Automata-based Symbolic Representations of Polyhedra[⋆]

Bernard Boigelot, Julien Brusten, and Jean-François Degbomont

Institut Montefiore, B28,
Université de Liège,
B-4000 Liège, Belgium
{boigelot,brusten,degbomont}@montefiore.ulg.ac.be

Abstract. This work describes a data structure, the *Implicit Real-Vector Automaton (IRVA)*, suited for representing symbolically polyhedra, i.e., regions of n-dimensional space defined by finite Boolean combinations of linear inequalities. IRVA can represent exactly arbitrary convex and non-convex polyhedra, including features such as open and closed boundaries, unconnected parts, and non-manifold components. In addition, they provide efficient procedures for deciding whether a point belongs to a given polyhedron, and determining the polyhedron component (vertex, edge, facet, ...) that contains a point. An advantage of IRVA is that they can easily be minimized into a canonical form, which leads to a simple and efficient test for equality between represented polyhedra. We also develop an algorithm for computing Boolean combinations of polyhedra represented by IRVA.

1 Introduction

The problem of designing a good data structure for representing and handling *polyhedra*, i.e., regions of n-dimensional space delimited by finitely many planar boundaries, has important applications in several areas of computer science. The precise class of polyhedra that needs to be covered and the range of necessary manipulation operations actually differ according to the application field.

Our historical motivation for studying this problem is related to *computer-aided verification*, where polyhedra are used for representing sets of system configurations that are manipulated during symbolic state-space exploration of hybrid automata [11, 6]. In this setting, polyhedra are defined as finite Boolean combinations of strict and/or non-strict linear inequalities. The operations that need to be performed on polyhedra include unions, intersections, projections, and linear transformations (for applying the transition relation of the system under study), and tests of inclusion or equality (for detecting that a fixed point has been reached).

[⋆] This work is partially supported by the *Interuniversity Attraction Poles* program *MoVES* of the Belgian Federal Science Policy Office, and by the grants 2.4607.08 and 2.4545.11 of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

The efficient manipulation of polyhedra is also essential to *computed-aided design*, where they yield convenient approximations of the shape of arbitrary objects. In this framework, the spatial dimension n is usually limited to 2 or 3, and polyhedra are often *regularized*, meaning that they are made equal to the topological closure of their interior. Intuitively, the regularization operation gets rid of polyhedron features that are considered to be negligible and problematic, such as isolated points, dangling facets or edges, and open boundaries. The operations applied on polyhedra include Boolean combinations in order to construct complex objects from elementary building blocks, geometric transformations and measurements, two and three-dimensional visualization, checking whether a point belongs to a given polyhedron (the *point location* problem), and computing the polyhedron component (vertex, edge, facet, ...) that contains a point (the *point classification* problem).

Finally, as last examples of applications, polyhedra are also used in *optimization theory* and *constraint programming* [16] for specifying systems of constraints. In those applications, the spatial dimension n corresponds to the number of variables implicated in the constraints, which is usually large, and the considered polyhedra are often convex, meaning that they can be expressed as finite conjunctions of linear inequalities. Typical problems there consist in searching inside a polyhedron for a point that maximizes a given objective function, and deciding whether a polyhedron is empty or not.

In this work, we consider the polyhedra defined as a finite Boolean combination of open and closed linear constraints, which are also known as *Nef polyhedra* [1, 10]. This class covers polyhedra with combinations of open and closed boundaries, non-convex or unconnected parts, and non-manifold components. Our aim is to obtain a data structure that is able to represent exactly those polyhedra, and for which efficient algorithms can be derived for computing their Boolean combinations, checking inclusion, equality, and emptiness, and solving the point location and point classification problems.

Several approaches have been proposed for tackling those problems. A first possibility is to represent a polyhedron by a logical formula expressed in the quantifier-free fragment of linear arithmetic, for which powerful solvers are available [9]. This solution has the advantage of being able to deal with large spatial dimensions, but does not provide efficient algorithms for checking set equality or inclusion, or for simplifying the representation of a polyhedron obtained as the result of complex operations. In the restricted case of *convex polyhedra*, formula-based representations can be augmented with redundant structural information (the so-called *vertices*, *extremal rays* and *lines* of polyhedra), which substantially simplifies comparison operations [13]. In computer-aided design applications, the main approaches consist in representing a solid object as an explicit Boolean combination of elementary primitives (*Constructive Solid Geometry (CSG)*) [15], or by a geometrical description of their boundary (*Boundary representations (B-rep)*). CSG methods can be generalized to non-polyhedral primitives such as spheres, toruses and shapes bounded by polynomial surfaces. They provide direct implementations of Boolean operators and an easy solution to the point location problem. However, they are usually restricted to regularized shapes, and do not make it possible to check easily inclusion or equality of objects. On the other hand, B-rep techniques are able to represent accurately features such as open and/or closed boundaries and non-manifold components, but do not admit efficient algorithms for applying Boolean operators or solving the point location problem. These drawbacks are addressed by *Selective Nef Complexes (SNC)*, which combine a geometrical description of the vertices, edges and facets that compose a polyhedron with a topological representation of the incidence relation between them. SNC data structures have the same expressive power as B-rep ones, but can be combined by means of Boolean operators. Algorithms have also been developed for solving the point location and point classification problems over these structures [10] in the case of small spatial dimensions ($n = 2$ or $n = 3$).

A different approach is to represent polyhedra using *Real-Vector Automata (RVA)*, which are a particular form of infinite-word automata recognizing encodings of points in $\mathbb{R}^n$ [2, 5]. It has been established that RVA are expressive enough for representing arbitrary polyhedra. The advantages of automata-based representations are that computing Boolean combinations of polyhedra reduce to carrying out similar operations on the languages accepted by the automata, for which simple algorithms are known. Furthermore, RVA can easily be minimized into a canonical form [14]. This leads to efficient comparison operations between represented sets, and allows to simplify the results of long chains of operations. RVA also provide a very efficient algorithm for solving the point location and classification problems. The main drawback of RVA is their size that can grow linearly with the coefficients of linear constraints, which makes those symbolic representations unmanageable in some applications. This drawback is alleviated by *Implicit Real-Vector Automata (IRVA)*, which intuitively operate on similar principles as RVA, but replace some of their unnecessarily large internal structures by more concise algebraic objects [3]. Interestingly enough, it has been shown that the RVA structures replaced in IRVA closely match the internal components of SNC representations of polyhedra, and that their reachability properties represent the incidence relation between them. The advantages of IRVA over SNC representations are threefold. First, they inherit the canonicity properties of RVA, which reduces equality testing between polyhedra to a simple isomorphism check. Second, like RVA, they admit very efficient algorithms for the point location and classification problems, which proceed by following a single path in a decision structure. Finally, IRVA are applicable to any spatial dimension n.

