On the Expressiveness of Real and Integer Arithmetic Automata (Extended Abstract)

Bernard Boigelot*, Stéphane Rassart and Pierre Wolper

Université de Liège
Institut Montefiore, B28
B-4000 Liège Sart-Tilman, Belgium
{boigelot,rassart,pw}@montefiore.ulg.ac.be

Abstract. If read digit by digit, a n-dimensional vector of integers represented in base r can be viewed as a word over the alphabet r^n. It has been known for some time that, under this encoding, the sets of integer vectors recognizable by finite automata are exactly those definable in Presburger arithmetic if independence with respect to the base is required, and those definable in a slight extension of Presburger arithmetic if only a specific base is considered.
Using the same encoding idea, but moving to infinite words, finite automata on infinite words can recognize sets of real vectors. This leads to the question of which sets of real vectors are recognizable by finite automata, which is the topic of this paper. We show that the recognizable sets of real vectors are those definable in the theory of reals and integers with addition and order, extended with a special base-dependent predicate that tests the value of a specified digit of a number. Furthermore, in the course of proving that sets of vectors defined in this theory are recognizable by finite automata, we show that linear equations and inequations have surprisingly compact representations by automata, which leads us to believe that automata accepting sets of real vectors can be of more than theoretical interest.

1 Introduction

The ability to represent and manipulate sets of integers and/or reals is a fundamental tool that has many applications. The specific problems motivating this paper come from the algorithmic verification of reactive systems where manipulating sets of integers or reals is important for verifying protocols [BW94], real-time systems [HNSY94, AD94] or hybrid systems [ACH$^+$95, Hen96, BBR97]. Of course, many well-established approaches exist for manipulating such sets, for instance using symbolic equations or various representations of polyhedra. However, each of these approaches has its limits and usually copes better with

* "Chargé de Recherches" (Post-Doctoral Researcher) for the National Fund for Scientific Research (Belgium).

sets of reals than with sets of integers or sets involving both reals and integers. This situation has prompted an effort to search for alternative representations, with Ordered Binary Decision Diagrams [Bry86, Bry92], a very successful representation of very large finite sets of values, as a leading inspiration.

An ordered binary decision diagram is a representation of a set of fixed-length bit-word values. It is a layered DAG in which each level corresponds to a fixed-position bit and separates between words in which this bit has value 0 and value 1. Abstractly, it is just a representation of a Boolean function, but one for which an effective algorithmic technology has been developed. A BDD can of course represent a set of fixed-length binary represented integers or reals, but in this context it has an interesting and powerful generalization. Indeed, observing that a BDD is just an acyclic finite automaton, one is naturally led to considering lifting this acyclicity restriction, i.e., to consider finite automata accepting the binary encoding of numbers. Actually, it is more than worthwhile to go one step further and consider automata operating on the encoding of vectors of numbers. For doing so, one makes the encoding of the numbers in the vector of uniform length and reads all numbers in the vector in parallel, bit by bit. An n-dimensional vector is thus seen as a word over the alphabet 2^n.

For integers, the representation of each integer is a finite word and one is thus dealing with traditional languages of finite words. The subject of finite automata accepting binary (or more generally base-r) encodings of integer vectors has been well studied, going back to work of Büchi [Büc60, Büc62]. Indeed, this use of finite automata has applications in logic. For instance, noticing that addition and order are easily represented by finite automata and that these are closed under the Boolean operations as well as projection, it is very easy to obtain a decision procedure for Presburger arithmetic. Going further, the question of which sets of integer vectors can be represented by finite automata has been remarkably answered by Cobham [Cob69] for 1-dimensional vectors and Semenov [Sem77] for n-dimensional vectors. The result is that sets that are representable independently of the chosen base (≥ 2) are exactly those that are Presburger definable. If one focuses on a given base, a predicate relating a number and the largest power of the base dividing it has to be added in order to capture the sets recognizable by finite automata (see [BHMV94] for a survey of these results).

When considering the reals, the situation is somewhat more complex. Indeed, to be able to represent all reals, one has to consider infinite representations, a natural one being the infinite base-r expansion of reals. A number or vector of numbers is thus now an infinite word and an automaton recognizing a set of reals is an automaton on infinite words [Büc62]. This idea was actually already familiar to Büchi himself and leads very simply to a decision procedure for the theory of reals and integers with addition and order predicates. We are, however, not aware of any further study of the subject since then.

This paper delves into this subject with the dual goal of understanding the theoretical limits of representing sets of reals and integers by automata on infinite words and of investigating the pragmatic algorithmic aspects of this approach. On the first of these topics, we settle the question of which sets of real vectors are representable by nondeterministic Büchi automata operating on base-r encodings, i.e., by ω-regular languages of base-r encodings. The result is that the representable sets are those definable in the theory of the integers and reals with addition and order predicates, as well as with a special predicate $X_r(x, u, k)$. This predicate is true if and only if u is a positive or negative integer power of the base r, k belongs to $\{0, \ldots, r-1\}$ (i.e., is a digit in base r), and the value of the digit of the representation of x appearing in the position corresponding to the power of the base u is k. In simpler terms, the predicate X_r lets one check which digit appears in a given position of the base-r encoding of a number x. The proof of this result, inspired by [Vil92], relies on an interesting encoding of the possible computations of a Büchi automaton by an arithmetic formula.

On the second topic, we will show that the sets representable by linear equations and inequations have remarkably simple and easy to construct representations by automata. Furthermore, in many cases, an optimal deterministic representation can be directly constructed. This improves on the results of [BBR97] and extends those of [BC96], which are limited to the positive integers.

