# An Effective Decision Procedure for Linear Arithmetic over the Integers and Reals *

BERNARD BOIGELOT, SÉBASTIEN JODOGNE †, and PIERRE WOLPER
Université de Liège
Institut Montefiore, B28
4000 Liège, Belgium

This paper considers finite-automata based algorithms for handling linear arithmetic with both real and integer variables. Previous work has shown that this theory can be dealt with by using finite automata on infinite words, but this involves some difficult and delicate to implement algorithms. The contribution of this paper is to show, using topological arguments, that only a restricted class of automata on infinite words are necessary for handling real and integer linear arithmetic. This allows the use of substantially simpler algorithms, which have been successfully implemented.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*; F.1.1 [**Computation by abstract devices**]: Models of computation—*Automata*; F.4.1 [**Mathematical Logic and formal languages**]: Mathematical Logic—*Computational logic*; F.4.3 [**Mathematical Logic and formal languages**]: Formal languages—*Classes defined by grammars or automata.*

General Terms: Algorithms, Theory.

Additional Key Words and Phrases: Decision procedure, Finite-state representations, Integer and real arithmetic, Weak $\omega-$automata.

## 1. INTRODUCTION

Among the techniques used to develop algorithms for deciding or checking logical formulas, finite automata have played an important role in a variety of cases. Classical examples are the use of infinite-word finite automata by Büchi [Büchi 1962] for obtaining decision procedures for the first and second-order monadic theories of one successor, as well as the use of tree automata by Rabin [Rabin 1969] for deciding the second-order monadic theory of $n$ successors. More recent examples

Authors' e-mail : {boigelot,jodogne,pw}@montefiore.ulg.ac.be
Authors' website : http://www.montefiore.ulg.ac.be/∼{boigelot,jodogne,pw}/

* This work was partially funded by a grant of the "Communauté française de Belgique - Direction de la recherche scientifique - Actions de recherche concertées" and by the European IST-FET project ADVANCE (IST-1999-29082).
A preliminary version of this paper appeared as [Boigelot et al. 2001].

† Research Fellow ("Aspirant") for the National Fund for Scientific Research (Belgium).

are the use of automata for obtaining decision and model-checking procedures for temporal and modal logics [Vardi and Wolper 1986a; 1986b; 1994; Kupferman et al. 2000]. In this last setting, automata-based procedures have the advantage of moving the combinatorial aspects of the procedures to the context of automata, which are simple graph-like structures well adapted to algorithmic developments. This separation of concerns between the logical and the algorithmic has been quite fruitful for instance in the implementation of model checkers for linear-time temporal logic [Courcoubetis et al. 1990; Holzmann 1997].

As already noticed by Büchi [Büchi 1962; 1960], automata-based approaches are not limited to sequential and modal logics, but can also be used for Presburger arithmetic. To achieve this, one adopts the usual encoding of integers in a base $r \geq 2$, thus representing an integer as a word over the alphabet $\{0, \ldots, r-1\}$. By extension, $n$-component integer vectors are represented by words over the alphabet $\{0, \ldots, r-1\}^n$ and a finite automaton operating over this alphabet represents a set of integer vectors. Given that addition and order are easily represented by finite automata and that these automata are closed under Boolean operations as well as projection, one easily obtains a decision procedure for Presburger arithmetic. This idea was first explored at the theoretical level, yielding for instance the very nice result that base-independent finite-automaton representable sets are exactly the Presburger sets [Cobham 1969; Semenov 1977; Bruyère et al. 1994]. Later, it has been proposed as a practical means of deciding and manipulating Presburger formulas [Boudet and Comon 1996; Boigelot 1998; Shiple et al. 1998; Wolper and Boigelot 2000]. The intuition behind this applied use of automata for Presburger arithmetic is that finite automata play with respect to Presburger arithmetic a role similar to the one of Binary Decision Diagrams (BDD) with respect to Boolean logic. These ideas have been implemented in the LASH tool [LASH ], which has been used successfully in the context of verifying systems with unbounded integer variables.

It almost immediately comes to mind that if a finite word over the alphabet $\{0, \ldots, r-1\}$ can represent an integer, an infinite word over the same alphabet extended with a fractional part separator (the usual dot) can represent a real number. Finite automata on infinite words can thus represent sets of real vectors, and serve as a means of obtaining a decision procedure for real additive arithmetic. Furthermore, since numbers with fractional parts equal to zero can easily be recognized by automata, the same technique can be used to obtain a decision procedure for a theory combining the integers and the reals. This was not previously handled by any tool, but can be of practical use, for instance in the verification of timed systems using integer variables [Boigelot et al. 1997]. However, turning this into an effective implemented system is not as easy as it might first seem. Indeed, projecting and complementing finite automata on infinite words is significantly more difficult than for automata on finite words. Projection yields nondeterministic automata and complementing or determinizing infinite-word automata is a notoriously difficult problem. A number of algorithms have been proposed for this [Büchi 1962; Sistla et al. 1987; Safra 1988; Klarlund 1991; Kupferman and Vardi 1997], but even though their theoretical complexity remains simply exponential as in the finite-word case, it moves up from $2^{O(n)}$ to $2^{O(n \log n)}$ and none of the proposed algorithms are

as easy to implement and fine-tune as the simple Rabin-Scott subset construction used in the finite-word case.

However, it is intuitively surprising that handling reals is so much more difficult than handling integers, especially in light of the fact that the usual polyhedra-based approach to handling arithmetic is both of lower complexity and easier to implement for the reals than for the integers [Ferrante and Rackoff 1979]. One would expect that handling reals with automata should be no more difficult than handling integers[1]. The conclusion that comes out of these observations is that infinite-word automata constructed from linear arithmetic formulas must have a special structure that makes them easier to manipulate than general automata on infinite words. That this special structure exists and that it can exploited to obtain simpler algorithms is precisely the subject of this paper.

As a starting point, let us look at the topological characterization of the sets definable by linear arithmetic formulas. Let us first consider a formula involving solely real variables. If the formula is quantifier free, it is a Boolean combination of linear constraints and thus defines a set which is a finite Boolean combination of open and closed sets. Now, since real linear arithmetic admits quantifier elimination, the same property also holds for quantified formulas. Then, looking at classes of automata on infinite words, one notices that the most restricted one that can accept Boolean combinations of open and closed sets is the class of deterministic weak automata [Staiger and Wagner 1974; Staiger 1983]. These accept all $\omega$-regular sets in the Borel class $F_\sigma \cap G_\delta$ and hence also finite Boolean combinations of open and closed sets. So, with some care about moving from the topology on vectors to the topology on their encoding as words, one can conclude that the sets representable by arithmetic formulas involving only real variables can always be accepted by deterministic weak automata on infinite words. If integers are also involved in the formula, a similar argument can be used, invoking a recently published quantifier elimination result for the combined theory [Weispfenning 1999]. However, initially unaware of this result, we developed a different argument to prove that sets definable by quantified linear arithmetic formulas involving both real and integer variables are within $F_\sigma \cap G_\delta$ and thus are representable by weak deterministic automata. This proof relies on separating the integer and fractional parts of variables and on topological properties of $F_\sigma \cap G_\delta$. It has the advantage of being much more direct than a proof relying on a quantifier elimination result.