2 Basic Notions and Notations

Let $n \in \mathbb{N}$ be a dimension. A *linear constraint* over points $\boldsymbol{x} \in \mathbb{R}^n$ is a constraint of the form $\boldsymbol{a}.\boldsymbol{x}\#b$, with $a \in \mathbb{Z}^n$, $b \in \mathbb{Z}$, and $\# \in \{<, \leq, =, \geq, >\}$. A finite Boolean combination of such constraints defines a *polyhedron*. A polyhedron Π is *convex* if for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Pi$ and $\lambda \in [0, 1]$, one has $\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \Pi$, i.e., the line segment joining $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is a subset of Π. Every convex polyhedron can be expressed as a finite conjunction of linear constraints. A polyhedron defined by a finite conjunction of linear equalities, i.e., constraints of the form $\boldsymbol{a}.\boldsymbol{x} = b$, is an *affine space*. An affine space that contains $\mathbf{0}$ is a *vector space*. The *dimension* $\dim(S) \leq n$ of an affine or vector space $S \subseteq \mathbb{R}^n$ is the largest number of linearly independent vectors it contains. A set $S \subseteq \mathbb{R}^n$ is *conical* with respect to the *apex* $\boldsymbol{v} \in \mathbb{R}^n$ if for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}_{>0}$, one has $\boldsymbol{x} \in S$ iff $\boldsymbol{v} + \lambda(\boldsymbol{x} - \boldsymbol{v}) \in S$, which intuitively states that the set S is not affected by a scaling transformation centered on the point $\boldsymbol{v}$. A polyhedron that is conical is a *pyramid*. The set of apexes of a pyramid always forms an affine space [1].

3 Polyhedra

3.1 Topological Components

The main idea behind the data structure discussed in this work is to exploit the specific topological properties of polyhedra. It has been observed that the structure of a polyhedron $\Pi \subseteq \mathbb{R}^n$ is pyramidal in arbitrarily small neighborhoods of any point $\boldsymbol{v} \in \mathbb{R}^n$ [1, 4]. This property can be formalized as follows.

Definition 1. *Let $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$ and $\varepsilon \in \mathbb{R}_{>0}$. The* cubic closed neighborhood *of size ε of $\boldsymbol{v}$ is the set*

$$N_\varepsilon(\boldsymbol{v}) = [v_1 - \frac{\varepsilon}{2}, v_1 + \frac{\varepsilon}{2}] \times [v_2 - \frac{\varepsilon}{2}, v_2 + \frac{\varepsilon}{2}] \times \cdots \times [v_n - \frac{\varepsilon}{2}, v_n + \frac{\varepsilon}{2}].$$

Theorem 2. *Let $\Pi \subseteq \mathbb{R}^n$ be a polyhedron. For every point $\boldsymbol{v} \in \mathbb{R}^n$, there exists $\varepsilon \in \mathbb{R}_{>0}$ such that Π coincides over $N_\varepsilon(\boldsymbol{v})$ with a pyramid of apex $\boldsymbol{v}$.*

Note that if Π coincides with a pyramid P in the neighborhood $N_\varepsilon(\boldsymbol{v})$ of a point $\boldsymbol{v}$, then the same pyramid P also describes its structure in all neighborhoods $N_{\varepsilon'}(\boldsymbol{v})$ such that $0 < \varepsilon' \leq \varepsilon$, since a pyramid is invariant by scaling transformations. It has additionally been established that a finite number of distinct pyramids suffices for describing the structure of Π in the neighborhood of all points in $\mathbb{R}^n$ [1, 4]. We have the following definition and theorem.

Definition 3. *Let $\Pi \subseteq \mathbb{R}^n$ be a polyhedron, and $\boldsymbol{v} \in \mathbb{R}^n$ be an arbitrary point. The* local pyramid *of Π with respect to $\boldsymbol{v}$ is the pyramid $P_\Pi(\boldsymbol{v})$ that coincides with Π over sufficiently small neighborhoods $N_\varepsilon(\boldsymbol{v})$ of $\boldsymbol{v}$.*

Theorem 4. *For each polyhedron $\Pi \subseteq \mathbb{R}^n$, the set $\{P_\Pi(\boldsymbol{v}) \mid \boldsymbol{v} \in \mathbb{R}^n\}$ is finite.*

This theorem states that a polyhedron Π partitions the space $\mathbb{R}^n$ into a finite number of equivalence classes, each described by a local pyramid. We call the equivalence class that contains a point $\boldsymbol{v}$ the *polyhedral component*, or more simply *component*, of Π associated to $\boldsymbol{v}$. Such a component is thus uniquely characterized by the local pyramid $P_\Pi(\boldsymbol{v})$. The set of apexes of this pyramid forms the *characteristic affine space* of the component, denoted $\text{aff}(C)$ for a component C, and the dimension of this space defines the *dimension* of the component, denoted $\dim(C)$. Intuitively, the dimension of a component characterizes the number of degrees of freedom among its points. Components of dimension 0, 1, 2 thus correspond to the classical notions of *vertices*, *edges* and *facets* of polyhedra. An illustration is given in Figure 1.

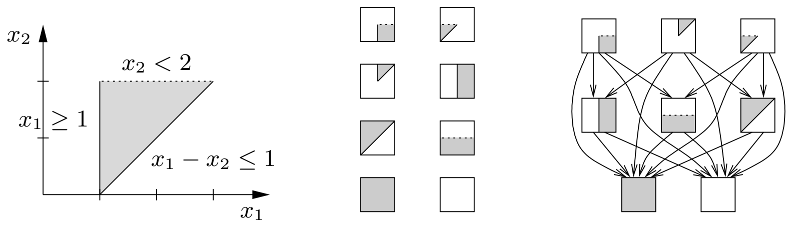

Fig. 1. Example of (a) polyhedron, (b) components, and (c) incidence relation

3.2 Incidence Relation

The components of a polyhedron are connected by an *incidence relation*. We have the following definition and theorem.

Definition 5. *A component C_2 of a polyhedron Π is* incident *to a component C_1, which is denoted $C_1 \preceq C_2$, if for every point $\boldsymbol{v} \in \mathbb{R}^n$ that belongs to C_1, there exist points that are arbitrarily close to $\boldsymbol{v}$ and that belong to C_2.*

Theorem 6. *For every polyhedron, the incidence relation $\preceq$ is a partial order over its components. This relation is such that $C_1 \preceq C_2$ implies $\mathrm{aff}(C_1) \subseteq \mathrm{aff}(C_2)$ (and thus $\dim(C_1) \leq \dim(C_2)$).*

3.3 Extension to Polyhedral Partitions

The notions of polyhedral component and incidence relation can be extended to more general structures than polyhedra. We define a *polyhedral partition* as a partition $\Pi = \{\Pi_1, \Pi_2, \ldots, \Pi_m\}$ of $\mathbb{R}^n$, with $m > 0$, such that each Π_i is a polyhedron. Such a partition can alternatively be specified as a *color function* $\Pi : \mathbb{R}^n \to \{1, 2, \ldots, m\}$ that maps every point $\boldsymbol{v} \in \mathbb{R}^n$ onto its index $\Pi(\boldsymbol{v})$ in the partition, which can be seen as a color assigned by the polyhedral partition out of m distinct possibilities. Polyhedral partitions are especially useful in the framework of the point classification problem, where the core issue is to describe such a partition by a data structure from which one can easily compute the index, or color, of arbitrary points. A polyhedron can be seen as a particular instance of a polyhedral partition, limited to two colors corresponding to the points that respectively belong and do not belong to the polyhedron. In the rest of this paper, we will thus indifferently use polyhedra and polyhedral partitions.

