ZPolyTrans: A library for computing and enumerating integer transformations of Z-Polyhedra

Rachid Seghir
Department of Computer Science
University of Batna
Algeria
seghir@univ-batna.dz

ABSTRACT

Many affine loop nest analysis and optimization techniques are based on the well-known polyhedral model framework. Namely, iterations and array references are represented by integer points in bounded polyhedra, or $\mathbb{Z}$-polyhedra. In this tool demonstration paper, we present our library *ZPolyTrans* dealing with: (i) computing integer affine transformations of $\mathbb{Z}$-polyhedra, (ii) counting integer solutions to such transformations.

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

Polyhedral model, parametric $\mathbb{Z}$-polytopes transformation, integer points counting, Counting solutions to Presburger formulas.

1. INTRODUCTION

In the polyhedral model, numerous code optimization and parallelization techniques raise the problem of counting integer points in parametric bounded $\mathbb{Z}$-polyhedra and in their transformations by affine integer functions. Through this work, we present a C library dedicated to manipulate integer transformations of parametric $\mathbb{Z}$-polytopes[1] based on *Polylib* [13] and *barvinok* [20] libraries. The main algorithms implemented in *ZPolyTrans*[2] are those we proposed in [17, 18]: (i) an algorithm for computing the integer affine transformation of a parametric polytope as a union of parametric $\mathbb{Z}$-polytopes, (ii) and an algorithm for counting the number of integer points in such a union. The result of this later algorithm is given by one or many multivariate polynomials in which the coefficients may be periodic numbers. These pseudo-polynomials, known as Ehrhart polynomials [2, 3], are defined on sub-sets (chambers) of the parameter values. Both algorithms have many applications in affine loop nest analysis and transformation, such as array linearization for hardware design [19], cache access optimization [8, 4, 14], memory size computation [24, 25], and data distribution for NUMA-machines [9].

The rest of this paper is organized as follows. In section 2, we present an overview of the theory we implemented in *ZPolyTrans*. We then show the way this library can be used in section 3. In section 4, we give some related work and experiments. Finally, the conclusions are given in section 5.

[1]Sub-lattices of bounded polyhedra.
[2]Available on http://zpolytrans.gforge.inria.fr

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

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

2. OVERVIEW OF THE THEORY IMPLEMENTED IN ZPOLYTRANS

In this section, we present an abstract of the two main algorithms that are implemented in *ZPolyTrans*. The full details about these algorithms are given in [18, 17].

2.1 $\mathbb{Z}$-polytope transformations and Presburger formulas

It has been shown that the integer affine transformation of a $\mathbb{Z}$-polytope can be written as a Presburger formula [15, 7]:

Let $\mathcal{Z} = P_{\mathbf{p}} \cap L$ be a parametric $\mathbb{Z}$-polytope:

$P_{\mathbf{p}} = \left\{\mathbf{x} \in \mathbb{Q}^d \mid A\mathbf{x} \geq B\mathbf{p} + \mathbf{c}\right\}$ a parametric rational polytope defined by m constraints, $L = \left\{A'\mathbf{z} + \mathbf{c}' \mid \mathbf{z} \in \mathbb{Z}^d\right\}$ a lattice in canonical form and

$$\begin{array}{rccl} T: & \mathbb{Z}^d & \rightarrow & \mathbb{Z}^k \\ & \mathbf{x} & \mapsto & \mathbf{x}' = A''\mathbf{x} + \mathbf{c}'' \end{array}$$

be an integer transformation, where A, B, A' and A'' are $(m \times d), (m \times n), (d \times d)$ and $(k \times d)$ integer matrices (respectively); $\mathbf{x}, \mathbf{x}', \mathbf{z}, \mathbf{c}, \mathbf{c}', \mathbf{c}''$ are vectors of variables; $\mathbf{p}$ is a vector of n parameters. Then the transformation of $\mathcal{Z}$ by T is:

$$T(\mathcal{Z}) = \Big\{\mathbf{x}' \in \mathbb{Z}^k \mid \exists \mathbf{x}, \mathbf{z} \in \mathbb{Z}^d, \\ A\mathbf{x} \geq B\mathbf{p} + \mathbf{c}, \mathbf{x} = A'\mathbf{z} + \mathbf{c}', \mathbf{x}' = A''\mathbf{x} + \mathbf{c}''\Big\},$$

which is a parametric Presburger formula. Solution $\mathbf{x}'$ to

this formula is obtained by eliminating its existential variables $\mathbf{x}$ and $\mathbf{z}$. We do this in two steps:

- The first step consists in removing all equalities implying existential variables. In this way, a number (equal to the number of non-redundant equalities) of existential variables are automatically eliminated. This must be done such that there exist integer values of the variables to be eliminated for each integer value of the other variables. The result of this step is a parametric $\mathbb{Z}$-polytope $\mathcal{Z}$, i.e., an intersection of a parametric polytope and an integer lattice which we call *validity lattice*.

- The second step consists in recursively eliminating the remaining existential variables from the $\mathbb{Z}$-polytope $\mathcal{Z}$ produced at the first step. Since all the constraints implying existential variables in $\mathcal{Z}$ are inequalities, we can rewrite them as 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 an existential variable chosen to be eliminated first, $l(\mathbf{x},\mathbf{p})$ 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. The result of projecting out variable z from each pair of bounds is given in the following form:

  dark shadow $\cup$ {*exact shadow* $\cap$ *sub-lattices of hyperplanes*}, where

  - the **exact shadow** corresponds to the *rational* projection of the points belonging to the pair of constraints.
  - the **dark shadow** corresponds to the convex part of the exact shadow in which any integer point has at least one integer preimage.
  - the **sub-lattices of hyperplanes** correspond to the sets containing the points of the integer projection that are outside the dark shadow. The integer points of the exact shadow having only rational preimages are called *holes*.

    Figure 1 shows the dark shadow, the exact shadow and a hole of the projection of a pair of constraints $\{x + 2 \leq 3y,\ 2y \leq x + 1\}$.

  The elimination of existential variable z from $\mathbb{Z}$-polytope $\mathcal{Z}$ is finally given by the intersection of the projections of all pairs of lower and upper bounds of the variable z. The result of this second step is than a union of, possibly non standard, parametric $\mathbb{Z}$-polytopes.

Note that our main contribution consists in computing the sub-lattices of hyperplanes. The notion of dark and exact shadows is introduced by W. Pugh [15].

2.2 Counting integer points in a union of parametric $\mathbb{Z}$-polytopes

We have seen in the previous section that the transformation of a parametric polytope by an integer affine function is usually given in the form of a union of parametric $\mathbb{Z}$-polytopes. In the following, we summarize the way in which

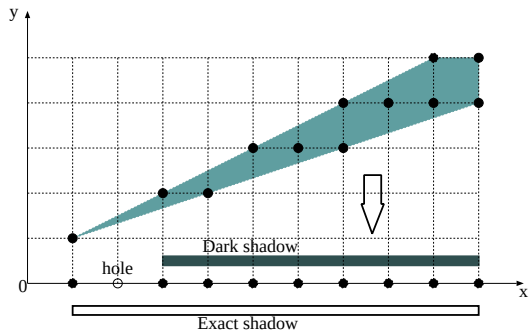

Figure 1: Integer projection of a pair of constraints.

we count integer points in a union of $\mathbb{Z}$-polytopes of the form $\mathcal{Z} = P \cap L$, with:

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

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

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 variables space and $\mathbf{p}$ is a vector of parameters.