The problematic part of the operations on automata used for deciding a first-order theory is the sequence of projections and complementations needed to eliminate a string of quantifiers alternating between existential and universal ones. The second result of this paper shows that for sets defined in linear arithmetic this can be done with constructions that are simple adaptations of the ones used for automata on finite words. Indeed, deterministic weak automata can be viewed as either Büchi or co-Büchi automata. The interesting fact is that co-Büchi automata can be determinized by the "breakpoint" construction [Miyano and Hayashi 1984; Kupferman and Vardi 1997], which basically amounts to a product of subset constructions.

[1]Note that one cannot expect reals to be easier to handle with automata than integers since, by nature, this representation includes explicit information about the existence of integer values satisfying the represented formula.

Thus, one has a simple construction to project and determinize a weak automaton, yielding a deterministic co-Büchi automaton, which is easily complemented into a deterministic Büchi automaton. In the general case, another round of projection will lead to a nondeterministic Büchi automaton, for which a general determinization procedure has to be used. However, we have the result that for automata obtained from linear arithmetic formulas, the represented sets stay within those accepted by deterministic weak automata. We prove that this implies that the automata obtained after determinization will always be weak.

Note that this cannot be directly concluded from the fact that the represented sets stay within those representable by deterministic weak automata. Indeed, even though the represented sets can be accepted by deterministic weak automata, the automata that are obtained by the determinization procedure might not have this form. Fortunately, we can prove that this is impossible. For this, we go back to the link between automata and the topology of the sets of infinite words they accept. The argument is that $\omega$-regular sets in $F_\sigma \cap G_\delta$ have a topological property that forces the automata accepting them to be inherently weak, i.e. not to have strongly connected components containing both accepting and non accepting cycles.

Finally, an important additional benefit of working with weak deterministic automata is that they admit a canonical minimal normal form that can be obtained efficiently [Maler and Staiger 1997; Löding 2001]. This brings us even closer to the situation of working with finite-work automata, and is a property that is not available when working either with general infinite-word automata, or with formulas as done in [Weispfenning 1999].

As a consequence of our results, we obtain a simple decision procedure for the theory combining integer and real linear arithmetic. The fact that this theory is decidable using automata-based methods was known [Boigelot et al. 1997], but the results of this paper make it possible to implement a tool that can handle it effectively.

## 2. AUTOMATA-THEORETIC AND TOPOLOGICAL BACKGROUND

In this section we recall some automata-theoretic and topological concepts that are used in the paper.

### 2.1 Automata on Infinite Words

An infinite word (or $\omega$-word) $w$ over an alphabet $\Sigma$ is a mapping $w : \mathbb{N} \mapsto \Sigma$ from the natural numbers to $\Sigma$. A Büchi automaton on infinite words is a five-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where

—$Q$ is a finite set of states;
—$\Sigma$ is the input alphabet;
—$\delta$ is the transition function and is of the form $\delta : Q \times \Sigma \mapsto 2^Q$ if the automaton is nondeterministic and of the form $\delta : Q \times \Sigma \mapsto Q$ if the automaton is deterministic;
—$q_0$ is the initial state;
—$F$ is a set of accepting states.

A run $\pi$ of a Büchi automaton $A = (Q, \Sigma, \delta, q_0, F)$ on an $\omega$-word $w$ is a mapping $\pi : \mathbb{N} \mapsto Q$ that satisfies the following conditions :

—$\pi(0) = q_0$, i.e. the run starts in the initial state;

—for all $i \geq 0$, $\pi(i+1) \in \delta(\pi(i), w(i))$ (nondeterministic automata) or $\pi(i+1) = \delta(\pi(i), w(i))$ (deterministic automata), i.e. the run respects the transition function.

Let $inf(\pi)$ be the set of states that occur infinitely often in a run $\pi$. A run $\pi$ is said to be accepting if $inf(\pi) \cap F \neq \emptyset$. An $\omega$-word $w$ is accepted by a Büchi automaton if that automaton has some accepting run on $w$. The language $L_\omega(A)$ of infinite words defined by a Büchi automaton $A$ is the set of $\omega$-words it accepts. The $\omega-$regular languages are defined as the languages of infinite words that can be accepted by a nondeterministic Büchi automaton.

A co-Büchi automaton is defined exactly as a Büchi automaton except that its accepting runs are those for which $inf(\pi) \cap F = \emptyset$.

We will also use the notion of *weak* automata [Muller et al. 1986]. For a Büchi automaton $A = (Q, \Sigma, \delta, q_0, F)$ to be weak, there has to be a partition of its state set $Q$ into disjoint subsets $Q_1, \ldots, Q_m$ such that

—for each of the $Q_i$ either $Q_i \subseteq F$ or $Q_i \cap F = \emptyset$, and

—there is a partial order $\leq$ on the sets $Q_1, \ldots, Q_m$ such that for every $q \in Q_i$ and $q' \in Q_j$ for which, for some $a \in \Sigma$, $q' \in \delta(q, a)$ ($q' = \delta(q, a)$ in the deterministic case), $Q_j \leq Q_i$.

Note that, in order to comply with this definition, each $Q_i$ has to be a union of strongly connected components. Thus, the strongly connected components of a weak automaton consist solely of either accepting or rejecting states.

For more details, a survey of automata on infinite words can be found in [Thomas 1990].

### 2.2 Topology

Given a set $S$, a distance $d(x, y)$ defined on this set induces a metric topology on subsets of $S$. A neighborhood $N_\varepsilon(x)$ of a point $x \in S$ with respect to $\varepsilon \in \mathbb{R}^+$ is the set $N_\varepsilon(x) = \{y \mid d(x, y) < \varepsilon\}$. A set $C \subseteq S$ is said to be open if for all $x \in C$, there exists $\varepsilon > 0$ such that the neighborhood $N_\varepsilon(x)$ is contained in $C$. A closed set is a set whose complement with respect to $S$ is open. We will be referring to the first few levels of the Borel hierarchy which are shown in Figure 1. The notations used are the following :

—$F$ are the closed sets,

—$G$ are the open sets,

—$F_\sigma$ is the class of countable unions of closed sets,

—$G_\delta$ is the class of countable intersections of open sets,

—$F_{\sigma\delta}$ is the class of countable intersections of $F_\sigma$ sets,

—$G_{\delta\sigma}$ is the class of countable unions of $G_\delta$ sets,

—$\mathcal{B}(X)$ represents the finite Boolean combinations of sets in $X$.

An arrow between classes indicates proper inclusion.

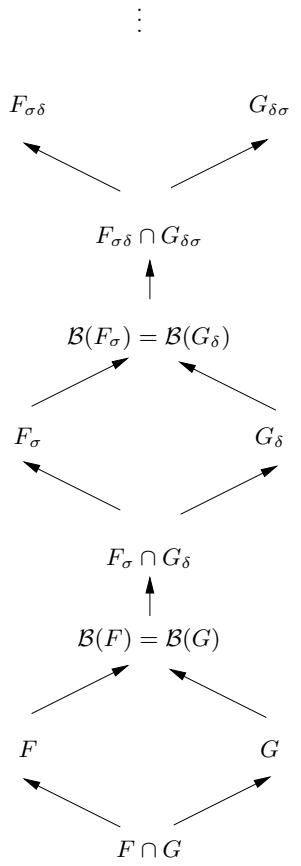


Fig. 1. The first few levels of the Borel hierarchy in a metric topology.

