PolyLib: A Library for Manipulating Parameterized Polyhedra

Vincent Loechner

ICPS, Université Louis Pasteur de Strasbourg
Bd. S. Brant, F-67400 ILLKIRCH

loechner@icps.u-strasbg.fr

March 17, 1999

1 Introduction

The polyhedral library (or *PolyLib* for short) operates on objects made up of unions of polyhedra of any dimension. It was first developed by Doran Wilde at IRISA, in Rennes, France, in connection with the ALPHA project. This document is a continuation of the technical report [Wil93], describing release 1.1 of the PolyLib. Version 1.1 manipulates non parameterized unions of polyhedra through the following operations: intersection, difference, union, convex hull, simplify, image and preimage, plus some input and output functions. The polyhedra are computed in their dual implicit and Minkowski representations, in homogeneous spaces. Each polyhedron is represented by two matrices: a matrix of lines and rays, and a matrix of equalities and inequalities. The first column of these matrices distinguishes lines from rays and equalities from inequalities respectively.

This document describes the functions and data structures that have been added to PolyLib1.1, in order to manipulate *parameterized polyhedra*. Version 2 of the PolyLib included parameterized vertices computation. The version including Ehrhart polynomials computation is PolyLib3.1. The latest release (PolyLib 4.03), still under development and not described in this document, includes 64 bits computations using the Arithmetic Library developped at CRI (ENSMP, France).

Parameterized polyhedra representation

PolyLib manipulates *rational* polyhedra. There are two dual representations of polyhedra: the implicit representation, as a set of constraints, and the Minkowski representation, as a set of lines, rays and vertices. A *parameterized polyhedron* is defined in the implicit form by a finite number of inequalities and equalities, which constant part depends linearly on a parameter vector p:

$$\mathcal{D}(p) = \{x \in \mathbb{Q}^n \mid Ax = A'p + a, \quad Bx \geq B'p + b\} \qquad \text{with } p \in \mathbb{Q}^m$$

where A is a $k \times n$ integer matrix, A' a $k \times m$ integer matrix, a is an integer k-vector, B is a $k' \times n$ integer matrix, B' a $k' \times m$ integer matrix and b is an integer k'-vector.

The Minkowski representation, as a set of lines, rays, and vertices, of a parameterized polyhedron is:

$$\mathcal{D}(p) = \left\{x \in \mathbb{Q}^n \mid x = L\lambda + R\mu + V(p)\nu, \ \forall\lambda, \ \forall\mu \geq 0, \ \forall\nu \geq 0, \ \sum \nu = 1\right\}$$

where L is the matrix containing the lines, R the matrix containing the rays, and $V(p)$ the matrix depending on the parameters p containing the vertices of the polyhedron.

PolyLib2 includes an algorithm, described below, computing the vertices $V(p)$ of a parameterized polyhedron.

Parameterized vertices representation

Each vertex of a parameterized polyhedron is an affine function of the parameters p, defined over a *validity domain*: each vertex exists only if p is included into the validity domain associated to this vertex [LW97]. There are two ways of representing such a set of parameterized vertices and validity domains:

- as a list of distinct vertices and their validity domain,
- as a list of distinct validity domains, and the complete matrix $V(p)$ associated to each validity domain.

There are two functions in PolyLib2 computing the parameterized vertices in these two representations, respectively `Polyhedron2Param_Vertices` and `Polyhedron2Param_Domain`.

Ehrhart polynomials representation

The *Ehrhart polynomials* associated to each of the distinct validity domains correspond to the number of integer points contained in a parameterized polytope, when the parameters are integers. They are *pseudo-polynomials*: polynomials whose coefficients are *periodic numbers*, i.e. numbers taking different values depending on the rest of the division of the parameters by the *period* of this periodic number. The function `Polyhedron_Enumerate`, in PolyLib3, returns a list of validity domains and each corresponding Ehrhart polynomial [Cla96, CLW97, CL98].

Summary

The next section describes the structures used for parameterized vertices and pseudo-polynomials internal representations. Section 3 is an overview of the different algorithms. For a complete formal description of these algorithms refer to [CL98, Loe97]. Section 4 contains the building instructions, a description of the standalone executables, and an example. Comments on distribution and authors are given in the last section.

2 Data stuctures

Parameterized vertices

There are two ways of representing the vertices of parameterized polyhedra. Both use the same data structure, `Param_Polyhedron`, described below.

```
typedef struct _Param_Poly
{
   int nbV;
   Param_Vertices *V;
   Param_Domain   *D;
}
Param_Polyhedron;
```

The fields of the `Param_Polyhedron` structure are:

`->nbV` number of parameterized vertices.