Suppose we need to compute $\mathcal{E}(\mathcal{Z}_1 \cup \mathcal{Z}_2)$, the number of integer points in the union of two $\mathbb{Z}$-polytopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$. To do this, we start by applying the inclusion-exclusion principle to $\mathcal{E}(\mathcal{Z}_1 \cup \mathcal{Z}_2)$, which gives $\mathcal{E}(\mathcal{Z}_1) + \mathcal{E}(\mathcal{Z}_2) - \mathcal{E}(\mathcal{Z}_1 \cap \mathcal{Z}_2)$. Of course, the generalization of this principle to many $\mathbb{Z}$-polytopes is straightforward. The number of points in each resulting $\mathbb{Z}$-polytope $\mathcal{Y}_i = P_i \cap L_i$ is calculated as follows:

We first transform the lattice L_i into an equivalent lattice L'_i in which the rows defining the lattice in the parameter space are *independent* of the variables. Then, we transform P_i by L'_i to get a polytope P' which is rewritten as a function of the original parameters. Finally, we use our counting algorithm [22] to calculate the Ehrhart quasi-polynomial corresponding to the number of integer points in the resulting polytope. The results coming from the different $\mathbb{Z}$-polytopes (lists of pairs (validity domain, Ehrhart polynomial)) are then combined into a single list.

3. HOW TO USE ZPOLYTRANS ?

In this section, we show through examples how to use the main C programs of *ZPolyTrans*.

3.1 Using the program `enumerate_image`

Suppose we are interested in computing the amount of data being accessed by the following loop nests (the cache size used by this computation could also be computed):

```
for i = 0 to n do
   for j = i+1 to n do
      A[4*i+2*j] = ...

for k = 0 to n do
   for l = 0 to n do
      A[k+l] = ...
```

The desired computation reduces to computing the integer points in the transformation of the first iteration domain ($0 \leq i \leq n \wedge i{+}1 \leq j \leq n$) by the first reference function ($4i{+}2j$) union the transformation of the second iteration domain ($0 \leq k \leq n \wedge 0 \leq l \leq n$) by the second reference function ($k + l$). This can be written in the form of a Presburger formula:

$$\{x \in \mathbb{Z} \mid \exists i \exists j \in \mathbb{Z}^2, 0 \leq i \leq n \wedge i{+}1 \leq j \leq n \wedge x = 4i{+}2j\} \cup$$
$$\{x \in \mathbb{Z} \mid \exists k \exists l \in \mathbb{Z}^2,\ 0 \leq k \leq n\ \wedge\ 0 \leq l \leq n\ \wedge\ x = k + l\}$$

To solve this formula, we construct the input (in *Polylib* format) for the program `enumerate_image` as follows:

```
2 # number of input polyhedra
# Polyhedron P1
5  6
#  i  j  x  n
0  4  2 -1  0  0   # x = 4*i+2*j
1  1  0  0  0  0   # i >= 0
1 -1  0  0  1  0   # i <= n
1 -1  1  0  0 -1   # j >= i+1
1  0 -1  0  1  0   # j <= n
2 # number of existential variables (i,j) of P1
# Polyhedron P2
5  6
#  k  l  x  n
0  1  1 -1  0  0   # x = k+l
1  1  0  0  0  0   # k >= 0
1 -1  0  0  1  0   # k <= n
1  0  1  0  0  0   # l >= 0
1  0 -1  0  1  0   # l <= n
2 # number of existential variables (k,l) of P2
# parameter context matrix
1  3
1  1  0   # n >= 0
# names of parameters
n
```

In *Polylib* format, the reference function and its iteration domain are combined to construct a single non full-dimensional polyhedron. The presence of 0 in the first column means that the constraint is an equality of the form $\mathbf{a}\mathbf{x} + c = 0$, and the presence of 1 indicates that the constraint is an inequality of the form $\mathbf{a}\mathbf{x} + c \geq 0$. The parameter context matrix defines the constraints on the parameters appearing in the loop nests (see [13] for more details).

