Centre Fédéré en Vérification

Technical Report number 2008.120

Computing Convex Hulls by Automata Iteration

François Cantin, Axel Legay, Pierre Wolper

This work was partially supported by a FRFC grant: 2.4530.02
and by the MoVES project. MoVES (P6/39) is part of the IAP-Phase VI Interuniversity
Attraction Poles Programme funded by the Belgian State, Belgian Science Policy

http://www.ulb.ac.be/di/ssd/cfv

Computing Convex Hulls by Automata Iteration

François Cantin[1], Axel Legay[2], and Pierre Wolper[1]

[1] Université de Liège, Institut Montefiore, Liège, Belgium
{cantin,pw}@montefiore.ulg.ac.be
[2] Carnegie Mellon University, Computer Science Department, Pittsburgh, PA
alegay@cs.cmu.edu

Abstract. This paper considers the problem of computing the real convex hull of a finite set of n-dimensional integer vectors. The starting point is a finite-automaton representation of the initial set of vectors. The proposed method consists in computing a sequence of automata representing approximations of the convex hull and using extrapolation techniques to compute the limit of this sequence. The convex hull can then be directly computed from this limit in the form of an automaton-based representation of the corresponding set of real vectors. The technique is quite general and has been implemented. Also, our result fits in a wider scheme whose objective is to improve the techniques for converting automata-based representation of constraints to formulas.

1 Introduction

Automata-based representations for sets of integer and real vectors have been a subject of growing interest in recent years [1,3,13,17,19]. While usually not optimal for specific problems, they provide much stronger generality and canonicity than other representations. For instance, in this context, combining real and integer constraints is very simple once the right framework has been set up [4]. The benefit of using automata-based representations for arithmetic sets could be even greater if one could, whenever appropriate, freely move between this and other representations such as explicit constraints. Going from constraints to automata has long been successfully studied [9,2,7], but going in the other direction is substantially more difficult. Nevertheless, it has been shown that it is possible [18] to construct constraint formulas from automata representing sets of integer vectors and that, under some restrictions, this can be done quite effectively [16].

One case that is not well handled though is that of finite sets of integer vectors. Indeed, imagine that a finite set of integers is represented by constraints and that an automaton representing this set is built from these. Since the set is finite, this acyclic automaton lacks the structure needed to construct the corresponding constraints. One is thus stuck with the automaton or with an enumerative representation of the set it defines, which is far from satisfactory. The work presented here was motivated by this problem with the idea of solving it along the following lines. The first step is to compute, as an automaton, a minimal dense set of

O.H. Ibarra and B. Ravikumar (Eds.): CIAA 2008, LNCS 5148, pp. 112–121, 2008.

real-vectors that contains the finite set of integers. On this automaton, techniques similar to those of [16,18] could then be applied to obtain constraints.

This paper proposes a solution for the first step in the form of a purely automata-based technique for computing the real convex hull (*i.e.* the convex hull over $\mathbb{R}^n$) of a finite automaton-represented finite set of integers. Note that, beyond the motivation outlined above, this is also a worthwhile challenge of independent interest in the area of automata-based representations. In simple terms, our approach proceeds as follows. We start with an automata-based representation of a finite set of integer vectors. We then repeatedly apply a transformation to this automaton that adds to the set the vectors that are mid-way between those it includes. This yields an infinite sequence of automata-represented sets. The limit of this infinite sequence is then computed as an automaton, using the extrapolation-based techniques of [5]. This limit is not quite the convex closure since we prove that it will only contain convex combinations of the initial vectors with coefficients that are multiples of a negative power of 2. This limit thus needs to be "completed" in order to obtain the convex hull and we show that this can be done by computing its topological closure. Bar a technical point due to the fact that some reals have two encodings in our framework, the computation of the topological closure is quite an easy step. This being done, the closure is obtained.