## 3. TOPOLOGICAL CHARACTERIZATION OF ARITHMETIC SETS

We consider the theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$, where $+$ represents the predicate $x + y = z$. Since any linear equality or order constraint can be encoded into this theory, we refer to it as additive or linear arithmetic over the reals and integers. It is the extension of Presburger arithmetic that includes both real and integer variables. We provide the space $\mathbb{R}^n$ $(n \geq 0)$ with the classical Euclidean distance between vectors defined by

$$d(\vec{x}, \vec{y}) = \left( \sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}.$$

The topology induced by this metric will be referred to as the *natural topology* of $\mathbb{R}^n$.

In this section, we prove that the sets representable in the additive linear arithmetic over the reals and integers belong to the topological class $F_\sigma \cap G_\delta$. This result is formalized by the following theorem.

THEOREM 3.1. *Let $S \subseteq \mathbb{R}^n$, with $n > 0$, be a set defined in the theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$. This set belongs to the class $F_\sigma \cap G_\delta$ of the natural topology of $\mathbb{R}^n$.*

PROOF. Since $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ is closed under negation, it is actually sufficient to show that each formula of this theory defines a set that belongs to $F_\sigma$, i.e., a set that can be expressed as a countable union of closed sets.

Let $\varphi$ be a formula of $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$. To simplify our argument, we will assume that all free variables of $\varphi$ are reals. This can be done without loss of generality since quantified variables can range over both $\mathbb{R}$ and $\mathbb{Z}$. We introduce $u < v$ as a shorthand for $u \leq v \wedge \neg(u = v)$.

The first step of our proof consists of modifying $\varphi$ in the following way. We replace each variable $x$ that appears in $\varphi$ by two variables $x_I$ and $x_F$ representing respectively the integer and the fractional part of $x$. Formally, this operation replaces each occurrence in $\varphi$ of a free variable $x$ by the sum $x_I + x_F$ while adding to $\varphi$ the constraints $0 \leq x_F$ and $x_F < 1$, and transforms the quantified variables of $\varphi$ according to the following rules :

$$\begin{aligned}
(\exists x \in \mathbb{R})\phi &\longrightarrow (\exists x_I \in \mathbb{Z})(\exists x_F \in \mathbb{R})(0 \leq x_F \wedge x_F < 1 \wedge \phi[x/x_I + x_F]) \\
(\forall x \in \mathbb{R})\phi &\longrightarrow (\forall x_I \in \mathbb{Z})(\forall x_F \in \mathbb{R})(x_F < 0 \vee 1 \leq x_F \vee \phi[x/x_I + x_F]) \\
(Qx \in \mathbb{Z})\phi &\longrightarrow (Qx_I \in \mathbb{Z})\phi[x/x_I],
\end{aligned}$$

where $Q \in \{\exists, \forall\}$, $\phi$ is a subformula, and $\phi[x/y]$ denotes the result of replacing by $y$ each occurrence of $x$ in $\phi$. The transformation has no influence on the set represented by $\varphi$, except that the integer and fractional parts of each value are now represented by two distinct variables.

Now, the atomic formulas of $\varphi$ are of the form $p = q + r$, $p = q$ or $p \leq q$, where $p, q$ and $r$ are either integer variables, sums of an integer and of a fractional variable, or integer constants. The second step consists of expanding these atomic formulas so as to send into distinct atoms the occurrences of the integer and of the fractional variables. This is easily done with the help of simple arithmetic rules, for the truth value of the atomic formulas that involve both types of variables has only to be preserved for values of the fractional variables that belong to the interval $[0, 1)$. The set of expansion rules[2] (up to commutability of members and terms) is given in Figure 2.

After the transformation, each atomic formula of $\varphi$ is either a formula $\phi_I$ involving only integer variables or a formula $\phi_F$ over fractional variables. We now distribute existential (resp. universal) quantifiers over disjunctions (resp. conjunctions), after rewriting their argument into disjunctive (resp. conjunctive) normal form, and then apply the simplification rules

$$\begin{aligned}
(Qx_I \in \mathbb{Z})(\phi_I \,\alpha\, \phi_F) &\longrightarrow (Qx_I \in \mathbb{Z})(\phi_I) \;\alpha\; \phi_F \\
(Qx_F \in \mathbb{R})(\phi_I \,\alpha\, \phi_F) &\longrightarrow \phi_I \;\alpha\; (Qx_F \in \mathbb{R})(\phi_F),
\end{aligned}$$

where $Q \in \{\exists, \forall\}$ and $\alpha \in \{\vee, \wedge\}$.

Repeating this operation, we eventually get a formula $\varphi'$ equivalent to $\varphi$ that

[2]In these rules, the expression $p = q + r + s$ is introduced as a shorthand for $(\exists u \in \mathbb{R})(u = q + r \wedge p = u + s)$.

$$
\begin{array}{rcl}
x_I = (y_I + y_F) & \longrightarrow & x_I = y_I \wedge y_F = 0 \\
(x_I + x_F) = (y_I + y_F) & \longrightarrow & x_I = y_I \wedge x_F = y_F \\
x_I = y_I + (z_I + z_F) & \longrightarrow & x_I = y_I + z_I \wedge z_F = 0 \\
x_I = (y_I + y_F) + (z_I + z_F) & \longrightarrow & (x_I = y_I + z_I \wedge y_F + z_F = 0) \vee \\
 & & (x_I = y_I + z_I + 1 \wedge y_F + z_F = 1) \\
(x_I + x_F) = y_I + z_I & \longrightarrow & x_I = y_I + z_I \wedge x_F = 0 \\
(x_I + x_F) = y_I + (z_I + z_F) & \longrightarrow & x_I = y_I + z_I \wedge x_F = z_F \\
(x_I + x_F) = (y_I + y_F) + (z_I + z_F) & \longrightarrow & (x_I = y_I + z_I \wedge x_F = y_F + z_F) \vee \\
 & & (x_I = y_I + z_I + 1 \wedge x_F = y_F + z_F - 1) \\
x_I \leq (y_I + y_F) & \longrightarrow & x_I \leq y_I \\
(x_I + x_F) \leq y_I & \longrightarrow & x_I < y_I \vee (x_I = y_I \wedge x_F = 0) \\
(x_I + x_F) \leq (y_I + y_F) & \longrightarrow & x_I < y_I \vee (x_I = y_I \wedge x_F \leq y_F)
\end{array}
$$

Fig. 2. Expansion rules.

takes the form of a finite Boolean combination

$$\mathcal{B}(\phi_I^{(1)}, \phi_I^{(2)}, \ldots, \phi_I^{(m)}, \phi_F^{(1)}, \phi_F^{(2)}, \ldots, \phi_F^{(m')})$$

of subformulas $\phi_I^{(i)}$ and $\phi_F^{(i)}$ that involve respectively only integer and fractional variables.

