Memory Optimization by Counting Points in Integer Transformations of Parametric Polytopes

Rachid Seghir
seghir@icps.u-strasbg.fr

Vincent Loechner
loechner@icps.u-strasbg.fr

LSIIT (UMR 7005 CNRS), ICPS
Université Louis Pasteur, Strasbourg, France

ABSTRACT

Memory size reduction and memory accesses optimization are crucial issues for embedded systems. In the context of affine programs, these two challenges are classically tackled by array linearization, cache access optimization and memory size computation. Their formalization in the polyhedral model reduce to solving the following problem: *count the number of solutions of a Presburger formula*.

In this paper we propose a novel algorithm that answers this question. We solve the Presburger formula whose solution is a union of parametric $\mathbb{Z}$-polytopes and we propose an algorithm to count points in such a union of parametric $\mathbb{Z}$-polytopes. These algorithms were implemented and we compare them to other existing methods.

Categories and Subject Descriptors

D.3.4 [**Programming Languages**]: Processors—*Compilers, Optimization*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Counting problems*

General Terms

Algorithms, Performance.

Keywords

Exact memory size computation, array linearization, cache access optimization, polytope model, counting lattice points in polytopes.

1. INTRODUCTION

Two parameters particularly impact the size and the cost of an embedded system: memory size, and power consumption. In many applications, and especially in real-time multimedia processing systems, a large part of power consumption is due to data transfers and memory access operations. For these two reasons -memory size reduction and memory accesses optimization- memory optimization is a crucial issue in compilation techniques for high performance embedded computing [19]. The class of *"affine programs"* (affine access functions to arrays in affine bounded loop nests) has been tackled in the past by many researchers, since they occur frequently and are resource-consuming, in applications like digital audio/video, imaging, graphics, compression/decompression, etc. In this context, array linearization [28], cache access optimization [11, 5, 15], and memory size computation [32, 33] reduce to the problem of counting the number of images of integer points in a polytope (or a $\mathbb{Z}$-polytope[1]) by an affine function, or equivalently counting the solutions to a Presburger formula. If the considered programs contain unknown variables at compile-time, this problem has to be solved analytically, and the solution has to be expressed as a function of these parameters. In such a case, we manipulate parameteric $\mathbb{Z}$-polytopes.

The first solutions proposed approximations or minima of this number [32, 25] or solved the subproblem of count the integer points in polytopes with one [1, 7] or many [4, 3, 30] parameters. Then, some exact methods to count transformation of polytopes were proposed for the non-parametric case [20, 2, 33].

Pugh [22] first proposed an algorithm to solve the general-case problem, based on the Fourier-Motzkin variable elimination [6]. However, it seems that complexity considerations prevented the algorithm to be ever implemented, as the number of *splinters* built by his method is a function of the coefficients of the formula. Moreover, it is not clear whether his method is able to eliminate more than one existential variable without scanning a large number of polytopes.

More recently, Verdoolaege et al. [29] proposed to apply simple rewriting rules (existential variables that are *unique* or *redundant*, decomposition in *independent splits*) to a disjoint union of parametric sets computed using the Omega Calculator [22], and to use the PIP library [9] if these rewriting rules fail. But the input of their algorithm, the disjoint union of sets computed from a Presburger formula is worst-case exponential; and PIP is worst-case exponential.

Another theoretical polynomial-time algorithm was proposed by Verdoolaege and Woods [31] but its implementation remains a challenge.

In this paper, we propose a new algorithm handling this problem, based on: (i) a polynomial method to count the number of integer points in a union of a fixed number of parametric $\mathbb{Z}$-polytopes (of fixed dimension), (ii) a method

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

CASES'06, October 23–25, 2006, Seoul, Korea.

to compute the integer projection of a union of parametric $\mathbb{Z}$-polytopes as a worst-case exponential union of parametric $\mathbb{Z}$-polytopes.