The output of `enumerate_image` for the above input is:

```
3 3
   2    0    0
   0    1    0
   0    0    1
POLYHEDRON Dimension:2
          Constraints:3  Equations:0  Rays:3  Lines:0
Constraints 3 4
Inequality: [  -1    6   -8  ]
Inequality: [   1    0   -2  ]
Inequality: [   0    0    1  ]
Rays 3 4
Ray:    [   0    1  ]
Ray:    [   6    1  ]
Vertex: [   6    5  ]/3
3 3
   2    0    0
   0    1    0
   0    0    1
POLYHEDRON Dimension:2
          Constraints:3  Equations:1  Rays:2  Lines:0
Constraints 3 4
Equality:   [   1   -6    4  ]
Inequality: [   0    1   -1  ]
Inequality: [   0    0    1  ]
Rays 2 4
Ray:    [   6    1  ]
Vertex: [   2    1  ]/1
3 3
   1    0    0
   0    1    0
   0    0    1
POLYHEDRON Dimension:2
          Constraints:3  Equations:0  Rays:3  Lines:0
Constraints 3 4
Inequality: [   1    0    0  ]
Inequality: [  -1    2    0  ]
Inequality: [   0    0    1  ]
Rays 3 4
Ray:    [   0    1  ]
Ray:    [   2    1  ]
Vertex: [   0    0  ]/1

         n  = 0
          1 >= 0
1
         n  -1 = 0
          1 >= 0
3
         n  -2 >= 0
          1 >= 0

( 4 * n + -2 )
```

This means that the result of the transformation is given by a union of three $\mathbb{Z}$-polytopes: $\{2 \leq x \leq 6n - 8 \wedge x \text{ even}\} \cup \{x = 6n - 4 \wedge n \geq 1 \wedge x \text{ even}\} \cup \{0 \leq x \leq 2n \wedge x \in \mathbb{Z}\}$. The number of points in this union equals to 1 when $n = 0$, 3 when $n = 1$ and $4n - 2$ when $n \geq 2$.

By default, the program `enumerate_image` computes the image and its cardinality and displays only this later one. When the user also needs to display the image (result of the transformation), it has to use the option `-v`. And when it needs to get the image only (without computing its cardinality), it has to use the option `-no-ep`.

3.2 Using the program `enumerate_zp_union`

In some applications (such as handling loop nests with non-unit stride [16, 10]), one might be interested in computing integer points in a union of $\mathbb{Z}$-polytopes. In this case, the program `enumerate_zp_union` can be used. For example, suppose that we need to count the number of array elements accessed by the following loop nests whose iteration domains are illustrated in Figure 2 (for N=6).

```
for (i=0; i<=N; i+=2)
   for (j=0; j<=N; j+=3)
       A[i][j] = ...

for (i=0; i<=N-1; i+=2)
   for (j=0; j<=i+1; j++)
       A[i][j] = ...
```

This computation reduces to counting the number of *different* iteration vectors in the two loop nests. To do this, we construct the input file for `enumerate_zp_union` as follows:

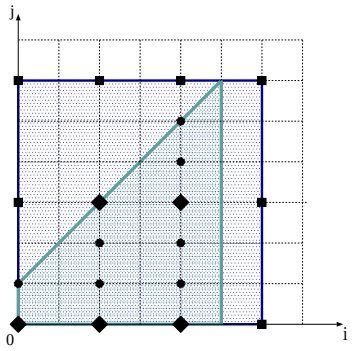

Figure 2: Example of a union of non-unit stride iteration domains, where squares belong to the first iteration domain, dots belong to the second iteration domain and diamonds belong to both iteration domains.

```
2 # number of Z-polytopes
      # Z-polytope ZP1
# Constraints matrix of ZP1
4  5
1  1  0  0  0
1 -1  0  1  0
1  0  1  0  0
1  0 -1  1  0
# lattice matrix of ZP1
4  4
2  0  0  0
0  3  0  0
0  0  1  0
0  0  0  1
      # Z-polytope ZP2
# Constraints matrix of ZP
4  5
1  1  0  0  0
1 -1  0  1 -1
1  0  1  0  0
1  1 -1  0  1
# lattice matrix of ZP
4  4
2  0  0  0
0  1  0  0
0  0  1  0
0  0  0  1
# parameter context matrix
1  3
1  1  0
# parameter names
N
```

The corresponding output is given by the following Ehrhart quasi-polynomial:

```
        N  >= 0
         1 >= 0

( 1/3 * N^2 + [ 1, 1, 2/3, 7/6, 5/6, 5/6 ]_N * N
            + [ 1, 2/3, 1/3, 1/2, 1/3, 1/2 ]_N )
```

Which means that the number of integer points in the above union of $\mathbb{Z}$-polytopes is: $\frac{1}{3}N^2+N$ when $N \bmod 6 = 0$, $\frac{1}{3}N^2+N+\frac{2}{3}$ when $N \bmod 6 = 1$, ... , $\frac{1}{3}N^2+\frac{5}{6}N+\frac{1}{2}$ when $N \bmod 6 = 5$.