2 Recognizing Sets of Real Vectors with Automata

In this section, we recall the encoding of real vectors by words introduced in [BBR97] and define the type of finite automata that will be used for recognizing sets of such encodings.

Let $x \in \mathbf{R}$ be a real number and $r > 1$ be an integer. We encode x in base r, most significant digit first, using r's complement for negative numbers. The result is a word of the form $w = w_I \star w_F$, where w_I encodes the integer part x_I of x as a finite word over the alphabet $\{0, \ldots, r-1\}$, the symbol "$\star$" is a separator, and w_F encodes the fractional part x_F of x as an infinite word over the alphabet $\{0, \ldots, r-1\}$. We do not fix the length p of w_I, but only require it to be nonzero and large enough for $-r^{p-1} \le x_I < r^{p-1}$ to hold. Hence, the most significant digit of a number will be "0" if this number is positive or equal to zero, and "$r-1$" otherwise. The length $|w_I|$ of w_I will be called the *integer-part length* of the encoding of x by w. For simplicity, we require that the length of w_F always be infinite (this is not a real restriction, since an infinite number of "0" symbols can always be appended harmlessly to w_F).

It is important to note that some numbers $x \in \mathbf{R}$ have two distinct encodings with the same integer-part length. For example, in base 10, the number $x = 11/2$ has the following two encodings with integer-part length 3 : $005 \star 5(0)^\omega$ and $005 \star 4(9)^\omega$ ($^\omega$ denotes infinite repetition). Such encodings are said to be *dual*. The encoding which ends with an infinite succession of "0" digits is said to be a *high* encoding of x. The encoding which ends with an infinite succession of "$r-1$" digits is said to be a *low* encoding of x. If there is only one encoding of x that has a given integer-part length, this encoding is said to be both high and low.

To encode a vector of real numbers, we encode each of its components with words of identical integer-part length. This length can be chosen arbitrarily, provided that it is sufficient for encoding the vector component with the highest magnitude. It follows that any vector has an infinite number of possible encodings. An encoding of a vector of reals $\mathbf{x} = (x_1, \ldots, x_n)$ can indifferently be viewed either as a tuple $(w_1, \ldots, w_n)$ of words of identical integer-part length over the alphabet $\{0, \ldots, r-1, \star\}$, or as a single word w over the alphabet $\{0, \ldots, r-1\}^n \cup \{\star\}$. For convenience, the real vector represented by a word w interpreted in base r is denoted $[w]_r$.

Since a real vector has several possible encodings, we have to choose which of these the automata we define will recognize. A natural choice is to accept all encodings. This leads to the following definition.

Definition 1 *Let $n > 0$ and $r > 1$ be integers. A* Real Vector Automaton (RVA) *$\mathcal{A}$ in base r for vectors in $\mathbf{R}^n$ is a Büchi automaton [Büc62] over the alphabet $\{0, \ldots, r-1\}^n \cup \{\star\}$, such that:*

- *Every word w accepted by $\mathcal{A}$ is of the form $w = w_I \star w_F$, with $w_I \in (\{0, r-1\}^n)(\{0, \ldots, r-1\}^n)^*$ and $w_F \in (\{0, \ldots, r-1\}^n)^\omega$.*
- *For every vector $\mathbf{x} \in \mathbf{R}^n$, $\mathcal{A}$ accepts either all the encodings of $\mathbf{x}$ in base r, or none of them.*

An RVA is said to *represent* the set of vectors encoded by the words belonging to its accepted language. Note that this notion of representation is not canonical since different Büchi automata may accept the same language. Any subset of $\mathbf{R}^n$ that can be represented by an RVA in base r is said to be *r-recognizable*.

3 The Expressive Power of Real Vector Automata

In this section, we introduce a logical theory in which all sets of real vectors that are recognizable by Real Vector Automata can be defined. Precisely, we prove that for any base $r > 1$, the r-recognizable subsets of $\mathbf{R}^n$ are definable in the first-order theory $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$, where Z is a predicate that tests whether a real number belongs to $\mathbf{Z}$, and X_r is a predicate that tests the value of the digit occurring at a given position in the development of a real number in base r. As will be shown in Section 6, the converse translation also holds and thus the theory $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$ exactly characterizes the r-recognizable sets of real vectors.

The predicate X_r over $\mathbf{R}^3$ is such that $X_r(x, u, k) = \mathbf{T}$ if and only if u is a (positive or negative) integer power of r, and there exists an encoding of x such that the digit at the position specified by u is k (which implies that $k \in \{0, \ldots, r-1\}$). Formally, $\mathbf{N}_0$ denoting the strictly positive integers, we have

$$\begin{aligned} X_r(x, u, k) \equiv (\exists p \in \mathbf{N}_0, a_p \in \{0, r-1\}, a_{p-1}, a_{p-2}, \ldots \in \{0, 1, \ldots, r-1\}) \\ (x = -(a_p/(r-1))r^p + a_{p-1}r^{p-1} + a_{p-2}r^{p-2} + \cdots \\ \wedge (\exists q \in \mathbf{Z})(q \leq p \wedge r^q = u \wedge a_q = k)). \end{aligned}$$

A subset of $\mathbf{R}^n$ that can be defined in the theory $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$ is said to be *r-definable*.

3.1 Some r-Definable Functions and Predicates

We show here the r-definability of predicates that will enable us cope with the existence of dual encodings. For any integer base $r > 1$ we have the following results.