We compared our implementation to the one of Verdoolaege [29], and showed that, for an important set of examples, it is more efficient. Moreover, the resulting quasi-polynomials are significantly smaller using our algorithm, and, as these polynomials are used in the optimized programs, their code size is smaller, which also is an important issue in the context of memory for embedded systems.

In section 2 we show the motivation of this work by means of an example: array linearization for cache optimization and memory compression. We present our algorithm for computing the integer transformation of a $\mathbb{Z}$-polytope in section 3. Section 4 presents our algorithm for counting integer points in arbitrary unions of parametric $\mathbb{Z}$-polytopes. In section 5 we provide experiments comparing our implementation to the one of Verdoolaege. Finally, the conclusions are given in section 6.

2. MOTIVATING EXAMPLE

In this section, we will optimize the loop nest in Figure 1.(a). We chose such a complex example to show the generality of our method, even if such array references occur seldom in real applications. Notice however that when the loop contains many references to an array, there are no other methods than array linearization to achieve memory compression.

The new access function (b) as compared to the original one, requires less memory: the size of array A is $18N^2 - 25N + 8$, whereas the number of array elements that are actually accessed is $3N^2 - 2N$ (it is the size of array B, computed using the algorithm described in Section 3). Under the assumption that the array element size is 4 bytes, array A requires $4(15N^2 - 23N + 8)$ bytes more than B, which is more than 576KB when $N = 100$ and more than 57MB 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 probably leads to a cache miss at each iteration when N is large, and page misses are very frequent. 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% of stride-one accesses, 23.5% of accesses of stride 200 and less than 2% of accesses of stride greater than 200 and smaller than 30000, for $N = 100$. While the original program leads to 99% of strides 1194 and 1% of strides greater than 100000.

When there is no data reuse, the new access function that we calculate always ensures stride-one accesses. In our example there is data reuse, but the number of stride-one accesses is improved. Let us 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 layout. Of course, later accesses to the same data have to be transformed as well using the same access function. A general framework for optimizing many references to arrays in many loops is out of the scope of this paper, but can be found in [15].

Let us now present the linearization method for this example. The iteration domain of loop 1.(a) 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 $T(i, j, k) = (x = 3i + 6k, y = 2j + 5)$. The elements of the array accessed by this loop nest are given by the transformation by T of the integer points of 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 [22]:

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

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

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

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

with $N \geq 1$. The number of integer points in this polytope is given by the following Ehrhart quasi-polynomial [8, 4, 30]:

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

While the *exact* number of images of the integer points in P, computed as explained in Section 3, is:

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

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 of this presentation, 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 quasi-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 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:

