A library to manipulate Z-polyhedra in image representation

Guillaume Iooss
Colorado State University and
Laboratoire de l'Informatique du Parallelisme
guillaume.iooss@gmail.com

Sanjay Rajopadhye
Colorado State University
Sanjay.Rajopadhye@ColoState.Edu

ABSTRACT

The polyhedral model is a powerful mathematical framework used for program analysis (such as program transformation, automatic parallelization of nested loops, etc). It is well known that this model becomes somewhat cumbersome when we encounter modulos, "holes" in the computation domain, or non-unit-stride loops. An extension, called the *Z-polyhedral* model, has been proposed to manage these situations. A Z-polyhedron is the intersection of an affine lattice and an integer polyhedron, but it can also be represented as the affine image of an integer polyhedron.

In this paper, we present a Java implementation of a Z-polyhedral library, based on the image representation of Z-polyhedron. The basic operations were implemented (intersection, difference, image, etc), most of them based on the algorithms of Gautam and Rajopadhye (PPoPP 2007) and a recent article by Seghir et al. (to appear in ACM TACO). We also present some performance data about our implementation.

1. INTRODUCTION

The polyhedral model is a mathematical framework widely used for program analysis and transformation, especially in the context of loop parallelization. For example, it is commonly used to represent iteration domains in regular nested loops with affine dependences, and allows us to do dependence analysis, automatic parallelization [4], loop transformations, etc

However, although this model works well with regular, dense, and unit-stride nested loops, it becomes somewhat cumbersome when we have modulos, "holes" in the iteration space, e.g. by non-unit stride loops, although some adaptations can be made (for example, by adding extra dimensions to deal with modulo). To deal with such situations, an extension of integer polyhedron has been proposed, called the *Z-polyhedral* model [6].

This material is based upon work supported in part, by the National Science Foundation under Grant No. 0917319.

IMPACT 2012
Second International Workshop on Polyhedral Compilation Techniques
Jan 23, 2012, Paris, France
In conjunction with HiPEAC 2012.

http://impact.gforge.inria.fr/impact2012

A Z-polyhedron is the intersection of an integer polyhedron with an integer lattice, which allows us to manage more easily regular holes inside its domain and to extend the algorithms based in the polyhedral model [16, 5, 8]. It is also possible to represent a Z-polyhedron as the affine image of an integer polyhedron and to operate on this image representation [6].

Some libraries to manipulate Z-polyhedra based on the intersection representation already exist, such as Polylib [9] and ISL [23] which implicitly handles Z-polyhedra as special cases of other, more general objects like relations. However, depending the representation considered, the complexities of the associated algorithms are different. For example, the image representation is more suitable for the image operation (especially by an injective affine function), and the intersection representation is more suitable for the intersection or the difference operation, although all of these operations can be made in both representations. In this paper, we present the implementation of a library in Java[1], to manipulate Z-polyhedra, based on the image representation and on the adaptation of the algorithms described in [6].

The rest of this paper is organized as follows. We first introduce some mathematical background needed in the rest of this paper. Then, we describe the algorithms used in this library in section 3 3. In section 4, we describe the implementation of the library and we evaluate its performance. After presenting the related work in section 5, we finally conclude in section 6.

2. MATHEMATICAL BACKGROUND

2.1 Some notions of linear algebra

We assume that every matrix and vector encountered is integral. A matrix is said to be *full-column rank* if its number of columns is equal to its rank. An *unimodular matrix* is a square matrix A whose determinant is ± 1.

The *Hermite Normal Form* (HNF) H of an integer matrix A is an integer matrix such that there exists an integer unimodular matrix U with $H = A.U$ and H satisfies the following properties:

- If they exist, the zero-columns of H are the last ones.

[1]Available at `http://www.cs.colostate.edu/AlphaZsvn/Development/trunk/mde/`

- For each non-zero column, its first non-zero value is greater than the first non-zero value of the previous column.
- The first non-zero value of a column is strictly maximal on its row, and all the coefficients on this row are non-negative.

For example, the matrix $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ -4 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 \\ 5 & 2 & 7 & 0 \\ 5 & 1 & 2 & 9 \end{bmatrix}$ is in HNF.

For every integer matrix A, there exists a unique Hermite Normal Form H. However, the associated unimodular matrix U might not be unique. The HNF of a matrix can be computed in polynomial time [19].

A *Diophantine equation* is an equation in which we are only interested in the integer solutions. A system of linear Diophantine equations $A.x = b$ can be solved in polynomial time, and the set of its solutions is in the form $\{x_0 + \lambda_1.x_1 + ... + \lambda_t.x_t \mid \lambda_i \in \mathbb{Z}\}$, with x_1, ..., x_t linearly independent integer vectors.

2.2 Integer Polyhedra

An *integer polyhedron* is a subset of $\mathbb{Z}^n$, which satisfies a finite number of linear *constraints*. In its matrix representation, an integer polyhedron can be written as:

$$\mathcal{P} = \{z \mid A.z + b = 0 \wedge Q.z + q \geq 0\} \subset \mathbb{Z}^n,$$

with A and Q integral matrices and b and q integral vectors. This representation is called the *implicit representation* of an integer polyhedron.

It is also possible to define an integer polyhedron by $\mathcal{P} = C + R + L$, with:

- $L = Vect(z_1, \ldots, z_t)$ a vectorial space called *linearity space*,
- $R = \{\mu_1.y_1 + \cdots + \mu_r.y_r \mid \mu_i \geq 0\}$ and
- $C = conv(x_1, ..., x_s)$.