`->V` linked list of parameterized vertices.

`->D` linked list of validity domains.

```
typedef struct _Param_V
{
   struct \_Param\_V *next;
   Matrix *Vertex;
   Matrix *Domain;
}
Param_Vertices;
```

The fields of the `Param_Vertices` structure are:

`->next` pointer to the next vertex. The list is NULL-terminated.

`->Vertex` this matrix contains the coordinates of the vertex, as function of the parameters. Each line is a coordinate of the vertex. The m first columns are the coefficients of the parameters, the $(m+1)$th value is the constant, the $(m+2)$th value is the common denominator. It is an $n \times (m+2)$ matrix.

`->Domain` the constraints on parameters, `Polyhedron` format.

```
typedef struct _Param_Domain
{
   struct _Param_Domain *next;
   Polyhedron *Domain;
   int *F;
} Param_Domain;
```

The fields of the `Param_Domain` structure are:

`->next` pointer to the next domain. The list is NULL-terminated.

`->Domain` the validity domain: constraints on the parameters, `Polyhedron` format.

`->F` a bit array of the vertices in `Param_Vertices->Vertex` (ordered list).

Figure 1 shows these structures dependencies. In the first representation, as a list of vertices and a validity domain associated to each vertex, the `Param_Domain *D` field of a `Param_Polyhedron` is not used. In the second representation, as distinct validity domains and a set of vertices associated to each of them, the `Matrix *Domain` of each vertex of the list `Param_Vertices` is not used.

There is a convenient macro to scan all the vertices of a parameterized polyhedron, that can be used as follows:

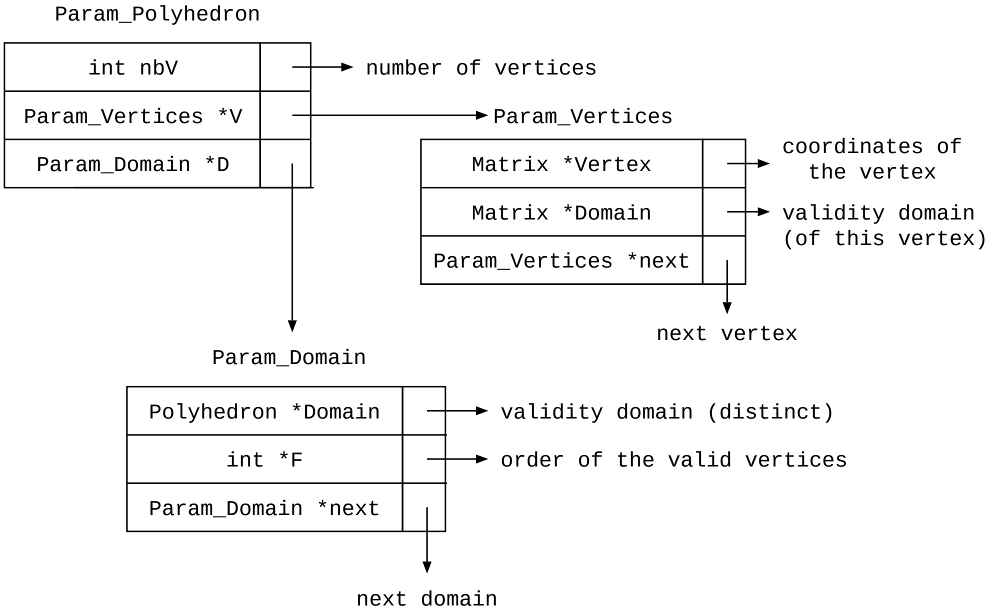

Figure 1: The `Param_Polyhedron` structure.

```
{
   /* declarations */
   Param_Polyhedron *P;
   Param_Domain     *D;
   Param_Vertices   *V;

      [...]

   /* Computation of P  */
   P = Polyhedron2Param_Domain(  [...]  );

   /* scan the validity domains */
   for( D=PA->D ; D ; D=D->next )
   {
      /* print the current validity domain */
      printf( "---------------------------------------\n" );
      printf( "Domain:\n");
      Print_Domain( stdout, P->Domain );
```

```
      /* scan the vertices */
      printf( "Vertices:\n");
      FORALL_PVertex_in_ParamPolyhedron(V, D, P)
      {
         /* print each vertex */
         Print_Vertex( stdout, V->Vertex );
         printf( "\n" );
      }
      END_FORALL_PVertex_in_ParamPolyhedron;
   }
}
```

