A Symbolic Approach to Bernstein Expansion for Program Analysis and Optimization

Philippe Clauss[1] and Irina Tchoupaeva[2]

[1] INRIA Rocquencourt, A3 Project
78153 Le Chesnay Cedex - France
clauss@icps.u-strasbg.fr
[2] ICPS/LSIIT, Université Louis Pasteur, Strasbourg
Pôle API, Bd Sébastien Brant
67400 Illkirch - France
irina@icps.u-strasbg.fr

Abstract. Several mathematical frameworks for static analysis of programs have been developed in the past decades. Although these tools are quite useful, they have still many limitations. In particular, integer multivariate polynomials arise in many situations while analyzing programs, and analysis systems are unable to handle such expressions. Although some dedicated methods have already been proposed, they only handle some subsets of such expressions. This paper presents an original and general approach to Bernstein expansion which is symbolic. Bernstein expansion allows bounding the range of a multivariate polynomial over a box and is generally more accurate than classic interval methods.

1 Introduction

Static program analysis has received a lot of attention in the past decades as mathematical frameworks for modeling programs have been developed. For example, the polytope model allows modeling nested loops as polyhedra whose contained integer points are associated to iterations. It provides mathematical tools for quantitative analysis and optimizing transformations. However this model is limited to loops whose loop bounds are affine functions over the enclosing loop indices, and array references are affine functions over the loop indices.

Linearity of the considered functions is one of the main limitations in formal analysis. Many behavior modeling issues where interactions between hardware and software are considered, as cache behavior of programs, involve nonlinear expressions. Compilers implementing advanced optimization transformations are also limited by this fact. In particular, multivariate polynomial functions arise in many situations resulting from linearized subscripts in array reference functions, or from induction variable recognition, and compilers fail in handling such expressions.

In this paper, we propose an extension of the theory of Bernstein expansion [3,4] to handle parameterized multivariate polynomial expressions. Bernstein expansion allows bounding the range of a multivariate polynomial considered over a box [2,11]. Numerical applications of this theory have been proposed to the resolution of system of strict polynomial inequalities [12,13]. It has been shown that Bernstein expansion is generally more accurate than classic interval methods [14]. Moreover, Stahl has shown in [17] that for *sufficiently small* boxes, the exact range is obtained.

In this paper, we generalize Bernstein expansion by considering parameterized multivariate polynomials for which ranges are bound by polynomials over the parameters. Sufficient conditions over the parameters for strict parameterized inequalities solutions can therefore be constructed.

Bernstein polynomials are particular polynomials that form a basis for the space of polynomials. Hence any polynomial can be expressed into this basis through coefficients, the Bernstein coefficients, that have interesting properties and that can be computed through a direct formula. Due to the Bernstein convex hull property [10], the value of the polynomial is then bounded by the values of the minimum and maximum Bernstein coefficients. The direct formula allows symbolic computation of these Bernstein coefficients giving a supplementary interest to the use of this theory. Another main interesting consequence is that the involved computations have quite low complexity. Moreover an appropriate instrumentation of this model allows automatic and inexpensive resolutions of complex program analysis issues.

This paper is organized as follows. Following a short and motivating application example in section 2, some notations are introduced and the handled polynomials are defined in section 3. In section 4, basic notions about Bernstein expansion are recalled as well as the general principles of its numerical application. Our symbolic approach is presented in section 5 where handled parameterized multivariate polynomials inequalities over parameterized boxes are considered. Several application examples are described in section 6: dependence testing between references with linearized subscripts, dead code elimination of conditional statements and computation of the minimum memory amount needed for array contraction. In section 7, we point out some current limits of our framework and discuss about some further developments. In this section, we also compare our approach to related works. Conclusions are finally given in section 8.

2 A Motivating Example

Consider the following loop, where "..." symbolizes some instructions having no effect on the considered ones:

```
q=alpha-25 ;
for (i=0 ; i<11 ; i++) {
  ...
  a[q]+=a[i+alpha+1] ;
  ...
  q-=i ;
  ...
  q+=9 ;
  ...
  q-=i ;
  ... }
```

In order to determine whether this loop can be parallelized, it has to be determined if array reference `a[q]` overlaps reference `a[i+alpha+1]`. For this, some induction variable recognition is first achieved [16] in order to get an expression of `q` depending on `i`:

```
for (i=0 ; i<11 ; i++) {
  ...
  a[-i*i+10*i+alpha-25]+=a[i+alpha+1] ;
  ... }
```

