Minimizing Memory Strides using Integer Transformations of Parametric Polytopes

Rachid Seghir, Vincent Loechner and Benoît Meister*

ICPS/LSIIT UMR CNRS-ULP 7005
Pôle API, Bd Sébastien Brant
F-67400 Illkirch
{seghir,loechner}@icps.u-strasbg.fr

*Reservoir Labs, USA
meister@reservoir.com

Abstract

The polyhedral model is a well-known framework for the analysis and transformation of affine loop nests. In this paper, we present a new method concerning one of the most delicate geometric operations that is raised by this model, namely the transformation of integer points in parametric polytopes, which is equivalent to eliminating the existential variables from a Presburger formula. The result of such a transformation is given by a union of $\mathbb{Z}$-polytopes.

We also propose a polynomial algorithm, in the input size (for fixed dimension), to count integer points in arbitrary unions of a fixed number of parametric $\mathbb{Z}$-polytopes. These results allow us to compute data layout transformations improving memory compression and spatial locality: only data that are actually used by a loop nest are allocated in memory, and they are allocated in the same order as they are accessed for the first time, ensuring stride-one accesses when possible.

1 Introduction

Many affine loop nests analyses and optimizations raise the problem of counting the number of images by an affine integer transformation of the integer points contained in a parametric polytope (a bounded polyhedron depending on parameters), or equivalently counting the solutions to a Presburger formula. The following works, among many others, raise this problem.

[11] and [5] propose a method to compute the number of cache misses by solving cache miss equations. [12] optimize the data distribution of parallel programs through volume of distant accesses for NUMA-machines. [28] calculate the number of memory locations touched by a loop nest to compress memory. [7] quantify data misses depending on the cache line size to minimize power consumption in embedded systems with adaptative memory hierarchy. [2] compute reuse distances to improve cache effectiveness.

In this work, the integer transformation of parametric polytopes is used to extend and improve the data layout remapping, discussed in [15]. The basic idea is the *array*

linearization: the data are stored in a first-access ordered one-dimensional array. In the new layout, array elements are remapped to main memory in the order in which they are accessed for the first time. As a result, almost all subsequent iterations refer to adjacent elements, which ensures spatial locality improvement, and memory compression. Indeed, only data that are actually used are allocated in memory.

Many approaches to count the image of integer points in polytopes have been proposed [21, 4, 1, 16, 19, 3], but none of them definitively solves this problem, particularly when the polytopes are *parametric*. Recently, Verdoolaege et al. [25] proposed to apply a number of simple rewriting rules to a disjoint union of parametric polytopes, and to use PIP [9] if these rules fail. Note that the preprocessing step of computing the disjoint union, using the Omega Calculator [20], is worst-case exponential, and so is PIP. A theoretical polynomial-time algorithm was proposed in [27] but its implementation remains a challenge.

Pugh et al. [20, 21, 22] proposed an algorithm based on Fourier-Motzkin variable elimination. Their algorithm allows to decide whether a Presburger formula is valid, i.e., if it admits an integer solution. In certain cases, this algorithm introduces *extra* existential variables which simplify the decision of the existence of a solution. Introducing new constraints with extra existential variables is not adapted to the problem of *counting* integer points after projecting out more than one variable of a polytope, and no implementation of this algorithm has been reported.

The integer transformation of a polytope by an affine function consists in eliminating equalities from the corresponding Presburger formula first and then projecting it along possibly many dimensions. In this paper we propose a new projection method, based on Pugh and Wonnacott's work [22], whose result is given by a union of parametric $\mathbb{Z}$-polytopes.[1] The number of resulting $\mathbb{Z}$-polytopes is worst-case exponential, when the coefficients of the existential variables are large. Note that in our context (array references linearization), these coefficients are usually not too large. We also provide an algorithm for counting integer points in such unions. When the number of $\mathbb{Z}$-polytopes in the union is fixed, this algorithm computes the solution in a polynomial time in the input size (for fixed dimension). Finally, we show through experiments that, for an important set of representative examples, our method is more efficient than the one of Verdoolaege [25].

This paper is organized as follows: in Section 2 we show the motivation of this work by means of an example. The data layout transformations for strides minimization is presented in Section 3. In Section 4 we present our method for calculating the transformation of parametric polytopes. Section 5 discusses a new algorithm for counting integer points in arbitrary unions of parametric $\mathbb{Z}$-polytopes. In Section 6, we provide experiments comparing our method to the one of Verdoolaege [25]. Finally, the conclusions are given in Section 7.

2 Motivating example

In this section, we will optimize the loop nest in Figure 1.(a) whose iteration domain corresponds to the parametric polytope $P = \{(i,j,k) \in \mathbb{Z}^3 \mid 1 \leq i \leq N \ \wedge 1 \leq j \leq N \ \wedge 1 \leq k \leq N\}$, and the affine access function to the elements of array A is $T(i,j,k) = (x = 3i + 6k, y = 2j + 5)$.

[1] A $\mathbb{Z}$-polytope is the intersection of a polytope with a non-standard integer lattice.

```
do i=1, N
  do j=1, N
    do k=1, N
      A(3*i+6*k, 2*j+5)= ...
    enddo
  enddo
enddo
```

(a) Original loop nest

```
do i=1, N
  do j=1, N
    do k=1, N
      if (i+2*k>=2*N+3)
        B((i+2*k-3)*N+j-1)=...
      else if (i+2*k<=2*N+1 & i mod 2 =1)
        B((j-1)*N+(i+2k-3)/2)=...
      else B((N+j-1)*N+i/2+k-2)=...
    enddo
  enddo
enddo
```

(b) Transformed loop nest

Figure 1: Data layout transformation: array A in loop nest (a) is transformed into a one-dimensional array B in loop nest (b).

The elements of array A accessed by this loop nest are given by the transformation by T of the integer points of the polytope P. The integer points of the rational image of P which do not have integer preimages in P are not accessed by this loop nest. We call these points the *holes*.

The transformation by T of the integer points of P is given by the following Presburger formula [21]:

$$S = \{(x,y) \in \mathbb{Z}^2 \mid \exists (i,j,k) \in \mathbb{Z}^3 : \\ 1 \leq i \leq N \ \wedge \ 1 \leq j \leq N \ \wedge \ 1 \leq k \leq N \ \wedge \ x = 3i + 6k \ \wedge \ y = 2j + 5\}.$$

The problem of calculating the integer image reduces then to the elimination of the existential variables i, j and k.

The transformation of the polytope P without taking into account the problem of the holes, i.e., directly applying Fourier-Motzkin variable elimination, gives the following result:

$$T(P) = \{(x,y) \in \mathbb{Z}^2 \mid 9 \leq x \leq 9N \ \wedge \ 7 \leq y \leq 2N + 5\} \text{ with } N \geq 1.$$

The number of integer points in this polytope is given by the following Ehrhart polynomial [8, 6, 26]:

$$\mathcal{E}(T(P)) = 18N^2 - 25N + 8 \ \text{ if } \ N \geq 1 \ \text{ and } \ 0 \ \text{ otherwise.}$$

While the *exact* number of images of the integer points in P, computed as explained in Section 4, is:

$$\mathcal{E}(S) = 3N^2 - 2N \ \text{ if } N \geq 1 \ \text{ and } 0 \text{ otherwise.}$$

That is to say, the number of array elements actually accessed is $3N^2 - 2N$, which means that the size of unused allocated memory is $15N^2 - 23N + 8$ array elements.

We now discuss the linearization of the above array. For simplicity, we process on an equivalent set of the accessed elements. This set is obtained applying a variable compression [16] to the access function $T(i,j,k) = (x = 3i + 6k, \ y = 2j + 5)$, which gives $T'(i,j,k) = (x' = i + 2k, \ y' = j + 2)$. The reference in Figure 1.(a) is assumed to be

`A(i+2k, j+2)=...` Each datum $A(x_0, y_0)$ will be remapped to $B(\mathcal{E}(x_0, y_0, N))$, where $\mathcal{E}(x_0, y_0, N)$ is the Ehrhart polynomial corresponding to the number of all array elements accessed before $A(x_0, y_0)$. This polynomial is obtained by counting integer points in the *exact integer transformation* of all iterations that are lexicographically smaller ($\prec$) than the first iteration accessing $A(x_0, y_0)$. Let $T'^{-1}(x_0, y_0) \cap P$ be the set of all iterations referencing $A(x_0, y_0)$. The first iteration accessing the datum $A(x_0, y_0)$ is equal to the lexicographic minimum $\mathbf{i}_{min}(x_0, y_0, N)$ of the set $T'^{-1}(x_0, y_0) \cap P$. PIP [9] allows to compute the integer lexicographic minimum of a set of parametric constraints. In our example, the lexicographic minimum is given by PIP as:

$$\mathbf{i}_{min}(x_0, y_0, N) = \begin{cases} \mathbf{i}_{min1}(x_0, y_0, N) = (x_0 - 2N,\ y_0 - 2,\ N) & \text{if } x_0 \geq 2N + 2 \\ \mathbf{i}_{min2}(x_0, y_0, N) = (x_0 - 2M + 2,\ y_0 - 2,\ M - 1) & \text{otherwise,} \end{cases}$$

where M (introduced by PIP) is equal to $\lfloor \frac{x_0+1}{2} \rfloor$, with $\lfloor . \rfloor$ denoting the lower integer part.

In the following we focus on the linearization of array A when the lexicographic minimum equals $\mathbf{i}_{min1}(x_0, y_0, N)$. Let $\mathcal{S}_1(x_0, y_0)$ be the set of iterations preceding the first one accessing the datum $A(x_0, y_0)$. This set is given by:

$$\mathcal{S}_1(x_0, y_0) = \{(i, j, k) \in P \mid (i, j, k) \prec \mathbf{i}_{min1}(x_0, y_0, N) = \\ i < x_0 - 2N \ \vee \ (i = x_0 - 2N \ \wedge \ j < y_0 - 2) \vee (i = x_0 - 2N \ \wedge \ j = y_0 - 2 \ \wedge \ k < N)\}$$

The array elements accessed by this set of iterations is given by the following Presburger formula:

$$\pi(\mathcal{S}_1(x_0, y_0)) = \{(x', y')\mathbb{Z}^2 \mid \exists (i, j, k) \in \mathcal{S}_1(x_0, y_0) : x' = i + 2k \ \wedge \ y' = j + 2\},$$

where x_0 and y_0 are now considered as parameters. The number of integer points in this Presburger formula is given by the Ehrhart polynomial:

$$\mathcal{E}_1(x_0, y_0, N) = \begin{cases} N^2 + (y_0 - 2)N - 1 & \text{if } \ x_0 = 2N + 2 \\ (x_0 - 3)N + y_0 - 3 & \text{if } \ x_0 \geq 2N + 3 \\ 0 \quad \text{otherwise.} \end{cases}$$

In the same way, we calculate the new access function, when the lexicographic minimum equals $\mathbf{i}_{min2}(x_0, y_0, N)$.