The definition of pyramids readily adapts to polyhedral partitions: A polyhedral partition Π over $\mathbb{R}^n$ is pyramidal with respect to the apex $\boldsymbol{v} \in \mathbb{R}^n$ if it is such that for every $\boldsymbol{x} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}_{>0}$, we have $\Pi(\boldsymbol{x}) = \Pi(\boldsymbol{v} + \lambda(\boldsymbol{x} - \boldsymbol{v}))$. A pyramidal partition of apex $\boldsymbol{v}$ is thus one that classifies the points of $\mathbb{R}^n$ on the sole basis of the direction in which they are seen from $\boldsymbol{v}$, which means that such a partition is not affected by scaling transformations centered on $\boldsymbol{v}$. From this extension of the concept of pyramid, one straightforwardly generalizes to polyhedral partitions the notions of local pyramid, components, and incidence relation.

4 Towards a Better Data Structure for Polyhedra

4.1 Real-Vector Automata

Selective Nef Complexes (SNC) are data structures that combine descriptions of the components of a polyhedron and of the incidence relation between them [10]. In this section, we study another class of representations, the Real-Vector Automata (RVA), and show that, even though RVA are based on different ideas from SNC representations, both data structures share some common principles of operation. Our aim will then be to define symbolic representations that are as concise as SNC, but inherit from RVA their canonicity property, as well as very efficient algorithms for solving the point location and point classification problems.

RVA are finite-state machines recognizing the coordinates of points, encoded into words [2, 5]. They depend on the choice of a *numeration base* $r \in \mathbb{N}_{>1}$, which provides an alphabet of digits $\Sigma_r = \{0, 1, \ldots, r - 1\}$, augmented with a distinguished symbol $\star$ used for separating the integer from the fractional part of encodings. In a given base r, a number $z \in \mathbb{R}_{\geq 0}$ is *encoded* by infinite words $a_{p-1}a_{p-2} \ldots a_0 \star a_{-1}a_{-2} \ldots$ such that $p > 0$, $a_i \in \Sigma_r$ for all $i < p$, and $z = \sum_{i<p} a_i r^i$. This scheme is extended to signed numbers by encoding negative numbers by their *r's complement*, which amounts to representing a number $z \in \mathbb{R}_{<0}$ by the encoding of $z + r^p$, where p is the length of its integer part. The value of p is not fixed, but has to satisfy the constraint $-r^{p-1} \leq z \leq r^{p-1}$. The integer part of an encoding can be increased at will, by repeating its leading digit (which is equal to 0 for positive and to $r - 1$ for negative numbers), hence every number admits infinitely many encodings.

Points in $\mathbb{R}^n$ are encoded by combining encodings of their components, which can always be chosen such that their integer parts share the same length. By reading those component encodings synchronously, one symbol at a time, one obtains a point encoding that takes the form of an infinite word over the alphabet $\{0, 1, \ldots, r-1\}^n \cup \{\star\}$ (since the separator symbol is read at the same time in all component encodings, it can be denoted by a single symbol). The exponential size of the alphabet can be avoided by *serializing* the encodings, which amounts to replacing each symbol $(d_1, d_2, \ldots, d_n) \in \{0, 1, \ldots, r - 1\}^n$ by the subword $d_1 d_2 \ldots d_n$ expressed over Σ_r.

Given a set $S \subseteq \mathbb{R}^n$ and a base $r \in \mathbb{N}_{>1}$, the base-r encodings of the points in S form a language. An infinite-word finite-state automaton that accepts this language is called a *Real-Vector Automaton (RVA)* representing S. It is known that RVA are expressive enough for representing all the sets that are definable in $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$ [2], i.e., the first-order additive theory of real and integer numbers, which covers our definition of polyhedra. Furthermore, a restricted form of infinite-word automata, *weak deterministic* ones, suffices for representing the sets definable in $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$, and the sets that are representable by these automata in every base exactly match those that are definable in that theory [5]. A weak automaton is a Büchi automaton such that each strongly connected component of its transition relation is entirely composed of accepting or non-accepting states. The advantages of using weak deterministic RVA is that they have efficient manipulation algorithms, in particular for applying to represented sets operations such as Boolean combinations, projections and Cartesian products, and that they admit an easily computable canonical form [14].

4.2 Point Decision in RVA

Consider a deterministic weak RVA $\mathcal{A}$ that represents a polyhedron $\Pi \subseteq \mathbb{R}^n$ in a base $r \in \mathbb{N}_{>1}$. Assessing whether a point $\boldsymbol{v} \in \mathbb{R}^n$ belongs or not to Π reduces to encoding $\boldsymbol{v}$ in base r, which yields a word $w \in \Sigma_r^+ \star \Sigma_r^\omega$, and then checking whether this word is accepted by $\mathcal{A}$. Since $\mathcal{A}$ is deterministic, this can be done by following a single path in its transition graph. In a weak automaton, determining whether a path is accepting or not amounts to checking the accepting status of the last strongly connected component (SCC) that it visits. In other words, the point decision problem for $\boldsymbol{v}$ is solved by following transitions labeled by the successive symbols in w, which moves through the SCC of $\mathcal{A}$, until a final SCC is reached.

It has been shown that the SCC of $\mathcal{A}$ are related to the polyhedral components of Π [4, 3, 7]. This can intuitively be explained as follows. Let $u \in \Sigma_r^+ \star \Sigma_r^*$ be a finite encoding prefix that contains k digits in the fractional part of each vector component, and s be the state of $\mathcal{A}$ reached after reading u. The points of $\mathbb{R}^n$ that admit an encoding prefixed by u form a n-cube C_u of size r^{-k}, and the shape of Π inside this n-cube is uniquely determined by the state s. Assume that s belongs to a non-trivial strongly connected component of $\mathcal{A}$, i.e., one containing a cycle from s to itself, labeled by a word u' that adds k' additional digits to the encoding of each vector component. The encodings prefixed by uu' determine a n-cube $C_{uu'}$ of size $r^{-(k+k')}$. In both cubes C_u and $C_{uu'}$, the shape of Π is identical. This shows that the linear scaling transformation that maps C_u to $C_{uu'}$ leaves Π invariant inside of this n-cube. It is established in [4] that this invariance property actually holds for arbitrary scaling factors, which implies that Π is pyramidal inside C_u. A correspondence between the local pyramidal structures induced by the strongly connected components of $\mathcal{A}$ and the components of Π has been discovered in [4], and is investigated in [7]. This correspondence is not exactly one-to-one: some polyhedral components can be split between several SCC, and components that have local pyramids that only differ by a translation are sometimes described by a common SCC.

The reachability relation between the SCC of $\mathcal{A}$ also loosely corresponds to the incidence relation between the components of Π. The intuition is as follows. Consider a non-trivial SCC $\mathcal{S}_2$ reachable from another one $\mathcal{S}_1$. For every point $\boldsymbol{v} \in \mathbb{R}^n$ with an encoding that ends up in $\mathcal{S}_1$, there exists a point $\boldsymbol{v}'$ that is arbitrarily close to $\boldsymbol{v}$ with an encoding ending up in $\mathcal{S}_2$. Indeed, one may follow in $\mathcal{S}_1$ the path reading the encoding of $\boldsymbol{v}$ for arbitrarily many transitions before deciding to divert towards one ending in $\mathcal{S}_2$.