The vectors z_i are called *bidirectional rays*, y_i are called *unidirectional rays* and C is the convex closure of the points x_i. This representation is called the *Minkowski representation* of an integer polyhedron and is equivalent to the implicit representation.

To get the parameter representation from the implicit representation, we can use Chernikova's algorithm [10]. This algorithm is exponential in dimension in the worst case, so it is not convenient for polyhedra with more than 10 to 12 dimensions.

The set of integer polyhedra satisfies several closure properties:

- The intersection of two integer polyhedra is an integer polyhedron.
- The difference of two integer polyhedra is a finite union of integer polyhedra.
- The preimage of an integer polyhedron by an integer affine function is an integer polyhedron.
- The image of an integer polyhedron by an integer affine function ($f : x \mapsto F.x + f_0$) is generally not an integer polyhedron. However, if the matrix of affine function F is unimodular, then the image is an integer polyhedron. In this case, f is called a *change of basis*.

All these operations can be done in polynomial time, except for the image by a non-unimodular function. The *context* of an integer polyhedron[2] is the linear part of the smallest affine subspace containing an integer polyhedron. However, for the implicit representation of an integer polyhedron, some equalities can be hidden among inequalities (example: $i \geq 0 \wedge j \geq 0 \wedge i + j \leq 0$), and finding this set is NP-hard (because this problem can be reduced to the emptiness check of an integer polyhedron).

The *integer hull* $\mathcal{P}_I$ of an integer polyhedron $\mathcal{P}$ is the convex hull of all the integer points of $\mathcal{P}$. Computing the integer hull of a polyhedron in an implicit representation is also a NP-complete problem. Indeed, if we can compute the integer hull of a polyhedron, then we have the vertices of this integer hull. However, determining if an integer point of $\mathcal{P}$ is a vertex of $\mathcal{P}_I$ is an NP-complete problem (see Schrijver [19, p 254]).

2.3 Affine Lattice

An *affine lattice* (also called $\mathbb{Z}$-module) is a set of the form $\mathcal{L} = \{L.z + l \mid z \in \mathbb{Z}^n\} \subset \mathbb{Z}^m$ with L an integer matrix called *generator* of $\mathcal{L}$, l an integer vector called *offset* of $\mathcal{L}$, n a positive integer called the *internal dimension* of $\mathcal{L}$ and m a positive integer called the *external dimension* of $\mathcal{L}$.

An affine lattice is said to be in *canonical form* if $\begin{bmatrix} 1 & 0 \\ l & L \end{bmatrix}$ is in Hermite Normal Form and L is full-column rank. The canonical form of a lattice $\mathcal{L}$ is unique.

The set of affine lattices satisfies several closure properties:

- $(\mathcal{L}_1 \cap \mathcal{L}_2)$ is either empty, or an affine lattice.
- $(\mathcal{L}_1 - \mathcal{L}_2)$ is a union of affine lattices. More precisely, the resulting union is finite, iff the internal dimension of $(\mathcal{L}_1 \cap \mathcal{L}_2)$ is equal to the internal dimension of $\mathcal{L}_1$.
- $f(\mathcal{L})$, with f an integer affine function, is an affine lattice.
- $f^{-1}(\mathcal{L})$, for an integer affine function, f is either empty, or an affine lattice.

[2]Not to be confused with the *context-domain* that Dupont de Dinechin [2] defines as the set of points where an Alpha expresion must be (or needs to be) evaluated.

2.4 Z-polyhedra

A *Linearly Bounded Lattice* (LBL) is the image of an integer polyhedron by an affine function[3], a set defined as

$$\{L.z + l \mid z \in \mathcal{P}\} \quad (1)$$

with L an integer matrix, l an integer vector and $\mathcal{P}_c$ an integer. Note that $\langle L, l\rangle$ define an affine lattice. We call $\mathcal{P}_c$ the *coordinate polyhedron*.

A *Z-polyhedron* $\mathcal{Z}$ is the intersection of an integer polyhedron $\mathcal{P}$ and an affine lattice $\mathcal{L}$, $\mathcal{Z} = \mathcal{P} \cap \mathcal{L}$. Intuitively, we a Z-polyhedron has "regular holes" in the domain. LeVerge [11] showed the inclusion: Polyhedra $\subset$ Zpolyhedra $\subset$ **LBL**. Therefore, a Z-polyhedron can also be written in the form of Eq. 1. Since the inclusion is strict, every LBL is not a Z-polyhedron. For example, $\{i + 3j \mid 0 \leq j \leq i \leq 3\}$, which corresponds to the set $[|0; 12|] - \{8, 10, 11\}$, is not a Z-polyhedron. However, it is a union of Z-polyhedra [20, 6].

To determine whether an LBL is a Z-polyhedron, we have a sufficient condition [11]:

THEOREM 2.1 (LEVERGE'S CONDITION). *The set* $\{L.z + l \mid Q.z + q \geq 0 \wedge A.z + b = 0\}$ *is a Z-polyhedron whenever* $Ker\begin{pmatrix} L \\ Q_0 \end{pmatrix} \subset Ker(Q)$ *with* $Ker(Q_0)$ *the context of the coordinate polyhedron* $\mathcal{P}_c = \{z \mid Q.z+q \geq 0 \wedge A.z+b = 0\}$.

However, because computing the context is hard, Gautam and Rajopadhye weaken this condition to obtain an other sufficient condition: $Ker(L) \subset Ker(Q)$.