Lemma 2. *The predicate P_r over $\mathbf{R}$ such that $P_r(x) = \mathbf{T}$ if and only if x is a (positive or negative) integer power of r is r-definable.*

Proof $P_r(x) \equiv (\exists k, y \in \mathbf{R})(X_r(y, x, k))$. □

Lemma 3. *The predicate D_r over $\mathbf{R}^2$, such that $D_r(x, u) = \mathbf{T}$ if and only if u is an integer power of r and x admits two encodings in base r that have the same integer-part length but have different digits at the position specified by u, is r-definable.*

Proof $D_r(x, u) \equiv (\exists k, l \in \mathbf{R})(k \neq l \wedge X_r(x, u, k) \wedge X_r(x, u, l))$. □

Lemma 4. *The predicate X_r^L over $\mathbf{R}^3$, such that $X_r^L(x, u, k) = \mathbf{T}$ if and only if u is an integer power of r and there exists a low encoding of x such that the digit at the position specified by u is k, is r-definable.*

Proof
$$\begin{aligned} X_r^L(x, u, k) \equiv\ & (X_r(x, u, k) \wedge \neg D_r(x, u)) \\ & \vee\ (X_r(x, u, k) \wedge X_r(x, u, k+1) \wedge \neg D_r(x, ru)) \\ & \vee\ (k = r - 1 \wedge X_r(x, u, 0) \wedge X_r(x, u, r-1) \wedge D_r(x, ru)). \end{aligned}$$
□

Lemma 5. *The predicate X_r^H over R^3, such that $X_r^H(x, u, k) = \mathbf{T}$ if and only if u is an integer power of r and there exists a high encoding of x such that the digit at the position specified by u is k, is r-definable.*

Proof
$$\begin{aligned} X_r^H(x, u, k) \equiv\ & (X_r(x, u, k) \wedge \neg D_r(x, u)) \\ & \vee\ (X_r(x, u, k) \wedge X_r(x, u, k-1) \wedge \neg D_r(x, ru)) \\ & \vee\ (k = 0 \wedge X_r(x, u, 0) \wedge X_r(x, u, r-1) \wedge D_r(x, ru)). \end{aligned}$$
□

Lemma 6. *The predicate X_r' over $\mathbf{R}^4$ such that $X_r'(x, u, k, a) = \mathbf{T}$ if and only if either $a = 1$ and $X_r^L(x, u, k) = \mathbf{T}$ or $a = 2$ and $X_r^H(x, u, k) = \mathbf{T}$ is r-definable.*

Proof $X_r'(x, u, k, a) \equiv (a = 1 \wedge X_r^L(x, u, k)) \vee (a = 2 \wedge X_r^H(x, u, k))$. □

3.2 From Recognizability to Definability

We are now ready to prove that every set of real vectors that is r-recognizable is definable in the theory $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$.

Theorem 7. *Let $n > 0$ and $r > 1$ be integers. Every r-recognizable subset of $\mathbf{R}^n$ is r-definable.*

Proof The idea of the proof is that a computation of a RVA can be encoded by an infinite word that can itself be seen as the encoding of a real vector. This makes it possible to express the existence of a computation of an RVA on a real vector within our target language.

Consider a r-recognizable set $U \subseteq \mathbf{R}^n$. By definition, there exists a Büchi automaton $\mathcal{A} = (\Sigma, S, \Delta, s_0, F)$ accepting all the encodings in base r of the elements of U. Let us show that U can be defined in $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$. We take $m \in \mathbf{N}$ such that $r^m > |S| + 1$, where $|S|$ denotes the number of states of $\mathcal{A}$. Each state $s \in S$ can be encoded by a tuple

$$E(s) = (e_1(s), e_2(s), \ldots, e_m(s)) \in \{0, 1, \ldots, r-1\}^m.$$

Without loss of generality, we can assume that there is no $s \in S$ such that $e_1(s) = e_2(s) = \cdots = e_m(s) = 0$ or such that $e_1(s) = e_2(s) = \cdots = e_m(s) = r-1$.

Using this encoding of states, a vector $(y_1, y_2, \ldots, y_m)$ of reals can be seen as representing, by means of its base-r encoding, a run $s_0, s_1, s_2, \ldots \in S^\omega$ of $\mathcal{A}$. However, given the multiplicity of encodings, this representation is ambiguous. There are two causes of ambiguity. The first is the fact that one can have various integer-part lengths. This is actually of no consequence since, if one restricts the y_i's to be positive or 0, going from one integer-part length to another just implies adding to the beginning of the encoding a number of tuples $\{0\}^m$ that by convention do not represent a state of $\mathcal{A}$.

The second cause of ambiguity is the existence of dual encodings. To solve this, we append to the vector $(y_1, y_2, \ldots, y_m)$ a second vector $(a_1, a_2, \ldots, a_m)$ whose elements are restricted to values in the set $\{1, 2\}$, with the convention that the value 1 for a_i expresses the fact that y_i should be represented using a low encoding and the value 2 specifies a high encoding. In summary, a vector

$$(y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m) \in \mathbf{R}_+^m \times \{1, 2\}^m$$

represents a run $s_0, s_1, s_2, \ldots \in S^\omega$ of $\mathcal{A}$ as follows.

- Let l be the shortest integer-part length that allows the base-r encoding of the real vector $(y_1, y_2, \ldots, y_m)$.
- For $i \in \{1, \ldots, m\}$ let w_i be the (low if $a_i = 1$ or high if $a_i = 2$) integer-part length l base-r encoding of y_i.
- The represented run is then the one such that, for all $j \geq 0$, the state s_j is the one whose encoding $E(s_j) = (e_1(s_j), e_2(s_j), \ldots, e_m(s_j))$ is given by the digits of the words $w_1, \ldots, w_m$ at the position corresponding to $r^{l-(j+1)}$. If at a given position the digits of the w_i do not represent a state, then the vector $(y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m)$ does not represent a run of $\mathcal{A}$.