Finally, the reference $A(3i + 6k, 2j + 5)$ is replaced by $B(\mathcal{E}(x_0, y_0, N))$, where B is a one-dimensional array, $x_0 = i + 2k$ and $y_0 = j + 2$ (see Figure 1):

$$\mathcal{E}(x_0, y_0, N) = \begin{cases} (x_0 - 3)N + y_0 - 3 & \text{if } \ x_0 \geq 2N + 3 \\ (y_0 - 3)N + (x_0 - 3)/2 & \text{if } \ x_0 \leq 2N + 1 \ \wedge \ x_0 \ \text{ odd} \\ N^2 + (y_0 - 3)N + (x_0 - 4)/2 & \text{if } \ x_0 \leq 2N + 2 \ \wedge \ x_0 \ \text{ even.} \end{cases}$$

Let us now discuss the benefit of the new access function as compared to the original one, first in terms of allocated memory and then in terms of spatial locality. As we mentioned at the beginning of this example, the size of the array A is $18N^2 - 25N + 8$, whereas that of B is $3N^2 - 2N$. Suppose the array element size is 4 bytes. This makes array A requiring $4(15N^2 - 23N + 8)$ bytes more than B, which is more than 576 KB when $N = 100$ and more than 57 MB when $N = 1000$.

We now focus on the spatial locality improvement, assuming row-major order storage of arrays. Obviously, for the original reference $(3i+6k, 2j+5)$, a jump of six rows is done each time the index of the innermost loop k changes. Since there are $2N-1$ elements per row, a jump of $12N-6$ elements is done when the index k changes, whatever the other indices. This could lead to a cache or TLB miss at each iteration (when N is large). In contrast, the new access function always ensures stride-one accesses for fixed i and j such that $i+2k \leq 2N+1$ (which is the case occurring most often). When $i+2k \geq 2N+3$ only a jump of $2N$ is done. Statistically speaking, this function leads to roughly 74.5 percent of stride-one accesses, 23.5 percent of accesses of stride $2N$ and less than 2 percent of accesses of stride more than $2N$. Note that when there is no data reuse, the new access function always ensures stride-one accesses. As an example, we can fix i to 3, j to 6 and N to 100. When $1 \leq k \leq 3$, the new access function respectively refers to elements $B(501)$, $B(502)$ and $B(503)$. One can also check the data coherence, i.e., if two iterations in the former layout access the same datum, they also do in the new one. Of course, later accesses to the same data have to be transformed as well using the same access function.

3 Data layout transformation

In [15], Loechner et al. presented an algorithm to optimize spatial locality through data layout transformation. The data are stored in memory in the same order as they are accessed for the first time by a loop nest. The previous example illustrates this method, for one loop nest accessing one array reference. The method consists in computing:

1. the first iteration $\mathbf{i}_{min}(\mathbf{x_0}, \mathbf{p})$ referencing an array element $\mathbf{x_0}$, where $\mathbf{p}$ is a vector of parameters: it is the lexicographic minimum of the set of iterations $I(\mathbf{x_0})$, referencing array element $\mathbf{x_0}$ through reference function T, $I(\mathbf{x_0}) = \{\mathbf{i} \in (P \cap \mathbb{Z}^d) \mid T(\mathbf{i}) = \mathbf{x_0}\}$, where P is a d-dimensional polytope representing the parametric constraints on the loop nest indices. $\mathbf{i}_{min}(\mathbf{x_0}, \mathbf{p})$ is computed from $I(\mathbf{x_0})$ using parametric polytope operations and PIP [9];

2. the number of array elements being accessed by the loop nest before iteration $\mathbf{i}_{min}(\mathbf{x_0}, \mathbf{p})$: the image by the access function T of the iterations lexicographically smaller than $\mathbf{i}_{min}(\mathbf{x_0}, \mathbf{p})$.

In general, when there are two references $\mathcal{R}_1$ and $\mathcal{R}_2$ to the same array in a loop nest, the data accessed by both access functions can not be optimized for both accesses. In [15], the proposed method was to choose one of the references to optimize for these data, and the other one will not be optimized. Let us consider that $\mathcal{R}_1$ is optimized. Then, all data accessed through $\mathcal{R}_1$ will be optimized as presented above. The data accessed through both $\mathcal{R}_1$ and $\mathcal{R}_2$ are using the same access function, not optimized for $\mathcal{R}_2$. And the data accessed through $\mathcal{R}_2$ but not through $\mathcal{R}_1$ can also be optimized, using another data layout. Generating code from this solution consists in either adding tests when accessing $\mathcal{R}_2$, or splitting the loop nest such that the iterations accessing $\mathcal{R}_2$ through the first computed function are separated from the iterations accessing $\mathcal{R}_2$ through the second function.

When there are two loop nests accessing the same data, the proposed solution is similar: for the data accessed by both loop nests, one has to be chosen to be optimized, and the

other one is split into two parts, one accessing the same data using the same not optimized access function, and the other one that can be optimized.

In the following, we propose a new data layout remapping policy, based on parametric $\mathbb{Z}$-polytope transformations. This technique has the advantage to deal with the particular cases for which the former method leads to an inefficient remapping or even fails to give a solution.

Let us consider two references $\mathcal{R}_1$ and $\mathcal{R}_2$ to the same array A, occurring at the same loop-level of a loop nest such that $\mathcal{R}_1$ precedes $\mathcal{R}_2$. We need to define two access functions $\mathcal{E}_1(\mathbf{x_0}, \mathbf{p})$ and $\mathcal{E}_2(\mathbf{x_0}, \mathbf{p})$, respectively optimizing the references $\mathcal{R}_1$ and $\mathcal{R}_2$, where $\mathbf{x_0}$ is the index vector of a referenced datum and $\mathbf{p}$ is the parameter vector. Let P be the polytope defining the iteration space of the loop nest, $\mathcal{R}_1 = A(T_1(\mathbf{i}))$ and $\mathcal{R}_2 = A(T_2(\mathbf{i}))$, where T_1 and T_2 are affine functions and $\mathbf{i}$ is a vector of the iteration space P. The first iteration accessing a datum $A(\mathbf{x_0})$ according to the reference $\mathcal{R}_1$, (resp. $\mathcal{R}_2$) is given by the lexicographic minimum of the set $T_1^{-1}(\mathbf{x_0}) \cap P$ (resp. $T_2^{-1}(\mathbf{x_0}) \cap P$). Let $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})$ and $\mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})$ be these minima, and D_1 and D_2 their respective validity domains.[2] Three cases are possible:

When $A(\mathbf{x_0})$ is only accessed by $\mathcal{R}_1$, i.e., $\mathbf{x_0} \in D_1 \setminus D_2$, the new access function for the reference $\mathcal{R}_1$ is equal to the number of array elements accessed before iteration $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})$ by both references $\mathcal{R}_1$ and $\mathcal{R}_2$. These elements are given by the union of the transformations by T_1 and T_2 of the iterations that are lexicographically smaller than $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})$.

$$\mathcal{E}_1(\mathbf{x_0}, \mathbf{p}) = \#(\{T_1(\mathbf{i}) \mid \mathbf{i} \in P \wedge \ \mathbf{i} \prec \mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})\} \cup \{T_2(\mathbf{i}) \mid \mathbf{i} \in P \wedge \ \mathbf{i} \prec \mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})\}).$$

When $A(\mathbf{x_0})$ is only accessed by $\mathcal{R}_2$, i.e., $\mathbf{x_0} \in D_2 \setminus D_1$, the new access function for the reference $\mathcal{R}_2$ is equal to the number of array elements accessed before iteration $\mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})$ by $\mathcal{R}_2$ union those accessed by $\mathcal{R}_1$ before iteration $\mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})$ (included).

$$\mathcal{E}_2(\mathbf{x_0}, \mathbf{p}) = \#(\{T_1(\mathbf{i}) \mid \mathbf{i} \in P \wedge \ \mathbf{i} \preceq \mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})\} \cup \{T_2(\mathbf{i}) \mid \mathbf{i} \in P \wedge \ \mathbf{i} \prec \mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})\}).$$

Finally, when $A(\mathbf{x_0})$ is accessed by both references $\mathcal{R}_1$ and $\mathcal{R}_2$, i.e., $\mathbf{x_0} \in D_1 \cap D_2$, the new access function $\mathcal{E}(\mathbf{x_0}, \mathbf{p})$, for $\mathcal{R}_1$ and $\mathcal{R}_2$, is given by the number of array elements accessed before the minimum of $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})$ and $\mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})$ by both references.

$$\mathcal{E}(\mathbf{x_0}, \mathbf{p}) = \#(\{T_1(\mathbf{i}) \mid \mathbf{i} \in P \ \wedge \ \mathbf{i} \prec \mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p}) \ \wedge \ \mathbf{i} \preceq \mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})\} \cup \{T_2(\mathbf{i}) \mid \mathbf{i} \in P \ \wedge \ \mathbf{i} \prec \mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p}) \ \wedge \ \mathbf{i} \prec \mathbf{i}_{min2}(\mathbf{x_0})\}).$$

We now consider two references $\mathcal{R}_1$ and $\mathcal{R}_2$, to the same array A, occurring in different loop nests such that $\mathcal{R}_1$ precedes $\mathcal{R}_2$. Let P_1 and P_2 be the polytopes defining the iteration spaces of $\mathcal{R}_1$ and $\mathcal{R}_2$, and consider again a datum $A(\mathbf{x_0})$ first accessed at iteration $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p}) \in P_1$ by the reference $\mathcal{R}_1$ (resp. at $\mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p}) \in P_2$ by $\mathcal{R}_2$). The reference $\mathcal{R}_1$ has to be optimized independently of the second one, since it is the first one accessing all the data it touches. In contrast the reference $\mathcal{R}_2$ is optimized in two different ways.