A Z-polyhedron is in *canonical form* if: (i) its affine lattice is in canonical form, and (ii) its coordinate polyhedron is full-dimensional, i.e., the dimension of its context is the dimension of its space.

An *affine lattice function* is an application of the form $(K.z+k \mapsto R.z + r)$ with K full-column rank. This is a generalization of the notion of affine function, which gives an image only for the points belonging to the affine lattice $\{K.z + k \mid z\}$.

Finally, the set of Z-polyhedra satisfies the following closure properties:

- $(\mathcal{Z}_1 \cap \mathcal{Z}_2)$ is a Z-polyhedron (which can be empty).
- $(\mathcal{Z}_1 - \mathcal{Z}_2)$ is a union of Z-polyhedra (which can be finite if the affine lattice difference is infinite).
- $f^{-1}(\mathcal{Z})$, with f an integer affine lattice function, is a Z-polyhedron.
- $f(\mathcal{Z})$, with f an integer affine lattice function, is an LBL in general. However, an LBL can always be expressed as a union of Z-polyhedra.

[3]Technically speaking, the term LBL is a misnomer since the words precisely describe *Z-polyhedra*. The name, introduced by Teich and Thiele [22] has remained in (mis) use.

Compared with the stability properties of integer polyhedra, we do not need to impose the affine function to be unimodular in the image closure property. However, the algorithms for manipulating Z-polyhedra are more sophisticated, and their complexity is increased.

For example, let us consider the following nested loop program:

```
for (int i=0; i<n; i=i++)
  for (int j=0; j<i; j=j+2)
      A[3i+2j,2i-j] = ...
```

Let us assume that we want to know which element of the array `A` is accessed (to do a dependence analysis, for example). We represent the iteration space by the $\mathcal{Z}$-polyhedron $\{(i, j) \mid 0 \leq i < n \wedge 0 \leq j < i\} \cap \{(i', 2.j')\}$. The corresponding image representation is $\{(i, 2.j') \mid 0 \leq i < n \wedge 0 \leq 2.j' < i\}$.

Then, we apply the affine function $((i, j) \mapsto (3i + 2j, 2i - j))$ to get the LBL $\{(3i+4j', 2i-2j') \mid 0 \leq i < n \wedge 0 \leq 2.j' < i\}$. Because we have $Ker(L) = \emptyset$, this LBL satisfies LeVerge's condition, and is a $\mathcal{Z}$-polyhedron.

3. ALGORITHMS FOR Z-POLYHEDRA

3.1 Operations on Z-polyhedra

Gautam and Rajopadhye previously presented some algorithms to manipulate Z-polyhedra in their image representation [6]. The algorithms presented are similar to the ones for the intersection representation [9], except that we have some extra work (or less work) to update each time the coordinate polyhedra. Moreover, as soon as we perform an operation, we put the obtained Z-polyhedron into canonical form.

About the difference algorithm, we restrict it to the situation where the difference between the associated affine lattices $(\mathcal{L}_1 - \mathcal{L}_2)$ is finite. However, if we have polytopes as coordinate polyhedron, we only need a finite set of affine lattice $(\mathcal{L}_i)_i$ that covers $(\mathcal{L}_1 - \mathcal{L}_2)$ over the polytopes's bounds, so we can manage some cases where the affine lattice difference is infinite.

However, these algorithms are not polynomial in general, because of the following reasons:

- These algorithm use Z-polyhedra in their canonical form. However, the second condition of the canonical form requires for a full-dimensional coordinate polyhedron and computing the context of an integer polyhedron is NP-hard.
- The image algorithm uses this condition, and does not work with an under-approximation of the context of an integer polyhedron (which could be obtained by considering the equalities of the integer polyhedron).

Therefore, we drop the second condition of the canonical form, to make the intersection, difference and preimage algorithms polynomial. However, we cannot compare two Z-polyhedra by looking separately at their affine lattices, then

at their coordinate polyhedrons. Let us consider the two following Z-polyhedra:

$$\mathcal{Z}_1 = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} . \begin{pmatrix} x \\ y \end{pmatrix} \mid y \geq 0 \wedge x \geq 4y \wedge 4 - x \geq 4y \right\}$$

$$\mathcal{Z}_2 = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} x' \\ y' \end{pmatrix} \;\middle|\; y' \geq 0 \wedge x' \geq 4y' \wedge 4 - x' \geq 4y' \right\}$$

These two polyhedra have different affine lattices, but are both equal to $[|0; 4|] \times \{0\} \times \{0\}$. Indeed, their coordinate polyhedron has an integer equality hidden, and is equal to $[|0; 4|] \times \{0\}$. Therefore, we still need to get the canonical form before comparing two Z-polyhedra.

Moreover, because the proposed image algorithm used the second condition of the canonical form, we need to find another algorithm to compute the image of a Z-polyhedron.

Image algorithm The image of a Z-polyhedron by an affine lattice function is an LBL in general. Therefore, the main part of the image algorithm consists in decomposing this LBL into an finite union of Z-polyhedra. More precisely, let us consider a Z-polyhedron $\{L.z + l \mid Q.z + q \geq 0 \wedge A.z + b = 0\}$ and an affine function $(x \mapsto R.x + r)$. The image of this Z-polyhedron by this affine lattice function is:

$$\{y \mid (\exists z) y - R.(L.z + l) - r = 0 \wedge A.z + b = 0 \wedge Q.z + q \geq 0\}$$

Then, we just have to eliminate all the existential variables in z. For this, we use the algorithm of Seghir et al. [21].

