, \cup, \Delta\}$, the (r,m)-language $\mathcal{L} = \mathcal{L}_1 \# \mathcal{L}_2$ is unambiguous and it satisfies $\rho_{r,m} \circ R_{r,m}(\mathcal{L}) = \rho_{r,m} \circ R_{r,m}(\mathcal{L}_1) \# \rho_{r,m} \circ R_{r,m}(\mathcal{L}_2)$. The set $X = \rho_{r,m} \circ R_{r,m}(\mathcal{L})$ is called the set *represented* by $\mathcal{L}$.

From lemma 3.1, we deduce the following corollary 5.2 that characterizes the SDA that recognizes an unambiguous language (by slightly modifying the proof of theorem 6.1 remarks that the problem of deciding if the language recognized by an SDA whose the final function is serialized is NP-complete).

Corollary 5.2. *A reachable (r,m)-SDA $\mathcal{A}$ recognizes an unambiguous language if and only if for any couple $(q,s) \in Q \times S_r^m$ either (1) only a finite set of $k \in \mathbb{N}$ are such that $s[k+1] \dots s[k+m] \not\in F(\delta(q, s[1] \dots s[k]))$, or (2) only a finite set of $k \in \mathbb{N}$ are such that $s[k+1] \dots s[k+m] \in F(\delta(q, s[1] \dots s[k]))$.*

The following theorem 5.3 shows in particular that the saturation of an unambiguous regular (r,m)-language is regular. That means the class of sets represented by unambiguous languages and the class of sets definable in FO $(\mathbb{Z}, \mathbb{N}, +, V_r)$ are equal.

Theorem 5.3. *Let $\mathcal{A}$ be a reachable (r,m)-SDA that recognizes an unambiguous (r,m)-language $\mathcal{L}$ and let us consider the final function $[F] : Q \rightarrow \mathcal{P}(S_r^m)$ such that $[F](q)$ is the set of (r,m)-sign vectors $s \in S_r^m$ such that only a finite set of $k \in \mathbb{N}$ are such that $s[k+1] \dots s[k+m] \not\in F(\delta(q, s[1] \dots s[k]))$. The final function $[F]$ is the unique final function F' such that the (r,m)-SDA obtained from $\mathcal{A}$ by replacing its final function F by F' recognizes the saturation $R_{r,m}(\mathcal{L})$.*

Proof. Let us first consider a set F' such that the (r,m)-SDA $\mathcal{A}'$ obtained from $\mathcal{A}$ by replacing F by F' recognizes $R_{r,m}(\mathcal{L})$ and let us prove that $F' = [F]$. Let us consider a state $q \in Q$. As $\mathcal{A}$ is reachable, there exists a path $q_0 \xrightarrow{\sigma} q$. Remark that $s \in F'(q)$ if and only if $(\sigma, s) \in R_{r,m}(\mathcal{L})$. From lemma 3.1 we deduce that $s \in F'(q)$ if and only if $s \in [F](q)$. From this same lemma, we also deduce that the (r,m)-SDA obtained from $\mathcal{A}$ by replacing its final function by $[F]$ recognizes $R_{r,m}(\mathcal{L})$. □

The following theorem shows that unambiguous languages partially solve the projection stability issue. We just say partially, because even if the projection of a saturated language is unambiguous, the class of unambiguous languages is unfortunately not stable by projection. That means we cannot compute on the fly complex operations involving alternation of quantifications (existential and universal). For instance, consider the unambiguous regular $(2,1)$-language $\mathcal{L} = ((0,1).(0,0))^* \times \{(0,0)\}$ and remark that $\mathcal{L}' = \Pi_{\{2\},2}(\mathcal{L})$ is the regular $(1,1)$-language $\mathcal{L}' = (0.0)^* \times \{0\}$ which is not unambiguous. The problem of discovering a class of regular (r,m)-language stable by boolean operations, projections, and such that a kind of operator $R_{r,m}$ can be defined is an open problem.