Let $x_I^{(1)}, x_I^{(2)}, \ldots, x_I^{(k)}$ be the free integer variables of $\varphi'$ ($k \leq m$). For each assignment of values to these variables, the subformulas $\phi_I^{(i)}$ are each identically true or false, hence we have

$$\varphi \equiv \bigvee_{(a_1,\ldots,a_k)\in\mathbb{Z}^k} \left( (x_I^{(1)}, \ldots, x_I^{(k)}) = (a_1, \ldots, a_k) \wedge \mathcal{B}_{(a_1,\ldots,a_k)}(\phi_F^{(1)}, \ldots, \phi_F^{(m')}) \right).$$

Each subformula $\phi_F^{(i)}$ belongs to the theory $\langle \mathbb{R}, +, \leq, 1 \rangle$, which admits the elimination of quantifiers [Ferrante and Rackoff 1979]. The sets of real vectors satisfying these formulas are thus finite Boolean combinations of linear constraints with open or closed boundaries. It follows that, for each $(a_1, \ldots, a_k) \in \mathbb{Z}^k$, the set described by $\mathcal{B}_{(a_1,\ldots,a_k)}$ is a finite Boolean combination of open and closed sets, that is a set belonging to the topological class $\mathcal{B}(F) = \mathcal{B}(G)$. Since, according to properties of the Borel hierarchy, this class forms a subset of $F_\sigma$, the set described by $\varphi$ is a countable union of countable unions of closed sets and also lies within $F_\sigma$. □

## 4. REPRESENTING SETS OF INTEGERS AND REALS WITH FINITE AUTOMATA

In this section, we recall the finite-state representation of sets of real vectors as introduced in [Boigelot et al. 1997]. A similar approach for representing vectors in the unit cube is also pursued in [Jürgensen and Staiger 2001].

In order to make a finite automaton recognize numbers, one needs to establish a mapping between these and words. Our encoding scheme corresponds to the usual notation for reals and relies on an arbitrary integer base $r > 1$. We encode a number $x$ in base $r$, most significant digit first, by words of the form $w_I \star w_F$, where $w_I$ encodes the integer part $x_I$ of $x$ as a finite word over $\{0, \ldots, r-1\}$, the special symbol "$\star$" is a separator, and $w_F$ encodes the fractional part $x_F$ of $x$ as an infinite word over $\{0, \ldots, r-1\}$. Negative numbers are represented by their $r$'s

complement. The length $p$ of $|w_I|$, which we refer to as the *integer-part length* of $w$, is not fixed but must be large enough for $-r^{p-1} \leq x_I < r^{p-1}$ to hold.

According to this scheme, each number has an infinite number of encodings, since their integer-part length can be increased unboundedly. In addition, the rational numbers whose denominator has only prime factors that are also factors of $r$ have two distinct encodings with the same integer-part length. For example, in base 10, the number 11/2 has the encodings $005 \star 5(0)^\omega$ and $005 \star 4(9)^\omega$, "$^\omega$" denoting infinite repetition.

To encode a vector of real numbers, we represent each of its components by 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. An encoding of a vector $\vec{x} \in \mathbb{R}^n$ can indifferently be viewed either as a $n$-tuple of words of identical integer-part length over the alphabet $\{0, \ldots, r-1, \star\}$, or as a single word $w$ over the alphabet $\{0, \ldots, r-1\}^n \cup \{\star\}$.

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

*Definition* 4.1. Let $n > 0$ and $r > 1$ be integers. A *Real Vector Automaton (RVA)* $A$ in base $r$ for vectors in $\mathbb{R}^n$ is a Büchi automaton over the alphabet $\{0, \ldots, r-1\}^n \cup \{\star\}$, such that

—every word accepted by $A$ is an encoding in base $r$ of a vector in $\mathbb{R}^n$, and

—for every vector $\vec{x} \in \mathbb{R}^n$, $A$ accepts either all the encodings of $\vec{x}$ in base $r$, or none of them.

An RVA is said to *represent* the set of vectors encoded by the words that belong to its accepted language.

Efficient algorithms have been developed for constructing RVA representing the sets of solutions of systems of linear equations and inequations [Boigelot et al. 1998]. Boolean operations can easily be achieved on RVA by applying the corresponding existing algorithms for infinite-word automata.

Furthermore, a set represented as an RVA can be quantified existentially with respect to its $i-$th vector component over the real domain, by replacing each symbol in $\{0, \ldots, r-1\}^n$ read by the automaton with the same symbol out of which the $i-$th component has been removed. This produces a nondeterministic automaton that may only accept some encodings of each vector in the quantified set, but generally not all of them. Such a situation can arise if the component of highest magnitude for some vectors in the set is projected out[3]. The second step consists thus of modifying the automaton so as to make it accept every encoding of each vector that it recognizes. Algorithms have been developed for this purpose in the case of finite-word automata [Boigelot 1998; Boigelot and Latour 2001]. These algorithms also apply to RVA, since the behavior of the underlying Büchi automaton before reading the separator "$\star$" is identical to that of a finite-word automaton recognizing the integer part of the vectors in the represented set.

[3]For instance, projecting out the first component of the set $\{(8, 1)\}$ in binary would produce an automaton that does not accept encodings of 1 having less than five bits in their integer part.

Finally, since it is immediate to constrain a number to be an integer with an RVA by imposing its fractional part to be either $0^\omega$ or $(r-1)^\omega$ (i.e. by intersecting its accepted language with $\{0, r-1\}^n \cdot (\{0, \ldots, r-1\}^n)^* \cdot \{\star\} \cdot \{0, r-1\}^n$), it follows that one can construct an RVA for any formula of the arithmetic theory we are considering.

## 5. WEAK AUTOMATA AND THEIR PROPERTIES

If one examines the constructions given in [Boigelot et al. 1998] to build RVA for linear equations and inequations, one notices that they have the property that all states within the same strongly connected component are either accepting or non accepting. This implies that these automata are *weak* in the sense of [Muller et al. 1986] (see Section 2.1).

### 5.1 Determinizing Weak Automata

Weak automata have a number of interesting properties. A first one is that they can be represented both as Büchi and co-Büchi. Indeed, a weak automaton $A = (Q, \Sigma, \delta, q_0, F)$ is equivalent to the co-Büchi automaton $A = (Q, \Sigma, \delta, q_0, Q \setminus F)$, since a run eventually remains within a single component $Q_i$ in which all states have the same status with respect to being accepting. A consequence of this is that weak automata can be determinized by the fairly simple "breakpoint" construction [Kupferman and Vardi 1997; Miyano and Hayashi 1984] that can be used for co-Büchi automata. This construction is the following.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic co-Büchi automaton. The deterministic co-Büchi automaton $A' = (Q', \Sigma, \delta', q'_0, F')$ defined as follows accepts the same $\omega$-language :

—$Q' = 2^Q \times 2^Q$, i.e. the states of $A'$ are pairs of sets of states of $A$.

—$q'_0 = (\{q_0\}, \emptyset)$.

—For $(S, R) \in Q'$ and $a \in \Sigma$, the transition function is defined by

  —if $R = \emptyset$, then $\delta((S, R), a) = (T, T \setminus F)$ where $T = \{q \mid (\exists p \in S)\ q \in \delta(p, a)\}$ : $T$ is obtained from $S$ as in the classical subset construction, and the second component of the pair of sets of states is obtained from $T$ by eliminating states in $F$;

  —if $R \neq \emptyset$, then $\delta((S, R), a) = (T, U \setminus F)$ where $T = \{q \mid (\exists p \in S)\ q \in \delta(p, a)\}$, and $U = \{q \mid (\exists p \in R)\ q \in \delta(p, a)\}$ : the subset construction set is now applied to both $S$ and $R$ and the states in $F$ are removed from $U$.

—$F' = 2^Q \times \{\emptyset\}$.