Ehrhart polynomials

The structure `Enumeration` described below is used to represent Ehrhart polynomials over validity domains. It is a linked list of distinct validity domains and Ehrhart polynomials:

```
typedef struct _enumeration
{
  struct _enumeration *next;  /* next in the list         */
  Polyhedron *ValidityDomain; /* contraints (on params)   */
  evalue EP;                  /* Ehrhart Polynomial       */
} Enumeration;

typedef struct _evalue
{
   int d;
   union
   {
      int n;
      struct _enode *p;
   } x;
} evalue;
```

An `evalue` is either a constant rational, or a pointer to an `enode`. The fields of the `evalue` structure are:

`->d` the denominator of the constant rational if not equal to 0. If d is equal to 0, the pointer to an enode is used.

`->x.n` the numerator of the constant rational.

->x.p a pointer to an enode.

```
typedef enum { polynomial, periodic, evector } enode_type;

typedef struct _enode
{
   enode_type type;
   int size;
   int pos;
   evalue arr [1];
} enode;
```

An enode is either a polynomial, or a periodic (the evector type is used internally). The fields of the enode structure are:

->type the type of the enode: polynomial or periodic.

->size the number of attached pointers in the array arr. If the enode is of type periodic, size is the period of the periodic number. If the enode is of type polynomial, size is the degree plus 1 of the polynomial.

->pos the number of the parameter used in this enode. Parameters are numbered from 1 to m.

->arr an array of size evalues. Do not trust the declaration, the number of evalues is *not* one: when an enode structure is allocated, depending on the number of necessary evalues, the necessary amount of memory is allocated (*Doran optimisation :)*.

The enode structure corresponding to the Ehrhart polynomial $[1,2]p^2 + 3p + \frac{5}{2}$ is given figure 2.

3 Description of operations

Parameterized vertices computation

As mentioned section 1, the vertices of a parameterized polyhedron can be represented in two ways: the first form as a set of vertices and their validity domains, and the second one as a set of distinct validity domains and all the corresponding vertices. These two forms are respectively computed by the following functions:

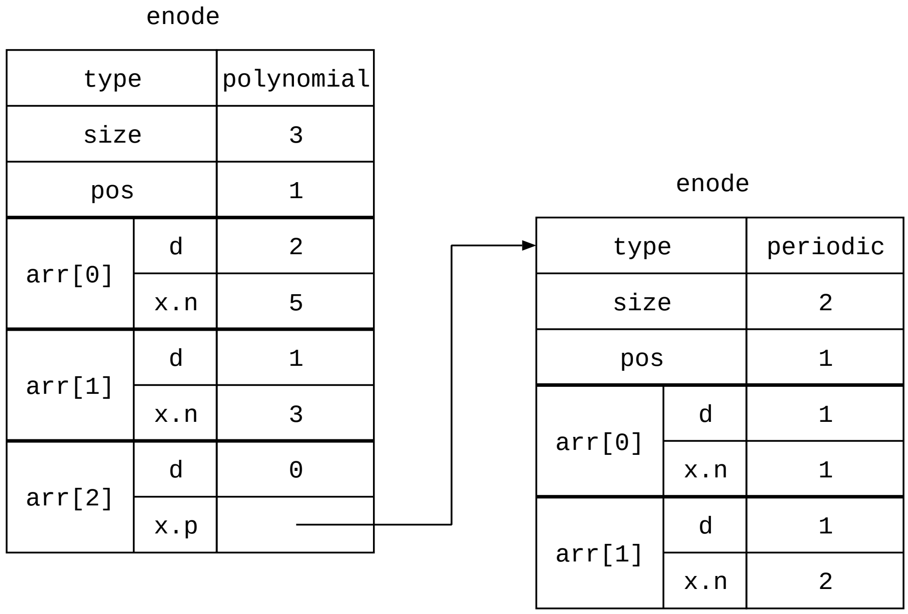

Figure 2: The Ehrhart polynomial $[1,2]p^2 + 3p + \frac{5}{2}$.

```
Param_Polyhedron *Polyhedron2Param_Vertices (
    Polyhedron *D, Polyhedron *C, int MaxRays );

Param_Polyhedron *Polyhedron2Param_Domain (
    Polyhedron *D, Polyhedron *C, int MaxRays );
```

Both functions take as input:

- `D`: the combined polyhedron of dimension $n + m$ where n is the size of the index space, and m the number of parameters.
- `C`: the context polyhedron, of size m: this polyhedron contains the constraints on the parameters. If there are no constraints, `C` must be equal to the universe polyhedron of dimension m.
- `MaxRays`: an integer, which gives the maximum number of rays in a (non parameterized) polyhedron. This value is used by the functions of the PolyLib1.1.