Theorem 5.4. *Let X be the set represented by a saturated (r,m)-language $\mathcal{L}$. The (r,m')-language $\mathcal{L}' = \Pi_{J,m}(\mathcal{L})$ is unambiguous and the set represented by $\mathcal{L}'$ is equal to $\Pi_{J,m}(X)$.*

Proof. Let $\mathcal{L}$ be a saturated (r,m)-language and let $\mathcal{L}' = \Pi_{J,m}(\mathcal{L})$. We denote by $X = \rho_{r,m} \circ R_{r,m}(\mathcal{L})$ the set represented by $\mathcal{L}$. As $\mathcal{L}$ is saturated then $R_{r,m}(\mathcal{L}) = \mathcal{L}$ and $X = \rho_{r,m}(\mathcal{L})$. Let us consider $x' \in \mathbb{Z}^{m'}$ and assume that there exists a (r,m')-decomposition (σ', s') of x' such that $(\sigma', s') \in \mathcal{L}'$. By definition, there exists a (r,m)-decomposition $(\sigma, s) \in \mathcal{L}$ such that $(\sigma', s') = \Pi_{J,m}(\sigma, s)$. As $\mathcal{L}$ is saturated, we deduce that $(\sigma.s[1] \dots s[k], s[k+1] \dots s[k+m]) \in \mathcal{L}$ for any $k \in \mathbb{N}$. Therefore $(\sigma'.s'[1] \dots s'[k'], s'[k+1] \dots s'[k'+m']) \in \mathcal{L}'$ for any $k' \in \mathbb{N}$. We have proved that the set of (r,m')-decomposition of x' that are not in $\mathcal{L}'$ is finite. That means $\mathcal{L}'$ is unambiguous. Remark that we have also proved that $X' = \rho_{r,m'} \circ R_{r,m'}(\mathcal{L}')$ is equal to $\rho_{r,m'} \circ \Pi_{J,m}(\mathcal{L}) = \Pi_{J,m}(X)$. □

Remark 5.5. Given a set $X' \subseteq \mathbb{Z}^{m'}$ defined by a formula $\phi'(x[1], \dots x[m'])$ in $\mathrm{FO}\,(\mathbb{Z}, \mathbb{N}, +, V_r)$, remark that the formula $\phi(x[1], \dots, x[m]) := \phi'(x[j_1], \dots, x[j_m])$ defines a set $X \subseteq \mathbb{Z}^m$ equal to $X = \Pi^{-1}_{J,m}(X')$. Naturally the (r,m)-language $\mathcal{L} = \Pi^{-1}_{J,m}(\mathcal{L}')$ is regular for any regular (r,m')-language $\mathcal{L}'$, and a (r,m)-SDA that recognizes $\mathcal{L}$ is computable in polynomial time from a (r,m')-SDA that recognizes $\mathcal{L}'$. Moreover, if $\mathcal{L}$ is unambiguous or saturated then $\Pi^{-1}_{J,m}(\mathcal{L})$ remains respectively unambiguous or saturated, and the set X represented by $\mathcal{L}$ is equal to $X = \Pi^{-1}_{J,m}(X')$ where X' is the set represented by $\mathcal{L}'$.

6 Symbolic saturation computation

In this section, we provide an algorithm that effectively computes the final function $[F]$ introduced by theorem 5.3. Naturally, by enumerating the (r,m)-sign vectors $s \in S^m_r$, the computation can be done in time $O(2^m.|Q|)$ since $|S^m_r| = 2^m$

and since the set of states $q \in Q$ such that $s \in [F](q)$ is computable in linear time. Remark that such an enumerative algorithm can be easily improved for the class of (r,m)-SDA that represent subsets of $\mathbb{N}^m$ since in this case $[F](q) \subseteq \{(0,\dots,0)\}$ and hence $[F]$ is computable in linear time. In the general case, we are interested in a *symbolic algorithm* that computes a serialization of $[F]$ from a serialization of F without enumerating all the possible (r,m)-sign vectors $s \in S_r^m$.