When the automaton $A'$ is in a state $(S, R)$, $R$ represents the states of $A$ that can be reached by a run that has not gone through a state in $F$ since the last "breakpoint", i.e. state of the form $(S, \emptyset)$. So, for a given word, $A$ has a run that does not go infinitely often through a state in $F$ if and only if $A'$ has a run that does not go infinitely often through a state in $F'$. Notice that the difficulty that exists for determinizing Büchi automata, which is to make sure that the *same* run repeatedly reaches an accepting state, disappears since, for co-Büchi automata, we are just looking for a run that eventually avoids accepting states.

It is interesting to notice that the construction implies that all reachable states $(S, R)$ of $A'$ satisfy $R \subseteq S$. The breakpoint construction can thus be implemented as a subset construction in which the states in $R$ are simply tagged, which implies that the worst-case complexity of the construction is $2^{\mathcal{O}(n)}$. This makes the construction behave in practice very similarly to the traditional subset construction for finite-word automata.

### 5.2 Topological Characterization

Another property of weak automata that will be of particular interest to us is the topological characterization of the sets of words that they can accept. We consider the topology on the sets of infinite words over an alphabet $\Sigma$ induced by the distance on the $\omega-$words

$$d(w, w') = \begin{cases} \frac{1}{|common(w,w')|+1} & \text{if } w \neq w' \\ 0 & \text{if } w = w', \end{cases}$$

where $|common(w, w')|$ denotes the length of the longest common prefix of $w$ and $w'$. The open sets in such a topological space are the sets of the form $X \cdot \Sigma^\omega$, where $X \subseteq \Sigma^+$ is a language of finite words. Relations between this topology and automata are well understood. For instance, it has been proved that the languages of infinite words that can be accepted by a deterministic Büchi automaton are exactly the $\omega-$rational languages belonging to the class $G_\delta$ [Landweber 1969]. By duality, deterministic co-Büchi automata accept exactly the $\omega$-regular languages that belong to $F_\sigma$.

As weak deterministic automata can be seen both as deterministic Büchi and deterministic co-Büchi, they accept exactly the $\omega$-regular languages that are in $F_\sigma \cap G_\delta$. This follows from the results on the Staiger-Wagner class of automata [Staiger and Wagner 1974; Staiger 1983], which coincides with the class of deterministic weak automata, as can be inferred from [Staiger and Wagner 1974] and is shown explicitly in [Maler and Staiger 1997].

### 5.3 Inherently Weak Automata

Given the result proved in Section 3, it is tempting to conclude that the encodings of sets definable in the theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ can always be accepted by weak deterministic automata. This conclusion is correct, but requires shifting the result from the topology on numbers to the topology on words, which we will do in the next section. In the meantime, we need one more result in order to be able to benefit algorithmically from the fact that we are dealing with $F_\sigma \cap G_\delta$ sets, i.e. that any deterministic automaton accepting a $F_\sigma \cap G_\delta$ set is essentially a weak automaton.

Consider the following definition.

*Definition* 5.1. A Büchi automaton is *inherently weak* if none of the reachable strongly connected components of its transition graph contains both accepting (including at least one accepting state) and non accepting (not including any accepting state) cycles.

Clearly, if an automaton is inherently weak, it can directly be transformed into a weak automaton : the partition of the state set is its partition into strongly

connected components and all the states of a component are made accepting or not, depending on whether the cycles in that component are accepting or not.

The following theorem can be inferred from results in [Landweber 1969; Wagner 1979]. We give a direct proof.

THEOREM 5.2. *Any deterministic Büchi automaton that accepts a language in $F_\sigma \cap G_\delta$ is inherently weak.*

To prove this, we use the fact that the language accepted by an automaton that is not inherently weak must have the following property.

*Definition* 5.3. A language $L \subseteq \Sigma^\omega$ has the *dense oscillating sequence* property if, $w_1, w_2, w_3, \ldots$ being words and $\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots$ being distances, one has that $\exists w_1 \forall \varepsilon_1 \exists w_2 \forall \varepsilon_2 \ldots$ such that $d(w_i, w_{i+1}) \leq \varepsilon_i$ for all $i \geq 1$, $w_i \in L$ for all odd $i$, and $w_i \notin L$ for all even $i$.

Showing that this infinitesimal oscillation is incompatible with the structure of weak deterministic automata will allow us to conclude. The proof of Theorem 5.2 can thus be split into the two following lemmas.

LEMMA 5.4. *Each $\omega-$language accepted by an Büchi automaton that is not inherently weak has the dense oscillating sequence property.*

PROOF. Consider a reachable strongly component that contains both an accepting and a non accepting cycle, and call $p$ a finite word that allows to reach the first state of the accepting cycle from the initial state of the automaton. Let $c_A$ (resp. $c_N$) be the finite word that labels the accepting (resp. non accepting) cycle, and $t_A$ (resp. $t_N$) a finite word that labels the path from the first state of the accepting (resp. non accepting) cycle to the first state of the non accepting (resp. accepting) cycle.

Given an infinite sequence of distances $\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots$, we are now ready to construct a dense oscillating sequence for the language $L$ accepted by the automaton. If $k_2, k_3, k_4, \ldots$ is a sequence of natural numbers, define $u_1 = p$, and for all $i > 1$ :

$$u_i = \begin{cases} u_{i-1}\ c_N^{k_i}\ t_N & \text{if } i \text{ is odd} \\ u_{i-1}\ c_A^{k_i}\ t_A & \text{if } i \text{ is even.} \end{cases}$$

$w_i$ $(i \geq 1)$ is then defined as follows :

$$w_i = \begin{cases} u_i\ c_A^\omega & \text{if } i \text{ is odd} \\ u_i\ c_N^\omega & \text{if } i \text{ is even.} \end{cases}$$

Given $i \geq 1$, it is always possible to find an integer $k_{i+1}$ large enough for $d(w_i, w_{i+1}) < \varepsilon_i$ to hold. Indeed, the length of the common prefix between $w_i$ and $w_{i+1}$ increases with $k_{i+1}$. Furthermore, $w_i$ loops either in an accepting cycle if $i$ is odd, or in a non accepting cycle if $i$ is even, hence, $w_i \in L$ if and only if $i$ is odd. Thus, the sequence of $w_i$'s is dense oscillating for the language accepted by the automaton. □

LEMMA 5.5. *An $\omega$-regular language that has the dense oscillating sequence property cannot be accepted by a weak deterministic automaton and hence is not in $F_\sigma \cap G_\delta$.*

PROOF. We proceed by contradiction. Assume that a language $L$ having the dense oscillating sequence property is accepted by a weak deterministic automaton $A$. Consider the first word $w_1$ in a dense oscillating sequence for $L$. This word eventually reaches an accepting component $Q_{i_1}$ of the partition of the state set of $A$ and will stay within this component. Since $\varepsilon_1$ can be chosen freely, it can be taken small enough for the run of $A$ on $w_2$ to also reach the component $Q_{i_1}$ before it starts to differ from $w_1$. Since $w_2$ is not in $L$, the run of $A$ on $w_2$ has to eventually leave the component $Q_{i_1}$ and will eventually reach and stay within a non accepting component $Q_{i_2} < Q_{i_1}$. Repeating a similar argument, one can conclude that the run of $A$ on $w_3$ eventually reaches and stays within an accepting component $Q_{i_3} < Q_{i_2}$. Carrying on with this line of reasoning, one concludes that the state set of $A$ must contain an infinite decreasing sequence of distinct components, which is impossible given that it is finite. □

### 5.4 Minimizing Weak Deterministic Automata

The breakpoint construction reduces much of the determinization of weak automata to that of finite-word automata. The similarity can be carried on. Indeed, like finite-word automata, weak deterministic automata admit a normal form unique up to isomorphism [Staiger 1983; Maler and Staiger 1997].