When $A(\mathbf{x_0})$ is also accessed by $\mathcal{R}_1$, i.e., $\mathbf{x_0} \in D_1 \cap D_2$, the new access function for $\mathcal{R}_2$ is equal to that of $\mathcal{R}_1$, which is given by the number of array elements accessed by $\mathcal{R}_1$ before iteration $\mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})$.

$$\mathcal{E}_2(\mathbf{x_0}, \mathbf{p}) = \mathcal{E}_1(\mathbf{x_0}, \mathbf{p}) = \#\{T_1(\mathbf{i}) \mid \mathbf{i} \in P_1 \ \wedge \ \mathbf{i} \prec \mathbf{i}_{min1}(\mathbf{x_0}, \mathbf{p})\}.$$

[2]The constraints on $\mathbf{x_0}$ and $\mathbf{p}$ for which the lexicographic minimum $\mathbf{i}_{minj}(\mathbf{x_0}, \mathbf{p})$ exists.

When $A(\mathbf{x_0})$ is only accessed by $\mathcal{R}_2$, i.e., $\mathbf{x_0} \in D_2 \setminus D_1$, the access function $\mathcal{E}_2(\mathbf{x_0}, \mathbf{p})$ is given by the number of all array elements accessed by $\mathcal{R}_1$ union those accessed before $\mathbf{i_{min2}}(\mathbf{x_0})$ by the reference $\mathcal{R}_2$.

$$\mathcal{E}_2(\mathbf{x_0}, \mathbf{p}) = \#(\{T_1(\mathbf{i}) \mid \mathbf{i} \in P_1\} \cup \{T_2(\mathbf{i}) \mid \mathbf{i} \in P_2 \ \wedge \ \mathbf{i} \prec \mathbf{i}_{min2}(\mathbf{x_0}, \mathbf{p})\}).$$

Of course, this generalizes to more than two references, appearing in possibly many different loop nests and accessing the same array, as described in [15].

Many improvements are ensured by this new data layout remapping method. First, it deals effectively with the general case for which the method in [15] fails to give a solution. This concerns the case where holes occur in the data space being accessed by a loop nest. It occurs, for example, when the coefficients of the loop indices in the access function are not coprime, as in $T(i, j) = 2i + 4j$ for $i, j \in \{1, 2\}$, where only even elements between 6 and 12 are accessed. This is solved using $\mathbb{Z}$-polyhedra integer transformations.

When two references to the same array occur in a loop nest, the old remapping technique does not optimize both references when they access the same data. Moreover, it optimizes each reference independently of the other, i.e., it omits the fact that data are alternatively accessed by both references, which leads to non-strict first-access ordered arrays. This has been solved by enumerating unions of transformed parametric $\mathbb{Z}$-polytopes. Finally, since it is difficult to separate the iteration space into convex regions, when data are accessed by both references, loop splittings are replaced by tests checking whether a datum has a lexicographic minimum according to each reference.

We have seen that the data layout transformation is based on counting integer points in unions of transformations of parametric polytopes. The following section describes the way we calculate these integer transformations.

4 Integer transformations of parametric polytopes

As we mentioned in Section 2, the problem of calculating the affine transformation of a polytope is equivalent to the elimination of the existential variables from the Presburger formula defining the transformation. The current section describes the way we deal with *parametric* Presburger formulas of the form:

$$\{\mathbf{x} \in \mathbb{Z}^d \mid \exists \mathbf{x}' \in \mathbb{Z}^{d'} : \ A\mathbf{x}' + B\mathbf{x} + C\mathbf{p} + \mathbf{c} = 0, \ A'\mathbf{x}' + B'\mathbf{x} + C'\mathbf{p} + \mathbf{c}' \geq 0\}, \qquad (1)$$

where A, B, C, A', B', C' are integer matrices, $\mathbf{x}$ and $\mathbf{x}'$ are subvectors of the data space, $\mathbf{c}$ and $\mathbf{c}'$ are constant vectors and $\mathbf{p}$ is a parameter vector.

4.1 Non-full-dimensional polytope preprocessing

The Presburger formulas we are interested in consist of a conjunction of affine equalities and inequalities in which a number of variables are linked with the existential quantifier $\exists$, as shown by formula (1). The conjunction of equalities and inequalities is called a non-full-dimensional polytope (when it is bounded).

A common way to handle existential variables is to project them out of a polytope. The first thing to do before projecting out the existential variables from a non-full-dimensional polytope is to remove its equalities, which automatically eliminates a certain number of

the existential variables.[3] The removal of a set of equalities from such a polytope must be done so that there exist integer values of the variables to be eliminated for each integer value of the other variables. In the following, we summarize an existing technique [17] for transforming a non-full dimensional polytope into a full-dimensional one. For simplicity, we consider here a non-parametric case, i.e., parameters are considered as regular variables. Indeed, the removal of equalities from a non-full-dimensional *parametric* polytope is done in the same way as for a non-parametric polytope. Notice, however, that *parameters* are never eliminated whatever the number of equalities.

Definition 4.1 *[23] A* d**-dimensional integer lattice** *is a subset of* $\mathbb{Z}^d$ *defined by linear combinations of linearly independent integer vectors, called lattice-generating vectors (or lattice basis), plus an affine part.*

$$L = \left\{ A\mathbf{x} + \mathbf{c} \mid \mathbf{x} \in \mathbb{Z}^d \right\}, \tag{2}$$

where A is a $d \times d$ integer matrix whose column vectors are the generators of the lattice and $\mathbf{c}$ is a constant integer vector. The integer lattice (2) can also be given in the *homogeneous* form:

$$L = \left\{ \left(\begin{array}{c|c} A & \mathbf{c} \\ \hline \mathbf{0} & 1 \end{array} \right) \left(\begin{array}{c} \mathbf{x} \\ 1 \end{array} \right) \;\middle|\; \mathbf{x} \in \mathbb{Z}^d \right\}, \tag{3}$$

where $\left(\begin{array}{c|c} A & \mathbf{c} \\ \hline \mathbf{0} & 1 \end{array} \right)$ is the *homogeneous* matrix generating latice L.

Consider a d-dimensional polytope P, defined by a non-redundant system of e equalities and e' inequalities, and let $E(\mathbf{x}_e, \mathbf{x}_{d'}) = 0$ be the subsystem of equalities, where $\mathbf{x}_e$ is a vector of e variables to be eliminated and $\mathbf{x}_{d'}$ is a vector of d' remaining variables ($d' = d - e$). In order to obtain a d'-full-dimensional polytope, it suffices to solve the system $E(\mathbf{x}_e, \mathbf{x}_{d'}) = 0$ in $\mathbf{x}_e$ as a function of $\mathbf{x}_{d'}$, and to substitute the result in the subsystem of inequalities. The resulting polytope is then intersected with an integer lattice defining the valid values of $\mathbf{x}_{d'}$ (we call this a $\mathbb{Z}$-*polytope*). Indeed, the solutions $\mathbf{x}_e$ to $E(\mathbf{x}_e, \mathbf{x}_{d'}) = 0$ may be *integers* for *only* a subset of $\mathbb{Z}^{d'}$. This subset is defined by an integer lattice, obtained by solving a system of modulo equalities.

The system of equalities $E(\mathbf{x}_e, \mathbf{x}_{d'}) = 0$ can be written in the matrix form:

$$A\mathbf{x}_e + B\mathbf{x}_{d'} + C = 0, \tag{4}$$

where A, B and C are respectively $e \times e$, $e \times d'$ and $e \times 1$ matrices. There is an *integer* solution in $\mathbf{x}_e$ to (4) if the d'-dimensional point $B\mathbf{x}_{d'} + C$ belongs to the lattice of integer points spanned by the column vectors of A.

Let $A = [H_A\ 0]U_A$ be the Hermit normal form of A. As the column vectors of A and those of H_A span the same lattice [24], the solutions to (4) are also solutions to $H_A\mathbf{x}_e + B\mathbf{x}_{d'} + C = 0$ and vis versa. Since H_A is invertible, the integer solutions to (4) are given by $\mathbf{x}_e = -H_A^{-1}(B\mathbf{x}_{d'} + C)$. As we only look for integer solutions, $H_A^{-1}(B\mathbf{x}_{d'} + C)$ must be integer. The general solution is given by an integer lattice [17]:

$$\mathbf{x}_{d'} = \{G\mathbf{x}' + \mathbf{x}_0 \mid \mathbf{x}' \in \mathbb{Z}^{d'}\}.$$

Two cases are possible once the equalities are eliminated.

[3]The number of eliminated variables is equal to the number of removed non-redundant equalities.

- There are no remaining existential variables (this is the case when the number of existential variables is small or equal to the number of equalities). In this case, the resulting full-dimensional $\mathbb{Z}$-polytope corresponds to the transformation.

- There still remain existential variables. In this case, we first calculate the preimage of the resulting full-dimensional $\mathbb{Z}$-polytope by the homogeneous matrix generating its lattice. Then, we eliminate the remaining existential variables as explained next in subsection 4.3. In order to preserve the original coordinates, we calculate the image of the resulting polytopes by the submatrix obtained by removing the lines and columns corresponding to existential variables from the above homogeneous matrix. The result is finally intersected with the sublattice defined by this latter submatrix.

In the following, we consider without loss of generality, that our formulas do not contain equalities, and we focus on eliminating the remaining existential variables.

4.2 Existential variable elimination and integer projection

The Fourier-Motzkin variable elimination procedure allows the elimination of an existential variable from a system of affine inequalities, defined on the set of rational numbers. Its main idea consists in rewriting the original system in the form of a set of lower and upper bounds on the variable to be eliminated. Then, each pair of lower and upper bounds of the form: $\{l(\mathbf{x},\mathbf{p}) \leq \beta z,\ \alpha z \leq u(\mathbf{x},\mathbf{p})\}$ is to be replaced by its equivalent $\alpha l(\mathbf{x},\mathbf{p}) \leq \beta u(\mathbf{x},\mathbf{p})$, where z is the existential variable chosen to be eliminated first, $l(\mathbf{x},\mathbf{p})$ and $u(\mathbf{x},\mathbf{p})$ are affine functions of the variables and parameters independent of z, and α and β are strictly positive integer constants. This procedure has been extended to integers by W. Pugh et al. [20, 21, 22] as follows:

Any pair of lower and upper bounds $\{l(\mathbf{x},\mathbf{p}) \leq \beta z,\ \alpha z \leq u(\mathbf{x},\mathbf{p})\}$ defines:

- an **exact shadow**, corresponding to the *rational* projection of the points which belong to this pair of constraints. This is given by: $\alpha l(\mathbf{x},\mathbf{p}) \leq \beta u(\mathbf{x},\mathbf{p})$.