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

```
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).

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

Let us focus on the linearization of array A in the first case, when the lexicographic minimum equals (x_0-2N, y_0-2, N). Let $\mathcal{S}_1(x_0, y_0)$ be the set of iterations preceding the first one accessing datum $A(x_0, y_0)$. This set is given by:

$$\begin{aligned}&\mathcal{S}_1(x_0, y_0)\\ &= \{(i,j,k) \in P \mid (i,j,k) \prec (x_0-2N, y_0-2, N)\}\\ &= \{(i,j,k) \in P \mid i < x_0-2N \ \vee \ (i = x_0-2N \ \wedge\\ &\quad j < y_0-2) \vee (i = x_0-2N \ \wedge \ j = y_0-2 \ \wedge \ k < N)\}\end{aligned}$$

The array elements accessed by this set of iterations are 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 solutions to this Presburger formula is given by the Ehrhart quasi-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 in the second case, when the lexicographic minimum equals $(x_0 - 2M + 2, y_0 - 2, M - 1)$.

Finally, 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):

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

3. INTEGER TRANSFORMATIONS OF PARAMETRIC POLYTOPES

The problem of calculating the affine transformation of a polytope is equivalent to the elimination of the existential variables from the Presburger formula representing the transformation. The first step consists in removing all the equalities of the formula, which automatically eliminates a number (equal to the number of non-redundant equalities) of existential variables. The removal of an equality from a polytope defined on the set of integers, must be done so that there exist integer values of the variables to be eliminated for each integer value of the other variables. We use Meister's technique described in [16] to eliminate the equalities. Note that since this algorithm modifies the coordinates of the considered polytopes, we must rewrite each transformed polytope as a function of its original variables and parameters before counting the integer points in the result. In the following, we consider without loss of generality, that our formulas do not contain equalities, and we focus on the remaining existential variables. We can then rewrite the formulas in the form of a set of lower and upper bounds $\{l(\mathbf{x}, \mathbf{p}) \leq \beta z, \ \alpha z \leq u(\mathbf{x}, \mathbf{p})\}$, where z is a variable chosen to be eliminated, $l(\mathbf{x}, \mathbf{p})$ and $u(\mathbf{x}, \mathbf{p})$ are affine functions of variables $\mathbf{x}$ and parameters $\mathbf{p}$ independent of z, and α and β are strictly positive integer constants.

3.1 Existential variable elimination and integer projection

W. Pugh et al. [21, 22, 23] proposed an extension of the Fourier-Motzkin existential variable elimination [6] from real numbers to integers, 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 [21] 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*,

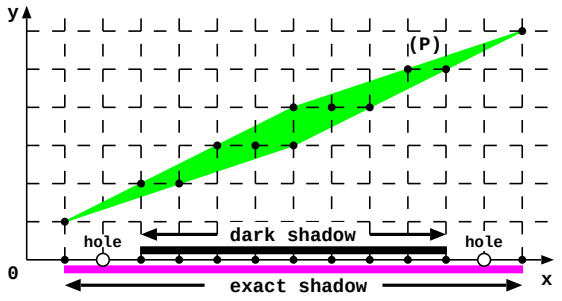

Figure 2: The integer projection of a polytope.

- 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.

In order to answer this latter question, Pugh and Wonnacott [23] 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 (the *splinters*) which complicate 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?

Note that the splinters provided by Pugh's method are somewhat similar to our $\mathbb{Z}$-polytopes (when the coefficients of the existential variable are not coprime, see Section 3.2.2). But it is not clear whether these splinters can be projected along another dimension without scanning a possibly large number of polytopes. Also, when the coefficients are coprime, Pugh's method does not propose a simple solution as we do, and no explanation is given for non-coprime coefficients nor for the parametric case.