The output is the same `Param_Polyhedron` structure, but with different contents as described above.

Polyhedron2Param_Domain makes a call to Polyhedron2Param_Vertices, and then finds a partition of the parameter space given the validity domains of the vertices, using a function named Compute_PDomains.

The other functions used in this context are the following:

```
void Param_Polyhedron_Free ( Param_Polyhedron *P );
```

This function frees all memory allocated by a Param_Polyhedron structure.

```
void Print_Domain ( FILE *DST, Polyhedron *D );

void Print_Vertex ( FILE *DST, Matrix *V );
```

These two functions make pretty printouts on the file DST, of a validity domain and of a parameterized vertex respectively.

All these functions are defined in the C source files polyparam.[ch].

Ehrhart polynomials computation

The main function computing a set of validity domains and Ehrhart polynomials is:

```
Enumeration *Polyhedron_Enumerate ( Polyhedron *P,
                      Polyhedron *C, int MaxRays );
```

It takes as input:

- P: the combined polyhedron of dimension $n + m$. It must of course be a parameterized *polytope*.
- C: the context polyhedron, of size m. This polyhedron contains the constraints on the parameters. If there are no supplementary constraints, you have to provide the universe polyhedron of dimension m.
- MaxRays: an integer, which gives the maximum number of rays in a (non parameterized) polyhedron. This value is used by the functions of the PolyLib1.1.

The output is an Enumeration structure as described above.

Polyhedron_Enumerate makes a call to Polyhedron2Param_Domain to compute the validity domains and the vertices of the parameterized polyhedron. It then computes the denominator of the Ehrhart polynomials, and then, for each of the validity domains, makes a call to the function P_Enum which computes the Ehrhart polynomial corresponding to this validity domain. For a description of this algorithm, see [Loe97]. The C source files are ehrhart.[ch].

The other important function defined in this context is:

```
int compute_poly ( Enumeration *E, int *p );
```

This function computes the value of an Ehrhart polynomial given the parameters. It first looks for the right validity domain in `E`. If there is no validity domain corresponding to the value of p, the polyhedron is empty, and the function returns 0. If a validity domain is found, it computes the corresponding Ehrhart polynomial and returns its integer value. It is defined in the C source files `eval_ehrhart.[ch]`.

4 Usage

Building the library

Edit the main Makefile included in the distribution to comply with your target architecture.

"make" builds the archive (polylib.a) and the executables. There may be some warnings during the compilation.

"make test" and "make 64test" tests the library on a set of examples in the Test/ directory.

If something fails, please send a report to `loechner@icps.u-strasbg.fr` or to the mailing-list.

Web page and mailing-list

The main Web page is : `http://icps.u-strasbg.fr/~loechner/polylib`

To join the PolyLib mailing-list, send an email to `sympa@u-strasbg.fr` containing the following message :

SUB PolyLib firstname lastname organization

You will be asked to confirm your subscription by replying to the returned message.

Using the library

The PolyLib can be used as a linked C library, or can also be used as one of the standalone executables described below. Three executables manipulating parameterized polyhedra are created by the makefile:

- `findv`: computes the list of vertices of a parameterized polyhedron and their validity domains, and prints them on the standard output.

- `pp`: computes the list of distinct validity domains and their associated sets of parameterized vertices, and prints them on the standard output.
- `ehrhart`: computes the list of distinct validity domains, their associated sets of parameterized vertices, and their associated Ehrhart polynomials, and prints them on the standard output.

By default, the parameter names are P, Q, R, ... The inputs of these programs are given as two matrices, the combined and the context polyhedra, in PolyLib format.