- a **dark shadow**, corresponding to the convex part of the exact shadow in which any integer point has at least one integer preimage. This is given by: $\alpha l(\mathbf{x},\mathbf{p}) + (\alpha - 1)(\beta - 1) \leq \beta u(\mathbf{x},\mathbf{p})$. Notice that if $\alpha = 1$ or $\beta = 1$, the dark shadow is equal to the exact shadow.

The part of the exact shadow which does not belong to the dark shadow may contain integer points having integer preimages, and other integer points having only rational preimages, i.e., defining so-called holes (see Figure 2).

The Omega test [20] answers the question: "is there an integer point in the projection having at least an integer preimage?" as follows:

- if the exact shadow does not contain any integer point, the answer is: *no*,

- if the dark shadow contains at least one integer point, the answer is: *yes*,

- else, the answer is not obvious. In this case, we have to know whether the part of the exact shadow which does not belong to the dark shadow contains an integer point having integer preimages.

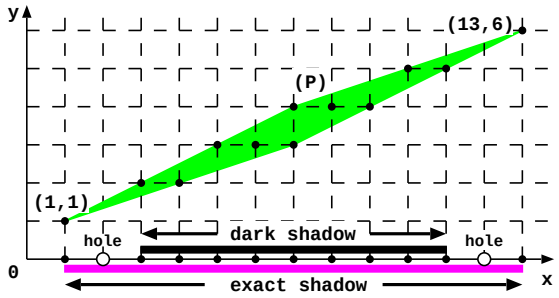

Figure 2: *The integer projection of a polytope.*

In order to answer this latter question, Pugh and Wonnacott [22] check whether the intersection of a certain number (function of α and β) of hyperplanes with the constraints of the original system contains an integer point. This solution provides new constraints with possibly extra existential variables, which complicates the answer to the two following questions:

- how many integer points are contained in the integer projection of the polytope?
- how to project the result along another dimension?

Example 1 *Consider the following example (introduced in [22]):*

$$\mathcal{S} = \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : \ 0 \leq 3y - x \leq 7 \wedge 1 \leq x - 2y \leq 5\}.$$

The exact shadow defined by the elimination of y is given by: $3 \leq x \leq 29$ and the dark shadow is given by: $5 \leq x \leq 27$.

Pugh and Wonnacott's algorithm [22] calculates the set of constraints containing the images which do not belong to the dark shadow as follows:

$$\begin{array}{l} \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : \ x = 3y \wedge 1 \leq y \leq 5 \ \} \cup \\ \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : \ x = 3y - 1 \wedge 2 \leq y \leq 6 \ \} \cup \\ \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} : \ x = 2y + 5 \wedge 5 \leq y \leq 12\}. \end{array}$$

While our method gives these images directly in the form: $\{x = 3, \ x = 29\}$.

4.3 Projection method

In this section, we will focus on the projection of a single pair of lower and upper bounds on the existential variable chosen to be eliminated first. Indeed, the projection of the whole polytope is simply obtained by intersecting the projections of all its pairs of bounds.[4]

Consider a pair of lower and upper bounds $\{l(\mathbf{x}, \mathbf{p}) \leq \beta z, \ \alpha z \leq u(\mathbf{x}, \mathbf{p})\}$. The projection of such a pair is given by the union of its dark shadow $(\alpha l(\mathbf{x}, \mathbf{p}) - \beta u(\mathbf{x}, \mathbf{p}) + (\alpha - 1)(\beta - 1) \leq 0)$ and another set of integer points which can not be obtained by applying simple rules. The following theorem defines the hyperplanes on which these latter points lie.

[4]The result is also intersected with the constraints that are independent of the eliminated variable.

Lemma 4.2 *Let x, y be two rational numbers and $\lceil x \rceil$, $\lceil y \rceil$ (resp. $\lfloor x \rfloor$, $\lfloor y \rfloor$) be their upper (resp. lower) integer parts. By definition, the following properties are equivalent.*

1. *$\exists n \in \mathbb{Z}$ such that $x \leq n \leq y$,*
2. *$\lceil x \rceil \leq \lfloor y \rfloor$,*
3. *$\lceil x \rceil \leq y$,*
4. *$x \leq \lfloor y \rfloor$.*

Theorem 4.3 *Consider the pair of bounds $\{l(\mathbf{x},\mathbf{p}) \leq \beta z,\ \alpha z \leq u(\mathbf{x},\mathbf{p})\}$ and let $l(\mathbf{x},\mathbf{p}) = l_l(\mathbf{x},\mathbf{p}) + c_l$ and $u(\mathbf{x},\mathbf{p}) = l_u(\mathbf{x},\mathbf{p}) + c_u$, where $l_l(\mathbf{x},\mathbf{p})$ and $l_u(\mathbf{x},\mathbf{p})$ are linear functions, c_l and c_u are integer constants and g is the greatest common divisor (gcd) of the coefficients of the variables and parameters in the linear function $\alpha l_l(\mathbf{x},\mathbf{p}) - \beta l_u(\mathbf{x},\mathbf{p})$. Then*

- *the points outside the dark shadow which belong to the integer projection, with respect to this pair of bounds, lie on hyperplanes of the form:*

$$\alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) + \gamma = 0, \tag{5}$$

  with $\gamma \in \mathbb{Z}$, $0 \leq \gamma \leq \alpha\beta - \alpha - \beta$ and g divides $(\beta c_u - \alpha c_l - \gamma)$.

- *the values of γ for which the hyperplane (5) contains the points we are interested in are those verifying the following inequality:*

$$\alpha(-l(\mathbf{x},\mathbf{p}) \bmod \beta) \leq \gamma, \tag{6}$$

  which is equivalent to:

$$\beta(u(\mathbf{x},\mathbf{p}) \bmod \alpha) \leq \gamma. \tag{7}$$

Proof 1 *We recall that the exact and the dark shadows are respectively given by $\alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) \leq 0$ and $\alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) \leq -(\alpha-1)(\beta-1)$ [22]. By definition, The part of the exact shadow containing the points outside the dark shadow which belong to the projection (see Figure 2) is given by: $-(\alpha\beta - \alpha - \beta) \leq \alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) \leq 0$ which is equivalent to: $\alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) + \gamma = 0$, where γ is an integer constant such that $0 \leq \gamma \leq \alpha\beta - \alpha - \beta$.*

Let $l(\mathbf{x},\mathbf{p}) = l_l(\mathbf{x},\mathbf{p}) + c_l$ and $u(\mathbf{x},\mathbf{p}) = l_u(\mathbf{x},\mathbf{p}) + c_u$, where $l_l(\mathbf{x},\mathbf{p})$ and $l_u(\mathbf{x},\mathbf{p})$ are linear functions of the variables and parameters, and c_l and c_u are integer constants. Substituting the values of $l(\mathbf{x},\mathbf{p})$ and $u(\mathbf{x},\mathbf{p})$ in (5) we obtain: $\alpha l_l(\mathbf{x},\mathbf{p}) - \beta l_u(\mathbf{x},\mathbf{p}) = \beta c_u - \alpha c_l - \gamma$, where $(\alpha l_l(\mathbf{x},\mathbf{p}) - \beta l_u(\mathbf{x},\mathbf{p}))$ is a linear function and $(\beta c_u - \alpha c_l - \gamma)$ is an integer constant. A necessary and sufficient condition for this hyperplane to contain integer points is that the gcd of the coefficients in the linear function $(\alpha l_l(\mathbf{x},\mathbf{p}) - \beta l_u(\mathbf{x},\mathbf{p}))$ divides the constant $(\beta c_u - \alpha c_l - \gamma)$.

On another hand, $\{l(\mathbf{x},\mathbf{p}) \leq \beta z,\ \alpha z \leq u(\mathbf{x},\mathbf{p})\}$ is equivalent to $\frac{l(\mathbf{x},\mathbf{p})}{\beta} \leq z \leq \frac{u(\mathbf{x},\mathbf{p})}{\alpha}$ (since $\alpha, \beta > 0$), where $\frac{l(\mathbf{x},\mathbf{p})}{\beta}$ and $\frac{u(\mathbf{x},\mathbf{p})}{\alpha}$ are rational functions. According to the properties 1 and 2 of Lemma 4.2, there exists an integer z such that $\frac{l(\mathbf{x},\mathbf{p})}{\beta} \leq z \leq \frac{u(\mathbf{x},\mathbf{p})}{\alpha}$ if and only if:

$$\left\lceil \frac{l(\mathbf{x},\mathbf{p})}{\beta} \right\rceil \leq \frac{u(\mathbf{x},\mathbf{p})}{\alpha}, \tag{8}$$

with $\left\lceil \frac{l(\mathbf{x},\mathbf{p})}{\beta} \right\rceil = \frac{1}{\beta}(l(\mathbf{x},\mathbf{p}) + (-l(\mathbf{x},\mathbf{p})) \bmod \beta)$. *Simplifying inequality (8), we obtain:* $\alpha(-l(\mathbf{x},\mathbf{p}) \bmod \beta) \leq \beta u(\mathbf{x},\mathbf{p}) - \alpha l(\mathbf{x},\mathbf{p})$, *with* $\beta u(\mathbf{x},\mathbf{p}) - \alpha l(\mathbf{x},\mathbf{p}) = \gamma$ *(according to equality (5)). Hence inequality (6) is satisfied. Applying Lemma 4.2, we can similarly prove inequality (7).*

Note 1 *If one of the bounds* $l(\mathbf{x},\mathbf{p})$ *or* $u(\mathbf{x},\mathbf{p})$ *is independent of the variables and the parameters, the elimination of the existential variable results in only the dark shadow, because in this case,* $\left\lceil \frac{l(\mathbf{x},\mathbf{p})}{\beta} \right\rceil$ *or* $\left\lfloor \frac{u(\mathbf{x},\mathbf{p})}{\alpha} \right\rfloor$ *is simply replaced by a constant.*

We calculate the solutions of inequality (6) or (7) in two different ways, depending on whether the coefficients α and β are coprime or not.