To prove our theorem, its is sufficient to show that the predicate $R_{\mathcal{A}}(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m)$ which is satisfied if and only if the tuple $(y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m)$ encodes an execution of $\mathcal{A}$ which accepts an encoding in base r of the real vector $(x_1, x_2, \ldots, x_n)$ is expressible in $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$.

Indeed, using this predicate $R_{\mathcal{A}}$, the set U of real vectors whose encodings are accepted by $\mathcal{A}$ can be expressed as follows

$$U = \{(x_1, x_2, \ldots, x_n) \in \mathbf{R}^n \mid (\exists y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m \in \mathbf{R}) \\ (R_{\mathcal{A}}(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m))\}.$$

We now turn to expressing $R_{\mathcal{A}}$ in $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$. We make use of the r-definable functions and predicates obtained in Section 3.1. The idea is to express that $R_{\mathcal{A}}(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m, a_1, a_2, \ldots, a_m) = \mathbf{T}$ if and only if there exist $z \in \mathbf{R}$ and $b_1, b_2, \ldots, b_n \in \{1, 2\}$ satisfying the conditions expressed below (z is used to represent the highest significant position used in the y_i's and the $b_1, b_2, \ldots, b_n$ to make explicit the fact that the encoding of the x_i's is high or low).

- The y_i's are positive, and the number z is the highest power of r appearing in the encodings in base r of the y_i's (with the convention that the encoding of a y_i is supposed to be low if $a_i = 1$, and high otherwise). This is expressed by the following formula.

$$\begin{aligned} &\phi_1(x_1, \ldots, x_n, b_1, \ldots, b_n, y_1, \ldots, y_m, a_1, \ldots, a_m, z) \\ &\quad \equiv P_r(z) \wedge \neg \bigwedge_{i \in \{1, \ldots, m\}} X'_r(y_i, z, 0, a_i) \\ &\qquad \wedge (\forall u \in \mathbf{R})((u > z \wedge P_r(u)) \Rightarrow \bigwedge_{i \in \{1, \ldots, m\}} X'_r(y_i, u, 0, a_i)). \end{aligned}$$

- In the encodings of the y_i's, the digits at the position specified by z correspond to the encoding of the initial state s_0 of $\mathcal{A}$. The formula expressing this is

$$\begin{aligned} &\phi_2(x_1, \ldots, x_n, b_1, \ldots, b_n, y_1, \ldots, y_m, a_1, \ldots, a_m, z) \\ &\quad \equiv \bigwedge_{i \in \{1, \ldots, m\}} X'_r(y_i, z, e_i(s_0), a_i). \end{aligned}$$

- There exists an accepting state of $\mathcal{A}$ whose encoding as a tuple of digits appears infinitely often in the digits of the y_i's. The corresponding formula is

$$\begin{aligned} &\phi_3(x_1, \ldots, x_n, b_1, \ldots, b_n, y_1, \ldots, y_m, a_1, \ldots, a_m, z) \\ &\quad \equiv \bigvee_{s \in F} (\forall u \in \mathbf{R})((u < z \wedge P_r(u)) \Rightarrow (\exists v \in \mathbf{R})(v < u \wedge P_r(v) \\ &\qquad\qquad \wedge \bigwedge_{i \in \{1, \ldots, m\}} X'_r(y_i, v, e_i(s), a_i))). \end{aligned}$$

- At any two successive positions, the digits of the y_i's encode two states of $\mathcal{A}$ linked together by a transition. The label of the transition is given by the corresponding digits of an encoding of the x_i's, except that reading the separator $\star$ introduces a shift. Precisely, for a transition whose origin is encoded at position $u \geq 1$, the label is given by the digits of the x_i's at position u. For a transition whose origin is encoded at position $u = r^{-1}$, the label is the separator $\star$. Finally, for a transition whose origin is encoded at position $u < r^{-1}$, the label is given by the digits of the x_i's at position ru. This is expressed by the following formula where the b_i's are used to avoid mixing dual encodings of the x_i's.

$$
\begin{aligned}
&\phi_4(x_1,\ldots,x_n,b_1,\ldots,b_n,y_1,\ldots,y_m,a_1,\ldots,a_m,z)\\
&\quad\equiv\ (\forall u\in\mathbf{R})((u\le z\ \wedge\ P_r(u))\Rightarrow\\
&\qquad\bigvee_{s_1,s_2\in S}\ (\ \bigwedge_{i\in\{1,\ldots,m\}}\ (X'_r(y_i,u,e_i(s_1),a_i)\ \wedge\ X'_r(y_i,u/r,e_i(s_2),a_i))\\
&\qquad\qquad\wedge((u\ge 1\ \wedge\ \bigvee_{d_1,\ldots,d_n\in\{0,\ldots,r-1\}}\ ((s_1,(d_1,\ldots,d_n),s_2)\in\Delta\\
&\qquad\qquad\qquad\wedge\ \bigwedge_{i\in\{1,\ldots,n\}}\ X'_r(x_i,u,d_i,b_i)))\\
&\qquad\qquad\vee(u=r^{-1}\ \wedge\ (s_1,\star,s_2)\in\Delta)\\
&\qquad\qquad\vee(u<r^{-1}\ \wedge\ \bigvee_{d_1,\ldots,d_n\in\{0,\ldots,r-1\}}\ ((s_1,(d_1,\ldots,d_n),s_2)\in\Delta\\
&\qquad\qquad\qquad\wedge\ \bigwedge_{i\in\{1,\ldots,n\}}\ X'_r(x_i,ru,d_i,b_i)))))\\
&\qquad\wedge(u>z\ \wedge\ P_r(u))\Rightarrow(\ \bigwedge_{i\in\{1,\ldots,n\}}\ \bigvee_{k\in\{0,r-1\}}\\
&\qquad\qquad\qquad(X'_r(x_i,u,k,b_i)\ \wedge\ X'_r(x_i,u/r,k,b_i)))).
\end{aligned}
$$