By applying conventional value range analysis on both polynomials $-i^2 + 10i + \alpha - 25$ and $i + \alpha + 1$, we get the following results:

$$\begin{aligned} -[0,10]^2 + 10[0,10] + \alpha - 25 &= [\alpha - 125, \alpha + 75] \\ [0,10] + \alpha + 1 &= [\alpha + 1, \alpha + 11] \end{aligned}$$

It yields the conclusion that the loop cannot be parallelized, since the value range of $i + \alpha + 1$ is entirely included in the value range of $-i^2 + 10i + \alpha - 25$. But since the form in which the polynomial is expressed can affect the result, we now use the Horner form of the polynomial $-i^2 + 10i + \alpha - 25 = \alpha - 25 + (10 - i)i$ which gives the more accurate bound $[\alpha - 25, \alpha + 75]$, yielding anyway to the same conclusion as before.

By applying our symbolic Bernstein expansion approach presented in this paper, we compute the Bernstein coefficients of $-i^2 + 10i + \alpha - 25$: $b_0 = \alpha$, $b_1 = \alpha - 25$ and $b_2 = \alpha$. Since the value is bounded by the minimum and maximum coefficients, it yields the *exact* value range $[\alpha - 25, \alpha]$. Moreover, it yields the conclusion that ranges of both references do not overlap at all and hence that the loop *can be parallelized*.

Notice that other more accurate methods recently proposed [5,6,9] also would have failed in this case, since the considered polynomial $-i^2 + 10i + \alpha - 25$ is not monotonous over the interval $[0, 10]$.

3 Considered Polynomials and Notations

Let $x = (x_1, \ldots, x_{l_x})$ denote a l_x-dimensional vector of rational variables and $u = (u_1, \ldots, u_{l_u})$ a l_u-dimensional vector of rational parameters.

$$\begin{aligned} \mathbb{Q}^{l_x} &= \{\bar{x} = (x_1, \ldots, x_{l_x}) \mid x_i \in \mathbb{Q},\ 1 \leq i \leq l_x\} \\ \mathbb{Q}^{l_u} &= \{\bar{u} = (u_1, \ldots, u_{l_u}) \mid u_i \in \mathbb{Q},\ 1 \leq i \leq l_u\} \end{aligned}$$

Any term $x_1^{i_1} x_2^{i_2} \ldots x_{l_x}^{i_{l_x}}$ is noted $\bar{x}^{I_x}$, where $I_x = (i_1, \ldots, i_{l_x})$ is a vector of l_x positive integer values. In the same way, $\bar{u}^{I_u}$ denotes any term $u_1^{i_i} \ldots u_{l_u}^{l_u}$ with $I_u = (i_1, \ldots, i_{l_u})$. I_x, respectively I_u, is called the *multi-index* of variable x, respectively u. The null vector $(0, \ldots, 0)$ is noted $\bar{0}$. In the following, an inequality $I = (i_1, \ldots, i_l) \leq J = (j_1, \ldots, j_l)$ means that for each $0 \leq k \leq l$ we have $i_k \leq j_k$.

Let $\mathbb{Q}[\bar{u}]$ be the set of polynomials defined by:

$$\mathbb{Q}[\bar{u}] = \{p(\bar{u}) \mid p(\bar{u}) = \sum_{\bar{0} \leq I_u \leq N_u} a_{I_u} \cdot \bar{u}^{I_u}\}$$

where coefficients $a_{I_u} \in \mathbb{Q}$ and $N_u = (n_{u,1}, \ldots, n_{u,l_u})$ is called the *maximal multi-degree* of u-variables, *i.e.*, the vector of the greatest degrees of the u-variables.

In this work, we consider polynomials over x-variables with coefficients in $\mathbb{Q}[\bar{u}]$ and with maximal multi-degree $N_x = (n_{x,1}, \ldots, n_{x,l_x})$:

$$Q[\bar{u}][\bar{x}] = \{p(\bar{x}, \bar{u}) \mid p(\bar{x}, \bar{u}) = \sum_{\bar{0} \leq I_x \leq N_x} p_{I_x}(\bar{u}) \cdot \bar{x}^{I_x},\ p_{I_x}(\bar{u}) \in \mathbb{Q}[\bar{u}]\}.$$