This normal form can be obtained efficiently using an algorithm proposed in [Löding 2001]. The minimization algorithm consists in locating the strongly connected components of the graph of the automaton that do not contain any cycle, then attributing them a new accepting status, according to a rule involving strongly connected components that are deeper in the graph. This operation does not affect the language accepted by the automaton, since for any run $\pi$ of the automaton, $\pi$ cannot loop in such strongly connected components, leaving $inf(\pi)$ unchanged. Hopcroft's classical algorithm for minimizing finite-word automata [Hopcroft 1971] can then be applied directly to the modified weak deterministic automaton in order to get an equivalent minimal weak deterministic automaton.

When suitably implemented, this algorithm can be run in time $\mathcal{O}(n \log n)$, moving us still closer to the case of automata on finite words.

## 6. DECIDING LINEAR ARITHMETIC WITH REAL AND INTEGER VARIABLES

Let us show that the result of Section 3 also applies to the sets of words that encode sets defined in $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$. In order to do so, we need to establish that the topological class $F_\sigma \cap G_\delta$ defined over sets of reals is mapped to its $\omega$-word counterpart by the encoding relation described in Section 4.

THEOREM 6.1. *Let $n > 0$ and $r > 1$ be integers, and let $L(S) \subseteq (\{0, \ldots, r-1\}^n \cup \{\star\})^\omega$ be the set of all the encodings in base $r$ of the vectors belonging to the set $S \subseteq \mathbb{R}^n$. If the set $S$ belongs to $F_\sigma \cap G_\delta$ (with respect to Euclidean distance), then the language $L(S)$ belongs to $F_\sigma \cap G_\delta$ (with respect to $\omega$-word distance).*

PROOF. Not all infinite words over the alphabet $\Sigma = \{0, \ldots, r-1\}^n \cup \{\star\}$ encode a real vector. Actually, every arbitrary small neighborhood of a word encoding validly a vector of $\mathbb{R}^n$ contains words that are not valid encodings, namely the ones containing multiple occurrences of the separator "$\star$" that are far enough in the word.

Let $V$ be the set of all the valid encodings of vectors in base $r$. The mapping $V \to \mathbb{R}^n$ that transforms each word in $V$ into the real vector it encodes is continuous, i.e., for each open set (w.r.t. Euclidean distance) $S \subseteq \mathbb{R}^n$, the language $L(S)$ is open (w.r.t. $\omega$-word distance) in $V$. Equivalently, for each closed set $S \subseteq \mathbb{R}^n$, the language $L(S)$ is closed in $V$. Hence, for each $S \subseteq \mathbb{R}^n$ that belongs to $F_\sigma \cap G_\delta$, the language $L(S)$ belongs to $F_\sigma \cap G_\delta$ in $V$.

The language $V$ can be expressed as the intersection of an open set (the language of all the words starting with valid sign digits and containing at least one occurrence of the separator "$\star$") and of a closed set (the language of all the words containing less than two occurrences of the separator). Therefore, $V$ belongs to $F_\sigma \cap G_\delta$ in $\Sigma^\omega$, hence each language that is $F_\sigma \cap G_\delta$ in $V$ also belongs to $F_\sigma \cap G_\delta$ in $\Sigma^\omega$. Thus, for each $S \subseteq \mathbb{R}^n$ that is $F_\sigma \cap G_\delta$, the language $L(S)$ belongs to $F_\sigma \cap G_\delta$ in $\Sigma^\omega$. □

Knowing that the language of the encodings of any set definable in the theory $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ belongs to $F_\sigma \cap G_\delta$, we use the results of Section 5 to conclude the following.

THEOREM 6.2. *Every deterministic RVA representing a set definable in* $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ *is inherently weak.*

This property has the important consequence that the construction and the manipulation of RVA obtained from arithmetic formulas can be performed effectively by algorithms operating on weak deterministic automata. Precisely, to obtain an RVA for an arithmetic formula one can proceed as follows.

For equations and inequations, one uses the constructions given in [Boigelot et al. 1998] to build weak RVA. Computing the intersection, union, and Cartesian product of sets represented by RVA simply reduces to performing similar operations with the languages accepted by the underlying automata, which can be done by simple product constructions. These operations preserve the weak nature of the automata. To complement a weak RVA, one determinizes it using the breakpoint construction, which is guaranteed to yield an inherently weak automaton (Theorem 6.2) that is easily converted to a weak one. This deterministic weak RVA is then complemented by inverting the accepting or non-accepting status of each of its components, and then removing from its accepted language the words that do not encode validly a vector (which is done by means of an intersection operation).

An existential quantifier can be applied to a set represented as an RVA by using the construction detailed in Section 4. This operation does not affect the weak nature of the automaton, which can then be determinized by the breakpoint construction. The determinization algorithm has to produce an inherently weak RVA easily converted to a weak automaton.

Thus, in order to decide whether a formula of $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ is satisfiable, one simply builds an RVA representing its set of solutions, and then check whether this automaton accepts a nonempty language. This also makes it possible to check the inclusion or the equivalence of sets represented by RVA. The main result of this paper is that, at every point of the interpretation of a formula, the constructed automaton remains weak and thus only the simple breakpoint construction is needed as a determinization procedure.

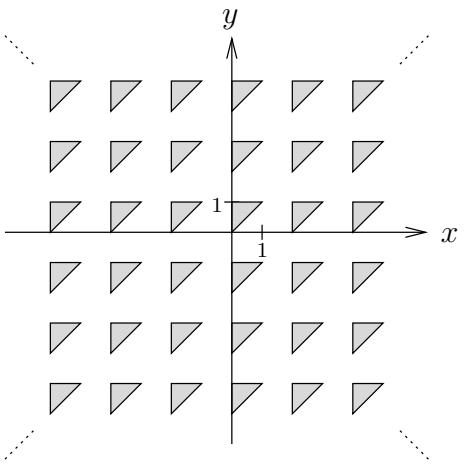


Fig. 3. Periodic tiling with triangles.

Finally, as weak deterministic automata can be efficiently minimized, each constructed automaton can be reduced down to a normal form. This is particularly useful from a practical point of view, since it speeds up the comparisons between sets by reducing them to structural tests on the automata, and since it prevents the representations from becoming unnecessarily large.

## 7. EXPERIMENTS

The decision procedure proposed in this paper has been implemented successfully in the LASH toolset, a package based on finite-state automata for representing infinite sets and exploring infinite state spaces [LASH ].

Various experiments have been achieved with the RVA package. For instance, it is possible to represent the set of Figure 3, which combines discrete and continuous features, by a weak RVA. Indeed, this set is defined by the following formula of the additive theory over the reals and integers :

$$\{(x_1, x_2) \in \mathbb{R}^2 \mid (\exists x_3, x_4 \in \mathbb{R})(\exists x_5, x_6 \in \mathbb{Z}) \\ (x_1 = x_3 + 2x_5 \wedge x_2 = x_4 + 2x_6 \wedge x_3 \geq 0 \wedge x4 \leq 1 \wedge x_4 \geq x_3)\}.$$

This set admits the compact minimal representation of Figure 4.