In summary, we have

$$
\begin{aligned}
&R_{\mathcal{A}}(x_1,x_2,\ldots,x_n,y_1,y_2,\ldots,y_m,a_1,a_2,\ldots,a_m)\\
&\quad\equiv\ (\exists z,b_1,b_2,\ldots,b_n\in\mathbf{R})\\
&\qquad\bigwedge_{i\in\{1,\ldots,4\}}\ \phi_i(x_1,\ldots,x_n,b_1,\ldots,b_n,y_1,\ldots,y_m,a_1,\ldots,a_m,z).
\end{aligned}
$$

□

4 Representing Linear Equations by Automata

The problem addressed in this section consists of constructing an RVA that represents the set S of all the solutions $\mathbf{x}\in\mathbf{R}^n$ of an equation of the form $\mathbf{a}.\mathbf{x}=b$, given $n\ge 0$, $\mathbf{a}\in\mathbf{Z}^n$ and $b\in\mathbf{Z}$.

4.1 A Decomposition of the Problem

The basic idea is to build the automaton corresponding to a linear equation in two parts : one that accepts the integer part of solutions of the equation and one that accepts the part of the solution that belongs to $[0,1]^n$.

More precisely, let $\mathbf{x}\in S$, and let $w_I\star w_F$ be an encoding of $\mathbf{x}$ in a base $r>1$, with $w_I\in\Sigma^*$, $w_F\in\Sigma^\omega$, and $\Sigma=\{0,\ldots,r-1\}^n$. The vectors $\mathbf{x}_I$ and $\mathbf{x}_F$ respectively encoded by the words $w_I\star\mathbf{0}^\omega$ and $\mathbf{0}\star w_F$, where $\mathbf{0}=(0,\ldots,0)$, are such that $\mathbf{x}_I\in\mathbf{Z}^n$, $\mathbf{x}_F\in[0,1]^n$, and $\mathbf{x}=\mathbf{x}_I+\mathbf{x}_F$. Since $\mathbf{a}.\mathbf{x}=b$, we have $\mathbf{a}.\mathbf{x}_I+$

$\mathbf{a}.\mathbf{x}_F = b$. Moreover, writing $\mathbf{a}$ as $(a_1, \ldots, a_n)$, we have $\alpha \leq \mathbf{a}.\mathbf{x}_F \leq \alpha'$, where $\alpha = \sum_{a_i<0} a_i$ and $\alpha' = \sum_{a_i>0} a_i$, which implies $b - \alpha' \leq \mathbf{a}.\mathbf{x}_I \leq b - \alpha$. Another immediate property of interest is that $\mathbf{a}.\mathbf{x}_I$ is divisible by $\gcd(a_1, \ldots, a_n)$.

From those results, we obtain that the language L of the encodings of all the elements of S satisfies

$$L = \bigcup_{\varphi(\beta)} \{w_I \in \Sigma^* \mid \mathbf{a}.[w_I \star \mathbf{0}^\omega]_r = \beta\} \cdot \{\star\} \cdot \{w_F \in \Sigma^\omega \mid \mathbf{a}.[\mathbf{0} \star w_F]_r = b - \beta\},$$

where "$\cdot$" denotes concatenation and $\varphi(\beta)$ stands for $b-\alpha' \leq \beta \leq b-\alpha \wedge (\exists m \in \mathbf{Z})(\beta = \gcd(a_1, \ldots, a_n) \times m)$.

This decomposition of L reduces the computation of an RVA representing S to the following problems:

- building an automaton on finite words accepting all the words $w_I \in \Sigma^*$ such that $[w_I \star \mathbf{0}^\omega]_r$ is a solution of a given linear equation;
- building a Büchi automaton accepting all the words $w_F \in \Sigma^\omega$ such that $[\mathbf{0} \star w_F]_r$ is a solution of a given linear equation.

These problems are addressed in the two following sections.

4.2 Recognizing Integer Solutions

Our goal is, given an equation $\mathbf{a}.\mathbf{x} = b$ where $\mathbf{a} \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$, to construct a finite automaton $\mathcal{A}_{\mathbf{a},b}$ that accepts all the finite words encoding in a given base r the integer solutions of that equation.

The construction proceeds as follows. The initial state of $\mathcal{A}_{\mathbf{a},b}$ is denoted s_0. All the other states s are in one-to-one correspondence with an integer $\beta(s)$, with the property that the vectors $\mathbf{x} \in \mathbf{Z}^n$ accepted by the paths leading from s_0 to s are exactly the solutions of the equation $\mathbf{a}.\mathbf{x} = \beta(s)$. The only accepting state s_F of $\mathcal{A}_{\mathbf{a},b}$ is the one such that $\beta(s_F) = b$.