Any polynomial $p(\bar{x}, \bar{u})$ is considered over a box $D_x \times \mathbb{Q}^{l_u}$ where $D_x \subset \mathbb{Q}^{l_x}$. Considered boxes $D_x = [a_1, b_1] \times ... \times [a_l, b_l]$ are such that the a_i's or the b_i's can be polynomials over parameters or variables and such that the knowledge of the q first variables values determines the parametric values of a_{q+1} and b_{q+1}, $q < l$:

$$\begin{aligned} D_x = {} & [a_2 = pa_2(x_1, \bar{u}), b_2 = pb_2(x_1, \bar{u})] \times [a_3 = pa_3(x_1, x_2, \bar{u}), b_3 = pb_3(x_1, x_2, \bar{u})] \\ & \times ... \times [a_l = pa_l(x_1, x_2, ..., x_{l-1}, \bar{u}), b_l = pb_l(x_1, x_2, ..., x_{l-1}, \bar{u})] \end{aligned}$$

where the pa_i's and pb_i's are polynomials over some variables and parameters.

Our aim is to find integer solutions to the inequality:

$$p(\bar{x}, \bar{u}) > 0, \qquad \text{where } (\bar{x}, \bar{u}) \in D_x \times \mathbb{Q}^{l_u} \cap \mathbb{Z}^{l_x + l_u}$$

Solutions are expressed as sets D_0 and D_1, where $D = (D_x \times \mathbb{Q}^{l_u}) = (D_0 \sqcup D_1 \sqcup D_\epsilon)$ such that for all $(\bar{x}, \bar{u}) \in D_0 \cap \mathbb{Z}^{l_x + l_u}$ $p(\bar{x}, \bar{u}) > 0$, for all $(\bar{x}, \bar{u}) \in D_1 \cap \mathbb{Z}^{l_x + l_u}$ $p(\bar{x}, \bar{u}) \leq 0$, and D_ϵ denotes the set where the sign of $p(\bar{x}, \bar{u})$ is indeterminate.

In this paper we generalize the numerical Bernstein method by applying it to parameterized polynomials of the set $Q[\bar{u}][\bar{x}]$. We first recall the Bernstein theory and the numerical method.

4 Numerical Bernstein Method

Let

$$C_N^K = C_{n_1}^{k_1} \cdot \cdots \cdot C_{n_l}^{k_l}, \text{ where } \bar{0} \leq K = (k_1, .., k_l) \leq N = (n_1, ..., n_l),\ (K, N) \in \mathbb{N}_0^l,$$

$$\text{and } C_n^k = \frac{n!}{k!(n-k)!}, 0 \leq k \leq n,\ (k, n) \in \mathbb{N}_0$$

$$\begin{aligned} & I - J = (i_1 - j_1, \ldots, i_l - j_l), && \text{where } I = (i_1, \ldots, i_l),\ J = (j_1, \ldots, j_l),\ I, J \in \mathbb{N}_0^l \\ & \alpha^I = \alpha_1^{i_1} \cdot \cdots \cdot \alpha_l^{i_l}, && \text{where } \alpha = (\alpha_1, \ldots, \alpha_l) \in \mathbb{R}^l \end{aligned}$$

For polynomials with constant coefficients:

$$p(\bar{x}) = \sum_{\bar{0} \le I \le N} a_I \cdot \bar{x}^I \in \mathbb{R}[\bar{x}], \quad \bar{x} \in D \subset \mathbb{R}^l, a_I \in \mathbb{R}$$

and with maximal multi-degree N, Bernstein polynomials and Bernstein coefficients are defined in the following way.

In the univariate case, the i-th Bernstein polynomial of degree n on the unit interval $[0,1]$ is defined as:

$$b_{n,i}^{[0,1]}(x) = C_n^i \cdot x^i (1-x)^{n-i}, 0 \le i \le n, x \in \mathbb{R}$$

In the multivariate case, the I-th Bernstein polynomial on the unit box $[0,1]^l$, where $I = (i_1, \ldots, i_l)$ and $N = (n_1, \ldots, n_l)$ is defined by:

$$B_{N,I}^{[0,1]^l}(\bar{x}) = b_{n_1,i_1}^{[0,1]}(x_1) \cdot \cdots \cdot b_{n_l,i_l}^{[0,1]}(x_l), \bar{0} \le I \le N, \bar{x} = (x_1, \ldots, x_l) \in \mathbb{R}^l$$