Example

The following example is taken from [Loe97] chapter II.2. Consider the polyhedron:

$$\mathcal{A}(P,Q) = \{(i,j,k) \mid 0 \le i \le P, \quad 0 \le j \le i, \quad 0 \le k \le i-j, \quad i+j+k = Q\}$$

The correponding input is:

```
#-------------------------------
# Dimension of the matrix:
7 7
# Constraints:
#      i  j  k    P  Q    cte
1      1  0  0    0  0     0    # 0 <= i
1     -1  0  0    1  0     0    #      i <= P
1      0  1  0    0  0     0    # 0 <= j
1      1 -1  0    0  0     0    #      j <= i
1      0  0  1    0  0     0    # 0 <= k
1      1 -1 -1    0  0     0    #      k <= i-j
0      1  1  1    0 -1     0    # Q = i + j + k

# 2 parameters, no constraints.
0 4
```

And this is the output of the Ehrhart program:

```
---------------------------------------
Domain:
         2P - Q  >= 0
         - P + Q  >= 0
          1 >= 0
```

```
Vertices:
[ Q/2,  Q/2,  0 ]
[ Q/2,  0,  Q/2 ]
[ P,  -P+Q,  0 ]
[ P,  0,  -P+Q ]

Ehrhart Polynomial:
( -1/2 * P^2 + ( 1 * Q + 1/2 )
 * P + ( -3/8 * Q^2 + [ 1/4, 0 ]_Q * Q + [ 1, 3/8 ]_Q )
 )

---------------------------------------
Domain:
         P - Q  >= 0
         Q  >= 0
          1 >= 0

Vertices:
[ Q/2,  Q/2,  0 ]
[ Q/2,  0,  Q/2 ]
[ Q,  0,  0 ]

Ehrhart Polynomial:
( 1/8 * Q^2 + [ 3/4, 1/2 ]_Q * Q + [ 1, 3/8 ]_Q )
```

There are two validity domains. This output corresponds to:

$$\begin{cases} -\frac{1}{2}P^2 + QP - \frac{3}{8}Q^2 + \frac{1}{2}P + \left[\frac{1}{4}, 0\right]_Q Q + \left[1, \frac{3}{8}\right]_Q & \text{if } P \leq Q \leq 2P \\ \frac{1}{8}Q^2 + \left[\frac{3}{4}, \frac{1}{2}\right]_Q Q + \left[1, \frac{3}{8}\right]_Q & \text{if } 0 \leq Q \leq P \end{cases}$$

The evaluation function is then called, if the input of the program is the standard input (and not a file).

You may find many other examples in the `Test/` directory.

5 Conclusion

Authors

Various authors have contributed to the PolyLib. Version 1.1 has been designed and programmed by Doran Wilde and Hervé Le Verge in the API group at IRISA (France). Since version 1.1, the main contributors are Doran Wilde (`wilde@ee.buy.edu`, Brigham Young University, Provo, Utah), and Vincent Loechner (`loechner@icps.u-strasbg.fr`, ICPS, Université Louis Pasteur, Strasbourg, France). The parameterized polyhedra and validity domains computation functions were first written by Vincent Loechner. Doran Wilde wrote the Ehrhart polynomials computation. Both sets of functions were tested, debugged and optimized by Doran Wilde and Vincent Loechner. Emmanuel Jeannot (`Emmanuel.Jeannot@ens-lyon.fr`, ENS, Lyon, France) wrote the function evaluating an Ehrhart polynomial, and as beta-tester Xavier Redon (`Xavier.Redon@eudil.univ-lille1.fr`, LIFL, Université de Lille, France) contributed to numerous bug corrections.

Distribution

The last release of the PolyLib is freely distributed on:

`http://icps.u-strasbg.fr/~loechner/polylib/`

References

[CL98] Ph. Clauss and V. Loechner. Parametric analysis of polyhedral iteration spaces. *Journal of VLSI Signal Processing*, 19(2), July 1998. http://icps.u-strasbg.fr/pub-98.

[Cla96] Ph. Clauss. Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: Applications to analyze and transform scientific programs. In *10th ACM Int. Conf. on Supercomputing, Philadelphia*, 1996.

[CLW97] Ph. Clauss, V. Loechner, and D. K. Wilde. Deriving formulae to count solutions to parameterized linear systems using Ehrhart polynomials: Applications to the analysis of nested-loop programs. Technical Report 97-05, ICPS, http://icps.u-strasbg.fr/pub-97, 1997.

[Loe97] V. Loechner. *Contribution à l'étude des polyèdres paramétrés et applications en parallélisation automatique.* PhD thesis, Université Louis Pasteur, Strasbourg, 1997. http://icps.u-strasbg.fr/pub-97/.

[LW97] V. Loechner and D. K. Wilde. Parameterized polyhedra and their vertices. *International Journal of Parallel Programming*, 25(6), December 1997.

[Wil93] D.K. Wilde. A library for doing polyhedral operations. Technical Report 785, IRISA, Rennes, France, 1993. http://www.irisa.fr/EXTERNE/bibli/pi/pi785.html.