One might fear that the exponential worst-case complexity of the breakpoint determinization algorithm makes our decision procedure unusable. Experimental results however show that such a blow-up does not frequently occur in practical applications. As an illustration, Figure 5 shows the cost of projecting and then determinizing the finite-state representations of some periodic subsets of $\mathbb{R}^3$ obtained by combining linear constraints with arbitrary coefficients, and then by inducing a periodicity by means of an integer quantification. The interesting observation is that the finite-state representations have always less states after the projection than before, whereas an exponential blow-up could have been feared.

Another finite-state representation system, the NDD (*Number Decision Diagram*) [Wolper and Boigelot 1995; Boigelot 1998], is based on finite-word automata

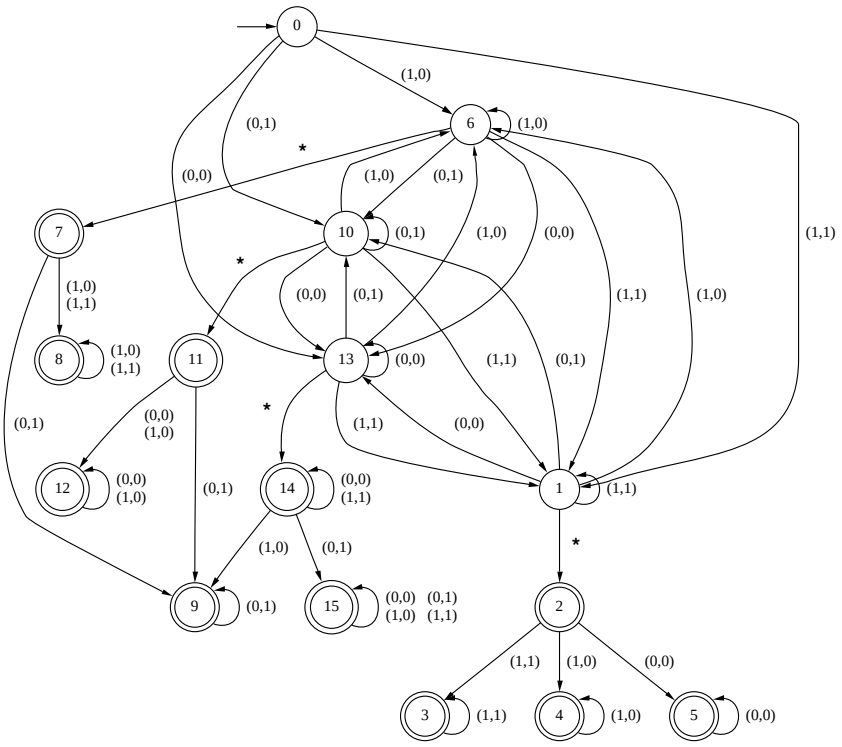


Fig. 4. Weak RVA representing the periodic tiling in binary.

and is able to represent the subsets of $\mathbb{Z}^n$ that can be expressed in an extension of the first-order theory $\langle \mathbb{Z}, +, \leq \rangle$. Figure 6 compares the size of weak RVA with that of NDD representing the same subsets of $\mathbb{Z}^3$ obtained by combining linear constraints with arbitrary coefficients. One notices that the behavior of RVA is very similar to that of NDD, that are reputed to behave quite well in practice [Wolper and Boigelot 2000].

These observations make one think that the pathological conditions that lead the breakpoint construction to blow-up are seldom met in practice.

## 8. CONCLUSIONS

A probably unusual aspect of this paper is that it does not introduce new algorithms, but rather shows that existing algorithms can be used in a situation where *a priori* they could not be expected to operate correctly. To put it in other words, the contribution is not the algorithm but the proof of its correctness.

The critical reader might be wondering if all this is really necessary. After all, algorithms for complementing Büchi automata exist, either through determinization [Safra 1988] or directly [Büchi 1962; Sistla et al. 1987; Kupferman and Vardi 1997; Klarlund 1991] and the more recent of these are even fairly simple and potentially implementable. There are no perfectly objective grounds on which to evaluate "simplicity" and "ease of implementation", but it is not difficult to convince oneself

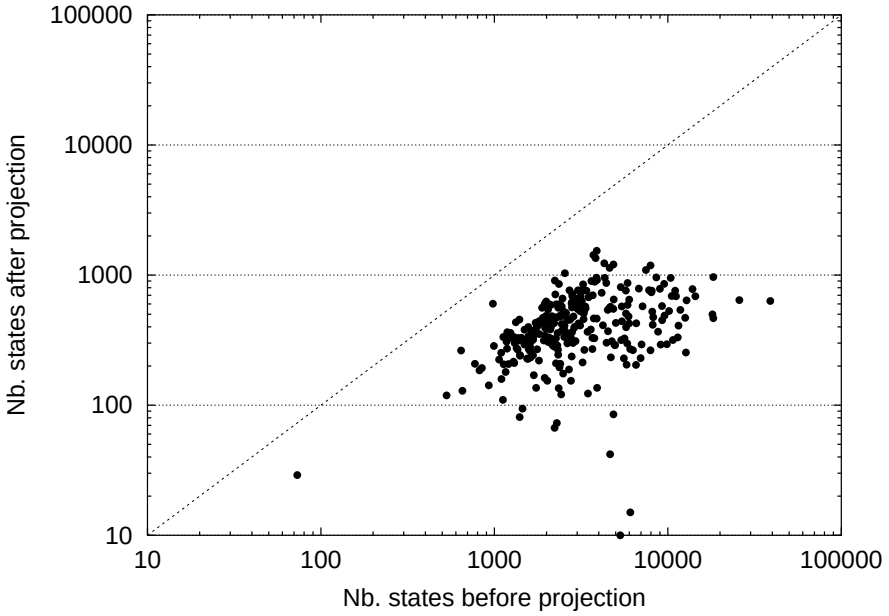


Fig. 5. The effect of projection-determinization on RVA.

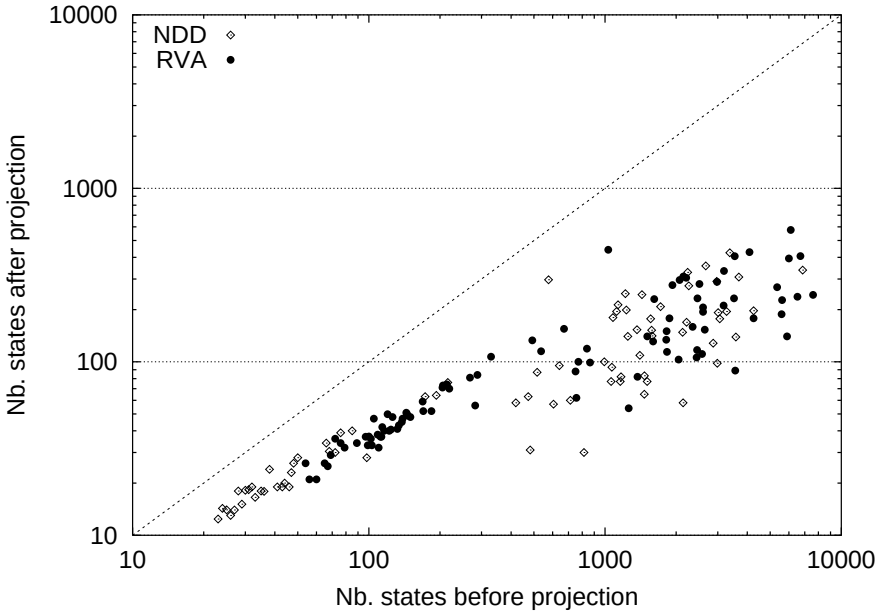


Fig. 6. The effect of projection-determinization on NDD and RVA.

that the breakpoint construction for determinizing weak automata is simpler than anything proposed for determinizing or complementing Büchi automata. Indeed, it is but one step of the probably simplest complementation procedure proposed so far, that of [Kupferman and Vardi 1997]. Furthermore, there is a complexity improvement from $2^{O(n \log n)}$ to $2^{O(n)}$, and being able to work with deterministic weak automata allows minimization [Löding 2001], which leads to a normal form. Those claims to simplicity and ease of implementation are substantiated by the experimental results.