Bernstein polynomials are defined on the whole l-dimensional space $\mathbb{R}^l$ and define a basis for polynomials $p(\bar{x})$:

$$p(\bar{x}) = \sum_{\bar{0} \le I \le N} b_I^{[0,1]^l} B_{N,I}^{[0,1]^l}(\bar{x}) = \sum_{\bar{0} \le I \le N} a_I \cdot \bar{x}^I,$$

where coefficients $b_I^{[0,1]^l}$ are the Bernstein coefficients of the polynomial $p(\bar{x})$ on the unit box $[0,1]^l$. They are determined by:

$$b_I^{[0,1]^l} = \sum_{\bar{0} \le J \le I} \frac{C_I^J}{C_N^J} a_J, \quad \bar{0} \le I \le N.$$

Theorem. *Let $p(\bar{x})$ be a polynomial of multi-degree N on the unit box $[0,1]^l$. For all $\bar{x} \in [0,1]^l$:*

$$\min_{\bar{0} \le I \le N} b_I^{[0,1]^l} \le p(\bar{x}) \le \max_{\bar{0} \le I \le N} b_I^{[0,1]^l}$$

Polynomials on any boxes $D \in \mathbb{R}^l$ of any sizes can also be considered since without loss of generality, any nonempty box can be mapped linearly onto the unit box $[0,1]^l$. Let ϕ be the linear transformation of the box D onto the unit box $[0,1]^l$. As a polynomial $p(\bar{x})$ is considered on a box D, then the corresponding polynomial on the unit box is $\tilde{p}(\bar{y}) = p(\phi^{-1}(\bar{y})), \bar{y} \in [0,1]^l$, and the Bernstein coefficients of the polynomial p on the box D are equal to the Bernstein coefficients of the polynomial $\tilde{p}$ on the unit box $[0,1]^l$: $b_I^D = b_I^{[0,1]^l}$.

Transformation ϕ is defined by:

$$\phi : \begin{array}{rl} \bar{x} = (x_1, \ldots, x_l) & \to \ \bar{y} = (y_1, \ldots, y_l) \\ [a_1, b_1] \times \cdots \times [a_l, b_l] & \to [0,1]^l, \quad a_i, b_i \in \mathbb{R} \end{array} \quad \text{where } y_i = \frac{x_i - a_i}{b_i - a_i}, 0 \le i \le l$$

Hence the inverse transformation is:

$$\begin{array}{rl} \phi^{-1} : \bar{y} = (y_1, \dots, y_l) \rightarrow & \bar{x} = (x_1, \dots, x_l) \\ [0,1]^l & \rightarrow [a_1, b_1] \times \cdots \times [a_l, b_l], \quad a_i, b_i \in \mathbb{R} \end{array}$$
$$\text{where } x_i = \alpha_i \cdot y_i + \beta_i, \alpha_i = (b_i - a_i), \beta_i = a_i, 0 \le i \le l, 0 \le i \le l$$

And $\tilde{p}(\bar{y})$ is such that:

$$\tilde{p}(\bar{y}) = \sum_{\bar{0} \le P \le N} \tilde{a}_P \bar{y}^P \quad \text{where } \tilde{a}_P = \alpha^P \cdot \sum_{P \le K \le N} a_K \cdot C_K^P \cdot \beta^{K-P}$$

Note that the maximal multi-degree N is the same for $p(\bar{x})$ and $\tilde{p}(\bar{y})$.

To apply the Bernstein method for polynomials over an arbitrary box D, the mapping ϕ is constructed and the Bernstein method is applied to the assigned polynomial $\tilde{p}(\bar{y})$ over the unit box $[0,1]^l$.

Using the theorem, an algorithm to find solutions to the inequality $p(\bar{x}) > 0, x \in [0,1]^l$, can be constructed. Using properties of the Bernstein coefficients of the polynomial on the box D and since $b_I^D = b_I^{[0,1]^l}$ for all $\bar{0} \le I \le N$, we have:

- if $\min_{\bar{0} \le I \le N} b_I^D > 0$, then for all $x \in D, p(x) > 0$
- if $\max_{\bar{0} \le I \le N} b_I^D \le 0$, then for all $x \in D, p(x) \le 0$

The complete algorithm can be found in [12,13].

5 Symbolic Approach

Our extension considers polynomials $p(\bar{x}, \bar{u})$ with parametric coefficients over boxes whose bounds can also be parametric. Moreover these bounds can also depend on variables.