4.3.1 Case of coprime coefficients

Theorem 4.4 *Consider the pair of bounds* $\{l(\mathbf{x},\mathbf{p}) \leq \beta z,\ \alpha z \leq u(\mathbf{x},\mathbf{p})\}$. *The calculation of the values of* γ, *for which the hyperplane (5) contains the points outside the dark shadow which belong to the integer projection, does not depend on the variables and the parameters, i.e., it depends only on the constants* α *and* β. *In this case, the inequalities (6) and (7) are respectively equivalent to (9) and (10).*

$$\alpha((c_1\gamma) \bmod \beta) \leq \gamma, \tag{9}$$

$$\beta((c_2\gamma) \bmod \alpha) \leq \gamma, \tag{10}$$

where c_1 *and* c_2 *are integer constants such that* $c_1\alpha + c_2\beta = 1$.

Proof 2 *When the coefficients of the existential variables,* α *and* β *are coprime, the following property follows immediately from Bezout's identity theorem. There exists two integer constants* c_1 *and* c_2 *such that:*

$$c_1\alpha + c_2\beta = 1. \tag{11}$$

Multiplying the equality (5) by c_1, *we obtain:* $c_1\alpha l(\mathbf{x},\mathbf{p}) - c_1\beta u(\mathbf{x},\mathbf{p}) + c_1\gamma = 0$. *This is equivalent to* $(1 - c_2\beta)l(\mathbf{x},\mathbf{p}) - c_1\beta u(\mathbf{x},\mathbf{p}) + c_1\gamma = 0$ *(according to (11)). Hence*

$$l(\mathbf{x},\mathbf{p}) = \beta(c_2 l(\mathbf{x},\mathbf{p}) + c_1 u(\mathbf{x},\mathbf{p})) - c_1\gamma.$$

Substituting the value of $l(\mathbf{x},\mathbf{p})$ *in inequality (6), we obtain:*

$$\alpha((-\beta(c_2 l(\mathbf{x},\mathbf{p}) + c_1 u(\mathbf{x},\mathbf{p})) + c_1\gamma) \bmod \beta) \leq \gamma \Leftrightarrow \alpha(c_1\gamma \bmod \beta) \leq \gamma,$$

since $-\beta(c_2 l(\mathbf{x},\mathbf{p}) + c_1 u(\mathbf{x},\mathbf{p}))$ *is a multiple of* β. *In the same way, one can prove (10), starting from inequality (7).*

Example 2 *Consider the following Presburger formula:*

$$\mathcal{S} = \{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z} :\ 2 \leq 3y - x \leq 5 \wedge -1 \leq x - 2y \leq N - 1\}.$$

This set is equivalent to projecting out the variable y *from the polytope* P *pictured in Figure 2 (when* $N = 2$*), where* N *is a positive integer parameter.*

According to the pair of bounds $\{x - N + 1 \leq 2y,\ 3y \leq x + 5\}$*, we have:*

$$l(x) = x - N + 1,\ u(x) = x + 5,\ \alpha = 3,\ \beta = 2 \Rightarrow c_1 = 1,\ c_2 = -1.$$

The constraint on the dark shadow, corresponding to this pair of bounds, is $x \leq 3N + 5$*. The points of the projection which do not belong to the dark shadow are given by:*

$$\alpha l(x) - \beta u(x) + \gamma = 0,\ 0 \leq \gamma \leq \alpha\beta - \alpha - \beta \text{ and } \alpha((c_1\gamma) \bmod \beta) \leq \gamma$$

$$\Rightarrow x - 3N - 7 + \gamma = 0,\ 0 \leq \gamma \leq 1 \text{ and } 3(\gamma \bmod 2) \leq \gamma.$$

The only value of γ *satisfying these constraints is:* $\gamma = 0$*. The corresponding point (a hyperplane in the general case) is* $x = 3N + 7$*. Similarly, one can calculate the point* $x = 1$ *from the other pair of bounds* $\{x + 2 \leq 3y,\ 2y \leq x + 1\}$ *generating the constraint* $x \geq 3$ *on the dark shadow.*

The integer projection of the polytope P *can then be obtained by intersecting the projections of the two pairs, i.e.,* $S = \{x = 3N + 7 \cup x \leq 3N + 5\} \cap \{x = 1 \cup x \geq 3\}$*. Since* N *is positive, this set is equal to:*

$$\mathcal{S} = \{x \in \mathbb{Z} \mid x = 1 \vee 3 \leq x \leq 3N + 5 \vee x = 3N + 7\}.$$

4.3.2 Case of non-coprime coefficients

In the previous subsection, we showed the way we calculate the projection of a pair of bounds when the coefficients α and β are coprime. Let us now consider the case of non-coprime coefficients. In this case, the calculation of the values of γ, for which the hyperplane (5) contains the points of the projection, which are outside the dark shadow, depends on the constants α and β, and furthermore depends on the variables and parameters. Let $g' = gcd(\alpha, \beta)$, $\alpha' = \alpha/g'$, $\beta' = \beta/g'$ and g be the *gcd* of the coefficients of the variables and parameters in the linear function $\alpha' l_l(\mathbf{x}, \mathbf{p}) - \beta' l_u(\mathbf{x}, \mathbf{p})$, with $l(\mathbf{x}, \mathbf{p}) = l_l(\mathbf{x}, \mathbf{p}) + c_l$ and $u(\mathbf{x}, \mathbf{p}) = l_u(\mathbf{x}, \mathbf{p}) + c_u$ (see Theorem 4.3). One can then rewrite the equation of the hyperplane (5) as follows:

$$\alpha' l(\mathbf{x}, \mathbf{p}) - \beta' u(\mathbf{x}, \mathbf{p}) + \gamma = 0, \tag{12}$$

with $\gamma \in \mathbb{Z}$, $0 \leq \gamma \leq \alpha\beta' - \alpha' - \beta'$ and g divides $(\beta' c_u - \alpha' c_l - \gamma)$.
The inequalities (6) and (7) can be rewritten respectively in the form (13) and (14):

$$\alpha'(-l(\mathbf{x}, \mathbf{p}) \bmod \beta) \leq \gamma, \tag{13}$$

$$\beta'(u(\mathbf{x}, \mathbf{p}) \bmod \alpha) \leq \gamma. \tag{14}$$

In this case, it may happen that only a subset of the integer points of the hyperplane (12) belong to the projection. These points are defined by the intersection of the hyperplane with a union of lattices obtained by solving one of the following modulo equalities.

$$-l(\mathbf{x}, \mathbf{p}) \bmod \beta = \gamma',\ \text{with } 0 \leq \gamma' \leq \min\left(\left\lfloor \frac{\gamma}{\alpha'} \right\rfloor, \beta\right), \tag{15}$$

$$u(\mathbf{x}, \mathbf{p}) \bmod \alpha = \gamma',\ \text{with } 0 \leq \gamma' \leq \min\left(\left\lfloor \frac{\gamma}{\beta'} \right\rfloor, \alpha\right). \tag{16}$$

In practice, it is worth to consider equality (15) when $\beta < \alpha$ and equality (16) otherwise.

The solution to a modulo equality $f(\mathbf{x}, \mathbf{p}) \bmod a = b$ is a lattice of the form:

$$L = \left\{ \left(\begin{array}{c|c} A_{\mathbf{x}} & A_{\mathbf{p}} \end{array} \right) \left(\begin{array}{c} \mathbf{x} \\ \mathbf{p} \end{array} \right) + \mathbf{c} \;\middle|\; \mathbf{x} \in \mathbb{Z}^d,\ \mathbf{p} \in \mathbb{Z}^n \right\}, \tag{17}$$

where $A_{\mathbf{x}}, A_{\mathbf{p}}$ are integer matrices, $\mathbf{c}$ is an integer vector, $\mathbf{x}$ is a vector of the data space and $\mathbf{p}$ is a vector of parameters. We calculate this solution using the technique presented in [17] and summarized in subsection 4.1. Of course, only non-empty lattices and hyperplanes are taken into account.

Example 3 *Consider a pair of bounds in which the coefficients of the existential variable y are not coprime $\{x - N - 2 \leq 2y,\ 4y \leq x + 5\}$. We have:*

$$l(x) = x - N - 2,\ u(x) = x + 5,\ \alpha = 4,\ \beta = 2 \Rightarrow gcd(\alpha, \beta) = 2,\ \alpha' = 2,\ \beta' = 1.$$

The corresponding constraint on the dark shadow is $2x \leq 4N + 15 \Leftrightarrow x \leq 2N + 7$. The points outside the dark shadow and belonging to the projection lie on the following hyperplane:

$$x - 2N - 9 + \gamma = 0,\ \textit{such that}\ 0 \leq \gamma \leq 1\ \textit{and}\ 2((x - N - 2) \bmod 2) \leq \gamma.$$

For both values of γ, the solution to the above modulo inequality is a lattice:

$$L = \left\{ \left(\begin{array}{cc} 2 & 1 \\ 0 & 1 \end{array} \right) \left(\begin{array}{c} x \\ N \end{array} \right) + \left(\begin{array}{c} -2 \\ 0 \end{array} \right) \;\middle|\; x \in \mathbb{Z},\ N \in \mathbb{Z} \right\}.$$

Hence, the points $x = 2N + 9$ and $x = 2N + 8$ (obtained by substituting the values of γ in equality $x - 2N - 9 + \gamma = 0$) belong to the projection only if x and N belong to the lattice L.

The whole projection of the pair of bounds is then given by:

$$\mathcal{S} = \{x \in \mathbb{Z} \mid (x = 2N + 8 \wedge (x, N) \in L) \vee (x = 2N + 9 \wedge (x, N) \in L) \vee x \leq 2N + 7\}.$$

Projecting out a first existential variable may result in a union of $\mathbb{Z}$-polytopes, i.e., a union of intersections of polytopes and lattices of the form (17). This may occur when the the coefficients of the existential variable are not coprime. Therefore, in order to project out a second variable, this union has to be projected again, and so on. The projection of a $\mathbb{Z}$-polytope is obtained by first transforming the polytope according to its associated lattice, then projecting it as explained before and finally rewriting it as a function of its original variables and parameters.

In the following section, we will be interested in counting integer points in arbitrary unions of parametric $\mathbb{Z}$-polytopes.