EXAMPLE 1. *Consider the following example (introduced in [23]):*

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

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 [23] 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 rules provide these images directly in the form: $\{x = 3,\ x = 29\}$.

3.2 Projection method

In this section, we will focus on the projection of a single pair of lower and upper bounds on an existential variable chosen to be eliminated. Indeed, the projection of the whole polytope is simply obtained by intersecting the projections of all its pairs of bounds (as for the well-known Fourier-Motzkin procedure).[2]

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 cannot be obtained by applying simple rules. We calculate these points as follows.

The set of points we are searching for lie on the region given by the difference between the exact shadow and the dark shadow. This region 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:

$$\alpha l(\mathbf{x},\mathbf{p}) - \beta u(\mathbf{x},\mathbf{p}) + \gamma = 0, \qquad (1)$$

where γ is an integer constant such that $0 \leq \gamma \leq \alpha\beta - \alpha - \beta$. The values of γ for which the hyperplane (1) contains the points we are interested in are those verifying the inequality:

$$\alpha(-l(\mathbf{x},\mathbf{p}) \bmod \beta) \leq \gamma \ \Leftrightarrow\ \beta(u(\mathbf{x},\mathbf{p}) \bmod \alpha) \leq \gamma \qquad (2)$$

NOTE 1. *The values of γ for which the hyperplane (1) contains no integer points are excluded from the search set. On another hand, if one of the bounds $l(\mathbf{x},\mathbf{p})$ or $u(\mathbf{x},\mathbf{p})$ is a constant, the elimination of the existential variable results in the dark shadow only.*

We calculate the solutions of inequality (2) in two different ways, depending on whether the coefficients α and β are coprime or not.

3.2.1 Case of coprime coefficients

When the coefficients of the existential variable are coprime, the calculation of the values of γ for which the hyperplane (1) contains the points we are searching for, depends neither on the variables nor on the parameters, i.e., it depends only on the constants α and β. In this case, the inequality (2) is equivalent to:

$$\alpha((c_1\gamma) \bmod \beta) \leq \gamma \ \Leftrightarrow\ \beta((c_2\gamma) \bmod \alpha) \leq \gamma, \qquad (3)$$

where c_1 and c_2 are integer constants such that $c_1\alpha + c_2\beta = 1$. Proofs are given in an extended technical report [27].

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:

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

Let $c_1 = 1,\ c_2 = -1$. The constraint on the dark shadow, corresponding to this pair of bounds, is $x \leq 3N + 5$. The

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

points of the projection which do not belong to the dark shadow are given by:

$$\begin{array}{l}\alpha l(x,N) - \beta u(x,N) + \gamma = 0,\ 0 \le \gamma \le \alpha\beta - \alpha - \beta \\ \qquad\qquad\qquad\qquad \textit{and } \alpha((c_1\gamma) \bmod \beta) \le \gamma \\ \Rightarrow x - 3N - 7 + \gamma = 0,\ 0 \le \gamma \le 1 \textit{ and } 3(\gamma \bmod 2) \le \gamma.\end{array}$$

Scanning the values of γ, we find that the only one 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 $\{x + 2 \le 3y,\ 2y \le x + 1\}$, generating the constraint $x \ge 3$ on the dark shadow. The integer projection of the polytope P can then be obtained by intersecting the projections of the two pairs. Since N is positive, the result is:

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

3.2.2 Case of non-coprime coefficients