5.1 Parameterized Multivariate Polynomials

The transformation of any polynomial:

$$p(\bar{x}, \bar{u}) = \sum_{\bar{0} \le I_x \le N_x} p_{I_x}(\bar{u}) \cdot \bar{x}^{I_x}, \quad \bar{x} \in D_x, \ \bar{u} \in \mathbb{Q}^{l_u}$$

into the polynomial:

$$\tilde{p}(\bar{y}, \bar{u}) = \sum_{\bar{0} \le I_x \le N_x} \tilde{p}_{I_x}(\bar{u}) \cdot \bar{y}^{I_x}, \quad \bar{y} \in [0,1]^{l_x}, \ \bar{u} \in \mathbb{Q}^{l_u}$$

can also be found in the case of parameterized coefficients $p_{I_x}(\bar{u})$ since the formulas are the same:

$$\tilde{p}_{I_x}(\bar{u}) = \alpha^{I_x} \cdot \sum_{I_x \le K_x \le N_x} p_{K_x}(\bar{u}) \cdot C_{K_x}^{I_x} \cdot \beta^{K_x - I_x}$$

where like above $D_x = [a_1, b_1] \times \cdots \times [a_{l_x}, b_{l_x}], \alpha_i = (b_i - a_i), \beta_i = a_i, 0 \leq i \leq l_x$. Hence parameterized Bernstein coefficients can be computed by:

$$b_{I_x}^{[0,1]^l} = \sum_{\bar{0} \leq J \leq I_x} \frac{C_{I_x}^J}{C_{N_x}^J} \tilde{p}_J(\bar{u}), \quad \bar{0} \leq I_x \leq N_x.$$

and the inequality $p(\bar{x}, \bar{u}) > 0, \quad \bar{x} \in D_x, \bar{u} \in \mathbb{Q}^{l_u}$ holds if $\min_{\bar{0} \leq I_x \leq N_x} b_{I_x}^{D_x} > 0$. But since the Bernstein coefficients are polynomials over the parameters $\bar{u}$, the minimum of these coefficients is generally difficult to determine. Verifying that all the coefficients are strictly positive is equivalent and the following system is considered: $b_{I_x}^{D_x} > 0$, for all $\bar{0} \leq I_x \leq N_x$.

Moreover, if coefficients $p_{I_x}(\bar{u})$ are linear functions over the parameters $\bar{u}$, the Bernstein coefficients of the polynomial p on the box D_x are also linear functions of the parameters and the latter system is a system of linear inequalities. Such a system is handled by several tools manipulating linear functions like for example the Polyhedral library *PolyLib* [1].

Example 1. Consider the polynomial $p(\bar{x}, \bar{u}) = (N + M)x_1x_2 + Nx_1 + Mx_2$ with $\bar{x} = (x_1, x_2), \bar{u} = (N, M)$ and $\bar{x} \in D_x = [0, 100] \times [0, 100]$. Hence multidegree $N_x = (1, 1)$. Box D_x is mapped onto the unit box $[0, 1]^2$ by transformation ϕ defined by $\phi(\bar{x}) = \bar{y} = (y_1 = x_1/100, y_2 = x_2/100)$. Hence polynomial $\tilde{p}(\bar{y}, \bar{u}) = (10000N + 10000M)y_1y_2 + 100Ny_1 + 100My_2$ with $\bar{y} \in [0, 1]^2$. Bernstein coefficients are computed for all $(0, 0) \leq I_x \leq (1, 1)$:

- for $I_x = (0, 0)$: $b_{(0,0)}^{D_x} = \frac{C_0^0 C_0^0}{C_1^0 C_1^0} 0 = 0$;
- for $I_x = (0, 1)$: $b_{(0,1)}^{D_x} = \frac{C_0^0 C_1^0}{C_1^0 C_1^0} 0 + \frac{C_0^0 C_1^1}{C_1^0 C_1^1} 100M = 100M$;
- for $I_x = (1, 0)$: $b_{(1,0)}^{D_x} = \frac{C_0^0 C_1^0}{C_1^0 C_1^0} 0 + \frac{C_1^1 C_0^0}{C_1^1 C_1^0} 100N = 100N$;
- for $I_x = (1, 1)$: $b_{(0,1)}^{D_x} = \frac{C_1^0 C_1^0}{C_1^0 C_1^0} 0 + \frac{C_1^0 C_1^1}{C_1^0 C_1^1} 100M + \frac{C_1^1 C_1^0}{C_1^1 C_1^0} 100N + \frac{C_1^1 C_1^1}{C_1^1 C_1^1} (10000N + 10000M) = 10100N + 10100M$;

Inequality $p(\bar{x}, \bar{u}) \geq 0$ holds when all these Bernstein coefficients are positive. In this example, it can be easily deduced that it holds for $N \geq 0$ and $M \geq 0$.