5 Counting integer points in unions of parametric $\mathbb{Z}$-polytopes

Counting integer points in unions of parametric $\mathbb{Z}$-polytopes is very useful, for counting points in integer transformations of polytopes, but also to solve many other problems,

such as non-unit stride loop nest analyses. In the following, we will discuss an algorithm dealing with general parametric $\mathbb{Z}$-polytopes of the form $\mathcal{Z} = P \cap L$, with:

$$P = \left\{ \mathbf{x} \in \mathbb{Q}^d, \mathbf{p} \in \mathbb{Z}^n \;\middle|\; \left(\begin{array}{c|c} A_{\mathbf{x}} & A_{\mathbf{p}} \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} + \mathbf{a} \geq 0 \right\},$$
$$L = \left\{ \left(\begin{array}{c|c} B_{\mathbf{x}} & B_{\mathbf{p}} \end{array} \right) \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} + \mathbf{b} \;\middle|\; \mathbf{x} \in \mathbb{Z}^d,\ \mathbf{p} \in \mathbb{Z}^n \right\},$$

where P is a parametric polytope, L is a parametric integer lattice, $A_{\mathbf{x}}, A_{\mathbf{p}}, B_{\mathbf{x}}$ and $B_{\mathbf{p}}$ are integer matrices, $\mathbf{a}$ and $\mathbf{b}$ are integer vectors, $\mathbf{x}$ is a vector of the data space and $\mathbf{p}$ is a vector of parameters.

Let $\mathcal{Z}_1 = P_1 \cap L_1$ and $\mathcal{Z}_2 = P_2 \cap L_2$ be two $\mathbb{Z}$-polytopes. The number of integer points in $\mathcal{Z}_1$ (resp. in $\mathcal{Z}_2$) is equal to the number of points in P_1' (resp. in P_2'), where P_1' and P_2' are the transformations of P_1 and P_2 respectively by the matrices defining the lattices L_1 and L_2. Unfortunately, the number of integer points in $\mathcal{Z}_1 \cup \mathcal{Z}_2$ is obviously not equal to that in $P_1' \cup P_2'$ since the applied transformation preserves the number of points in $\mathcal{Z}_1$ and $\mathcal{Z}_2$ but does not preserve their original coordinates.

Example 4 *Consider the two $\mathbb{Z}$-polytopes pictured in Figure 3, $\mathcal{Z}_1 = P_1 \cap L_1$ and $\mathcal{Z}_2 = P_2 \cap L_2$, where dots belong to $\mathcal{Z}_1$, squares belong to $\mathcal{Z}_2$ and diamonds belong to both $\mathbb{Z}$-polytopes:*

$$P_1 = \{(x, y) \in \mathbb{Q}^2 \mid 1 \leq x \leq 10 \ \wedge \ 3 \leq y \leq 7\},$$

$$L_1 = \left\{ \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \;\middle|\; (x', y') \in \mathbb{Z}^2 \right\},$$

$$P_2 = \{(x, y) \in \mathbb{Q}^2 \mid 3 \leq x \leq 12 \ \wedge \ 1 \leq y \leq 6\},$$

$$L_2 = \left\{ \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \;\middle|\; (x', y') \in \mathbb{Z}^2 \right\}.$$

Substituting $x = 2x' + 1$ and $y = 2y' + 1$ in P_1 (resp. $x = 3x'$ and $y = 2y' + 1$ in P_2) we respectively obtain P_1' and P_2' in which the numbers of integer points are respectively 15 *and* 12*:*

$$P_1' = \{(x', y') \in \mathbb{Z}^2 \mid 0 \leq 2x' \leq 9 \ \wedge \ 1 \leq y' \leq 3\},$$

$$P_2' = \{(x', y') \in \mathbb{Z}^2 \mid 1 \leq x' \leq 4 \ \wedge \ 0 \leq 2y' \leq 5\}.$$

One can check that the number of integer points in $P_1' \cup P_2'$ is 19*, whereas that in $\mathcal{Z}_1 \cup \mathcal{Z}_2$ is* 23 *as showed in Figure 3.*

To the best of our knowledge, the only technique describing how to enumerate such unions is due to B.Meister [16]. His method is *lattice-union* based, which is exponential in the size of lattice generators and their least common multiple. This method has not been implemented. In contrast, our method is *lattice-intersection* based, which is polynomial since the intersection of two lattices results in only one lattice, whatever their generators. We provide a sketch of complexity analysis after a short description of the algorithm.

In our algorithm, we process on a set of signed $\mathbb{Z}$-polytopes. The number of integer points in the union of two $\mathbb{Z}$-polytopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$, (say $\mathcal{E}(\mathcal{Z}_1 \cup \mathcal{Z}_2)$) is equal to the number

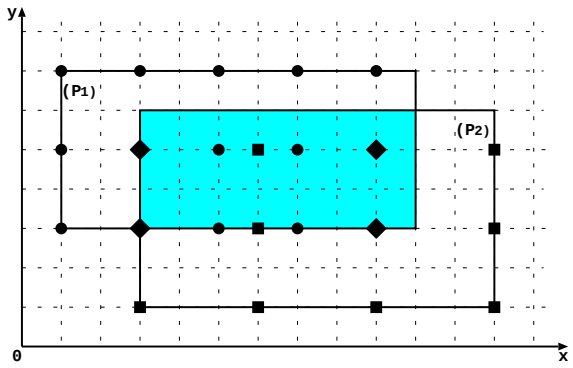

Figure 3: *Union of two $\mathbb{Z}$-polytopes*

of points in $\mathcal{Z}_1$, plus the number of points in $\mathcal{Z}_2$, minus the number of points in $\mathcal{Z}_1 \cap \mathcal{Z}_2$. Therefore, the signs of $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are set to $+1$ and that of $\mathcal{Z}_1 \cap \mathcal{Z}_2$ is set to -1.

The integer points in a union of three $\mathbb{Z}$-polytopes is counted as follows:

$$\begin{aligned}
&\mathcal{E}(\mathcal{Z}_1 \cup (\mathcal{Z}_2 \cup \mathcal{Z}_3)) = \mathcal{E}(\mathcal{Z}_1) + \mathcal{E}(\mathcal{Z}_2 \cup \mathcal{Z}_3) - \mathcal{E}(\mathcal{Z}_1 \cap (\mathcal{Z}_2 \cup \mathcal{Z}_3)) = \\
&\quad \mathcal{E}(\mathcal{Z}_1) + (\mathcal{E}(\mathcal{Z}_2) + \mathcal{E}(\mathcal{Z}_3) - \mathcal{E}(\mathcal{Z}_2 \cap \mathcal{Z}_3)) - \mathcal{E}((\mathcal{Z}_1 \cap \mathcal{Z}_2) \cup (\mathcal{Z}_1 \cap \mathcal{Z}_3)) = \\
&\quad\quad \mathcal{E}(\mathcal{Z}_1) + \mathcal{E}(\mathcal{Z}_2) + \mathcal{E}(\mathcal{Z}_3) - \mathcal{E}(\mathcal{Z}_2 \cap \mathcal{Z}_3) - \mathcal{E}(\mathcal{Z}_1 \cap \mathcal{Z}_2) - \mathcal{E}(\mathcal{Z}_1 \cap \mathcal{Z}_3) + \mathcal{E}(\mathcal{Z}_1 \cap \mathcal{Z}_2 \cap \mathcal{Z}_3).
\end{aligned}$$

Of course, this generalizes to any number of $\mathbb{Z}$-polytopes. Finally, the number of points in the union is simply given by summing the number of points in each of the resulting $\mathbb{Z}$-polytopes.

Before counting points in a $\mathbb{Z}$-polytope $\mathcal{Y}_i = P_i \cap L_i$ we need to transform the matrix generating the lattice: $L_i = \left\{ \left(\begin{array}{c|c} B_{\mathbf{x}} & B_{\mathbf{p}} \end{array} \right) \left(\begin{array}{c} \mathbf{x} \\ \mathbf{p} \end{array} \right) + \left(\begin{array}{c} \mathbf{b_x} \\ \mathbf{b_p} \end{array} \right) \;\middle|\; \mathbf{x} \in \mathbb{Z}^d, \mathbf{p} \in \mathbb{Z}^n \right\}$ into a new matrix of the form: $M = \left(\begin{array}{c|c} B_{\mathbf{xx}} & B_{\mathbf{xp}} \\ \hline 0 & B_{\mathbf{pp}} \end{array} \right)$. I.e., M is a matrix in which the rows which transform the parameter space are *independent* of the variables.[5] This is required to keep the data space compressed when rewriting the transformed polytope as a function of its original parameters. We calculate matrix M from the Hermit normal form [24] of $\left(\begin{array}{c|c} B_{\mathbf{x}} & B_{\mathbf{p}} \end{array} \right)$. We then apply the affine transformation $\left(\begin{array}{c|c} M & \left(\begin{array}{c} \mathbf{b_x} \\ \mathbf{b_p} \end{array} \right) \end{array} \right)$ to P_i to get an ordinary polytope P'. Then, P' is rewritten as a function of the original parameters using the submatrix $(B_{\mathbf{pp}}|\mathbf{b_p})$ and finally, we use our counting algorithm [26] to calculate the Ehrhart polynomial corresponding to the number of integer points in the resulting polytope. Note that when the submatrix $B_{\mathbf{pp}}$ is not equal to the identity matrix, the polytope is valid for only the parameter values generated by the lattice $L_{\mathbf{p}}$ whose basis is $B_{\mathbf{pp}}$ and affine part is $\mathbf{b_p}$. In this case, the resulting Ehrhart polynomial is to be multiplied by one if the parameter values are valid (i.e. if they belong to the points spanned by the lattice $L_{\mathbf{p}}$) and zero otherwise.

[5]The matrix M generates the same integer points as the original matrix.