The extrapolation-based techniques of [5], which have so far only been applied in the context of "regular model checking" [8], are semi-algorithms that tackle the undecidable problem of computing the limit of an infinite sequence by extrapolating finite prefixes of the sequence. For the procedure above to work correctly, we thus depend on the result of the extrapolation being exact, which is not guaranteed a priori. Nevertheless, this can be checked as described in [5], but one interesting twist is that checking safety (enough is obtained) can be done much more easily (and just as correctly) after computing the topological closure. This is due to the fact that taking the topological closure yields an automaton that falls within an easier to handle class. Checking preciseness (nothing is added) with the techniques of [5] is probably not practical, but in the present situation one can exploit the properties of the extrapolation and make this check just as simple as the safety check.

Our approach has been implemented and the implementation has actually served as a guide to hone our results. The implementation has been tested and performs well, within the bounds allowed by the automata manipulations needed for the computation of the limit of the sequence of approximations. We certainly do not claim to outperform more traditional methods when they apply, our goal being to establish the basis of a different approach with interesting characteristics, performance gains not being part of our initial agenda. Also note that complexity analysis would not yield useful information since, at the heart of our approach, lies the extrapolation procedure which is only a semi-amgorithm.

Related Work. Computing convex hulls is of course a well studied problem of independent interest. There are quite a few known techniques for computing convex hulls of a set of vectors in a non automata-theoretic setting. Among these a long series of algorithms specialized to the 2D and 3D case and widely used

and studied in computational geometry. Algorithms for the general case (any dimensions) have also been studied [12]. All those algorithms, which are generally more efficient than an automata-based approach, require an enumeration of the set, which we avoid here. In [14], Finkel and Leroux show that the convex hull of a (possibly infinite) set of integer vector represented by an automaton is a computable polyhedron. The algorithm in [14] can be applied to infinite sets and is guaranteed to terminate. On the other hand, this algorithm, may require to enumerate the set represented by the automaton and is restricted to work in $\mathbb{Z}^n$.

Some proofs had to be omitted due to space constraints. A self-contained long version of this paper is available at [11].

2 Automata-Theoretic Background

2.1 Automata on Infinite Words

An infinite word (or ω-word) w over an alphabet Σ is a mapping $w : \mathbb{N} \to \Sigma$ from the natural numbers to Σ. The length-k prefix of an infinite word w, i.e. the finite-word $w(0), w(1), \ldots, w(k-1)$, will be denoted by $pref_k(w)$.

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, Σ is the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a *transition function* ($\delta : Q \times \Sigma \to Q$ if the automaton is deterministic), q_0 is the initial state, and F is a set of accepting states. A *run* π of a Büchi automaton $A = (Q, \Sigma, \delta, q_0, F)$ on an ω-word w is a mapping $\pi : \mathbb{N} \to Q$ such that $\pi(0) = q_0$ and 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). Let $inf(\pi)$ be the set of states that occur infinitely often in a run π. A run π is said to be accepting if $inf(\pi) \cap F \neq \emptyset$. An ω-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 ω-words it accepts.

We will also use the notion of *weak* automata [21]. Roughly speaking, a weak automaton is a Büchi automaton such that each of the strongly connected components of its graph contains either only accepting or only non-accepting states. Not all omega-regular languages can be accepted by weak deterministic Büchi automata, nor even by weak nondeterministic automata. However, there are algorithmic advantages to working with weak automata. Indeed, weak deterministic automata can be complemented simply by inverting their accepting and non-accepting states, while the complementation operation for Büchi automata requires intricate algorithms that not only are worst-case exponential, but are also hard to implement and optimize [24]. There exists a simple determinization procedure for weak automata [22], which produces Büchi automata that are deterministic, but not necessarily weak. However, we will be working in a context in which the obtained automata are always easily transformed into weak auatomata [4]. A final advantage of weak deterministic Büchi automata is that they admit a normal form, which is unique up to isomorphism [20].