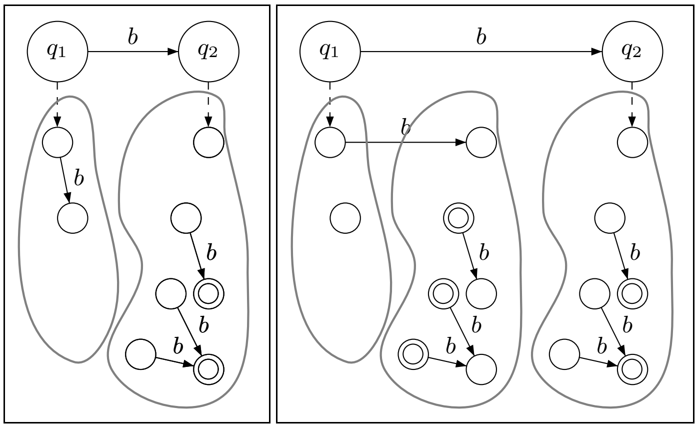

Fig. 4. On the left $\mathcal{A}_F(q_1)$ and $\mathcal{A}_F(q_2)$ and on the right $\mathcal{A}_{F_t}(q_1)$ and $\mathcal{A}_{F_t}(q_2)$ where $t = (q_1 \xrightarrow{b} q_2)$.

The following theorem shows that such a computation cannot be done in polynomial time (if P≠NP).

Theorem 6.1. *Let $\mathcal{A}$ be a (r,m)-SDA whose the final function is serialized, and assume that $\mathcal{A}$ recognizes an unambiguous (r,m)-language. The problem of deciding the emptiness of the set represented by $\mathcal{A}$ is NP-complete.*

Proof. Let us consider such a (r,m)-SDA and let $\mathcal{A}_F$ be a serialization of F. We denote by X the set represented by $\mathcal{A}$. Remark that X is non-empty if and only if there exists $s \in S_r^m$ and a reachable state $q \in Q$ satisfying $s \in [F](q)$. As the set of states $q \in Q$ such that $s \in [F](q)$ is computable in linear time, we deduce that we can decide in NP if $X \neq \emptyset$. The NP-hardness comes from a direct reduction of a 3SAT problem to our decision problem. Recall that the 3SAT problem consists in deciding if a formula ϕ of the form $\phi(s) := \bigwedge_{j=0}^{k-1} \phi_j(s)$ where ϕ_j is a disjunction of exactly 3 terms of the form $s[i] = 0$ or $s[i] = r-1$ has a solution $s \in S_r^m$. Consider the graph $G_0 = (Q_0, \Sigma_r^m, \delta_0)$ given by the set of states $Q_0 = \{q_j;\ j \in \{0,\dots,k-1\}\} \cup \{q_\perp\}$, the transition function δ_0 defined by $\delta_0(q_\perp, \sigma) = q_\perp$ for any $\sigma \in \Sigma_r^m$, $\delta_0(q_{\theta_m(j)}, \sigma) = q_j$ if $\sigma \in S_r^m$ and satisfies ϕ_j, and defined by $\delta_0(q_{\theta_m(j)}, \sigma) = q_\perp$ otherwise. Naturally, we can compute in linear time from ϕ a graph $G = (Q, \Sigma_r, \delta)$ such that $Q_0 \subseteq Q$ and $\delta(q,\sigma) = \delta_0(q,\sigma)$ for any $\sigma \in \Sigma_r^m$. Let us consider the (r,m)-SDA $\mathcal{A} = (Q, \Sigma_r, \delta, q_1, m, F)$ such that $F(q) = S_r^m$ if $q \in \delta(q_1, \Sigma_r^*) \backslash \delta(q_\perp, \Sigma_r^*)$ and $F(q) = \emptyset$ otherwise. Naturally a serialization $\mathcal{A}_F$ of F can be computed in linear time from ϕ. Now, just remark that $\mathcal{A}$ recognizes an unambiguous (r, m)-language $\mathcal{L}$ such that ϕ is satisfiable if and only if $R_{r,m}(\mathcal{L})$ is non empty. □

We are going to prove that the algorithm given in figure 5 is *correct and terminating*, its step by step complexity is *linear in time*, and there exists a *strategy* for selecting the transitions t such that it terminates in $2.|Q|^2$ steps.