In short, this algorithm first eliminates existential variables by using equalities. For this, it does a rational elimination of this variable to obtain an integer polyhedron $\mathcal{Y}$, then it computes the validity lattice $\mathcal{L}$, which corresponds to the integer solutions of this elimination. Therefore, we obtain a Z-polyhedron $(\mathcal{Y} \cap \mathcal{L})$. Then, we get the image representation of this Z-polyhedron and we continue the elimination on its coordinate polyhedron.

When we do not have enough equalities to suppress all the existential variables, we have to use inequalities to eliminate them. For this, we arbitrarily choose an existential variable z to eliminate, we list all inequalities involving this existential variable, and we separate them into two sets: the upper bound (of the form $\alpha.z \leq u(x, p)$) and the lower bound (of the form $l(x, p) \leq \beta.z$). For every pair of lower and upper bound, we have to compute its projection along the dimension of z, then we will intersect all these projections to get the integer projection of the polyhedron.

Let us consider an upper bound and a lower bound: $\alpha.z \leq u(x, p)$ and $l(x, p) \leq \beta.z$. We can consider two sets in relation to this integer projection (first introduced by [14]):

- The *Exact shadow*, which is the rational projection of these constraints. Its equation is $\alpha.l(x, p) \leq \beta.u(x, p)$.
- The *Dark shadow*, which is the convex part in which we are sure that every integer point has at least a preimage. The idea is that the gap between the two constraints has to be at least one (along the dimension of z), to be sure that we have at least one integer point (cf Figure 1).

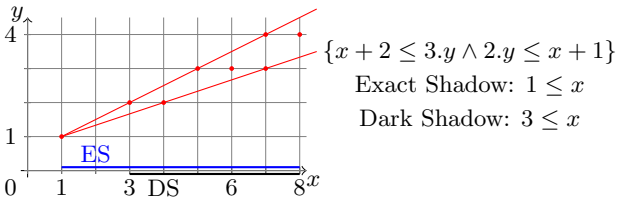

Figure 1: Exact and Dark shadow

The exact shadow contains the integer projection we are looking for, but it might also contain extra points which are not on the integer projection. On the other hand, the dark shadow contains only points in the integer projection, but it might miss some extra points outside. Therefore, the integer projection is the union of the dark shadow, and these "extra points" that are in the exact shadow and not in the dark shadow. A way to get these points though equations is described in [21]. Because of the presence of a modulo at some point, we might introduce a new existential variable, which can be immediately suppressed, using the corresponding introduced equality.

Therefore, the number of existential variables strictly decreases in the generated Z-polyhedron and the method to get these new LBL by eliminating an existential variable in a single LBL is polynomial. However, the number of sets generated might be exponential [6, 20], so the whole algorithm is exponential.

To adapt this algorithm to the image representation, when we eliminate several existential variables by using equalities, we transform the Z-polyhedron $(\mathcal{Y} \cap \mathcal{L})$ into its image representation $\{L.x + l \mid x \in \mathcal{P}_c\}$ and we continue the algorithm with its coordinate polyhedron $\mathcal{P}_c$. Therefore, if several affine lattices are generated, we just have to compose them, rather than taking their intersection.

3.2 Parametric Z-polyhedron

We have two different ways to define the notion of parametric Z-polyhedra. Intuitively, we want to define a parametric Z-polyhedron as a conventional Z-polyhedron, where parameters replace some indexes. Therefore, if we consider the image representation, a parametric Z-polyhedron should be a set of the form:

$$\left\{ L. \begin{pmatrix} p \\ z \end{pmatrix} + l \;\middle|\; Q. \begin{pmatrix} p \\ z \end{pmatrix} + q >= 0 \wedge A. \begin{pmatrix} p \\ z \end{pmatrix} + b = 0 \right\},$$

with p the vector of parameters and z the vector of indices.

Consider the intersection between the two parametrized Z-polyhedra $\{N + 2k \mid k \in \mathbb{Z}\}$ and $\{2k + 1 \mid k \in \mathbb{Z}\}$. Among the parity of N, the intersection is empty (if N is even), or equal to $2\mathbb{Z} + 1$ (if N is odd). Indeed, the parameter part shifts the offset of the affine lattice where the Z-polyhedron live, and, depending on the intersection between the different affine lattices, we can have either an empty result, or

a Z-polyhedron. Therefore, the algorithms used for non-parametric Z-polyhedron are not directly applicable to the parametric case.

If we go back to the intersection representation, the current definition is equivalent to $\mathcal{Z}(p) = \mathcal{L}(p) \cap \mathcal{P}(p)$, where $\mathcal{L}$ is a parametrized affine lattice and $\mathcal{P}$ a parametrized integer polyhedron. Therefore, we can weaken the definition of parametric Z-polyhedron by imposing that $\mathcal{L}$ is not parametrized anymore. Therefore, the affine lattice is fixed, and we do not have conditions on the parametric part to decide which Z-polyhedron we have.

In terms of image representation, if a Z-polyhedron has parameters on the affine lattice part of its image representation, we have to ensure that its affine lattice is fixed. In this case, we always can translate the coordinate polyhedron to suppress the parameter contribution from the affine lattice.

PROOF.

Let $\mathcal{Z} = \left\{ \begin{pmatrix} L_p & L_x \end{pmatrix} . \begin{pmatrix} p \\ x \end{pmatrix} + l \;\middle|\; x \in \mathcal{P}_c(p) \right\}$. Let N be the first parameter of p. We want to nullify the column C_N of L_p corresponding to N.