2.2 Automata-Based Representations of Sets of Integers and Reals

In this section, we briefly introduce the representation of sets of integer and real vectors by finite automata. Details are only given for the case of real vectors, the case of integer vectors being a simplification of the former where automata on finite words replace automata on infinite words. A survey on this topic can be found in [7].

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. We call these respectively the *high* and *low* encodings and refer collectively to them as *dual* encodings.

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 $\mathbf{x} \in \mathbb{R}^n$ 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 $\{0, \ldots, r-1\}^n \cup \{\star\}$[1].

Real vectors being encoded by infinite words, a set of vectors can be represented by an infinite-word automaton accepting the corresponding encodings. 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 1. *Let $n > 0$ and $r > 1$ be integers. A base-r n-dimension* Real Vector Automaton (RVA) *[6] is a Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ over the alphabet $\Sigma = \{0, \ldots, r-1\}^n \cup \{\star\}$, such that (1) Every word accepted by $\mathcal{A}$ is an encoding in base r of a vector in $\mathbb{R}^n$, and (2) For every vector $\mathbf{x} \in \mathbb{R}^n$, $\mathcal{A}$ 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 that belong to its accepted language. In [4], it is shown that if the set represented by the

[1] In practice, one reads the bits of the vector components in a round robin way, which avoids an exponential-size alphabet. However, for presentation purposes, it is easier to view all same-position bits of the vector components as being read simultaneously.

RVA can be defined in the first-order theory of linear constraints, then this RVA can be transformed into an equivalent weak deterministic Büchi automata. If not explicitly mentioned, we assume that the RVAs we manipulate are minimal weak deterministic Büchi automata. Also, since our implementation works with a base 2 representation, we will present all our results in this context, knowing that they can be generalized to other bases.

3 Convex Hulls and Topological Concepts

We recall a few notations and definitions that are used throughout the paper.

Let $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ be respectively the sets of integers, rational, and reals, and let $\mathbb{Z}^n$, $\mathbb{Q}^n$, and $\mathbb{R}^n$ denote the usual n-dimensional Euclidean vector spaces. Vectors are written in boldface, e.g. $\mathbf{x}$, and scalars without emphasis, e.g. a. The *ith* component of a vector $\mathbf{x} \in \mathbb{R}^n$ is denoted by $\mathbf{x}[i]$. We say that a set $E \in \mathbb{R}^n$ is *convex* iff for each $\mathbf{x_1}, \mathbf{x_2} \in E$, we have $\{\alpha \mathbf{x_1} + (1-\alpha)\mathbf{x_2} \mid \alpha \in [0,1]\} \subseteq E$. We will also use the following usual definitions.

Definition 2. *Given a set $E \subseteq \mathbb{R}^n$, the* convex hull *of E is the set $Conv(E) \subseteq \mathbb{R}^n$ defined by*

$$Conv(E) = \{\mathbf{x} \mid \exists \mathbf{x}_1, \ldots, \mathbf{x}_k \in E \; \exists \lambda_1, \ldots, \lambda_k \in [0,1] \; \mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{x}_i \; \wedge \sum_{i=1}^{k} \lambda_i = 1\}$$

The *Euclidean distance* between two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, denoted by $|\mathbf{x} - \mathbf{x}'|$ is the real number $\sqrt{\sum_{i=1}^{n} (\mathbf{x}[i] - \mathbf{x}'[i])^2}$. The *open ball* centered in $\mathbf{x} \in \mathbb{R}^n$ with a radius $\epsilon > 0$ is the subset $B_{(\mathbf{x},\epsilon)} = \{\mathbf{x}' \mid |\mathbf{x} - \mathbf{x}'| < \epsilon\}$. A set $E \subseteq \mathbb{R}^n$ is said to be *open* if for any $\mathbf{x} \in E$ there exists $\epsilon > 0$ such that $B_{(\mathbf{x},\epsilon)} \subseteq E$. A *closed* set E is a subset of $\mathbb{R}^n$ such that $\mathbb{R}^n \setminus E$ is an open set. A *compact* set in $\mathbb{R}^n$ is a bounded and closed set. We use the concept of *topological closure* of a set.