Our implementation makes it possible to represent possibly non convex periodic sets containing both integers and reals, and to manipulate those sets using Boolean operations and quantification, and to check relations existing between them. To the best of our knowledge, doing this is beyond the scope of any other implemented tool. The potential application field of RVA is wide and range from symbolic analysis of linear hybrid systems [Alur et al. 1995] to temporal databases [Chomicki and Imieliński 1988; Kabanza et al. 1990].

## REFERENCES


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

Boigelot, B. 1998. Symbolic methods for exploring infinite state spaces. Ph.D. thesis, Université de Liège.

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

Boigelot, B., Jodogne, S., and Wolper, P. 2001. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. International Joint Conference on Automated Reasoning (IJCAR)*. Lecture Notes in Computer Science, vol. 2083. Springer-Verlag, Siena, Italy, 611–625.

Boigelot, B. and Latour, L. 2001. Counting the solutions of Presburger equations without enumerating them. In *Proc. International Conference on Implementations and Applications of Automata*. Lecture Notes in Computer Science, vol. 2494. Springer-Verlag, Pretoria, 40–51.

Boigelot, B., Rassart, S., and Wolper, P. 1998. On the expressiveness of real and integer arithmetic automata. In *Proc. 25th Colloq. on Automata, Programming, and Languages (ICALP)*. Lecture Notes in Computer Science, vol. 1443. Springer-Verlag, Aalborg, 152–163.

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

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

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

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

Chomicki, J. and Imieliński, T. 1988. Temporal deductive databases and infinite objects. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems*. ACM Press, Austin, Texas, 61–73.

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

Courcoubetis, C., Vardi, M. Y., Wolper, P., and Yannakakis, M. 1990. Memory efficient algorithms for the verification of temporal properties. In *Proc. 2nd Workshop on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 531. Springer-Verlag, Rutgers, 233–242.

Ferrante, J. and Rackoff, C. W. 1979. *The Computational Complexity of Logical Theories*. Lecture Notes in Mathematics, vol. 718. Springer-Verlag, Berlin-Heidelberg-New York.

Holzmann, G. J. 1997. The model checker SPIN. *IEEE Transactions on Software Engineering 23,* 5 (May), 279–295. Special Issue: Formal Methods in Software Practice.

Hopcroft, J. E. 1971. An $n \log n$ algorithm for minimizing states in a finite automaton. *Theory of Machines and Computation*, 189–196.

Jürgensen, H. and Staiger, L. 2001. Finite automata encoding geometric figures. In *Proc. 4th International Workshop on Implementing Automata (WIA'99), Revised Papers*, O. Boldt and H. Jürgensen, Eds. Lecture Notes in Computer Science, vol. 2214. Springer-Verlag, Potsdam, Germany, 101–108.

Kabanza, F., Stévenne, J.-M., and Wolper, P. 1990. Handling infinite temporal data. In *Proc. of the 9th ACM Symposium on Principles of Database Systems*. ACM Press, Nashville, Tennessee, 392–403.

Klarlund, N. 1991. Progress measures for complementation of $\omega$-automata with applications to temporal logic. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, San Juan, Puerto Rico, 358–367.

Kupferman, O. and Vardi, M. 1997. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symposium on Theory of Computing and Systems*. IEEE Computer Society Press, Ramat-Gan, Israel, 147–158.

Kupferman, O., Vardi, M. Y., and Wolper, P. 2000. An automata-theoretic approach to branching-time model checking. *Journal of the ACM 47,* 2 (March), 312–360.

Landweber, L. H. 1969. Decision problems for $\omega$-automata. *Mathematical Systems Theory 3,* 4, 376–384.

LASH. The Liège Automata-based Symbolic Handler (LASH). Available at : `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

Löding, C. 2001. Efficient minimization of deterministic weak $\omega-$automata. *Information Processing Letters 79,* 3, 105–109.

Maler, O. and Staiger, L. 1997. On syntactic congruences for $\omega$-languages. *Theoretical Computer Science 183,* 1, 93–112.

Miyano, S. and Hayashi, T. 1984. Alternating finite automata on $\omega$-words. *Theoretical Computer Science 32*, 321–330.

Muller, D. E., Saoudi, A., and Schupp, P. E. 1986. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Colloquium on Automata, Languages and Programming*. Springer-Verlag, Rennes, 275–283.

Rabin, M. O. 1969. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS 141*, 1–35.

Safra, S. 1988. On the complexity of omega-automata. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society Press, White Plains, 319–327.

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

Shiple, T. R., Kukula, J. H., and Ranjan, R. K. 1998. A comparison of Presburger engines for EFSM reachability. In *Proceedings of the 10th Intl. Conf. on Computer-Aided Verification*. Lecture Notes in Computer Science, vol. 1427. Springer-Verlag, Vancouver, 280–292.

Sistla, A. P., Vardi, M. Y., and Wolper, P. 1987. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science 49*, 217–237.

Staiger, L. 1983. Finite-state $\omega$-languages. *Journal of Computer and System Sciences 27,* 3, 434–448.

Staiger, L. and Wagner, K. 1974. Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regulärer folgenmengen. *Elektron. Informationsverarbeitung und Kybernetik EIK 10*, 379–392.

Thomas, W. 1990. Automata on infinite objects. In *Handbook of Theoretical Computer Science – Volume B: Formal Models and Semantics*, J. Van Leeuwen, Ed. Elsevier, Amsterdam, Chapter 4, 133–191.

Vardi, M. Y. and Wolper, P. 1986a. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*. IEEE Computer Society Press, Cambridge, 322–331.

Vardi, M. Y. and Wolper, P. 1986b. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science 32,* 2 (April), 183–221.

Vardi, M. Y. and Wolper, P. 1994. Reasoning about infinite computations. *Information and Computation 115,* 1 (November), 1–37.

Wagner, K. 1979. On *omega*-regular sets. *Information and Control 43,* 2 (November), 123–177.

Weispfenning, V. 1999. Mixed real-integer linear quantifier elimination. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*. ACM Press, Vancouver, 129–136.

Wolper, P. and Boigelot, B. 1995. An automata-theoretic approach to Presburger arithmetic constraints. In *Proc. Static Analysis Symposium*. Lecture Notes in Computer Science, vol. 983. Springer-Verlag, Glasgow, 21–32.

Wolper, P. and Boigelot, B. 2000. On the construction of automata from linear arithmetic constraints. In *Proc. 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Lecture Notes in Computer Science, vol. 1785. Springer-Verlag, Berlin, 1–19.

Received March 2003; revised February 2004; accepted February 2004