- If we cannot express C_N as an integer linear combination of the columns of L_x, then $\mathcal{L}(N, p') \neq \mathcal{L}(N+1, p')$. Indeed, for all x, $\mathcal{L}(N+1, p')(x) = C_N + \mathcal{L}(N, p')(x) \notin \mathcal{L}(N, p')$. Therefore, the affine lattice is not fixed.

- Otherwise, if $C_N = L_x.\lambda$ then, we have:

$$\begin{aligned} &\begin{pmatrix} C_N & L'_p & L_x \end{pmatrix} . \begin{pmatrix} N \\ p' \\ x \end{pmatrix} + l \\ &\qquad = (L_x.\lambda).N + \begin{pmatrix} L'_p & L_x \end{pmatrix} . \begin{pmatrix} p' \\ x \end{pmatrix} + l \\ &\qquad = \begin{pmatrix} L'_p & L_x \end{pmatrix} . \begin{pmatrix} p' \\ x' \end{pmatrix} + l, \text{ with } x' = x + N.\lambda. \end{aligned}$$

So, $\mathcal{Z} = \left\{ \begin{pmatrix} L'_p & L_x \end{pmatrix} . \begin{pmatrix} p' \\ x' \end{pmatrix} + l \;\middle|\; x' \in \tau_{(N.\lambda)}(\mathcal{P}_c(p)) \right\}$, with $\tau_{N.\lambda}(\mathcal{P}_c(p))$ the translation of $\mathcal{P}_c(p)$ along the vector $N.\lambda$, which is also a parametrized polyhedron.

So, we can rewrite $\mathcal{Z}$ in such a way that the parameter N is no longer present on the affine lattice part. □

From this constructive proof, we can deduce an algorithm which verifies when a parametric affine lattice on its image representation corresponds to this definition, and which eliminates the parametric part of the affine lattice. Therefore, in the library, the internal dimension of the affine lattice of a Z-polyhedron is equal the number of indices of the coordinate polyhedron.

Although the non-parametric algorithms no longer work for the more general definition of parametric Z-polyhedra, it is still possible to adapt them. Such algorithms would manipulate QUAST (QUasi-Affine Selection Tree) [3] whose nodes are modulo linear conditions on the parameters (of the form $C.p + c \in \lambda\mathbb{Z}$), and whose leaves are Z-polyhedra.

4. IMPLEMENTATION

4.1 Implementation

This library is implemented in Java to be compatible with the AlphaZ system being developed at CSU and to use Z-polyhedron in the *Alpha* language[15, 12]. To manage polyhedra, we use an interface called *Polymodel*, which calls a polyhedron library, such as ISL [23].

We first have implemented an affine lattice library to do the basic operations on affine lattices (intersection, difference, image, preimage). For the difference algorithm, we restrict the algorithm to the case where $(\mathcal{L}_1 - \mathcal{L}_2)$ is a finite union (i.e., when the internal dimension of $\mathcal{L}_1$ and $(\mathcal{L}_1 \cap \mathcal{L}_2)$ are equal). Moreover, in the finite case, no optimizations have been done to limit the number of generated affine lattices.

The HNF algorithm that we are using is the "standard" polynomial algorithm (described in Schrijver [19, p 56-57]), based on pivoting Gauss and Euclidean algorithm to suppress all the values of a row. This algorithm has a complexity of $O(n^4.log||A||)$, with $||A|| = max|A_{i,j}|$. Moreover, because we are doing only column operations, we can compute at the same time the associated unimodular matrix U, such that $H = A.U$. To reduce the numerical explosion for the computation of U, we use Blankinship's version of Euclidian algorithm [7]. We could use more complex and efficient HNF algorithm to reduce the time or the space complexity (such as the one by Micciancio and Warinschi [13]).

The operations currently implemented for the Z-polyhedral library are intersection, difference (only in the finite case for the affine lattices), preimage and image. Moreover, we can deal with parametric Z-polyhedra, as long as the associated affine lattice does not depend on the values of the parameters.

Compared to the image algorithm described by Seghir and Loechner [21], no heuristic to select existential variable has been implemented. Indeed, a possible heuristic as "selecting the existential variable with the least constraint on it", might help us to reduce or to delay the combinatorial explosion of the number of Z-polyhedra to describe an LBL. In the library, we just try to detect and remove empty Z-polyhedra by quickly looking at their set of equalities and inequalities, and by trying to find incoherences.

4.2 Non-polynomial operations

About the emptiness check of a Z-polyhedron, we just have to check if the associated coordinate polyhedron is empty. Because this polyhedron is integer, this problem is NP-hard and the associated algorithm has an exponential complexity.

For testing the equality, we need the canonical form first (with the full-dimensionality condition on the coordinate polyhedron) to be able to compare separately the two affine lattices, then the two coordinate polyhedra. Otherwise, another possible way is to do two emptiness check on both differences (assuming than none of them becomes infinite).

To get the canonical form, we need to compute the context of an integer polyhedron, i.e. the smallest affine space which contains the coordinate polyhedron. This problem is equivalent to finding all the equalities of an integer polyhedron. However, some of them might be hidden among the inequalities (such as $a.x + b \geq 0$ and $a.x + b \leq 0$). There are two types of hidden equalities in an integer polyhedron $\mathcal{P}$:

- The *rational equalities*, which are the equalities that exist for the rational polyhedron corresponding to $\mathcal{P}$,
- The *integer equalities*, which are the equalities that do not occur in the rational instance, but in the integer case (such as $y \geq 0 \wedge x \geq 4y \wedge 4 - x \geq 4y$).