Definition 3. *Given a set $E \subseteq \mathbb{R}^n$, the* topological closure *$TC(E)$ of E is the smallest closed set that contains E.*

When dealing with infinite words, we will be working with the topology on words induced by the distance defined by

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

where $common(w, w')$ denotes the longest common prefix of w and w'. Notice that, among words that validly encode vectors, words that are topologically close encode vectors that are close according to the Euclidean distance, the reverse also being true except for the cases where dual encodings can appear.

4 Computing Convex Hulls

In this section, we describe a technique to compute the convex hull over $\mathbb{R}^n$ of a *finite* set $E = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$ defined over $\mathbb{Z}^n$.

The technique proceeds by constructing a sequence of approximations of the convex hull by adding the vectors that are mid-way between those obtained so far. This is quite an obvious way to proceed, but in order to exploit it, we need to formalize its exact properties. We use the following definitions.

Definition 4. *The median sequence of E is the* infinite *sequence $E_0, E_1, E_2, \ldots$ such that (1) $E_0 = E$ and (2) $E_{i+1} = E_i \cup \{(\mathbf{x_1} + \mathbf{x_2})/2 \mid \mathbf{x_1}, \mathbf{x_2} \in E_i\}$ for each $i \in \mathbb{N}$.*

The *limit* of the median sequence of E, denoted by E^*, is defined by $\bigcup_{i=0}^{\infty} E_i$. It is easy to see that each vector $\mathbf{v}$ of E^* is also a vector of $Conv(E)$. However, E^* is not the complete convex hull, but can be characterized using the following definition.

Definition 5. *The 2-chopped convex hull of a finite subset E=$\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$ of $\mathbb{Z}^n$ is the* maximal *subset $Conv_{2^*}(E)$ of $Conv(E)$, where for each $\mathbf{v} \in Conv_{2^*}(E)$, $\mathbf{v} = \sum_{i=1}^{k} \lambda_i \mathbf{x_i}$ with $\lambda_i \in [0,1]$, $\sum_{i=1}^{k} \lambda_i = 1$, and $\lambda_i = \frac{k_i}{2^{m_i}}$ for $k_i, m_i \in \mathbb{N}$ and $i \in [1, \ldots, k]$.*

Theorem 1. *For any finite subset $E = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$ of $\mathbb{Z}^n$, the limit of its median sequence and its 2-chopped convex hull coincide, i.e $E^* = Conv_{2^*}(E)$.*

Even though the 2-chopped convex hull of a set E is not quite its real convex hull, it contains vectors that are arbitrarily close to any element of the full convex closure. In fact, the convex hull of E is included in the topological closure of its 2-chopped hull. The following theorem states that these two sets coincide.

Theorem 2. *For any finite subset $E = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$ of $\mathbb{Z}^n$, we have that $TC(Conv_{2^*}(E)) = Conv(E)$.*

Computing the real convex hull of a finite set of integer vectors can thus be reduced to compute the topological closure of the limit of its median sequence. We now investigate how to compute $Conv_{2^*}(E)$ and $TC(E)$ for a set E described by an *RVA*.

5 Algorithmic Issues

We consider a finite subset $E = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$ of $\mathbb{Z}^n$ that is represented by a (weak deterministic) RVA A_E. Our goal is to compute an RVA that represents the convex hull over $\mathbb{R}^n$ of E. According to the results in Section 4, this can be done by computing an RVA A_{E^*} representing the limit E^* of the median sequence of E, and then computing an RVA representing the topological closure of E^*. We now show how these two problems can be tackled by automata-based semi-algorithms.

5.1 Computing an RVA for the 2-Chopped Hull

Computing the elements of the median sequence. We notice that since E is finite and represented by a weak deterministic RVA, each element in its median sequence can also be represented in the same way (see [11] for details).