The next step is to define the transitions of $\mathcal{A}_{\mathbf{a},b}$. Consider moving from a state s to a state s' while reading a tuple $\mathbf{d} = (d_1, \ldots, d_n)$ of digits. This has the effect of appending these digits to the number that has been read so far and thus the number $\mathbf{x}'$ read when reaching s' is related to the number $\mathbf{x}$ that had been read when reaching s by $\mathbf{x}' = r\mathbf{x} + \mathbf{d}$. Therefore, for states s and s' other than s_0 to be linked by a transition labeled d, the number $\beta(s')$ associated with the state s' has to be given by

$$\beta(s') = \mathbf{a}.\mathbf{x}' = r\,\mathbf{a}.\mathbf{x} + \mathbf{a}.\mathbf{d} = r.\beta(s) + \mathbf{a}.\mathbf{d}.$$

For transitions from s_0, the relation is similar, except that the bits that are read can only be sign bits (0 or $r-1$), that a bit $r-1$ has to be interpreted as -1 when computing $\mathbf{a}.\mathbf{d}$, and that the fictitious $\beta(s_0)$ is taken to be 0.

In practice, we will compute the automaton $\mathcal{A}_{\mathbf{a},b}$ backwards, starting from the accepting state and moving backwards along transitions. Thus, the transition reaching a given state s' and labeled by a vector of bits $\mathbf{d}$ will originate from the state s such that

$$\beta(s) = \frac{\beta(s') - \mathbf{a}.\mathbf{d}}{r}.$$

Note that $\beta(s)$ must be an integer, and must be divisible by $\gcd(a_1, \ldots, a_n)$, otherwise there would be no integer solution to $\mathbf{a}.\mathbf{x} = \beta(s)$. If this requirement is not satisfied, then there is no ingoing transition to s' labeled by $\mathbf{d}$.

From an algorithmic perspective, the automaton $\mathcal{A}_{\mathbf{a},b}$ can be constructed by starting from the state s_F such that $\beta(s_F) = b$, and then repeatedly computing the ingoing transitions to the current states until stabilization occurs (we will shortly show that it always does). If one wishes to construct k automata $\mathcal{A}_{\mathbf{a},b_1}, \mathcal{A}_{\mathbf{a},b_2}, \ldots, \mathcal{A}_{\mathbf{a},b_k}$, with $b_1, \ldots, b_k \in \mathbf{Z}$ (for instance, as an application of the method presented in Section 4.1, in which the b_i are all the integers satisfying φ), then a technique more efficient than repeating the construction k times consists of starting from the set $\{s_1, \ldots, s_k\}$ such that $\beta(s_i) = b_i$ for each $i \in \{1, \ldots, k\}$, rather than from a set containing a single state. The states and transitions computed during the construction will then be shared between the different $\mathcal{A}_{\mathbf{a},b_i}$, and each s_i will be the only accepting state of the corresponding $\mathcal{A}_{\mathbf{a},b_i}$.

Let us now show that the computation terminates. The immediate predecessors $s \neq s_0$ of a state $s' \neq s_0$ are such that $(\beta(s') - (r-1)\alpha')/r \leq \beta(s) \leq (\beta(s') - (r-1)\alpha)/r$, where α and α' are as defined in Section 4.1. As a consequence, if there exists a path of length $k \geq 0$ from a state s to the accepting state s_F, we have

$$\frac{1}{r^k}\beta(s_F) - (r-1)\alpha' \sum_{1 \leq i \leq k} \frac{1}{r^i} \leq \beta(s) \leq \frac{1}{r^k}\beta(s_F) - (r-1)\alpha \sum_{1 \leq i \leq k} \frac{1}{r^i}.$$

It follows that if k is such that $r^k > \beta(s_F)$, then we have $-(r-1)\alpha' \leq \beta(s) \leq -(r-1)\alpha$. Thus during the construction of the automaton, the only non-initial states s that have to be considered are those belonging to the finite union of intervals

$$\bigcup_{0 \leq k \leq l} [\frac{1}{r^k}\beta(s_F) - (r-1)\alpha', \frac{1}{r^k}\beta(s_F) - (r-1)\alpha],$$

where $l = \log_r(\beta(s_F)) + 1$. The total number of states of the automaton is then bounded by $l(r-1)(\alpha' - \alpha) + 1$. If as described above, the construction is done simultaneously for a set $b_1, \ldots, b_k \in \mathbf{Z}$ of right-hand side values for the equation, the computation above should be reworked taking into account the maximum β_{max} and the minimum β_{min} of these values and the bound on the number of states of the automaton becomes $l(r-1)(\alpha' - \alpha) + (\beta_{max} - \beta_{min} + 1)$ with $l = \log_r(\max(|\beta_{min}|, |\beta_{max}|)) + 1$.

4.3 Recognizing Fractional Solutions

We now address the computation of a Büchi automaton $\mathcal{A}'_{\mathbf{a},b}$ that accepts all the infinite words $w \in \Sigma^\omega$ such that $\mathbf{0} \star w$ encodes a solution $\mathbf{x} \in [0,1]^n$ of the equation $\mathbf{a}.\mathbf{x} = b$.

The construction is similar to the one of Section 4.2, except that we are now dealing with the expansion of fractional numbers. All the states s of $\mathcal{A}'_{\mathbf{a},b}$ are in a one-to-one correspondence with an integer $\beta'(s)$, such that the vectors $\mathbf{x} \in \mathbf{Z}^n$ accepted by the infinite paths starting from s are exactly the solutions of the equation $\mathbf{a}.\mathbf{x} = \beta'(s)$. The initial state s_0 is the one such that $\beta'(s_0) = b$. All the states are accepting.