To deal with the rational equalities, we can use Chernikova's algorithm [10] to get the dual representation of $\mathcal{P}$: $\mathcal{P} = C + R + L$, with $L = Vect(z_1, \ldots, z_t)$, $R = Cone(y_1, \ldots, y_r)$ and $C = Conv(x_1, \ldots, x_s)$. The y_j and z_k can be integer vectors, but the x_i are rational points. Then, the context of the rational version of $\mathcal{P}$ is:

$$Vect(z_1, \ldots, z_t, y_1, \ldots, y_r, x_2 - x_1, \ldots, x_s - x_1),$$

and we can get the rational equalities by taking an implicit equation of this vectorial space.

To deal with the integer equalities, we can use an integer hull algorithm to get $C = Conv(x'_1, \ldots, x'_{s'})$ with x'_i integer points. The context of the integer polyhedron $\mathcal{P}$ is:

$$Vect(z_1, \ldots, z_t, y_1, \ldots, y_r, x'_2 - x'_1, \ldots, x'_{s'} - x'_1).$$

However, we do not need the total integer hull to get the context, so we do not need to do all the algorithm, as soon as we have points along all the dimensions. For the algorithm presented in [1], we can stop with a sub-approximation of the integer hull as soon as we are sure that we cannot get more dimensions.

4.3 Comparison of complexity

Between the intersection and the image representations, the algorithms remain the same, modulo some changes of bases to obtain the right coordinate polyhedron for the image representation. Therefore, the asymptotic complexities of the intersection, difference, pre-image and image algorithms (by an affine function) are the same.

However, the intersection, difference and preimage algorithm will be slightly slower for the image representation, because of the extra matrix multiplication. On the other hand, the image algorithm will be faster for the image representation, because we compose lattices (in $O(n^3)$) instead of intersecting them (complexity of the HNF algorithm).

It is easy to get the image representation from the intersection representation (possible in $O(n^3)$). To do the reverse operation, we start form a Z-polyhedron in its image representation $\{L.z + l \mid z \in \mathcal{P}\}$ and we want to find the corresponding polyhedron $\mathcal{P}'$ in the intersection representation. We have $(z \mapsto L.z + l)(\mathcal{P}) = \mathcal{P}'$, so this translation has the same complexity than the computation of the affine image of an integer polyhedron. Because LeVerge condition is verified, $\mathcal{P}'$ is a Z-polyhedron and can be obtained in intersection representation. Then, we have to intersect it with the lattice $\{L.z+l \mid z\}$ to get the intersection representation of $\mathcal{P}$. The translation from the image representation to the intersection representation has not been implemented.

Operations	Polyhedra	Z-polyhedra
Intersection	$O(N_{constraints})$	$O(n^4.log(\|\|L\|\|))$ (HNF)
Difference	$O(N^2_{constraints})$	$O(n^4.log(\|\|L\|\|))$ (HNF)
Preimage	$O(n^3)$	$O(n^4.log(\|\|L\|\|))$ ($\cap$)
Image (unimodular)	$O(n^3)$	$O(n^3)$ (matrix mult)
Image (non unimodular)	-	Exponential

Figure 2: Comparison of the computational complexity of operations on integer polyhedra and Z-polyhedra

Figure 2 shows the comparison between the asymptotic complexities of the algorithms for integer polyhedron and Z-polyhedron. $||L||$ corresponds to the maximum of the absolute value of the coefficients on the matrix part of both affine lattices. As expected, the Z-polyhedron algorithms cost more than the integer polyhedron ones. Therefore, Z-polyhedron should be used only when the expressiveness provides by integer polyhedron is not sufficient.

4.4 Performance

In the rest of this section, we will interest ourself in analyzing the dependences of the following program:

```
for i1 = 0 to N
   for j1 = 0 to i1
     S1:  A[N + 2i1 + 3j1, 2i1 + 2j1 + 1] = f1(i1,j1)
for i2 = N to 2N
   for j2 = i2 to 2N
     m if (i2%3==0 ∧ j2%2==0)
        S2:  A[i2,j2] = f2(A[i2 - 2j2, -2i2 + 5j2+2])
```

The following measures were made on a laptop equipped equipped with two 1.73 GHz processors (Pentium Dual-core) and 2.9 GB of Ram.

Construction Z-polyhedron: We want to compute the set of indexes that are accessed by `S1`. This set is defined as: $\left\{ \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} N \\ i_1 \\ j_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;\middle|\; 0 \leq j_1 \leq i_1 \leq N \right\}$, where N is the parameter. The first column (corresponding to N) is a linear combination of the other columns, so the affine lattice does not depend on the parameter. Moreover, we can check that this LBL satisfies LeVerge's condition: it is a Z-polyhedron.

The algorithm runs in about 23 ms and returns:

$$\left\{ \begin{bmatrix} 2 & 3 \\ 2 & 2 \end{bmatrix} \begin{pmatrix} i_1 \\ j_1 \end{pmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;\middle|\; -i1 \geq 0 \wedge 2N + i_1 - j_1 \geq 0 \wedge -N + j_1 \geq 0 \right\}$$

Construction Z-polyhedron (intersection): We want to compute the set of indexes written by `S2`. This set is:

$$\left\{ \begin{bmatrix} i_2 \\ j_2 \end{bmatrix} \;\middle|\; N \le i_2 \le 2N \wedge i_2 \le j_2 \le 2N \right\} \cap \left\{ \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \right\}.$$

The algorithm runs in about 8 ms and returns:

$$\left\{ \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{pmatrix} k \\ l \end{pmatrix} \;\middle|\; -N+3k \ge 0 \;\wedge\; 2*N-3k \ge 0 \;\wedge\; -3k+2l \ge 0 \;\wedge\; 2*N-2l \ge 0 \right\}$$

Image (unimodular): We want to study the dependence between `S1` and `S2`. For this, we first need to compute the set of indexes that are read by `S2`. This set is the image of the previous Z-polyhedron by the integer affine function: $\left(\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 & -2 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right)$. This function is unimodular, which simplifies the algorithm we have to use.