The procedure for deciding whether a point $\boldsymbol{v} \in \mathbb{R}^n$ belongs to a polyhedron represented by a RVA can thus be summarized as follows. One follows in the transition graph of the automaton the path given by an encoding w of $\boldsymbol{v}$, until reaching a non-trivial strongly connected component $\mathcal{S}_1$. If the remaining suffix of w can be read without leaving $\mathcal{S}_1$, which means that $\boldsymbol{v}$ belongs to a polyhedral component C_1 represented by $\mathcal{S}_1$, then the accepting status of this component provides the answer to the point decision problem. If this suffix leaves $\mathcal{S}_1$, then it eventually reaches another non-trivial SCC $\mathcal{S}_2$, which intuitively corresponds to moving from C_1 to another polyhedral component C_2 that is incident to C_1. The same procedure is repeated in $\mathcal{S}_2$, and so on until the path finally reaches a SCC that it does not leave anymore.

4.3 Principles of IRVA

The idea behind *Implicit Real-Vector Automata* is to define a data structure representing the components of a polyhedron and the incidence relation between them, in such a way that the point decision problem can be solved by following deterministically a single path in the structure, similarly to RVA. Compared to RVA, the main advantage of IRVA will be their substantially more efficient representation of the polyhedral components.

Informally, an IRVA is a graph composed of *implicit states*, which correspond to the components of its represented polyhedron, and *explicit states* and *transitions*, that provide an acyclic and deterministic decision structure linking the implicit states. In order to decide whether a point $\boldsymbol{v} \in \mathbb{R}^n$ belongs to a polyhedron represented by an IRVA, one will start from a first implicit state s_1. If $\boldsymbol{v}$ belongs to the corresponding polyhedral component C_1, then the search is over and the answer to the decision problem depends on whether C_1 is a subset of the polyhedron (which may be indicated by a flag attached to s_1). If $\boldsymbol{v}$ does not belong to C_1, then the procedure follows the decision structure that leaves s_1, until reaching another implicit state s_2, representing a polyhedral component C_2 that is incident to C_1. The same procedure is repeated in this implicit state, and so on until reaching the component that finally contains $\boldsymbol{v}$. Note that this idea straightforwardly generalizes to representations of polyhedral partitions, by associating a color to each implicit state instead of a binary flag.

The information that needs to be associated to an implicit state s includes the characteristic affine space $\mathrm{aff}(C)$ of its corresponding polyhedral component C, and either the color associated to this component or a binary acceptance flag.

The purpose of the deterministic decision structure leaving s is to represent the structure of the underlying local pyramid of C, by directing the vectors leaving C to the appropriate incident components. The development of such a decision structure will be addressed in Section 4.4.

A problem that remains to be addressed is to locate efficiently the first implicit state s_1 to be visited during the search for a point $\boldsymbol{v} \in \mathbb{R}^n$. This could be achieved by building a deterministic decision structure for classifying the points of $\mathbb{R}^n$. We choose to follow a simpler approach, which consists in considering only polyhedra in which the choice of the initial implicit state is trivial. We have the following result.

Theorem 7. *If a polyhedron Π is a pyramid, then it contains a unique component C that is minimum with respect to the incidence relation $\preceq$, i.e., such that $C \preceq C'$ for every component C' of Π. This minimum component corresponds to the set of apexes of Π.*

As a consequence, if a pyramid Π is represented by an IRVA, then the first implicit state visited during the search for a point can systematically be chosen to be the representation of the minimum component of Π, which eliminates the need for a special form of decision structure. Intuitively, this is possible because a pyramid of apex $\boldsymbol{v}$ is not affected by scaling transformations centered on $\boldsymbol{v}$. This property can be exploited for conducting the search in any arbitrarily small neighborhood of $\boldsymbol{v}$.

It is important to point out that moving from polyhedra to pyramids does not incur a loss of expressive power, for every polyhedron can be transformed into a pyramid that represents it without ambiguity, and vice-versa. We have the following definition.

Definition 8. *Let $\Pi \subseteq \mathbb{R}^n$ be a polyhedron. The* representing pyramid *of Π is the polyhedron $\overline{\Pi} \subseteq \mathbb{R}^{n+1} = \{\lambda(x_1, \ldots, x_n, 1) \mid \lambda \in \mathbb{R}_{>0} \wedge (x_1, \ldots, x_n) \in \Pi\}$.*

For every polyhedron $\Pi \subseteq \mathbb{R}^n$, the polyhedron $\overline{\Pi}$ is a pyramid of apex $\mathbf{0}$. The polyhedron Π can be recovered from $\overline{\Pi}$ by the transformation $\Pi = \{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n, 1) \in \overline{\Pi}\}$, which amounts to computing the section of $\overline{\Pi}$ by the planar constraint $x_{n+1} = 1$, and projecting the result over the n first vector components.

Note that the elementary operations over polyhedra, such as computing Boolean combinations, testing equality or inclusion, and solving the point decision and point classification problems, readily translate into identical or similar operations over their representing pyramids. The notion of representing pyramids also straightforwardly generalizes to polyhedral partitions.

In the sequel, we will thus only address without loss of generality the problem of designing a data structure for pyramids, or pyramidal partitions, that admit the apex $\mathbf{0}$. This choice brings the additional benefit of simplifying some structures. In particular, the characteristic affine spaces $\mathrm{aff}(C)$ of components become, in the case of such partitions, vector spaces since they systematically include $\mathbf{0}$. It follows that the implicit states of IRVA can actually be annotated by vector spaces instead of affine ones.

4.4 Decision Structure for Directions

Before defining formally IRVA, we discuss the details of the decision structures linking implicit states. This section is adapted from [3]. The problem that is tackled can be stated as follows. Let $\mathcal{A}$ be an IRVA representing a pyramidal partition Π of $\mathbb{R}^n$, with apex $\mathbf{0}$. Let s be an implicit state of $\mathcal{A}$, representing a polyhedral component C of Π. As explained in Section 4.3, a vector space $VS(s)$ is associated to s, and represents the set of apexes of the local pyramid $P_\Pi(C)$ of C. The problem consists in building a deterministic decision structure that classifies the points $\boldsymbol{v} \in \mathbb{R}^n$ such that $\boldsymbol{v} \notin VS(s)$ according to their polyhedral component in the pyramid $P_\Pi(C)$.

For every $\lambda \in \mathbb{R}_{>0}$, the decision taken for the point $\lambda\boldsymbol{v}$ has to match exactly the one taken for $\boldsymbol{v}$, since $P_\Pi(C)$ is invariant by scaling transformations centered on $\mathbf{0}$. We ensure that this property is satisfied by first *normalizing* the point $\boldsymbol{v}$, which intuitively corresponds to keeping only the direction in which it can be reached from $VS(s)$, and then encode this direction over a finite alphabet. The decision structure leaving s can then take the form of an acyclic graph, the edges of which are labeled by symbols of this alphabet.