Let us now consider the case of non-coprime coefficients ($gcd(\alpha, \beta) \neq 0$). The calculation of the values of γ, for which the hyperplane (1) 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 $g = gcd(\alpha, \beta)$, $\alpha' = \alpha/g$, $\beta' = \beta/g$. One can then rewrite the equation of the hyperplane (1) in the form:

$$\alpha' l(\mathbf{x}, \mathbf{p}) - \beta' u(\mathbf{x}, \mathbf{p}) + \gamma = 0, \tag{4}$$

with $\gamma \in \mathbb{Z}$, $0 \le \gamma \le \alpha\beta' - \alpha' - \beta'$, and the inequality (2) in the form:

$$\alpha'(-l(\mathbf{x}, \mathbf{p}) \bmod \beta) \le \gamma \ \Leftrightarrow\ \beta'(u(\mathbf{x}, \mathbf{p}) \bmod \alpha) \le \gamma. \tag{5}$$

Again, only the values of γ for which the hyperplane (4) contains integer points are taken into account.

In this case, it may happen that only a subset of the integer points of the hyperplane (4) belong to the projection. These points are defined by the intersection of the hyperplane with a union of integer lattices obtained by solving one of the following modulo equalities:

$$-l(\mathbf{x}, \mathbf{p}) \bmod \beta = \gamma',\ \text{with } 0 \le \gamma' \le \min\left(\left\lfloor \frac{\gamma}{\alpha'} \right\rfloor, \beta\right), \tag{6}$$

$$u(\mathbf{x}, \mathbf{p}) \bmod \alpha = \gamma',\ \text{with } 0 \le \gamma' \le \min\left(\left\lfloor \frac{\gamma}{\beta'} \right\rfloor, \alpha\right). \tag{7}$$

In practice, it is worth to consider equality (6) when $\beta < \alpha$ and equality (7) otherwise.

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

$$L = \left\{ \begin{pmatrix} A_{\mathbf{x}} \mid A_{\mathbf{p}} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{p} \end{pmatrix} + \mathbf{c} \;\middle|\; \mathbf{x} \in \mathbb{Z}^d, \mathbf{p} \in \mathbb{Z}^n \right\}, \tag{8}$$

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]. Of course, only non-empty lattices and hyperplanes are considered.

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

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

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

$$x - 2N - 9 + \gamma = 0,$$

such that $0 \le \gamma \le 1$ and $2((x - N - 2) \bmod 2) \le \gamma$.

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

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

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:

$$\mathcal{S} = \left\{ x \in \mathbb{Z} \;\middle|\; \begin{array}{l} (x = 2N + 8 \wedge (x, N) \in L) \\ \vee\ (x = 2N + 9 \wedge (x, N) \in L) \\ \vee\ x \le 2N + 7 \end{array} \right\}.$$

Projecting out an existential variable may result in a union of $\mathbb{Z}$-polytopes, i.e., a union of intersections of polytopes and lattices of the form (8). This may occur when 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.

The number of $\mathbb{Z}$-polytopes in the final resulting union, and thus the complexity of the algorithm, is worst-case exponential in the coefficients of the eliminated variables.

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

4. COUNTING POINTS IN UNIONS OF PARAMETRIC $\mathbb{Z}$-POLYTOPES

Counting integer points in unions of parametric $\mathbb{Z}$-polytopes is useful for counting points in integer transformations of polytopes, but also to handle non-unit stride loop nest analyses [24, 12]. In the following, we will discuss an algorithm dealing with general parametric $\mathbb{Z}$-polytopes of the form $\mathcal{Z} = P \cap L$, with:

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

where P is a parametric polytope, L is a parametric integer lattice, $A_{\mathbf{x}} \in \mathbb{Z}^{m \times d}, A_{\mathbf{p}} \in \mathbb{Z}^{m \times n}, B_{\mathbf{x}} \in \mathbb{Z}^{(d+n) \times d}$ and $B_{\mathbf{p}} \in \mathbb{Z}^{(d+n) \times n}$ are integer matrices, $\mathbf{a} \in \mathbb{Z}^m$ and $\mathbf{b} \in \mathbb{Z}^{d+n}$ are integer vectors, $\mathbf{x}$ is a vector of the data space and $\mathbf{p}$ is a vector of parameters.