In this algorithm, a step consists in choosing a transition $t = (q_1 \xrightarrow{b} q_2)$ and replacing a final function F by the final function F_t defined by $F_t(q) = F(q)$ for any $q \in Q \backslash \{q_1\}$ and defined by the following equality:

$$F_t(q_1) = (F(q_1) \backslash (b.S_r^{m-1})) \cup (b.F(q_2).b^{-1})$$

Figure 4 explains why an automaton $\mathcal{A}_{F_t}(q_1)$ labelled by S_r recognizing $F_t(q_1)$ is computable in *linear time* from two automata $\mathcal{A}_F(q_1)$ and $\mathcal{A}_F(q_2)$ labelled in S_r recognizing respectively $F(q_1)$ and $F(q_2)$. Remark that in practice such a construction provides an automaton $\mathcal{A}_{F_t}(q_1)$ that should be minimized in order to avoid an exponential blow up of the serialization during the execution of the algorithm.

input: A reachable (r, m)-SDA $\mathcal{A} = (G, q_0, m, F)$ that recognizes an unambiguous (r, m)-language.
output: The final function $[F]$.
begin
    let $F' := F$.
    while there exists a transition t such that $F'_t \neq F'$
        let $F' := F'_t$.
    end while
    return F'.
end

Fig. 5. An algorithm computing the final function $[F]$.

In the remaining of this paper, a reachable initialized graph (G, q_0) labelled by Σ_r and a dimension $m \in \mathbb{N} \backslash \{0\}$ are fixed. Let us introduce some definitions. Given $(q, s) \in Q \times S_r^m$ and an integer $k \in \mathbb{N}$, we denote by $q_{s,k}$ the state $q_{s,k} = \delta(q, s[1] \dots s[k])$, and we denote by s_k the (r, m)-sign vector $s_k = s[k+1] \dots s[k+m]$. A final function F is a function $F : Q \to \mathcal{P}(S_r^m)$. Given $(q, s) \in Q \times S_r^m$ and a final function F, let us consider the sequence $a(F) = (a^{q,s}(F))_{(q,s) \in Q \times S_r^m}$ where $a^{q,s}(F)$ is the sequence $a^{q,s}(F) = (a_k^{q,s}(F))_{k \in \mathbb{N}}$ of elements in $\{\top, \perp\}$ defined by $a_k^{q,s}(F) = \top$ if and only if $s_k \in F(q_{s,k})$. Given a sequence $a = (a_k)_{k \in \mathbb{N}}$ of elements in $\{\top, \perp\}$, let us denote by $K(a)$ the set of $k \in \mathbb{N}$ such that $a_k \neq a_{k+1}$. A final function F is said respectively *unambiguous* or *saturated* if the (r, m)-SDA $\mathcal{A} = (G, q_0, m, F)$ recognizes respectively an unambiguous or a saturated (r, m)-language. From corollaries 4.2 and 5.2, we deduce that F is respectively unambiguous or saturated if and only $K(a^{q,s}(F))$ is respectively finite or empty for any $(q,s)\in Q\times S^m$. Let us consider the partial order $\geq$ over the sequences in $\{\top,\bot\}^{\mathbb{N}}$ defined by $a^1\geq a^2$ if and only (1) there exists $\#\in\{\top,\bot\}$ and $k_0\in\mathbb{N}$ such that $a^1_k=\#=a^2_k$ for any $k\geq k_0$, and (2) the sets $K(a^1)=\{k^1_1,\dots,k^1_{d_1}\}$ and $K(a^2)=\{k^2_1,\dots,k^2_{d_2}\}$ are such that $d_1\geq d_2$ and $k^1_i\geq k^2_i$ for any $1\leq i\leq d_2$. Remark that the partially ordered set $(\{\top,\bot\}^{\mathbb{N}},\geq)$ is *finite height*: for any sequence $(a^n)_{n\in\mathbb{N}}$ such that $a^n\geq a^{n+1}$, there exists $n_0\in\mathbb{N}$ such that $a^n=a^{n_0}$ for any $n\geq n_0$. We also denote by $\geq$ the component-wise partial order over the sequences $(a^{q,s})_{(q,s)\in Q\times S_r^m}$ of sequences in $\{\top,\bot\}^{\mathbb{N}}$ that naturally remains finite height.