The algorithm runs in about 7 ms and returns:

$$\left\{ \begin{bmatrix} 3 & -4 \\ -6 & 10 \end{bmatrix} \begin{pmatrix} k \\ l \end{pmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;\middle|\; -N+3k \ge 0 \;\wedge\; 2*N-3k \ge 0 \;\wedge\; -3k+2l \ge 0 \;\wedge\; 2*N-2l \ge 0 \right\}$$

Intersection: Finally, to get the set of indexes of `A` that are read by `S2` and written by `S1`, we have to do the intersection of the two corresponding sets, previously computed.

The algorithm runs in about 20 ms and returns:

$$\left\{ \begin{bmatrix} -1 & 3 \\ 4 & -6 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;\middle|\; -7x+12y \ge 0 \;\wedge\; 2N+12x-21y \ge 0 \;\wedge\; -N-5x+9y \ge 0 \;\wedge -N+3x+3y \ge 0 \;\wedge\; 2N-3x-3y \ge 0 \;\wedge\; -x-3y \ge 0 \;\wedge\; 2N-2x \ge 0 \right\}$$

Difference: We want to compute the elements of `A` that were written by `S1` and not modified by `S2`. For this, we have to do the difference of the set of indexes written by `S1` by the set of indexes written by `S2`. These two sets were previously computed.

The algorithm runs in about 172 ms and returns an union of 3 Z-polyhedra: $\left\{ \begin{bmatrix} -1 & 3 \\ 4 & -6 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;\middle|\; \mathcal{P}_i \right\},\ i \in \{1,2,3\}$

$$\mathcal{P}_1 = \{x,y \mid -7x+12y \ge 0 \;\wedge\; 2N+12x-21y \ge 0 \;\wedge\; -N-5x+9y \ge 0 \;\wedge\; x+3y-1 \ge 0\}$$
$$\mathcal{P}_2 = \{x,y \mid -x \ge 0 \wedge 2N-3y-1 \ge 0 \wedge -N+x+3y+1 \ge 0\}$$
$$\mathcal{P}_3 = \{x,y \mid -i_1 \ge 0 \wedge 2N-3y-2 \ge 0 \wedge -N+x+3y+2 \ge 0\}$$

Image (non unimodular): In this library, the more costly operation implemented is the image algorithm, specifially, the transformation of an LBL into a union of Z-polyhedron (using Seghir and Loechner's algorithm [21]). To measure its speed, we run this part on the following Pressburger set:

$$\{x \mid (\exists i,j,k,l)\ 1 \le i \le n \;\wedge\; i+1 \le j \le n \;\wedge\; 1 \le k \le n \;\wedge\; 1 \le l \le k-1 \;\wedge\; x = 2i+3j \;\wedge\; x = k+2l\}.$$

The algorithm generates 20 Z-polyhedra, with a peak of about 300 LBLs generated at the middle of the algorithm (most of them are empty). In total, it takes 550 ms. However, as we saw in the previous section, the algorithm can still be improved with heuristics and we did not fully optimize our Java implementation.

All the previous operations can be done by using conventional integer polyhedron with additional tricks, such as adding extra dimensions. However, this solution is much more complex for the user. An interesting future work would be to compare the time taken by using such methods with the time taken by the associated Z-polyhedron algorithms.

5. RELATED WORK

PolyLib [24] is a polyhedral library currently maintained by Loechner, which allows the user to manipulate unions of parametric polyhedra. This library maintains both representations of integer polyhedron, chooses the most suitable when we perform an operation and gets the other representation by using Chernikova's algorithm. This library supports some Z-polyhedral algorithms, but is based on the intersection representation.

ISL (Integer Set Library) [23] is another polyhedral library developed by Verdoolaege. This library also supports some complex algorithms, such as integer hull computation, and since it supports relations, it implicitly supports Z-polyhedra. However, it does not provide specialized algorithms for Z-polyhedra. *Polymodel* is a generic interface to polyhedral libraries, developed at IRISA, which allows the user to change easily the underlying polyhedral library used (such as ISL, PolyLib, etc).

Seghir and Loechner presented an algorithm to count the number of points in unions of parametric Z-polytopes [21]. To count the number of points inside the union of 2 Z-polytopes, they add the number of points inside each Z-polytopes and they subtract the number of points in the intersection. These algorithms were implemented in the library *PolyTrans*[4]. The notion of Linearly Bounded Lattice (LBL) was introduced by Teich and Thiele [22]. This paper also proved some stability properties of this object.

Ramanujam [17, 18] studied non-unimodular loop transformations and focused on the related code generation issues. His generated code is based on `For` loops with steps, and various algorithms are provided to infer the loop bounds, the array access function or the steps of the loops.