To the best of our knowledge, the previous methods [16, 33] to count points in unions of $\mathbb{Z}$-polytopes are *lattice-union* based, which is exponential in the size of lattice generators and their least common multiple. Furthermore, Zhu et al.'s method [33] only deals with *non-parametric* $\mathbb{Z}$-polytopes.

In contrast, our method is *lattice-intersection* based: the lattice intersection is polynomial since the intersection of two

Counting the integer transformation of a polytope

Input:
  P: Polytope
  T: Transformation matrix
Output:
  L: List of (Validity domain, Ehrhart quasi-polynomial)
Variables:
  F, U: Presburger Formulas
  S, S': List of (sign, $\mathbb{Z}$-polytope)

F = PresburgerFormula (P, T)
F = EliminateEqualities (F)

// Existential variables elimination
For each v in variables to be eliminated in F
  F = ReduceLattice (v, F)
  U = Universe (Dim$(F) - 1$)
  For each $\alpha u, \beta l$ in pairs of bounds on v in F
    D = DarkShadow $(\alpha u, \beta l)$
    If $\alpha = 1$ or $\beta = 1$
      $U = U \cap D$
    Else
      E = ExactShadow $(\alpha u, \beta l)$
      $U = U \cap (D \cup RemoveHoles(E - D, \alpha u, \beta l))$
    End If
  End For
  $F = U$
End For
// F is now a union of $\mathbb{Z}$-polytopes

// Inclusion-exclusion principle
S = Empty
For each $\mathcal{Z}$ in F
  $S' = S$
  For each (s,$\mathcal{Y}$) in S
    $I = \mathcal{Z} \cap \mathcal{Y}$
    If Not Empty (I)
      $S' = S' + (-1 \times s, I)$
    End If
  End For
  $S = S' + (+1, \mathcal{Z})$
End For

// Enumeration of S
L= Empty ()
For each $(s, \mathcal{Y})$ in S
  L = AddAndSimplify $(L, s \times$Enumerate $(\mathcal{Y}))$
End For

Figure 3: Algorithm

lattices results in only one lattice, whatever their generators. Previous algorithms start by calculating a *disjoint* union of the input $\mathbb{Z}$-polytopes. It is usually very hard to separate a union of $\mathbb{Z}$-polytopes into a *disjoint* union [33], and it may be exponential even for a fixed number of $\mathbb{Z}$-polytopes.

We therefore rather use the well-known inclusion-exclusion principle to generate a set of $\mathbb{Z}$-polytopes, where only non-empty intersections are taken into account. 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 $\mathcal{E}(\mathcal{Z}_1) + \mathcal{E}(\mathcal{Z}_2) - \mathcal{E}(\mathcal{Z}_1 \cap \mathcal{Z}_2)$. Of course, this generalizes to any number of $\mathbb{Z}$-polytopes.

After applying the inclusion-exclusion principle, the number of points in each of the resulting $\mathbb{Z}$-polytope $\mathcal{Y}_i = P_i \cap L_i$ is calculated as follows. First, we transform the matrix $\left(\begin{array}{c|c} B_{\mathbf{x}} & B_{\mathbf{p}} \end{array} \right)$ 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: $M = \left(\begin{array}{c|c} B_{\mathbf{xx}} & B_{\mathbf{xp}} \\ \hline 0 & B_{\mathbf{pp}} \end{array} \right)$. In matrix M, the rows which transform the parameter space are *independent* of the variables.[3] This is required to keep the data space compressed when rewriting the transformed polytope as a function of its original parameters. Matrix M is calculated from the Hermit normal form [26] of $\left(\begin{array}{c|c} B_{\mathbf{x}} & B_{\mathbf{p}} \end{array} \right)$. Then, we 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'. Finally, P' is rewritten as a function of the original parameters using the submatrix $(B_{\mathbf{pp}}|\mathbf{b_p})$, and we use our counting algorithm [30] to calculate the Ehrhart quasi-polynomial corresponding to the number of integer points in the resulting polytope. Note that when submatrix $B_{\mathbf{pp}}$ is not equal to the identity matrix, the polytope is valid only for 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 quasi-polynomial is to be multiplied by one if the parameter values are valid and zero otherwise.