Algorithm 1 Parametric $\mathbb{Z}$-polytope union enumeration

Input: $List = \{(List[i].P,\ List[i].L)\}$ with $1 \le i \le size(List)$
List of pairs (Polyhedron, Lattice)
Output: $\mathcal{E}(List)$ = Number of integer points in $List$
Variables: $I = \{(I[i].sign,\ I[i].P,\ I[i].L)\}$, with $sign = \pm 1$
$O = \{(O[i].sign,\ O[i].P,\ O[i].L)\}$

1. For $i = 1$ to $size(List)$
   $I[i] = (+1,\ List[i].P,\ List[i].L)$
2. $O[1] = I[1],\ \mathcal{E}(List) = 0$
3. For $i = 2$ to $size(List)$
   (a) For $j = 1$ to $size(O)$
   If $O[j].P \cap I[i].P \neq \phi$ and $O[j].L \cap I[i].L \neq \phi$
   $O = O + (-O[j].sign \times I[i].sign,\ O[j].P \cap I[i].P,\ O[j].L \cap I[i].L)$
   (b) $O = O + I[i]$
4. For $i = 1$ to $size(O)$
   (a) $P = VariableSubstitution(O[i].P,\ O[i].L)$
   (b) $\mathcal{E}(List) = \mathcal{E}(List) + O[i].sign \times Enumerate(P)$

Example 5 *Consider parametric versions of $\mathbb{Z}$-polytopes $Z_1 = P_1 \cap L_1$ and $Z_2 = P_2 \cap L_2$ from Example 4, with*

$$P_1 = \{(x, y) \in \mathbb{Q}^2 \mid 1 \le x \le N + 5 \ \wedge\ 3 \le y \le 7\},$$

$$L_1 = \left\{ \left(\begin{array}{cc|c} 2 & 0 & 3 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x' \\ y' \\ N' \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \;\middle|\; (x', y') \in \mathbb{Z}^2,\ N' \in \mathbb{Z} \right\},$$

$$P_2 = \{(x, y) \in \mathbb{Q}^2 \mid 3 \le x \le 2N + 7 \ \wedge\ 1 \le y \le 6\},$$

$$L_2 = \left\{ \left(\begin{array}{cc|c} 3 & 0 & 0 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 3 \end{array}\right) \begin{pmatrix} x' \\ y' \\ N' \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} \;\middle|\; (x', y') \in \mathbb{Z}^2,\ N' \in \mathbb{Z} \right\},$$

where $N \in \mathbb{Z}^+$ is a parameter. The number of integer points in the union $Z_1 \cup Z_2$ is given by:

$$\mathcal{E}(Z_1 \cup Z_2) = \mathcal{E}(Z_1) + \mathcal{E}(Z_2) - \mathcal{E}(Z_1 \cap Z_2).$$

Let $(Z_1 \cap Z_2) = Z_3 = (P_3 \cap L_3)$*, with*

$$P_3 = P_1 \cap P_2 = \{(x, y) \in \mathbb{Q}^2 \mid 3 \leq x \leq N + 5 \ \wedge \ 3 \leq y \leq 6\},$$

$$L_3 = L_1 \cap L_2 = \left\{ \left(\begin{array}{cc|c} 3 & 0 & 0 \\ 0 & 2 & 0 \\ \hline 3 & 0 & 6 \end{array} \right) \left(\begin{array}{c} x' \\ y' \\ N' \end{array} \right) + \left(\begin{array}{c} 2 \\ 1 \\ 3 \end{array} \right) \ \middle| \ (x', y') \in \mathbb{Z}^2, \ N' \in \mathbb{Z} \right\}.$$

The number of integer points in Z_3 *is calculated as follows:*

First of all, the basis $\left(\begin{array}{cc|c} 3 & 0 & 0 \\ 0 & 2 & 0 \\ \hline 3 & 0 & 6 \end{array} \right)$ *of lattice* L_3 *is transformed into a new basis* $\left(\begin{array}{cc|c} 6 & 0 & 3 \\ 0 & 2 & 0 \\ \hline 0 & 0 & 3 \end{array} \right)$ *in which the variable coefficients in the parameter row are all equal to zero. The new basis spans the same integer points as the original one since it is calculated from its Hermit normal form.* $\mathbb{Z}$*-polytope* Z_3 *is then transformed into a regular polytope* P *given by the preimage of* P_3 *by the homogeneous matrix* $\left(\begin{array}{ccc|c} 6 & 0 & 3 & 2 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 3 & 3 \\ \hline 0 & 0 & 0 & 1 \end{array} \right)$*:*

$$P = \{(x', y') \in \mathbb{Q}^2 \mid -3N' + 1 \leq 6x' \leq 6 \ \wedge \ 2 \leq 2y' \leq 5\}.$$

Before counting points in P*, we need to write it as a function of the original parameter* N*. To do this, it suffices to calculate its image by the matrix* $M = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 3 \\ \hline 0 & 0 & 0 & 1 \end{array} \right)$*, and we obtain:*

$$P' = \{(x', y') \in \mathbb{Q}^2 \mid -N + 4 \leq 6x' \leq 6 \ \wedge \ 2 \leq 2y' \leq 5\}.$$

The number of points in this polytope is given [26] in the form:

$$\mathcal{E}(P') = \frac{1}{3}N + \left[2, \frac{5}{3}, \frac{4}{3}, 1, \frac{8}{3}, \frac{7}{3}\right]_N,$$

where $\left[2, \frac{5}{3}, \frac{4}{3}, 1, \frac{8}{3}, \frac{7}{3}\right]_N$ *is a periodic number whose value is* 2 *when* $N \bmod 6 = 0$*,* $\frac{5}{3}$ *when* $N \bmod 6 = 1$ *and so on.*

The third row in matrix M *states that* $N = 3N' + 3$*, with* $N' \in \mathbb{Z}$*. In other words,* N *must be a multiple of* 3 *(*$N \bmod 3 = 0$*). Therefore the resulting polynomial is multiplied by a periodic number* $[1, 0, 0]_N$ *which is equal to one when* $N \bmod 3 = 0$ *and zero otherwise. The result is*

$$\mathcal{E}(Z_3) = \left[\frac{1}{3}, 0, 0\right]_N N + [2, 0, 0, 1, 0, 0]_N$$

The numbers of points in Z_1 *and* Z_2 *are obtained in a similar way as:*

$$\mathcal{E}(Z_1) = \frac{3}{2}N + \left[9, \frac{15}{2}\right]_N,$$

$$\mathcal{E}(Z_2) = [2,0,0]_N N + [3,0,0]_N.$$

Finally, the number of points in $Z_1 \cup Z_2$ is given by:

$$\mathcal{E}(Z_1 \cup Z_2) = \mathcal{E}(Z_1) + \mathcal{E}(Z_2) - \mathcal{E}(Z_3) = \left[\frac{19}{6}, \frac{3}{2}, \frac{3}{2}\right]_N N + \left[10, \frac{15}{2}, 9, \frac{19}{2}, 9, \frac{15}{2}\right]_N.$$

The complexity of Algorithm 1 depends on the complexity of the significant $\mathbb{Z}$-polytope operations (step 3.(a)), the complexity of counting integer points in a parametric polytope (step 4.(b)) and the number of resulting $\mathbb{Z}$-polytopes in the set O (calculated in step 3). The only significant $\mathbb{Z}$-polytope operation used in this algorithm is the intersection, which is polynomial since the intersection of two polytopes is simply given by concatenating their constraints, and the intersection of two lattices reduces to solving a system of linear equalities [18], which is polynomial in the input size [24]. Counting integer points in a parametric polytope is also polynomial in the input size (for fixed dimension), as we showed in [26]. Finally, the resulting $\mathbb{Z}$-polytopes in the set O is given by the inclusion-exclusion principle. When the number of input $\mathbb{Z}$-polytopes, in the set I is fixed, the inclusion-exlusion principle provides a polynomial number of $\mathbb{Z}$-polytopes.[6] Hence the whole algorithm is polynomial in the input size (for fixed dimension and fixed number of input $\mathbb{Z}$-polytopes).

6 Experiments

These experiments were undertaken with PolyLib version 5.22 and Barvinok version 0.20. The first library is used to realize polyhedral operations, while the second one is used to count integer points in parametric polytopes. In addition, Verdoolaege et al. [25] use the PIP library [10] to handle the remaining existential variables (when their rewriting rules fail to eliminate them). Verdoolaege's implementation also (optionally) uses the Omega library [13] to simplify the input polytopes. In almost all our examples, Verdoolaege's method gives better results when it uses Omega preprocessing. We therefore activated this option for these experiments. The test set was built to be representative of a large number of cases. The dimension of the polytopes vary between 3 and 7, the number of parameters between 1 and 3, and the number of eliminated variables from 1 to 5. In almost all our tests, the exact shadow is not equal to the dark shadow, i.e., the integer projection is different from the rational one.

In these experiments, the execution times and the output sizes are plotted as a function of the number of $\mathbb{Z}$-polytopes that are generated by our method, which is proportional to the number of holes in the projection and the size of the existential variable coefficients. We compare our work against the one of Verdoolaege et al. [25] using Ehrhart quasi-polynomials representation as lookup-table and fractional enumerators. Let $f(\mathbf{p})$ be a rational affine function of the parameters. The lookup-table representation expresses a periodic number $(f(\mathbf{p}) \bmod n)$ as a multi-dimensional table, whose dimension is the number of parameters, and where the number of elements in each dimension is the *period* of the corresponding parameter. When the periods are large, the lookup-table representation is known to be exponential [26]. In order to avoid the exponential behavior of

[6] Only non-empty sets are taken into account.