Computing the limit of the median sequence. Computing A_{E^*} amounts to computing the limit of an infinite sequence of weak deterministic automata. To finitely compute this limit, we obviously need some form of "speed-up" technique. We will use the extrapolation-based technique proposed in [5]. A rough description of the technique is as follows. The technique proceeds by comparing successive automata in a prefix of the sequence, trying to identify the difference between these in the form of an "increment", and extrapolating the repetition of this increment by adding loops to the last automaton of the prefix. If the extrapolation is *correct*, then the limit is computed, else, one has to lengthen the prefix and restart the extrapolation process. Checking correctness of the extrapolation is a non trivial procedure whose description is, for technical reasons, postponed to Section 5.3. The technique has been implemented in a tool called T(O)RMC [23]. The tool relies on the LASH package [15] for automata manipulation procedures, but implements the specific algorithms given in [5]. There is no guarantee that T(O)RMC will produce a result since the general problem of computing the limit of a sequence of automata is undecidable.

It is worth mentioning that the automata produced by T(O)RMC are weak, but not necessarily deterministic [5]. Furthermore, if one tries to determinize these automata, one might end up combining accepting and non accepting connected components, which leads to an automaton that is not weak. This situation actually occurred systematically in our experiment, which is not surprising since the 2-chopped convex hull of a set of integer vectors is not definable in $\langle \mathbb{R}, +, \leq, Z \rangle$ and thus falls outside the guaranteed reach of weak deterministic automata given in [4].

5.2 Computing the Topological Closure of an RVA-Represented Set

In this section, we explicitely consider RVAs that may not be weak deterministic. Consider a set $E \subseteq \mathbb{R}^n$ represented by an RVA A_E. Our goal is to compute an RVA $A_{TC(E)}$ that represents the topological closure of E. The intuition behind the computation is that we need to add to the language accepted by A_E, all words that are arbitrarily close to words of this language. This is fairly straightforward to do since we only need to add words that have arbitrarily long common prefixes with accepted words. A simple step to do this is to make accepting all states of the fractional part of the automaton. Of course, this will compute the topological closure within the topology on infinite words, but this also almost computes the vector Euclidean topological closure as it is shown by the following result.

Theorem 3. *Let A_E be a RVA representing a vector set E. Let $\overline{A}_E$ be A_E with all states of its fractional part made accepting, and let $W(\mathbf{v}, n)$ be the set of all the encodings of a vector $\mathbf{v} \in \mathbb{R}^n$. For each vector $\mathbf{v} \in \mathbb{R}^n$, $W(\mathbf{v}, n) \cap L(\overline{A}_E) \neq \emptyset$ if and only if $\mathbf{v} \in TC(E)$.*

Theorem 3 guarantees that $\overline{A}_E$ contains at least one encoding for each vector in $TC(E)$. However the automaton $\overline{A}_E$ is not necessarily $A_{TC(E)}$. Indeed, there is no guarantee that $\overline{A}_E$ will contain *all* the encodings of each vector included in the

topological closure. We thus need an extra step that adds all missing encodings. To do this, we use the fact that an automaton that recognizes words that are dual encodings of the same numbers can be built with simple automata-based operations (see [10] for the detailed algorithm).

5.3 Correctness Criterion

After having constructed the extrapolation A_E^* of a finite sequence $A_E^{i_1}$, $A_E^{i_2}, \ldots,$ $A_E^{i_l}$ of automata representing elements in the median sequence of a set E, it remains to check whether it accurately corresponds to what we really intend to compute, i.e., A_{E^*}. This is done by first checking that the extrapolation is *safe*, in the sense that it captures all words accepted by A_{E^*} ($L(A_{E^*}) \subseteq L(A_E^*)$), and then checking that it is *precise*, i.e. that it accepts no more words than A_{E^*} ($L(A_E^*) \subseteq L(A_{E^*})$). To lighten the presentation, we will often use the notations and operations defined for sets of vectors directly on the automata that represent them.