The complexity of the proposed algorithm depends on the complexity of the significant $\mathbb{Z}$-polytope operations, the complexity of counting integer points in a parametric polytope and the number of resulting $\mathbb{Z}$-polytopes after application of the inclusion-exclusion principle. 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 [26]. Counting integer points in a parametric polytope is also polynomial in the input size (for fixed dimension), as we showed in [30]. Finally, when the number of input $\mathbb{Z}$-polytopes is fixed, the inclusion-exclusion principle provides a polynomial number of $\mathbb{Z}$-polytopes. Hence the whole algorithm is polynomial in the input size (for fixed dimension and fixed number of input $\mathbb{Z}$-polytopes).

The whole algorithm is summarized Figure 3.

[3]Matrix M generates the same integer points as the original matrix.

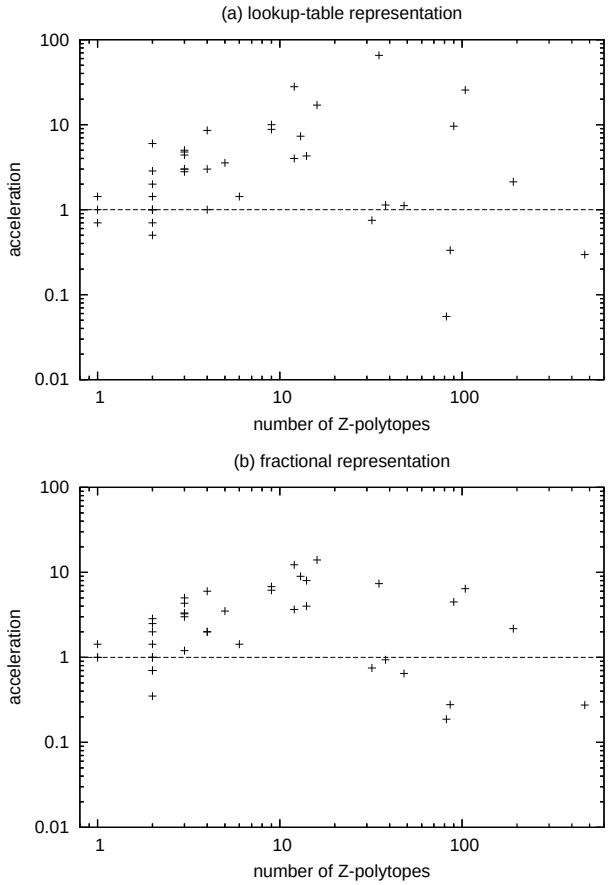

Figure 4: Execution time comparison with Verdoolaege's implementation.

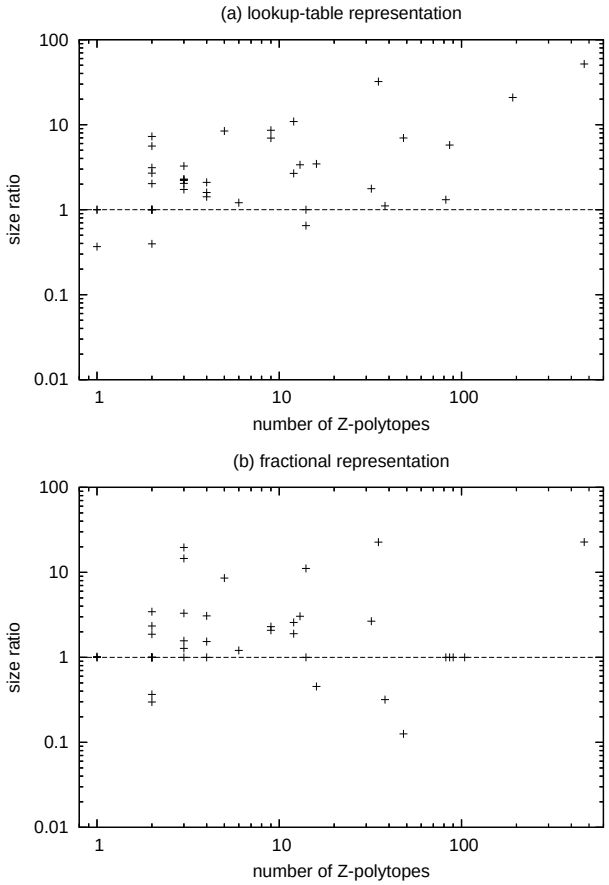

Figure 5: Output size comparison with Verdoolaege's implementation.

5. EXPERIMENTS

In this section, we compare our implementation to the one of Verdoolaege et al., since it is the most efficient known exact method to be implemented [29].

These experiments were undertaken with PolyLib version 5.22 and Barvinok version 0.20. In both implementations, the first library is used to realize polyhedral operations, and the second one to count integer points in parametric polytopes. In addition, Verdoolaege et al. use the PIP library [10], and the Omega library [13] to simplify the input polytopes.

This random test set is representative of an important number of complex cases (which do not reduce to the rational transformation, easily computed in polynomial time by both implementations). The dimension of the polytopes vary between 3 and 7 (corresponding to the loop depth plus the array dimension), the number of parameters vary between 1 and 3, and the number of eliminated variables from 1 to 5. We chose to plot the acceleration and output size ratio as a function of the number of $\mathbb{Z}$-polytopes generated after transformation by our algorithm, since it is the most significant parameter on its complexity. The comparison is performed using two representations of Ehrhart quasi-polynomials: as lookup-table and as fractional enumerators.

The lookup-table representation is known to be exponential [29] but its advantage, compared to the fractional representation, is that it can be fully simplified. Therefore, lookup-table quasi-polynomials may be of smaller size than fractional ones (when their periods are small).

Figure 4 shows that, for many of these tests, our execution times are significantly lower than Verdoolaege's ones (the scale is logarithmic). The acceleration is about 2.05 in the fractional case, and 2.71 when using lookup-table representation. Indeed, most of the quasi-polynomials generated by Verdoolaege's method have larger periods than those generated by ours. Therefore, when using the fractional representation, the performance of our algorithm does not change much, while Verdoolaege's algorithm is faster for some examples. Note that Verdoolaege's method does not calculate the actual projection, but an equivalent set of polytopes having the same number of integer points. These polytopes are sometimes of larger dimensions and may have larger constraints coefficients, which increases the execution time.

Figure 5 shows the size ratio of the output polynomials. Again, since our method generates smaller periods, the resulting polynomials are smaller than Verdoolaege's. The geometric mean of the size ratio in this set of examples is 1.94 in the fractional case and 3.23 in the periodic case.