The transitions of $\mathcal{A}'_{\mathbf{a},b}$ are defined as follows. Consider moving from a state s to a state s' while reading a tuple $\mathbf{d} = (d_1, \ldots, d_n)$ of digits. This amounts to prefixing the tuple $\mathbf{d}$ to the word that will be read from s'. The value $\mathbf{x}$ of the word read from s is thus related to the value of the word read from s' by $\mathbf{x} = (1/r)(\mathbf{x}' + \mathbf{d})$. Therefore, for states s and s' to be linked by a transition labeled $\mathbf{d}$, the number $\beta'(s')$ associated with the state s' has to be given by

$$\beta'(s') = \mathbf{a}.\mathbf{x}' = r\mathbf{a}.\mathbf{x} - \mathbf{a}.\mathbf{d} = r\beta'(s) - \mathbf{a}.\mathbf{d}.$$

This expression allows one to compute the value of $\beta'(s')$ given $\beta'(s)$ and $\mathbf{d}$, i.e., to determine the outgoing transitions from the state s. Note that $\beta'(s')$ must belong to the interval $[\alpha, \alpha']$, where α and α' are as defined in Section 4.1, otherwise there would be no solution in $[0, 1]^n$ to $\mathbf{a}.\mathbf{x}' = \beta'(s')$. If this requirement is not satisfied, then there is no outgoing transition from s labeled by $\mathbf{d}$.

The automaton $\mathcal{A}'_{\mathbf{a},b}$ can be constructed by starting from the state s such that $\beta'(s) = b$, and then repeatedly computing the outgoing transitions from the current states until stabilization occurs. Like in Section 4.2, the construction of k automata $\mathcal{A}'_{\mathbf{a},b_1}, \mathcal{A}'_{\mathbf{a},b_2}, \ldots, \mathcal{A}'_{\mathbf{a},b_k}$, with $b_1, \ldots, b_k \in \mathbf{Z}$ (for instance, as an application of the method presented in Section 4.1) can simply be done by starting from the set $\{s_1, \ldots, s_k\}$ such that $\beta'(s_i) = b_i$ for each $i \in \{1, \ldots, k\}$, rather than from a set containing a single state. The computation terminates, since for every state s, the integer $\beta'(s)$ belongs to the bounded interval $[\alpha, \alpha']$. The number of states of the resulting automaton is thus bounded by $\alpha' - \alpha + 1$.

4.4 Complexity

If the decomposition proposed in Section 4.1 and the algorithms presented in Sections 4.2 and 4.3 are used to build an RVA $\mathcal{A}$ representing the set of all the solutions of the equation $\mathbf{a}.\mathbf{x} = b$, with $n \geq 0$, $\mathbf{a} \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$, then the number of states N_S of $\mathcal{A}$ is bounded by $l(r-1)(\alpha' - \alpha) + (\beta_{max} - \beta_{min} + 1) + \alpha' - \alpha + 1$, where α, α' and l are as defined in Sections 4.2 and 4.3, $\beta_{min} = b - \alpha'$, and $\beta_{max} = b - \alpha$. As a consequence, we have $N_S = 0(\sum_{1 \leq i \leq n} |a_i| \log_r |b|)$. The numbers of ingoing and of outgoing transitions of each state are bounded independently of $\mathbf{a}$ and of b by r^n. The total size of $\mathcal{A}$ is thus asymptotically linear in the magnitude of the multiplicative coefficients a_i of the equation, and logarithmic in the magnitude of the additive constant b. The cost of the construction is linear in the size of $\mathcal{A}$.

It is worth mentioning that the automaton $\mathcal{A}$ is deterministic and minimal. Indeed, by construction, any pair of transitions outgoing from the same state and labeled by the same tuple of digits lead to the same state. Moreover, the sets of words labeling the paths that are accepted from two distinct states s_1 and s_2 correspond to the sets of solutions of two equations $\mathbf{a}.\mathbf{x} = \beta_1$ and $\mathbf{a}.\mathbf{x} = \beta_2$ such that $\beta_1 \neq \beta_2$, and are thus different.

5 Representing Linear Inequations by Automata

The method presented in Section 4 can be easily adapted to linear inequations. The problem consists of computing an RVA representing the set of all the solutions $\mathbf{x} \in \mathbf{R}^n$ of an inequation of the form $\mathbf{a}.\mathbf{x} \leq b$, given $n \geq 0$, $\mathbf{a} \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$.

The decomposition of the problem into the computation of representations of the sets of integer solutions and of solutions in $[0,1]^n$ of linear inequations is identical to the one proposed for equations in Section 4.1.

Given an inequation of the form $\mathbf{a}.\mathbf{x} \leq b$, where $\mathbf{a} \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$, the definition of an automaton $\mathcal{A}_{\mathbf{a},b}$ that accepts all the finite words $w \in \Sigma^*$ such that $w \star \mathbf{0}^\omega$ encodes an integer solution of $\mathbf{a}.\mathbf{x} \leq b$ is very similar to the one given for equations in Section 4.2. The only difference with the case of equations, is that we do not discard the states s for which the computed $\beta(s)$ is not an integer or is not divisible by $\gcd(a_1, \ldots, a_n)$. Instead, we round the value of $\beta(s)$ to the nearest lower integer β'' that is divisible by $\gcd(a_1, \ldots, a_n)$. This operation is correct since the sets of integer solutions of $\mathbf{a}.\mathbf{x}' \leq \beta(s')$ and of $\mathbf{a}.\mathbf{x}' \leq \beta''$ are in this case identical.

The construction of an automaton $\mathcal{A}_{\mathbf{a},b}$ that accepts all the infinite words $w \in \Sigma^\omega$ such that $\mathbf{0} \star w$ encodes a solution of $\mathbf{a}.\mathbf{x} \leq b$ that belongs to $[0,1]^n$ is again very similar to the one developed for equations in Section 4.3. The difference with the case of equations, in that we do not discard here the states s' for which the computed $\beta'(s')$ is greater than α'. Instead, we simply replace the value of $\beta'(s')$ by α', since the sets of solutions in $[0,1]^n$ of $\mathbf{a}.\mathbf{x}' \leq \beta'(s')$ and of $\mathbf{a}.\mathbf{x}' \leq \alpha'$ are in this case identical. On the other hand, we still discard the states s' for which the computed $\beta'(s')$ is lower than α, since this implies that the inequation $\mathbf{a}.\mathbf{x}' \leq \beta'(s')$ has no solution in $[0,1]^n$.