We first describe the normalization operation. Let $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m\}$, with $0 \leq m \leq n$, be a basis of the vector space $VS(s)$. If $m = n$, then the component C is universal, meaning that $P_\Pi(C)$ has a uniform color over $\mathbb{R}^n$. In this case, there is no need for a decision structure leaving s, since one cannot have $\boldsymbol{v} \notin VS(s)$. If $m < n$, then we introduce $n - m$ vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{n-m}$ such that $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{n-m}\}$ is a basis of $\mathbb{R}^n$. The vectors $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, $\ldots$, $\boldsymbol{z}_{n-m}$ can be chosen in a canonical way, by selecting among $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$, in that order, $n - m$ vectors linearly independent with $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_m\}$.

The next step for normalizing $\boldsymbol{v}$ is to express this point in the coordinate system $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n-m}\}$, obtaining a tuple $(y_1, \ldots, y_m, z_1, \ldots, z_{n-m})$. Clearly, $P_\Pi(C)$ is not affected by translations within $VS(s)$, hence only the coordinates $(z_1, z_2, \ldots, z_{n-m})$ are relevant for classifying $\boldsymbol{v}$.

Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_{n-m})$. The next operation is to get rid of the magnitude of this vector, keeping only its direction. This can be done by computing the intersection of the half-line $\{\lambda\boldsymbol{z} \mid \lambda \in \mathbb{R}_{>0}\}$ with the faces of the *normalization cube* $[-\frac{1}{2}, \frac{1}{2}]^{n-m}$. The resulting point $\boldsymbol{v}' \in \mathbb{R}^{n-m}$ then provides the normalization of our original point $\boldsymbol{v}$.

It remains to define an encoding scheme for mapping normalized points onto words over a finite alphabet. Instead of using the technique discussed in Section 4.1, we use an encoding relation that exploits the fact that the points to be encoded belong to the normalization cube. Precisely, an encoding of a normalized point $\boldsymbol{v}' = (v'_1, v'_2, \ldots, v'_{n-m}) \in \mathbb{R}^{n-m}$ begins with a symbol $a \in \{-(1), +(1), -(2), +(2), \ldots, -(n-m), +(n-m)\}$ that identifies the face of the normalization cube to which $\boldsymbol{v}'$ belongs: If $a = -(i)$, with $1 \leq i \leq n - m$, then $v'_i = -\frac{1}{2}$; if $a = +(i)$, then $v'_i = +\frac{1}{2}$. The leading symbol a is followed by a suffix $w \in \{0, 1\}^\omega$ that encodes the position of $\boldsymbol{v}'$ within the face of the normalization cube represented by a. The suffix w is obtained as follows. If

$a \in \{-(i), +(i)\}$, with $1 \leq i \leq n - m$, then the i-th component v'_i of $\boldsymbol{v}'$ is fixed, and it remains to encode the coordinates $\boldsymbol{v}'' = (v'_1, \ldots, v'_{i-1}, v'_{i+1}, \ldots, v'_{n-m})$. We then define $w \in \{0,1\}^\omega$ as the fractional part of a base-2 encoding of the point $\boldsymbol{v}'' + (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$, i.e., such that $0^{n-m-1} \star w$ is a binary encoding of this point.

This encoding scheme maps a normalized point $\boldsymbol{v}' \in \mathbb{R}^{n-m}$ onto words of the form aw, with $a \in \{-(1), +(1), -(2), +(2), \ldots, -(n-m), +(n-m)\}$ and $w \in \{0,1\}^\omega$. Note that some points have multiple encodings, because either they are located at the boundary between different faces of the normalization cube, or their position on a face of this cube admits more than one fractional binary encoding. This situation is not at all problematic, provided that the decision structure leaving the implicit state s handles all these encodings in the same way.

In order to obtain a deterministic decision structure rooted at s, we consider the successive digits of encodings w. The leading digit a of w characterizes a specific face of the normalization cube. The normalized points $\boldsymbol{v}' \in \mathbb{R}^{n-m}$ that admit an encoding prefixed by a form a convex pyramid, and it is easily shown that the points $\boldsymbol{v} \in \mathbb{R}^n$ that normalize into $\boldsymbol{v}'$ form a convex pyramid as well. This latter pyramid, which only depends on the vector space $V = \mathit{VS}(s)$ and the leading symbol a, will be denoted $R_{V,a}$. This pyramid corresponds to a region of $\mathbb{R}^n$ that has a non empty intersection with some subset S of the components of Π. If this subset contains a unique minimum component C' with respect to the incidence relation $\preceq$, then deciding whether a point $\boldsymbol{v} \in R_{V,a}$ belongs or not to Π can be carried out in the neighborhood of C', hence the decision branch labeled by a can be directed to the implicit state representing C'.

If, on the other hand, the subset S of components does not contain a minimum element with respect to $\preceq$, then the decision branch labeled by a has to be developed further. By reading an additional prefix $u \in \{0,1\}^*$, one refines the region $R_{V,a}$ into the pyramid $R_{V,au}$ containing all points $\boldsymbol{v} \in \mathbb{R}^n$ whose normalization admits an encoding prefixed by au. Once again, if $R_{V,au}$ covers a subset of components of Π with a unique minimum component C', then the decision branch labeled by au can be oriented to the implicit state representing C'. Otherwise, the refinement procedure has to be repeated until the prefix is long enough. Termination is ensured by the following result.

Theorem 9. *Let Π be a pyramidal partition of $\mathbb{R}^n$, with apex $\mathbf{0}$, C be one of its polyhedral components, and $V = \mathrm{aff}(C)$. There exists $k \in \mathbb{N}_{>0}$ such that for every encoding w of length k, the subset of components of Π that have a non-empty intersection with the region $R_{V,w}$ admits a unique minimum element with respect to the incidence relation $\preceq$.*

5 Implicit Real-Vector Automata

5.1 Syntax

We are now ready to define the syntax of IRVA. As discussed in Sections 3.3 and 4.3, the goal is to obtain a symbolic representation of a pyramidal partition, i.e., a generalized pyramid in which the components are labeled by a finite range of colors instead of a binary acceptance flag.

Definition 10. *An* Implicit Real-Vector Automaton (IRVA) *is a tuple* $(n, S_I, S_E, s_0, \delta, VS, col)$*, where:*

- $n \in \mathbb{N}$ *is a* dimension*,*
- S_I *is a finite set of* implicit states*,*
- S_E *is a finite set of* explicit states*,*
- $s_0 \in S_I$ *is an* initial state*,*
- $\delta : (S_I \times \pm(\mathbb{N}_{>0})) \cup (S_E \times \{0,1\}) \to S_I \cup S_E$ *is a (partial)* transition function*,*
- $VS : S_I \to 2^{\mathbb{R}^n}$ *associates a* vector space *to each implicit state,*
- $col : S_I \to \mathbb{N}_{>0}$ *associates a* color *to each implicit state.*

In order to be valid, IRVA have to satisfy some syntactic constraints. First, the transition function δ must be acyclic as well as complete, in the sense that for every implicit state $s \in S_I$, $\delta(s, -(i))$ and $\delta(s, +(i))$ are defined iff $i \in \{1, 2, \ldots, n - \dim(VS(s))\}$, and for every explicit state $s \in S_E$, both $\delta(s, 0)$ and $\delta(s, 1)$ are defined. Let us denote by $s_1 \xrightarrow{w} s_2$, or more simply $s_1 \to s_2$ if w is not of interest, the fact that the transition function δ leads from the implicit state $s_1 \in S_I$ to $s_2 \in S_I$, reading the word $w \in \pm(\mathbb{N}_{>0})\{0,1\}^*$, and visiting only explicit states between s_1 and s_2. The reflexive and transitive closure of the relation $\to$ is denoted $\to^*$. We impose the following additional restrictions on IRVA: each implicit state $s \in S_I$ must be reachable, i.e., such that $s_0 \to^* s$, and for every pair $s_1, s_2 \in S_I$ such that $s_1 \xrightarrow{w} s_2$ for some word w, one must have $VS(s_1) \subset VS(s_2)$ (which implies $\dim(VS(s_1)) < \dim(VS(s_2))$), as well as $R_{VS(s_1),w} \cap VS(s_2) \neq \emptyset$. These restrictions intuitively express that the decision structures linking the implicit states are consistent with the properties of the incidence relation between polyhedral components.