4. RELATED WORK AND EXPERIMENTS

The problem of computing the integer affine transformation of parametric $\mathbb{Z}$-polytopes is a difficult geometric operation that is raised by the polyhedral model. Therefore, many approaches trying to tackle this problem have been proposed. Among these approaches, we cite:

- the work of W. Pugh [15] on *integer* affine projections based on the Fourier-Motzkin variable elimination [5].
- The theoretical rational generating function based approach, first proposed by A. Barvinok [1], and than extended to parametric case by Verdoolaege and Woods [23] and Köppe et al.[11].
- the weak quantifier elimination approach by Lasaruk and Sturm [12] consisting in a generalization of Presburger arithmetic, where the coefficients are arbitrary polynomials in non-quantified variables.
- the $\mathbb{Z}$-polyhedral model proposed by Gautam and Rajopadhye [7] in which the transformation of a $\mathbb{Z}$-polytope consists in computing a union of translated subsets of a $\mathbb{Z}$-polyhedron, and then to cut each $\mathbb{Z}$-polyhedron of the resulting union with hyperplanes.
- the work of Verdoolaege et al. [21] consisting in applying simple rewriting rules (removing existential variables that are *unique* or *redundant* and to solve (when these rules fail) a parametric integer linear programming problem (using the PIP library [6]). To the best of our knowledge, this later work, distributed in the *barvinok* library [20], is the most robust and general-case approach. We therefore choose to compare it against our method. A quick comparison to the other related work is given in [18].

The particular advantage of our method over the one of Verdoolaege is its ability to handle efficiently *unions* of projections. Indeed, in such a case, our algorithm computes each projection independently of the others, even if the original sets are not pairwise disjoint projections. The final result is then easily obtained by doing the union of the resulting atomic projections. The number of lattice points contained in this union is computed straightforwardly using the algorithm we presented in section 2.2. Verdoolaege's method could not do so since, in contrast to our method, it does not compute the actual atomic projections but equivalent sets having the same number of lattice points. Hence, the original input sets have to be pairwise disjoint. Furthermore, the equivalent resulting sets may be of higher dimension than the actual projections. This, usually, leads to higher lattice points counting times and results in larger Ehrhart quasi-polynomials. For example, in the following two loop nests:

```
for i = 1 to n do
   for j = i+1 to n do
      A[2*i+3*j] = ...
```

```
for k = 1 to n do
   for l = 1 to k-1 do
      A[k+2*l] = ...
```

The computation of the number of accessed array elements is done by Verdoolaege's *iscc* interface to the *barvinok* library in 136 *ms*, and the size of its output (piecewise quasi-polynomial) is 767 bytes. While our method does this computation in only 16 *ms* and outputs 148 bytes.

We also did a simulation through 500 pseudo-randomly generated examples that model array accesses in perfect loop nests. The examples are constructed as follows:

- the depth of loop nests (number of the existential variables) varies from 1 to 6,
- the dimension of arrays (number of free-quantifier variables) varies from 1 to 4,
- the number of parameters varies from 1 to 4,
- the number of equalities equals the array dimension,
- the number of inequalities equals 2 times the depth of the loop nest,
- the coefficients of the variables and parameters are chosen such that they reflect what could be found in a real program.

Consequently, the sum of dimensions (regular varibles + existential variables + parameters) of the generated sets[3] varies from 3 to 14, and the number of constraints varies from 3 to 16.