5.2 Parameterized Boxes

Since D_x has bounds that can depend on parameters or variables, transformation ϕ to map D_x onto the unit box is no more linear and is defined from variables and parameters: $y_i = \frac{x_i - a_i}{b_i - a_i}$ where a_i and b_i can be polynomials over variables and parameters. Hence, ϕ is valid if and only if $b_i - a_i \neq 0$, which means that $a_i < b_i$. So a valid mapping on the unit box can only be defined for D_x where parameters and variables values such that $b_i - a_i = 0$ have been excluded. We note this reduced set D_x^* which is generally a union of disjoint boxes:

$$D_x^* = D_x - \{\bar{x} \in D_x \mid \exists\, 1 \leq i \leq l, x_i = \mathit{Root\_of}(b_i - a_i)\}$$

When the a_i's and b_i's are affine functions of variables and parameters, the roots of $(b_i - a_i)$ are easily computed. Otherwise, roots can also be easily computed for univariate polynomials of degree at most 4. Transformation ϕ is then defined and applied on D_x^* in the following way.

At first, the polynomial $\tilde{p}(\bar{y}, \bar{u})$ is computed as described above. Then an additional transformation consisting in propagating the new variables in $\tilde{p}(\bar{y}, \bar{u})$ is necessary. Finally, the Bernstein coefficients are computed from this polynomial on $[0, 1]^l$. A specific value-per-value analysis is finally done for all roots of the $(a_i - b_i)$'s polynomials.

Example 2. Consider the polynomial $p(i, j) = (i^2 + i)/2 + j$ defined on the box $D_x = [0, N] \times [0, i]$. Transformation ϕ is such that $\phi(i, j) = (i' = i/N, j' = j/i)$ which is not valid for $N = 0$ and $i = 0$. Hence point $(0, 0)$ is excluded from D_x: $D_x^* = D_x - \{(0, 0)\} = [1, N] \times [0, i]$.

On D_x^*, the polynomial $p(i, j)$ is first transformed into $(N^2 i'^2 + Ni')/2 + ij'$. Variable i' is then propagated and $\tilde{p}(i', j') = (N^2 i'^2 + Ni')/2 + Ni'j', (i', j') \in [0, 1] \times [0, 1]$.
□

We use this symbolic approach to get sufficient conditions for any considered parameterized multivariate polynomial to be strictly positive or negative. We show with some examples in next section how an appropriate instrumentation of this model allows automatic and inexpensive resolutions of complex program analysis issues.

6 Applications

A typical application of our mathematical framework is dependence testing between references with linearized subscripts. Linearized subscripts can result from optimizations like spatial data locality improvement [7]. Let us first consider an example given by Maslov and Pugh in [15] in order to show how our method can handle linearized subscripts. Although this example is also handled by Maslov and Pugh, we show in the following some cases that could not, because of the limitations of their method explained in section 7.

Example 3. The first example shown on figure 6 is a loop nest from the oil simulation program BOAST in the RiCEPS benchmark suite. Anyway this kind of loop nest is quite typical and met quite often in programs.

To be able to parallelize or to transform any of the `i`, `j` and `k` loops, it must be checked whether the flow dependence from the first statement to the second statement is loop-independent. Compilers fail to prove that this dependence is loop-independent.

We instrument our model in the following way: consider both iterations (i_1, j_1, k_1) and $(i_1 + \alpha, j_2, k_2)$. If we prove that:

$$MNi_1 + Nj_1 + k_1 < MN(i_1 + \alpha) + Nj_2 + k_2 \text{ for any } \alpha \geq 1$$

```
do i=p,p+L-1
 do j=q,q+M-1
  do k=r,r+N-1
   A(M*N*i+N*j+k)=...
   ...=A(M*N*i+N*j+k)
  enddo
 enddo
enddo
```

(a) First typical example of loop nest with linearized references.

```
do i=2,N
 do j=1,i-1
  if 4*i*i+j>=3*i*j+5*i+1
   ...
  else
   ...
  endif
 enddo
enddo
```

(b) Second example with a complex conditional statement which can be eliminated.

```
do i=1,N
 do j=i+1,(N-i)/2
  A(i,j)=A(j-1,N+i-j)
 enddo
 do j=(N-i)/2+1,N
  A(i,j)=A(i-1,j-i)
 enddo
enddo
```

(c) Third example of loop nest for which accessed array **A** has to be compacted.