Safety. We first investigate how to check whether A_E^* is safe. The idea is simply to perform one more mid-point adding step on A_E^* and to check that this does not change the accepted language. Given a set E, let $C_2(E)$ be the set $\{\mathbf{y} \mid \mathbf{y} = (\mathbf{x_1} + \mathbf{x_2})/2 \mid \mathbf{x_1}, \mathbf{x_2} \in E\}$. We have the following theorem.

Theorem 4. *Let A_E^* and A_{E^*} be respectively the extrapolation of a median automata sequence for a set E and a representation of the actual limit of this sequence. We have that, if $L(C_2(A_E^*)) \subseteq L(A_E^*)$, then $L(A_{E^*}) \subseteq L(A_E^*)$.*

The required computation step is thus to check that $L(C_2(A_E^*)) \subseteq L(A_E^*)$. This is simple except for the fact that, the result of the extrapolation is representable by an automaton which is weak but not necessarily deterministic (see Section 5.1), and hence testing inclusion requires to complement a Büchi automaton. The problem can be solved by first applying the topological closure step to A_E^* and then performing the safety check given by Lemma 4.

It is easy to see that doing this has no impact on the result of the test. However it has an impact on its efficiency since the strongly connected component added by T(O)RM are made uniformly accepting status by the procedure that computes the topological closure. This ensures that we only need to complement weak deterministic automata.

Preciseness. Checking preciseness could be performed with the techniques proposed in [5]. However, this solution (which involves counter automata) is computationally demanding and not really practical. In the present situation, one can however propose a much more efficient scheme that exploits the properties of the extrapolation. Due to space limitation, we only sketch the procedure here, details can be found in [11].

Definition 6. *Let $E \in \mathbb{R}$ be a convex set. The set of extreme points of E, denoted $S(E)$, is defined as* $\{\mathbf{x} \in E \mid (\neg\exists(\mathbf{x_1}, \mathbf{x_2}) \in E)(\mathbf{x_1} \neq \mathbf{x_2} \wedge \mathbf{x} = (\mathbf{x_1} + \mathbf{x_2})/2)\}$.

By extension we will also use the notation $S(A)$ on automata representing vector sets. We now present our preciseness check. Instead of checking whether $L(A_E^*) \subseteq L(A_{E^*})$, we check $L(TC(A_E^*)) \subseteq L(Conv(A_E))$. This is enough to ensure that we do not compute an overapproximation of the hull.

Theorem 5. *Let A_E^* be an RVA that represents a safe extrapolation of the limit of the median sequence of a finite set of integer vectors represented by the RVA A_E. If $L(S(TC(A_E^*))) \subseteq L(A_E)$, then $L(TC(A_E^*)) \subseteq L(Conv(A_E))$.*

In summary, to check the preciseness of an RVA A_E^* that represents a safe extrapolation of the limit of the median sequence of a finite set $E \subseteq \mathbb{Z}^n$, we first compute an RVA $TC(A_E^*)$ for the topological closure of the set represented by A_E^*. We then compute an automaton for $S(TC(A_E^*))$, which is easily done by computing the difference between $TC(A_E^*)$ and $C_2(TC(A_E^*))$. Finally, one checks whether the language of the resulting automaton is included in that of A_E. Again, all complementation operations are only applied to weak deterministic Büchi automata.

Infinite Sets. It is worth mentioning that our results do not extend as such to the computation of the real convex hull of an infinite set of integer vectors. Indeed, by relying on the computation of a topological closure, our methodology produces convex hulls which are closed sets. However there are infinite sets of integer vectors whose convex hull is not closed.

6 A Brief Note on the Experimental Results

The approach presented in this paper has been tested on several examples using a prototype implementation that relies on T(O)RMC. We computed the convex hull over $\mathbb{R}^n$ of finite convex sets in $\mathbb{Z}^n$, of the difference/union of finite convex sets in $\mathbb{Z}^n$, and of arbitrary finite sets of points in $\mathbb{Z}^n$. Some experiments that validate the fact that our approach performs well for sets for which the representation by automata remains manageable are reported in [11].