6. CONCLUSION AND FUTURE WORK

After introducing some mathematical background, we saw the two different representations of a Z-polyhedron. Then, we discussed algorithms manipulating Z-polyhedra and the notion of parametric Z-polyhedra. Finally, we presented the implementation of a Z-polyhedral library, based on the image representation and we discussed about some non implemented algorithms. We also compared the complexity of the implemented operations with the ones for the intersection representation of the Z-polyhedron, and with the ones for the integer polyhedron.

A Z-polyhedron is an extension of integer polyhedron that is more convenient to use in some cases. However, they

[4]http://zpolytrans.gforge.inria.fr

are mathematically more complex and their operation algorithms have a bigger complexity than integer polyhedron ones. The asymptotic complexities between the two possible representations of Z-polyhedra do not change, but the intersection representation is more efficient for the intersection and difference algorithms, whereas the image representation is better for the image algorithms.

There is still missing operations in the presented Z-polyhedra library, such as the algorithm to get back to the intersection representation, the algorithms to get the canonical form of a Z-polyhedron, the equality algorithm, etc. Moreover, some optimization are possible, such as using heuristics to select which existential variable we should eliminate first for the image algorithm, or such as limiting the number of Z-polyhedron we generate when we do a difference.

The next step is to take the algorithms using integer polyhedron and to translate them to Z-polyhedron. Because of the extra expressiveness provided and the enhanced stabilities properties, we might be able to improve some algorithms by using those properties. For example, because we are no more restricted to unimodular transformations with Z-polyhedron, we can do more loop transformations.

7. REFERENCES

[1] P. J. Charles, J. M. Howe, and A. King. Integer Polyhedra for Program Analysis. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management*, AAIM '09, pages 85–99, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] F. Dupont de Dincehcin. *Systèmes structurés d'équations récurrentes : mide en œuvre dans le langage Alpha et applications*. PhD thesis, Université de Rennes, IRISA, Rennes, janvier 1997.

[3] P. Feautrier. Parametric integer programming. *RAIRO Recherche Opérationnelle*, 22(3):243–268, 1988.

[4] P. Feautrier. Automatic Parallelization in the Polytope Model. In *The Data Parallel Programming Model: Foundations, HPF Realization, and Scientific Applications*, pages 79–103, London, UK, 1996. Springer-Verlag.

[5] Gautam, D. Kim, and S. Rajopadhye. Scheduling in the Z-Polyhedral Model. In *IEEE International Parallel and Distributed Processing Symposium*, page 39. IEEE Computer Society, 2007.

[6] Gautam and S. Rajopadhye. The Z-Polyhedral Model. In *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 237–248, 2007.

[7] G. Havas and B. S. Majewski. Hermite normal form computation for integer matrices. In *Congressus Numerantium 105*, pages 87–96, 1994.

[8] D. Kim, Gautam, and S. Rajopadhye. Value-based Dependence Analysis for the Z-Polyhedral Model. Technical report, Colorado State University, 2008.

[9] S. P. Kumar and T. Risset. A library for Z-polyhedral Operations. Technical report, IRISA, 2000.

[10] H. Le Verge. A note on Chernikova's Algorithm. Technical report, IRISA, 1994.

[11] H. Le Verge. Recurrences on lattice polyhedra and their applications to the synthesis of systolic arrays. Based on an unpublished manuscript written by H. Le Verge before his untimely death in 1994, Jan 1995.

[12] P. M. Lenders and S. V. Rajopadhye. Multirate VLSI Arrays and Their Synthesis. *IEEE Trans. Computers*, 46(5):515–529, 1997.

[13] D. Micciancio and B. Warinschi. A Linear Space Algorithm for Computing the Hermite Normal Form. In *Proceedings ISSAC 2001, Lecture Notes in Computer Sci., 2146*, pages 126–145. Springer, 2001.

[14] W. Pugh. The Omega test: a fast and practical integer programming algorithm for dependence analysis. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, Supercomputing '91, pages 4–13, New York, NY, USA, 1991. ACM.

[15] P. Quinton, S. V. Rajopadhye, and T. Risset. Extension of the ALPHA language to recurrences on sparse periodic domains. In *IEEE Conference on Application-specific Systems, Architectures and Processors*, Chicago, IL, Aug 1996.

[16] P. Quinton, S. V. Rajopadhye, and T. Risset. On Manipulating $\mathcal{Z}$-polyhedra using a Canonical Representation, June 1997.

[17] J. Ramanujam. Non-unimodular Transformations of Nested Loops. In *IN PROC. SUPERCOMPUTING 92*, pages 214–223. IEEE Computer Society Press, 1992.

[18] J. Ramanujam. Beyond unimodular transformations. *The Journal of Supercomputing*, 9:365–389, 1995. 10.1007/BF01206273.

[19] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-interscience series in discrete mathematics and optimisation, 1986.

[20] R. Seghir and V. Loechner. Memory optimization by counting points in integer transformations of parametric polytopes. In *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, CASES '06, pages 74–82, New York, NY, USA, 2006. ACM.

[21] R. Seghir, V. Loechner, and B. Meister. Integer Affine Transformations of Parametric Z-polytopes and Applications to Loop Nest Optimization. Rapport de recherche (to appear in TACO), May 2010.

[22] J. Teich and L. Thiele. Partitioning of processor arrays: a piecewise regular approach. *Integr. VLSI J.*, 14:297–332, February 1993.

[23] S. Verdoolaege. ISL: An integer set library for the polyhedral model. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Lecture Notes in Computer Science,*, pages 299–302. Springer, Sept. 2010.

[24] D. Wilde, K. A Library for doing polyhedral operations. Rapport de recherche RR-2157, INRIA, 1993.