The size of the resulting RVA and the cost of the construction are similar to those obtained for equations. However, in general, the constructed RVA is neither deterministic nor minimal in this case.

6 RVA Representing Arbitrary Formulas

RVA are not restricted to representing the set of solutions of linear equations and inequations. We have the following result [BBR97].

Theorem 8. *Let V_1, V_2 be sets of real vectors of respective arities (number of components per vector) n_1 and n_2, and A_1, A_2 be base-r RVA representing respectively V_1 and V_2. There exist algorithms for computing a base-r RVA representing:*

- *The union* $V_1 \cup V_2$ *and intersection* $V_1 \cap V_2$, *provided that* $n_1 = n_2$;
- *The complement* $\overline{V_1}$;
- *The Cartesian product* $V_1 \times V_2 = \{(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in V_1 \wedge \mathbf{x}_2 \in V_2\}$;
- *The projection* $\exists x_i V_1 = \{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_1}) \mid (\exists x_i)(x_1, \ldots, x_{n_1}) \in V_1\}$;
- *The reordering* $\pi V_1 = \{(x_{\pi(1)}, \ldots, x_{\pi(n_1)}) \mid (x_1, \ldots, x_{n_1}) \in V_1\}$, *where* π *is a permutation of* $\{1, \ldots, n_1\}$.

Furthermore, we can also prove the following.

Theorem 9. *Given a base* $r > 1$, *there exist RVA representing the sets* $\{(x, u, k) \in \mathbf{R}^3 \mid X_r(x, u, k)\}$ *as well as the sets* $\mathbf{R}^n$ *and* $\mathbf{Z}^n$.

Proof The RVA accepting $\mathbf{R}^n$ and $\mathbf{Z}^n$ are immediate to construct. The one representing the set $\{(x, u, k) \in \mathbf{R}^3 \mid X_r(x, u, k)\}$, though simple in its principle is rather lengthy due to the necessity to deal with dual encodings. It will be given in the full paper. □

As a consequence of the results of Sections 4 and 5 as well as of Theorems 8 and 9, one can build for every formula ψ of $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$ an RVA representing the set of real vectors that satisfies ψ. From this and the fact that RVA can be algorithmically tested for nonemptiness, we can establish the following results.

Theorem 10. *Let* $n, r \in \mathbf{N}$ *with* $r > 1$. *Every subset of* $\mathbf{R}^n$ *is* r*-recognizable if and only if it is* r*-definable.*

Theorem 11. *The first-order theory* $\langle \mathbf{R}, +, \leq, Z, X_r \rangle$ *is decidable.*

7 Conclusions

At first glance, studying the representation of sets of real vectors by Büchi automata might seem to be a rather odd idea. Indeed, real vectors are such a well studied subject that questioning the need for yet another representation is natural. However, the appeal of automata is their ability to deal easily with integers as well as reals, their simplicity and, foremost, the fact that they are a representation that is easily manipulated by algorithms, which for instance makes the existence of decision procedures almost obvious. The results of this paper show furthermore that the expressiveness of Büchi automata accepting the encodings of real vectors can be characterized quite naturally.

The procedures that were given for building automata corresponding to linear equations and inequations are not needed from a theoretical point of view (a direct expression of basic predicates would be sufficient), but show that the most commonly used description of sets of real vectors can be quite efficiently expressed by automata. This opens the path towards the actual use for practical purposes of this representation. From this point of view, a likely objection is that the size of the alphabet increases exponentially with the number of components of the vectors. However, this problem can be solved by sequentializing the reading of the digits of the vector components. That is, rather than reading a tuple of digits $\mathbf{d} = (d_1, \ldots, d_n)$ at each transition, one cycles through transitions reading in turn d_1, d_2, $\ldots d_n$, which is actually what would be done in a BDD.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, February 1995.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

[BBR97] B. Boigelot, L. Bronne, and S. Rassart. An improved reachability analysis method for strongly linear hybrid systems. In *Proceedings of the 9th International Conference on Computer-Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 167–177, Haifa, Israel, June 1997. Springer-Verlag.

[BC96] A. Boudet and H. Comon. Diophantine equations, Presburger arithmetic and finite automata. In *Proceedings of CAAP'96*, number 1059 in Lecture Notes in Computer Science, pages 30–43. Springer-Verlag, 1996.

[BHMV94] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. *Bulletin of the Belgian Mathematical Society*, 1(2):191–238, March 1994.

[Bry86] R. E. Bryant. Graph based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Bry92] R. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.

[Büc60] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift Math. Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[Büc62] J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress on Logic, Method, and Philosophy of Science*, pages 1–12, Stanford, CA, USA, 1962. Stanford University Press.

[BW94] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proc. 6rd Workshop on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 55–67, Stanford, June 1994. Springer-Verlag.

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

[Hen96] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994. Special issue for LICS 92.

[Sem77] A. L. Semenov. Presburgerness of predicates regular in two number systems. *Siberian Mathematical Journal*, 18:289–299, 1977.

[Vil92] R. Villemaire. The theory of $\langle \mathbf{N}, +, V_k, V_l \rangle$ is undecidable. *Theoretical Computer Science*, 106:337–349, 1992.

This article was processed using the LaTeX macro package with LLNCS style