References

1. Bartzis, C., Bultan, T.: Construction of efficient bdds for bounded arithmetic constraints. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 394–408. Springer, Heidelberg (2003)
2. Boigelot, B.: Symbolic Methods for Exploring Infinite State Spaces. Collection des publications de la Faculté des Sciences Appliquées, Liège (1999)
3. Boigelot, B., Herbreteau, F.: The power of hybrid acceleration. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 438–451. Springer, Heidelberg (2006)
4. Boigelot, B., Jodogne, S., Wolper, P.: An effective decision procedure for linear arithmetic over the integers and reals. ACM Transactions on Computational Logic 6(3), 614–633 (2005)

5. Boigelot, B., Legay, A., Wolper, P.: Omega-regular model checking. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 561–575. Springer, Heidelberg (2004)
6. Boigelot, B., Rassart, S., Wolper, P.: On the expressiveness of real and integer arithmetic automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 152–163. Springer, Heidelberg (1998)
7. Boigelot, B., Wolper, P.: Representing arithmetic constraints with finite automata: An overview. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 1–19. Springer, Heidelberg (2002)
8. Bouajjani, A., Jonsson, B., Nilsson, M., Touili, T.: Regular model checking. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 403–418. Springer, Heidelberg (2000)
9. Boudet, A., Comon, H.: Diophantine equations, presburger arithmetic and finite automata. In: Proc of ICALP. LNCS, vol. 1159, pp. 30–43. Springer, Heidelberg (1996)
10. Cantin, F.: Techniques d'extrapolation d'automates: Application au calcul de la fermeture convexe. Master's thesis, University of Liège, Belgium (2007)
11. Cantin, F., Legay, A., Wolper, P.: Computing convex hulls by automata iteration. Technical report, University of Liège (2008), `http://www.montefiore.ulg.ac.be/~legay/papers/ciaa08rep.pdf`
12. Chazelle, B.: An optimal convex hull algorithm in any fixed dimension. Discrete & Computational Geometry 10, 377–409 (1993)
13. Eisinger, J., Klaedtke, F.: Don't care words with an application to the automata-based approach for real addition. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 67–80. Springer, Heidelberg (2006)
14. Finkel, A., Leroux, J.: The convex hull of a regular set of integer vectors is polyhedral and effectively computable. IPL (96-1), 30–35 (2005)
15. The Liège Automata-based Symbolic Handler (LASH), `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`
16. Latour, L.: From automata to formulas: Convex integer polyhedra. In: Proc. of LICS, pp. 120–129. IEEE Computer Society, Los Alamitos (2004)
17. Leroux, J.: Algorithmique de la vérification des systèmes à compteurs. Approximation et accélération. Implémentation de l'outil FAST. PhD thesis, Cachan (2004)
18. Leroux, J.: A polynomial time presburger criterion and synthesis for number decision diagrams. In: Proc of LICS, pp. 147–156. IEEE Computer Society, Los Alamitos (2005)
19. Leroux, J., Sutre, G.: Flat counter automata almost everywhere! In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 489–503. Springer, Heidelberg (2005)
20. Löding, C.: Efficient minimization of deterministic weak automata. IPL (79-3), 105–109 (2001)
21. Muller, D.E., Saoudi, A., Schupp, P.E.: Alternating automata, the weak monadic theory of the tree and its complexity. In: Proc of ICALP, Rennes, pp. 275–283. Springer, Heidelberg (1986)
22. Safra, S.: Exponential determinization for ω-automata with strong-fairness acceptance condition. In: Proc. of POPL, Victoria (May 1992)
23. The T(O)RMC toolset, `http://www.montefiore.ulg.ac.be/~legay/TORMC/index-tormc.html`
24. Vardi, M.: The büchi complementation saga. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 12–22. Springer, Heidelberg (2007)