5.2 Semantics

Let $\mathcal{A} = (n, S_I, S_E, s_0, \delta, VS, col)$ be an IRVA and $\boldsymbol{v} \in \mathbb{R}^n$ be a point. We have the following definition.

Definition 11. *A* run *of* $\mathcal{A}$ *over* $\boldsymbol{v}$ *is a finite sequence* $s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \cdots \xrightarrow{w_m} s_m$*, where* $0 \leq m \leq n$*,* $s_0, s_1, \ldots, s_m \in S_I$*, and* $w_1, w_2, \ldots, w_m \in \pm(\mathbb{N}_{>0})\{0,1\}^*$*, such that* $\boldsymbol{v} \in VS(s_m)$*, and for every* $i \in \{0, 1, \ldots, m-1\}$*,* $\boldsymbol{v} \notin VS(s_i)$ *and* $\boldsymbol{v} \in R_{VS(s_i),w_{i+1}}$*.*

In other words, a run over a point $\boldsymbol{v}$ is obtained by starting from the initial implicit state s_0, and repeatedly moving from an implicit state to another according to the deterministic decision structures induced by the transition function δ, which amounts to following the relation $\to$. A run ends when it finally reaches an implicit state whose associated vector space contains $\boldsymbol{v}$. A point may admit multiple runs. In order to be able to solve the point decision and point classification problems by following a single run in an IRVA, we introduce the following semantical integrity constraint.

Definition 12. *An IRVA $(n, S_I, S_E, s_0, \delta, \mathit{VS}, \mathit{col})$ is* well-formed *if it is syntactically valid, and for every $\boldsymbol{v} \in \mathbb{R}^n$, each of its runs over $\boldsymbol{v}$ ends up in the same implicit state.*

From now on, we will only consider well-formed IRVA. Solving the point decision or classification problems on such IRVA thus reduce to following an arbitrary run over the point of interest. The answer is then provided by the color of the implicit state that is finally reached by this run.

We are now ready to define the semantics of an IRVA $\mathcal{A} = (n, S_I, S_E, s_0, \delta, \mathit{VS}, \mathit{col})$, i.e., to describe the partitioning, or coloring, function $\Pi : \mathbb{R}^n \to \mathbb{N}_{>0}$ that it represents. Since the transition relation $\to$ between implicit states is acyclic, we can consider these implicit states in bottom-up order, associating a coloring function Π_s to each $s \in S_I$. This procedure can be started from the states $s \in S_I$ such that $\dim(\mathit{VS}(s)) = n$, which do not have successors by $\to$, and will eventually end up in the initial state s_0. The coloring function Π_{s_0} of s_0 will then provide the partition represented by the IRVA. The procedure relies on the following result.

Theorem 13. *Let $s \in S_I$ be an implicit state of $\mathcal{A}$. Its coloring function Π_s is such that*

- *$\Pi_s(\boldsymbol{v}) = \mathit{col}(s)$ for all points $\boldsymbol{v} \in \mathit{VS}(s)$,*
- *$\Pi_s(\boldsymbol{v}) = \Pi_{s'}(\boldsymbol{v})$ for all states $s' \in S_I$ and points $\boldsymbol{v} \in \mathbb{R}^n$ such that $s \xrightarrow{w} s'$ and $\boldsymbol{v} \in R_{\mathit{VS}(s),w}$, for some word $w \in \pm(\mathbb{N}_{>0})\{0,1\}^*$.*

In summary, we have shown constructively how to build a coloring function that describes the semantics of a given IRVA. We thus have the following theorem.

Theorem 14. *Every well-formed IRVA $(n, S_I, S_E, s_0, \delta, \mathit{VS}, \mathit{col})$ represents a pyramidal partition of $\mathbb{R}^n$.*

6 Canonicity

6.1 Canonical IRVA Representations

We now show that, for every pyramidal partition Π of $\mathbb{R}^n$ with apex $\mathbf{0}$, there exists an IRVA $\mathcal{A}$ that represents it, and that this IRVA can be defined canonically up to equality of vector spaces and isomorphism of transition graphs. We will then develop an algorithm for transforming any IRVA that represents Π into this canonical form.

The first step consists in defining the set of implicit states S_I of $\mathcal{A}$, by creating one implicit state s_i for each polyhedral component C_i of Π. From Theorem 7, we know that Π admits a unique minimum component C_0 with respect to the incidence relation $\preceq$. The corresponding implicit state s_0 becomes the initial state of $\mathcal{A}$. For each $s_i \in S_I$, the vector space $\mathit{VS}(s_i)$ is then made equal to

$\mathrm{aff}(C_i)$, and the color $col(s_i)$ takes the (common) value of $\Pi(\boldsymbol{v})$ for the points $\boldsymbol{v} \in C_i$.

It remains to define the decision structures that link the implicit states. Recall that, for an implicit state $s_i \in S_I$ and a word $w \in \pm(\{1, 2, \ldots, n-m\})\{0, 1\}^*$, where $m = \dim(VS(s_i))$, it is possible to have $s_i \xrightarrow{w} s_j$, with $s_j \in S_I$, iff the set of components $\{C_k \mid (C_i \preceq C_k) \wedge (R_{VS(s_i),w} \cap C_k \neq \emptyset)\}$ admits C_j as unique minimum element (with respect to $\preceq$). Since our aim is to obtain a canonical decision structure, it is natural to only select the shortest words w for which such a decision is possible, i.e., the transition $s_i \xrightarrow{w} s_j$ will be considered only if there does not exist a shorter prefix u of w for which $s_i \xrightarrow{u} s_j$ is possible. By applying this reasoning to all pairs of implicit states, one finally obtains a canonical form of the acyclic labeled transition relation $\rightarrow$ linking these states.

From the labeled relation $\rightarrow$, it is straightforward to define the explicit states of the canonical IRVA representing Π. This can be achieved by building a deterministic finite-state automaton with a transition relation corresponding to $\rightarrow$, considering that each implicit state has a unique distinguished accepting status. This automaton can then be minimized into a canonical form using classical techniques [12]. The implicit states are preserved by this operation, and the other states of the minimized automaton become the explicit states of the canonical IRVA. The transition function δ is then directly given by the transition relation of the minimized automaton.