Fig. 1. Examples.

then we prove that $MNi_1+Nj_1+k_1$ can never be equal to $MN(i_1+\alpha)+Nj_2+k_2$ and so that the dependence is independent of loop i. We use our method to get sufficient conditions for:

$$\begin{aligned} p(\bar{x},\bar{u}) &= MNi_1 + Nj_1 + k_1 - (MN(i_1+\alpha) + Nj_2 + k_2) \\ &= Nj_1 + k_1 - Nj_2 - k_2 - MN\alpha < 0 \end{aligned}$$

with $\bar{x} = (j_1, k_1, j_2, k_2)$ and $\bar{u} = (\alpha, N, M, p, L, q, r)$. Variables are considered on the box $D_x = [q, q+M-1] \times [r, r+N-1] \times [q, q+M-1] \times [r, r+N-1]$.

Transformation ϕ to map D_x on the unit box is defined by:

$$\bar{y} = (\frac{j_1-q}{M-1}, \frac{k_1-r}{N-1}, \frac{j_2-q}{M-1}, \frac{k_2-r}{N-1})$$

which is valid for any variables values and for any parameters values strictly greater than one. It transforms polynomial $p(\bar{x},\bar{u})$ into $\tilde{p}(\bar{y},\bar{u})$:

$$\begin{aligned} \tilde{p}(\bar{y},\bar{u}) &= N((M-1)j_1' + q) + (N-1)k_1' + r - N((M-1)j_2' + q) \\ &\quad -((N-1)k_2' + r) - MN\alpha \\ &= (NM-N)j_1' + (N-1)k_1' - (NM-N)j_2' - (N-1)k_2' - MN\alpha \end{aligned}$$

with $\bar{y} = (j_1', k_1', j_2', k_2') \in [0,1]^4$, $M > 1$ and $N > 1$. We use our software to compute the Bernstein coefficients of $\tilde{p}(\bar{y},\bar{u})$ and get the following answer:

$$p(\bar{x},\bar{u}) < 0 \text{ if } \begin{cases} NM\alpha > 0 \\ NM\alpha + N - 1 > 0 \\ NM\alpha + NM - N > 0 \\ NM\alpha + NM - 1 > 0 \\ NM\alpha - N + 1 > 0 \\ NM\alpha + NM - 2N + 1 > 0 \\ NM\alpha - NM + N > 0 \\ NM\alpha - NM + 2N - 1 > 0 \\ NM\alpha - NM + 1 > 0 \end{cases}$$

It can be easily checked that all these conditions are equivalent to $\alpha \geq 1$, since we are only concerned with integer values. Hence we have proved that the dependence is independent of loop i which can be parallelized. In the perspective of a fully automated framework, Bernstein expansion could be used again to check the nonlinear conditions.

□

We now consider an example showing our framework usability for dead code elimination of conditional statements.

Example 4. Consider the loop nest shown on figure 6. It contains a complex conditional statement that can be the result of induction variable recognition [16]. We use our tool to check whether the condition is either always true or always false in order to eventually eliminate some dead code.

Checking if $4i^2+j \geq 3ij+5i+1$ is equivalent to check if $p(\bar{x},\bar{u}) = 4i^2-3ij-5i+j-1 \geq 0$. Here we have $\bar{x}=(i,j)$, $\bar{u}=N$ and $D_x=[2,N]\times[1,i-1]$. Hence ϕ is defined by $\bar{y}=(\frac{i-2}{N-2},\frac{j-1}{i-2})$ which is valid only if N is strictly greater than two and for all points excepting point $(2,1)$: $D_x^*=[3,N]\times[1,i-1]$. It comes that N has to be strictly greater than three since ϕ is redefined as $\bar{y}=(\frac{i-3}{N-3},\frac{j-1}{i-2})$. We compute $\tilde{p}(\bar{y},\bar{u})$:

$$\begin{aligned}\tilde{p}(\bar{y},\bar{u}) &= 4((N-3)i'+3)^2-3((N-3)i'+3)((i-2)j'+1)-5((N-3)i'+3)\\ &\quad+((i-2)j'+1)-1\\ &= 4((N-3)i'+3)^2-3((N-3)i'+3)((((N-3)i'+3)-2)j'+1)\\ &\quad-5((N-3)i'+3)+(((N-3)i'+3)-2)j'\\ &= (-24N+36+4N^2)i'^2+(-27-3N^2+18N)i'^2j'+(-48+16N)i'\\ &\quad+(-11N+33)i'j'-8j'+12\end{aligned}$$

with $\bar{y}=(i',j')\in[0,1]^2$ and $N>3$. We compute the Bernstein coefficients of $\tilde{p}(\bar{y},\bar{u})$ and get the following answer:

$$\begin{aligned}p(\bar{x},\bar{u})>0 \text{ if } \{\, &12>0, 4>0, 8N-12>0, \tfrac{5N-7}{2}>0,\\ &4N^2-8N>0, N^2-N-2>0\}\end{aligned}$$