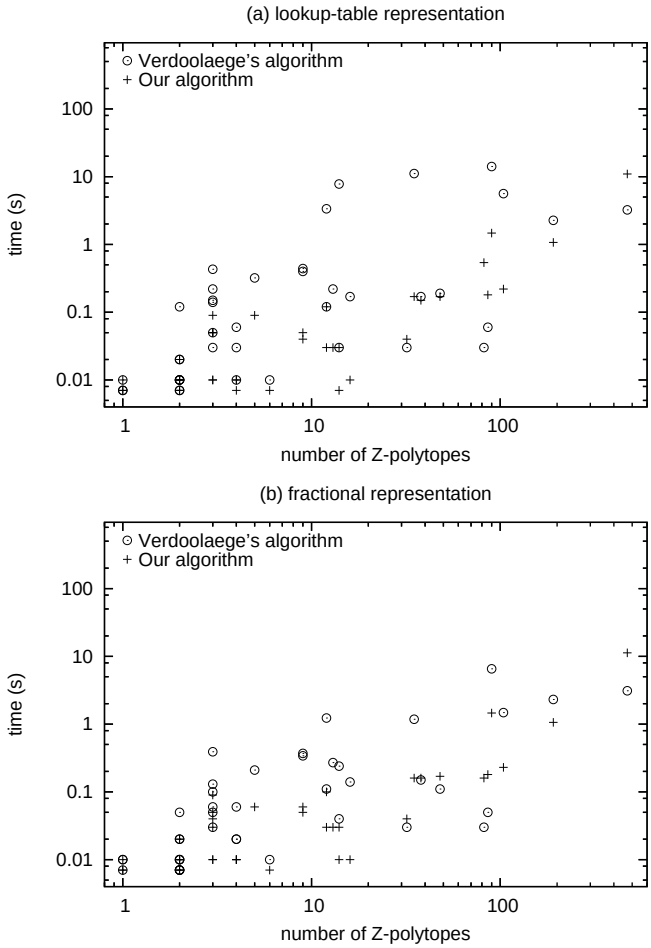

Figure 4: *Execution time comparison with Verdoolaege's implementation.*

lookup tables, a fractional representation was proposed [26]. In this representation, a periodic number $(f(\mathbf{p}) \bmod n)$ is expressed as a function of the fractional part of $f(\mathbf{p})$, with $frac(f(\mathbf{p})) = f(\mathbf{p}) - \lfloor f(\mathbf{p}) \rfloor$. The disadvantage of the fractional representation is that additions and multiplications of fractional numbers may not be fully simplified. Indeed, when the periods are not large, the result size of the fractional representation may be larger than the one of the lookup-table representation.

Figure 4 shows that, for almost all these tests, our execution times are lower than Verdoolaege's ones. The ratio is more significant when using lookup-table representation. Indeed, most of the polynomials generated by Verdoolaege's method have larger periods than those generated by ours. Therefore, when using the fractional representation, our times do not change much, while Verdoolaege's times are better fore some examples. Note that Verdoolaege's method does not calculate the real projection, but an equivalent set of polytopes, having the same number of integer points. These polytopes are sometimes of larger dimensions, which may increase the execution time.

Figure 5 shows the difference between the sizes of the output polynomials. Again,

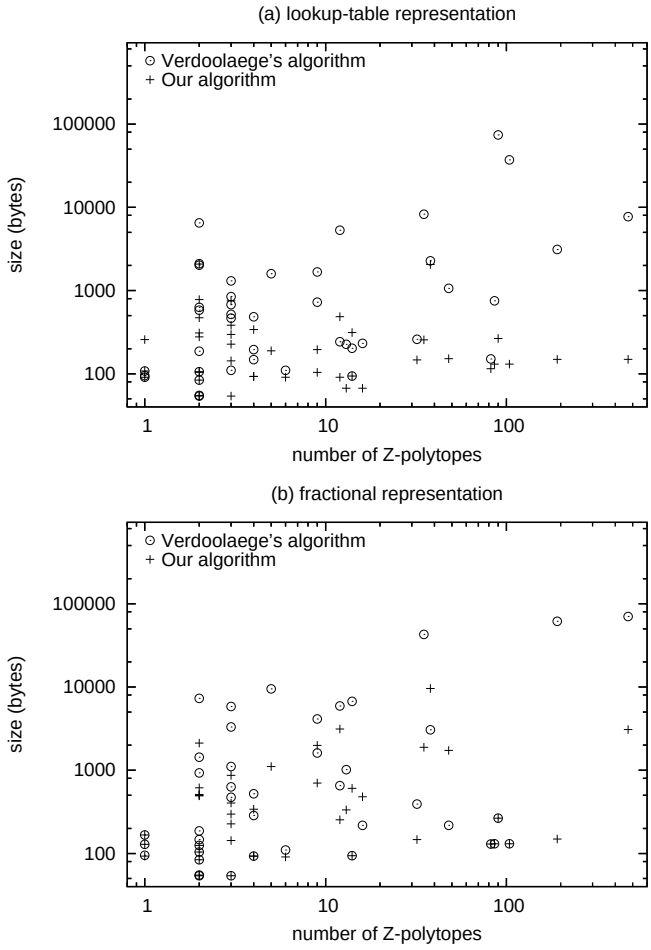

Figure 5: *Output size comparison with Verdoolaege's implementation.*

since our method generates smaller periods, the resulting polynomials are smaller than Verdoolaege's. Note that our output sizes are better with lookup-table representation, because there are no large periods in these tests.

7 Conclusion

We have presented in this paper a new technique for calculating the transformation of integer points in parametric polytopes. The basic idea is the Fourier-Motzkin variable elimination procedure, processing pairs of lower and upper bounds on the existential variables to be eliminated. The computation time of our method depends on the coefficients of the existential variable in each pair. If these coefficients are small, the elimination of the existential variable results in a relatively small number of $\mathbb{Z}$-polytopes. In contrast, if the coefficients are large, the result could be given by a possibly large set of $\mathbb{Z}$-polytopes, increasing the execution time. Further work goes on to minimize this number of $\mathbb{Z}$-polytopes, by choosing the best order and rules of variable eliminations. We also

presented a polynomial algorithm (for fixed dimension) to count integer points in unions of a fixed number of parametric $\mathbb{Z}$-polytopes. The proposed algorithms are implemented using the Polyhedral and Barvinok libraries [14, 26]. This work allows to compress efficiently the data space of arrays accessed by affine functions in loop nests: only data that are actually used are allocated in memory. We extended and improved the data locality optimization presented in [15]: the data are allocated to memory in the order they are accessed for the first time by the loop nests. Our method is more general and increases the number of stride-one accesses compared to the one of [15]. This work also has many other applications in parametric affine loop nest analysis and optimization.

References

[1] A. Barvinok and K. Woods. Short rational generating functions for lattice point problems. *Journal of the American Mathematical Society*, 16:657–979, 2003.

[2] K. Beyls and E. D'Hollander. Reuse distance as a metric for cache behavior. In *IASTED conference on Parallel and Distributed Computing and Systems 2001 (PDCS01)*, pages 617–662, 2001.

[3] B. Boigelot and L. Latour. Counting the solutions of Presburger equations without enumerating them. *Theoretical Computer Science*, 313(1):17–29, Feb. 2004.

[4] P. Boulet and X. Redon. Communication pre-evaluation in HPF. In *EUROPAR'98*, volume 1470 of *LNCS*, pages 263–272. Springer Verlag, 1998.

[5] S. Chatterjee, E. Parker, P. J. Hanlon, and A. R. Lebeck. Exact analysis of the cache behavior of nested loops. In *Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation*, pages 286–297. ACM Press, 2001.

[6] P. Clauss and V. Loechner. Parametric Analysis of Polyhedral Iteration Spaces. *Journal of VLSI Signal Processing*, 19(2):179–194, July 1998.

[7] P. D'Alberto, A. Veidembaum, A. Nicolau, and R. Gupta. Static analysis of parameterized loop nests for energy efficient use of data caches. In *Workshop on Compilers and Operating Systems for Low Power (COLP01)*, Sept. 2001.

[8] E. Ehrhart. Polynômes arithmétiques et méthode des polyèdres en combinatoire. *International Series of Numerical Mathematics*, 35, 1977.

[9] P. Feautrier. Parametric integer programming. *Operationnelle/Operations Research*, 22(3):243–268, 1988.

[10] P. Feautrier, J. Collard, and C. Bastoul. Solving systems of affine (in)equalities. Technical report, PRiSM, Versailles University, 2002.

[11] S. Ghosh, M. Martonosi, and S. Malik. Cache miss equations: a compiler framework for analyzing and tuning memory behavior. *ACM Transactions on Programming Languages and Systems*, 21(4):703–746, 1999.

[12] F. Heine and A. Slowik. Volume driven data distribution for NUMA-machines. In *Proceedings from the 6th International Euro-Par Conference on Parallel Processing*, pages 415–424, 2000.

[13] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott. The Omega Library. Technical report, Institue for advanced computer studies, University of Maryland, College Park, 1996.

[14] V. Loechner. Polylib: A library for manipulating parameterized polyhedra. Technical report, LSIIT - ICPS UMR7005 Univ. Louis Pasteur-CNRS, Mar. 1999.

[15] V. Loechner, B. Meister, and P. Clauss. Precise data locality optimization of nested loops. *Journal of Supercomputing*, 21(1):37–76, 2002.

[16] B. Meister. Projecting periodic polyhedra for loop nest analysis. In *Proceedings of the 11th Workshop on Compilers for Parallel Computers (CPC 04), Kloster Seeon, Germany*, pages 13–24, July 2004.

[17] B. Meister. *Stating and Manipulating Periodicity in the Polytope Model. Applications to Program Analysis and Optimization.* PhD thesis, December 2004.

[18] S. P. K. Nookala and T. Risset. A library for Z-polyhedral operations. Technical report, 1330, Irisa, 2000.

[19] E. Parker and S. Chatterjee. An automata-theoretic algorithm for counting solutions to Presburger formulas. In *Compiler Construction 2004*, volume 2985 of *Lecture Notes in Computer Science*, pages 104–119, Apr. 2004.

[20] 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*, pages 4–13. ACM Press, 1991.

[21] W. Pugh. Counting solutions to Presburger formulas: how and why. In *SIGPLAN Conference on Programming Language Design and Implementation (PLDI'94)*, pages 121–134, 1994.

[22] W. Pugh and D. Wonnacott. Experiences with constraint-based array dependence analysis. In *Principles and Practice of Constraint Programming*, pages 312–325, 1994.

[23] P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra using a canonical representation. *Parallel Processing Letters*, 7(2):181–194, 1997.

[24] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1986.

[25] S. Verdoolaege, K. Beyls, M. Bruynooghe, and F. Catthoor. Experiences with enumeration of integer projections of parametric polytopes. In R. Bodik, editor, *Compiler Construction: 14th International Conference*, volume 3443, pages 91–105, Edinburgh, 3 2005. Springer.

[26] S. Verdoolaege, R. Seghir, K. Beyls, V. Loechner, and M. Bruynooghe. Analytical computation of Ehrhart polynomials: Enabling more compiler analyses and optimizations. In *Proceedings of International Conference on Compilers, Architectures, and Synthesis for Embedded Systems, Washington D.C.*, pages 248–258, Sept. 2004.

[27] S. Verdoolaege and K. Woods. Counting with rational generating functions, 2005. http://arxiv.org/abs/math/0504059.

[28] Y. Zhao and S. Malik. Exact memory size estimation for array computations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(5):517–521, October 2000.