6.2 Minimization Algorithm

We now sketch an algorithm for computing, from an IRVA $\mathcal{A} = (n, S_I, S_E, s_0, \delta, VS, col)$, a canonical IRVA that represents the same pyramidal partition. The idea is to exploit the acyclic structure of the transition graph, by inspecting the implicit and explicit states of $\mathcal{A}$ one by one in bottom-up order. Each step of the minimization procedure consists in examining one state $s \in S_I \cup S_E$, in order to determine whether it can be merged with another state s' that has already been processed, or left otherwise unchanged. When a state s is merged into a state s', all its incoming transitions are redirected to s', and its outgoing transitions are orphaned. This may leave unreachable states in the resulting IRVA, which are easily removed by a subsequent cleaning step. Thanks to the order in which the states are processed during minimization, the states s considered at each step are such that their successors by the transition function have already undergone minimization. The minimization of IRVA thus proceeds quite similarly to the usual minimization algorithms for acyclic finite-state automata or for binary decision diagrams [8]. The precise rules for deciding whether states should be merged are however specific to IRVA.

A first situation occurs when the state s under scrutiny happens to have identical successors as an already processed state s', i.e., for every symbol a, one has $\delta(s, a) = s''$ iff $\delta(s', a) = s''$. In this case, if s is an explicit state, then it can be merged into s'. If s is an implicit state, one additionally has to check the conditions $VS(s) = VS(s')$ and $col(s) = col(s')$ before merging s with s'.

The next rule is more complex. Consider an explicit state $s \in S_E$, with an outgoing decision structure leading to the set of implicit states $\{s_1, s_2, \ldots, s_k\} \subset S_I$. If this set contains a unique minimum element s' with respect to the transition relation $\rightarrow^*$, i.e., if $s \rightarrow^* s_i$ for all $i \in \{1, 2, \ldots, k\}$, then the decision structure leaving s is redundant, since all its paths can be redirected towards s' without affecting the semantics of $\mathcal{A}$. In such a case, the state s itself is redundant and can be merged with s'. It can be shown that this situation only occurs when the implicit state s' is a direct successor of s; this property may be exploited for speeding up the search for the states s_i.

Finally, a similar rule applies to implicit states $s \in S_I$. Let $\{s_1, s_2, \ldots, s_k\} \subset S_I$ be the set of implicit states that are directly reachable from s by the transition relation $\rightarrow$. If this set admits a unique minimum element s' with respect to $\rightarrow^*$, then s can be merged with s' provided that two conditions are satisfied. First, the color of both states must match: $col(s) = col(s')$. Second, in the particular case where one has $\dim(VS(s')) = \dim(VS(s)) + 1$, it is essential to ensure that the state s' does not represent a boundary of the polyhedral component associated to s. This is done by checking that, among the words w such that $s \xrightarrow{w} s'$, at least two of them have leading symbols $-(i)$ and $+(i)$ with an identical face number and opposite polarities. This tricky particular case was overlooked in [3].

7 Combination Operation

Our goal is now to develop an algorithm for combining two IRVA $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively representing pyramidal partitions Π_1 and Π_2 of $\mathbb{R}^n$. Recall that these partitions can be seen as functions $\Pi_1 : \mathbb{R}^n \rightarrow \{1, 2, \ldots, m_1\}$ and $\Pi_2 : \mathbb{R}^n \rightarrow \{1, 2, \ldots, m_2\}$ that assign colors to the points in $\mathbb{R}^n$. In order to combine Π_1 and Π_2, we need a *combination function* $c : \{1, 2, \ldots, m_1\} \times \{1, 2, \ldots, m_2\} \rightarrow \{1, 2, \ldots, m\}$, where $m \in \mathbb{N}_{>0}$, that maps every pair of colors from Π_1 and Π_2 onto a single one. We have the following definition.

Definition 15. *The* combination *of Π_1 and Π_2 induced by c is the pyramidal partition $\Pi = \Pi_1 \bowtie_c \Pi_2$ over $\mathbb{R}^n$ such that $\Pi(\boldsymbol{v}) = c(\Pi_1(\boldsymbol{v}), \Pi_2(\boldsymbol{v}))$ for every $\boldsymbol{v} \in \mathbb{R}^n$.*

Note that, in the particular case of binary partitions, this definition covers the computation of intersections, unions, and differences of sets, by choosing a combination function that corresponds to the appropriate Boolean operator.

The construction that we are about to describe shares some similarities with the computation of the product of two finite-state automata. The idea is to construct incrementally an IRVA $\mathcal{A}$ representing $\Pi = \Pi_1 \bowtie_c \Pi_2$, starting from its initial state and developing its transition function step by step. Each implicit state s of $\mathcal{A}$ corresponds to a pair (s_1, s_2), where s_1 (resp. s_2) is an implicit state of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$). The polyhedral component of Π represented by s corresponds to the points $\boldsymbol{v} \in \mathbb{R}^n$ that simultaneously belong to the component C_1 of Π_1 represented by s_1, and to the component C_2 of Π_2 represented by s_2. As a consequence, one has $VS(s) = VS(s_1) \cap VS(s_2)$ and $col(s) = c(col(s_1), col(s_2))$.

The initial state of $\mathcal{A}$ is the first to be created, by pairing the initial states of $\mathcal{A}_1$ and $\mathcal{A}_2$.

During the construction of $\mathcal{A}$, the creation of a new implicit state s is followed by the development of its outgoing decision structure. This is done by exploring the prefixes that can be read from s in breadth-first order. For such a prefix w, one checks whether $\mathcal{A}$ admits an implicit state s' such that $s \xrightarrow{w} s'$. This check is carried out by first computing the convex region $R_{VS(s),w}$, and then determining whether this region covers a unique minimum component incident to C_1 in $\mathit{\Pi}_1$, as well as one incident to C_2 in $\mathit{\Pi}_2$ (with respect to the incidence relations $\preceq_1$ and $\preceq_2$ of these pyramids). If a suitable implicit state s' exists, then it either corresponds to a previously computed state of $\mathcal{A}$, or to a new state that needs to be created. Otherwise, the decision labeled by w has to be developed further.

A key operation in the previous procedure is thus, given the IRVA $\mathcal{A}_i$ representing the pyramid $\mathit{\Pi}_i$, with $i \in \{1, 2\}$, a convex region $R \subseteq \mathbb{R}^n$, and an implicit state s_i of $\mathcal{A}_i$ representing a component C_i of $\mathit{\Pi}_i$, to determine whether the set of components of $\mathit{\Pi}_i$ that are incident to C_i and have a non-empty intersection with R admits a unique minimum element C'_i. We encapsulate this operation in a function $\mathit{minel}(\mathcal{A}_i, R, s_i)$ that returns the implicit state of $\mathcal{A}_i$ representing C'_i if it exists, and $\perp$ otherwise. The value of $\mathit{minel}(\mathcal{A}_i, R, s_i)$ can be computed as follows. First, one explores the implicit states s'_i of $\mathcal{A}_i$ that are reachable from s_i, i.e., for which there exist words w_1, w_2, $\ldots w_k$ and implicit states q_1, $q_2, \ldots q_{k-1}$, with $k > 0$, such that $s_i \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \xrightarrow{w_3} \cdots \xrightarrow{w_{k-1}} q_{k-1} \xrightarrow{w_k} s'_i$. All such states s'_i for which the intersection $R \cap R_{VS(s_i),w_1} \cap R_{VS(q_1),w_2} \cap \ldots \cap R_{VS(q_{k-1}),w_k} \cap VS(s'_i)$ is non-empty are collected in a set U. If U contains a minimum state s'_i with respect to $\rightarrow^*$, i.e., such that $s'_i \rightarrow^* s''_i$ for every $s''_i \in U$, then one has $\mathit{minel}(\mathcal{A}_i, R, s_i) = s'_i$. Otherwise, the function returns $\mathit{minel}(\mathcal{A}_i, R, s_i) = \perp$.