6. CONCLUSION

We presented a new algorithm for calculating the transformation of integer points in parametric polytopes. The solution is given as a union of parametric $\mathbb{Z}$-polytopes, worst-case exponential but efficient in practical cases, and less complex compared to other existing methods. A general polynomial algorithm for computing the exact solution remains a challenge. We also proposed a new polynomial algorithm to count points in arbitrary unions of a fixed number of $\mathbb{Z}$-polytopes (of fixed dimension).

These algorithms have been implemented using the Polyhedral library [14] and Barvinok library [30].

These results have many applications in parametric affine loop nest analysis and optimization, for example array linearization, cache optimization, and memory size computation. The exact computation of the transformation of a $\mathbb{Z}$-polytope allows static generation of optimized code and hardware, reducing power consumption and memory size in embedded systems. Moreover, the smaller quasi-polynomials that we obtain (compared to other methods) leads to smaller optimized code.

7. ACKNOWLEDGEMENTS

We are grateful to the anonymous referees for their helpful comments.

We would like to thank Catherine Mongenet for careful reading of previous versions of this paper. Many thanks to Sven Verdoolaege and Benoît Meister for providing us with their implementations and for many fruitful discussions.

8. REFERENCES

[1] A. I. Barvinok. Computing the Ehrhart polynomial of a convex lattice polytope. *Discrete Comput. Geom.*, 12:35–48, 1994.

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

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

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

[5] 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.

[6] G. B. Dantzig and B. C. Eaves. Fourier-Motzkin elimination and its dual. *J. Comb. Theory, Ser. A*, 14(3):288–297, 1973.

[7] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. Effective lattice point counting in rational convex polytopes. *Journal of Symbolic Computation*, 38(4):1273–1302, 2004.

[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] P. Held. Hipars: a tool for automatic conversion of nested loop programs into single assignment programs. Technical report, Dept. Electrical Engineering, Delft University of Technology, 1994.

[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] P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle, and P. G. Kjeldsberg. Data and memory optimization techniques for embedded systems. *ACM Trans. Des. Autom. Electron. Syst.*, 6(2):149–206, 2001.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] J. Ramanujam, J. Hong, M. Kandemir, and A. Narayan. Reducing memory requirements of nested

loops for embedded systems. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 359–364, New York, NY, USA, 2001. ACM Press.

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

[27] R. Seghir and V. Loechner. Minimizing memory strides using integer transformations of parametric polytopes. Technical report, LSIIT - ICPS UMR7005 ULP-CNRS, oct 2006. http://icps.u-strasbg.fr.

[28] A. Turjan, B. Kienhuis, and E. Deprettere. Solving out-of-order communication in Kahn process networks. *J. VLSI Signal Process. Syst.*, 40(1):7–18, 2005.

[29] 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.

[30] 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.

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

[32] 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.

[33] H. Zhu, I. I. Luican, and F. Balasa. Memory size computation for multimedia processing applications. In *Proceedings of 11th Asia and South Pacific Design Automation Conference*, pages 802–807, Jan. 2006.