The following lemma 6.2 proves that the algorithm is correct and terminating. In fact as $\geq$ is finite height, the while loop can be executed only a finite number of times. Moreover, an immediate induction shows that the final function F' returned by the algorithm is such that $a(F)\geq a(F')$ and in particular there exists $k_0\in\mathbb{N}$ and $\#\in\{\top,\bot\}$ such that $a^{q,s}_k(F)=\#=a^{q,s}_k(F_t)$ for any $k\geq k_0$. In particular F' is unambiguous and $[F]=[F']$. As the while loop condition is not true for F' we deduce that $F'_t=F'$ for any transition t. From lemma 4.4 we deduce that F' is saturated. Therefore $[F']=F'$ and we have proved that the algorithm returns the final function $F'=[F]$.

Lemma 6.2. *We have $a(F)>a(F_t)$ for any unambiguous final function F and for any transition t such that $F_t\neq F$.*

Proof. Let us first show that $a^{q,s}(F)\geq a^{q,s}(F_t)$ for any $(q,s)\in Q\times S_r^m$. Let us consider the set K of integers $k\in K(a^{q,s}(F))$ such that $q_{s,k}\xrightarrow{s[k+1]}q_{s,k+1}$ is equal to the transition t. By definition of F, there exists $\#,\#'\in\{\top,\bot\}$ such that $a^{q,s}_k(F)=\#$ and $a^{q,s}_{k+1}(F)=\#'$ for any $k\in K$. By definition of F_t, we have $a^{q,s}_k(F_t)=a^{q,s}_k(F)$ for any $k\in\mathbb{N}\backslash K$ and $a^{q,s}_k(F_t)=\#'$ for any $k\in K$. We deduce that $a^{q,s}(F)\geq a^{q,s}(F_t)$.

Next let us prove that there exists $(q,s)\in Q\times S_r^m$ such that $a^{q,s}(F)\neq a^{q,s}(F_t)$. The transition t is denoted by $q\xrightarrow{b}q'$. As $F_t\neq F$, there exists $w\in S_r^{m-1}$ such that either (1) $b.w\in F(q)$ and $w.b\notin F(q')$ or (2) $b.w\notin F(q)$ and $w.b\in F(q')$. Let us denote by $s=b.w$ and remark that $s_0=b.w$, $s_2=w.b$, $q_{s,0}=q$ and $q_{s,1}=q'$. We have proved that $a^{q,s}_0(F)\neq a^{q,s}_1(F)$. Moreover, as $a^{q,s}(F_t)_0=a^{q,s}(F_t)_1=a^{q,s}(F)_1$, we deduce that $a^{q,s}(F)\neq a^{q,s}(F_t)$. □

The following proposition 6.3 provides a *strategy* for choosing the transitions t such that the algorithm only consider $2.|Q|^2$ transitions in the worst case. In fact, assume that the algorithm first select an order t_0, ..., $t_{2.|Q|-1}$ over the set $\{q\xrightarrow{b}q';\ q,q'\in Q;\ b\in S_r\}$, and apply these transitions sequentially (if a transition t is such that $F'_t=F'$ then it just skip it) and repeat this choice until the termination. We denote by F_n the final function obtained after the n-th iteration. From the proposition 6.3, we deduce that $n\leq|Q|$ since $\max(K(a^{q,s}(F)))\leq|Q|-1$ for any unambiguous final function F. Therefore, the algorithm terminates in $n\leq|Q|$ iterations, and it only considers $2.|Q|^2$ transitions.

Proposition 6.3. *For any $(q,s) \in Q \times S_r^m$, there exists a transition t such that* $\max(K(a^{q,s}(F_t))) \leq \max(K(a^{q,s}(F))) - 1$.

Proof. Naturally, if $K(a^{q,s}(F)) = \emptyset$ then from lemma 6.2 we deduce that $K(a^{q,s}(F_t))$ is also empty and in particular we get the required inequality. So, we can assume that $K(a^{q,s}(F))$ is not empty. Let $k_0 = \max(K(a^{q,s}(F)))$ and let us consider the transition $t = (q_{s,k_0} \xrightarrow{s[k_0+1]} q_{s,k_0+1})$. By definition F_t, we have $a_k^{q,s}(F_t) = a_{k_0+1}^{q,s}(F)$ for any $k \geq k_0$. Therefore $\max(K(a^{q,s}(F_t))) \leq k_0 - 1$ and we are done. □

Remark 6.4. If m is fixed, the algorithm with our strategy performs the computation of $[F]$ in time $O(|Q|^2)$ whereas an enumerative algorithm performs the same computation in time $O(|Q|)$. In particular it seems that our strategy is not optimal. Naturally we can easily optimize this strategy by decomposing the graph G into its *strongly connected components*. Unfortunately, the worst case time complexity remains $O(|Q|^2)$. A better choice (maybe probabilistic) is an open problem.