The procedure for combining two IRVA is formalized in Algorithm 1. In addition to *minel*, this procedure relies on a function $\mathit{succ}(s)$ that returns the alphabet of symbols that can potentially be read from a state s, a function $\mathit{new}()$ that instantiates new explicit states, and the usual *push*, *pop* and *empty*? operations on stacks. The algorithm relies on a stack for storing the states of $\mathcal{A}$ whose outgoing decision structures still need to be developed. This stack contains tuples (s_1, s_2, s, s_I, w), where s is such a state, s_1 and s_2 are the current states reached in respectively $\mathcal{A}_1$ and $\mathcal{A}_2$, s_I is the last visited implicit state in $\mathcal{A}$, and w is the word read from s_I to s.

8 Conclusions and Perspectives

We have studied a data structure, the Implicit Real Vector Automaton (IRVA), that can be used for representing symbolically polyhedra or polyhedral partitions in $\mathbb{R}^n$. IRVA are not limited to convex or regularized polyhedra, admit an easily computable minimal form, and are closed under Boolean operators.

IRVA imitate the principles of operation of Real-Vector Automata (RVA), another automata-based data structure suited for polyhedra, but can be considerably more concise. IRVA inherit from RVA very efficient algorithms for solving

```
input  : Two IRVA $\mathcal{A}_1 = (n_1, S_{I1}, S_{E1}, s_{01}, \delta_1, VS_1, col_1)$ and
         $\mathcal{A}_2 = (n_2, S_{I2}, S_{E2}, s_{02}, \delta_2, VS_2, col_2)$, a combination function $c$
output: An IRVA $\mathcal{A} = (n, S_I, S_E, s_0, \delta, VS, col)$
$s_0 := (s_{01}, s_{02})$;
$S_I := \{s_0\}$;
$S_E := \emptyset$;
$VS(s_0) := VS_1(s_{01}) \cap VS_2(s_{02})$;
$col(s_0) := c(col_1(s_{01}), col_2(s_{02}))$;
$stack := \emptyset$;
push $(stack, (s_{01}, s_{02}, s_0, s_0, \varepsilon))$;
while not(empty?($stack$)) do
    $(s_1, s_2, s, s_I, w) :=$ pop($stack$);
    foreach $a \in$ succ($s$) do
        $m_1 :=$ minel($\mathcal{A}_1, R_{VS(s_I),wa}, s_1$);
        $m_2 :=$ minel($\mathcal{A}_2, R_{VS(s_I),wa}, s_2$);
        if $m_1 \neq \bot$ and $m_2 \neq \bot$ then
            $R := VS_1(m_1) \cap VS_2(m_2)$;
            if $R \cap R_{VS(s_I),wa} = \emptyset$ or $\dim(VS(s_I)) \geq \dim(R)$ then
                $s_N :=$ new();
                $S_E := S_E \cup \{s_N\}$;
                $\delta(s, a) := s_N$;
                push($stack, (m_1, m_2, s_N, s_I, wa)$);
            else
                if $(m_1, m_2) \notin S_I$ then
                    $S_I := S_I \cup \{(m_1, m_2)\}$;
                    $VS((m_1, m_2)) := R$;
                    $col((m_1, m_2)) := c(col_1(m_1), col_2(m_2))$;
                    push($stack, (m_1, m_2, (m_1, m_2), (m_1, m_2), \varepsilon)$);
                end
                $\delta(s, a) := (m_1, m_2)$;
            end
        else
            if $m_1 = \bot$ then $m_1 := s_1$;
            if $m_2 = \bot$ then $m_2 := s_2$;
            $s_N :=$ new();
            $S_E := S_E \cup \{s_N\}$;
            $\delta(s, a) := s_N$;
            push($stack, (m_1, m_2, s_N, s_I, wa)$);
        end
    end
end
```

Algorithm 1: Computation of $\mathcal{A}_1 \bowtie_c \mathcal{A}_2$

the point decision and classification problems, which is a substantial advantage compared to other symbolic representations of polyhedra, in particular Selective Nef Complexes [10].

Future work will address the implementation of a package for building and manipulating IRVA, the assessment of their performances in actual applications, and the computation of additional operations such as projecting polyhedra and converting IRVA to and from other representations.

References

1. Bieri, H., Nef, W.: Elementary set operations with d-dimensional polyhedra. In: Workshop on Computational Geometry. Lecture Notes in Computer Science, vol. 333, pp. 97–112. Springer (1988)
2. Boigelot, B., Bronne, L., Rassart, S.: An improved reachability analysis method for strongly linear hybrid systems. In: Proc. CAV. Lecture Notes in Computer Science, vol. 1254, pp. 167–177. Springer (1997)
3. Boigelot, B., Brusten, J., Degbomont, J.F.: Implicit real vector automata. In: Proc. INFINITY. Electronic Proceedings in Theoretical Computer Science, vol. 39, pp. 63–76 (2010)
4. Boigelot, B., Brusten, J., Leroux, J.: A generalization of Semenov's theorem to automata over real numbers. In: Proc. CADE. Lecture Notes in Computer Science, vol. 5663, pp. 469–484. Springer (2009)
5. 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)
6. Bournez, O., Maler, O., Pnueli, A.: Orthogonal polyhedra: Representation and computation. In: Proc. HSCC. Lecture Notes in Computer Science, vol. 1569, pp. 46–60. Springer (1999)
7. Brusten, J.: On the sets of real vectors recognized by finite automata in multiple bases. Ph.D. thesis, University of Liège (2011)
8. Bryant, R.: Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys 24(3), 293–318 (1992)
9. Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: Proc. CAV. Lecture Notes in Computer Science, vol. 4144, pp. 81–94. Springer (2006)
10. Hachenberger, P., Kettner, L., Mehlhorn, K.: Boolean operations on 3D selective Nef complexes: Data structure, algorithms, optimized implementation and experiments. Comput. Geom. 38(1–2), 64–99 (2007)
11. Halbwachs, N., Raymond, P., Proy, Y.: Verification of linear hybrid systems by means of convex approximations. In: Proc. SAS. Lecture Notes in Computer Science, vol. 864, pp. 223–237. Springer-Verlag (1994)
12. Hopcroft, J.: An $n \log n$ algorithm for minimizing states in a finite automaton. Tech. rep., Stanford (1971)
13. Le Verge, H.: A note on Chernikova's algorithm. Tech. rep., IRISA, Rennes (1992)
14. Löding, C.: Efficient minimization of deterministic weak ω-automata. Information Processing Letters 79(3), 105–109 (2001)
15. Requicha, A.A.G.: Representations for rigid solids: Theory, methods, and systems. ACM Comput. Surv. 12(4), 437–464 (1980)
16. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of constraint programming. Elsevier (2006)