In these experiments, our implementation and Verdoolaege's compute the solution in $0.01s$ or less for 403 examples. The remaining examples are as follows:

- For 6 examples the two implementations have the same computation time (between $0.02s$ and $0.06s$).
- For 40 examples, our implementation does better than Verdoolaege's.
- For 28 examples, Verdoolaege's implementation does better than ours.
- For 23 examples, both implementations do not compute the solution in less than $30s$, which we set as timeout threshold.

5. CONCLUSION

We have presented a C library (*ZPolyTrans*) for computing and enumerating the integer affine transformations of parametric $\mathbb{Z}$-polytopes. The library basically implements two recent algorithms: the first one consists in computing the integer affine image of a $\mathbb{Z}$-polytope in the form of a worst-case exponential union of $\mathbb{Z}$-polytopes, and the second one computes the cardinality of such a union. When the number of input $\mathbb{Z}$-polytopes is fixed, this later algorithm is polynomial (for fixed dimension). The comparative study of our implementation with Verdoolaege's related work indicates that both approaches work very well for a large class of problems. But for some hard examples, it may be suitable to choose one approach rather than the other. Therefore we think that both implementations need some further work in order to handle efficiently such hard problems.

[3]We consider only non-empty sets.

6. REFERENCES

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

[2] Ph. Clauss. Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: Applications to analyse and transform scientific programs. In *Proceedings of the 10th International Conference on Supercomputing, ICS'96*, May 1996.

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

[4] 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)*, September 2001.

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

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

[7] Gautam and Sanjay Rajopadhye. The Z-polyhedral model. In *Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, PPoPP '07, pages 237–248. ACM, 2007.

[8] Somnath Ghosh, Margaret Martonosi, and Sharad 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.

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

[10] P.C. 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.

[11] Matthias Köppe, Sven Verdoolaege, and Kevin Woods. An implementation of the barvinok–woods integer projection algorithm. In *ITSL*, pages 53–59, 2008.

[12] Aless Lasaruk and Thomas Sturm. Weak quantifier elimination for the full linear theory of the integers: A uniform generalization of presburger arithmetic. *Appl. Algebra Eng., Commun. Comput.*, 18(6):545–574, 2007.

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

[14] Vincent Loechner, Benoit Meister, and Philippe Clauss. Precise data locality optimization of nested loops. *Journal of Supercomputing*, 21(1):37–76, 2002.

[15] William Pugh. Counting solutions to Presburger formulas: how and why. In *SIGPLAN Conference on*

Programming Language Design and Implementation (PLDI'94), pages 121–134, 1994.

[16] Patrice Quinton, Sanjay Rajopadhye, and Tanguy Risset. On manipulating Z-polyhedra using a canonical representation. *Parallel Processing Letters*, 7(2):181–194, 1997.

[17] Rachid Seghir and Vincent Loechner. Memory optimization by counting points in integer transformations of parametric polytopes. In *Proceedings of International Conference on Compilers, Architectures, and Synthesis for Embedded Systems, CASES 2006, Seoul, Korea*, pages 74–82, October 2006.

[18] Rachid Seghir, Vincent Loechner, and Benoit Meister. Integer affine transformations of parametric Z-polytopes and applications to loop nest optimization, to appear in transactions on architecture and code optimization (TACO). Technical report. http://hal.inria.fr/inria-00534686/en.

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

[20] Sven Verdoolaege. Barvinok: user guide, August 2006. http://www.kotnet.org/~skimo/barvinok/.

[21] Sven Verdoolaege, Kristof Beyls, Maurice Bruynooghe, and Francky 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, March 2005. Springer.

[22] Sven Verdoolaege, Rachid Seghir, Kristof Beyls, Vincent Loechner, and Maurice Bruynooghe. Counting integer points in parametric polytopes using barvinok's rational functions. *Algorithmica*, 48(1):37–66, 2007.

[23] Sven Verdoolaege and Kevin Woods. Counting with rational generating functions, 2005. http://arxiv.org/abs/math/0504059.

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

[25] Hongwei Zhu, Ilie I. Luican, and Florin Balasa. Memory size computation for real-time multimedia applications based on polyhedral decomposition. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences.*, E89-A(12):3378–3386, 2006.