Remark 6.5. A *number decision diagram* (NDD) [WB00] $\mathcal{A}$ that symbolically represents a set $X \subseteq \mathbb{Z}^m$ in basis r is an automaton $\mathcal{A}$ labelled by Σ_r that recognizes the language $\{\sigma.s;\ (\sigma,s) \in ((\Sigma_r^m)^* \times S_r^m) \cap \rho_{r,m}^{-1}(X)\}$. Remark that (1) a NDD recognizes only words of length multiple of m, and (2) the digit b of a transition $q \xrightarrow{b} q'$ can either correspond to a digit of σ or a digit of s. These two issues (1) and (2) prevent NDD having a *simple symbolic algorithm* for saturation computation. Up to our knowledge there does not even exist a paper presenting an *enumerative algorithm*.

References

[BB03] Constantinos Bartzis and Tevfik Bultan. Efficient symbolic representations for arithmetic constraints in verification. *International Journal of Foundations of Computer Science (IJFCS)*, 14(4):605–624, August 2003.

[BFLP03] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: Fast Acceleration of Symbolic Transition systems. In *Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003), Boulder, CO, USA, July 2003*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121. Springer, 2003.

[BHMV94] Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.*, 1(2):191–238, March 1994.

[Cob69] A. Cobham. On the base-dependance of sets of numbers recognizable by finite automata. *Mathematical Systems Theory*, 3:186–192, 1969.

[Cou04] Jean-Michel Couvreur. A bdd-like implementation of an automata package. In *Proc. 9th Int. Conf. on Implementation and Application of Automata (CIAA'2004), Kingston, Canada, July 22-24, 2004*, volume 3317 of *Lecture Notes in Computer Science*, pages 310–311. Springer, 2004.

[GBD02] Vijay Ganesh, Sergey Berezin, and David L. Dill. Deciding presburger arithmetic by model checking and comparisons with other methods. In *Proc. 4th Int. Conf. Formal Methods in Computer Aided Design (FMCAD'02), Portland, OR, USA, nov. 2002*, volume 2517 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2002.

[GS66] Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas and languages. *Pacific J. Math.*, 16(2):285–296, 1966.

[KMS02] Nils Klarlund, A. Møller, and M. I. Schwartzbach. MONA implementation secrets. *Int. J. of Foundations Computer Science*, 13(4):571–586, 2002.

[Las] LASH homepage. `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[Ler04] Jérôme Leroux. The affine hull of a binary automaton is computable in polynomial time. In Elsevier Science, editor, *Proc. 5th Int. Workshop on Verification of Infinite State Systems (INFINITY 2003), Marseille, France, Sep. 2003*, volume 98 of *Electronic Notes in Theor. Comp. Sci.*, pages 89–104, 2004.

[Mon] MONA homepage. `http://www.brics.dk/mona/index.html`.

[Ome] OMEGA homepage. `http://www.cs.umd.edu/projects/omega/`.

[Pre29] M. Presburger. Uber die volstandigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In *C. R. 1er congres des Mathematiciens des pays slaves, Varsovie*, pages 92–101, 1929.

[RV02] Tatiana Rybina and Andrei Voronkov. Brain: Backward reachability analysis with integers. In *Proc. 9th Int. Conf. Algebraic Methodology and Software Technology (AMAST'2002), Saint-Gilles-les-Bains, Reunion Island, France, Sep. 2002*, volume 2422 of *Lecture Notes in Computer Science*, pages 489–494. Springer, 2002.

[WB00] Pierre Wolper and Bernard Boigelot. On the construction of automata from linear arithmetic constraints. In *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), Berlin, Germany, Mar.-Apr. 2000*, volume 1785 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2000.