It can easily be checked that all these conditions always hold when $N>2$. For point $(2,1)$ which was excluded, we observe that $p(\bar{x},\bar{u})=0$. Hence $p(\bar{x},\bar{u})$ is always positive on D_x, the condition and the "else" parts of the code can be eliminated.

□

Example 5. This example aims to show that our polynomial evaluation method is also useful to achieve complex analysis and optimizations. Array contraction is an optimization that transforms an array into a smaller array within a loop in order to save memory and to increase locality. It is particularly useful for embedded systems.

Consider the loop nest shown on figure 6. We want to achieve contraction of array `A` assuming that `A` is a temporary array not further accessed in the rest of the program. An array element is alive when it has been accessed through read or write at least once and will be accessed again in the future. The minimum amount of memory needed is given by the maximum number of simultaneously alive array elements. Contraction of array `A` to this amount can be achieved by defining new access functions and data layout.

Determining the maximum number of simultaneously alive array elements generally translates in maximizing a parameterized multi-variate polynomial. It is shown that such objective can be reached by our method.

We first compute the live range of any array element `A(x,y)`, i.e., the number of iterations occurring between the first and the last references to `A(x,y)`. The maximum live range will also be the maximum number of simultaneously alive array elements, and also the minimum amount of memory needed, since at each iteration an array element is accessed for the first time and another one is accessed for the last time. Moreover, each array element is referenced at most once by each reference.

The live range of `A(x,y)` is computed in the following way: for each reference `A(i,j)`, `A(j-1,N+i-j)` and `A(i-1,j-1)`, we compute the number of iterations occurring before the one where the considered reference resolves in `A(x,y)`. For example, considering reference `A(i,j)`, `A(x,y)` is referenced when $i = x$ and $j = y$. Hence we compute the number of iterations occurring before iteration (x, y) included. When all the references have been considered, the live range of `A(x,y)` is given by the largest difference between these iteration counts.

These iteration counts are given by the Ehrhart polynomials of some appropriately defined polytopes [7]. We get the following results:

Reference	Last Iteration	Iteration count
`A(i,j)`	(x, y)	$ic_1 = (N - \frac{1}{2})x - \frac{1}{2}x^2 + y - N$
`A(j-1,N+i-j)`	$(x + y - N + 1, x + 1)$	$ic_2 = -\frac{1}{2}x^2 - xy + (2N - \frac{1}{2})x - \frac{1}{2}y^2 + (2N - \frac{3}{2})y - \frac{3}{2}N^2 + \frac{3}{2}N$
`A(i-1,j-1)`	$(x + 1, y + x + 1)$	$ic_3 = xN - \frac{1}{2}x^2 - \frac{1}{2}x + y$

Since `A(j-1,N+i-j)` and `A(i-1,j-1)` do not reference the same array elements, since reference to `A(x,y)` through `A(j-1,N+i-j)` occurs before reference to `A(x,y)` through `A(i,j)`, and since reference to `A(x,y)` through `A(j-1,N+i-j)` occurs after reference to `A(x,y)` through `A(i,j)`, we compute the differences $ic_1 - ic_2 + 1$ and $ic_3 - ic_1 + 1$:

$$\begin{aligned} ic_1 - ic_2 + 1 &= xy - Nx + \tfrac{1}{2}y^2 + (\tfrac{5}{2} - 2N)y - \tfrac{5}{2}N + \tfrac{3}{2}N^2 + 1 \\ ic_3 - ic_1 + 1 &= N + 1 \end{aligned}$$

The maximum number of simultaneously active array elements that are referenced through `A(i-1,j-1)` is $ic_3 - ic_1 + 1 = N + 1$. But in order to determine the maximum number of simultaneously active array elements that are referenced through `A(j-1,N+i-j)`, we need to determine the maximum value of $ic_1 - ic_2 + 1$ for any value of x and y. It is given by the largest Bernstein coefficient of $p(\bar{x}, \bar{u}) = ic_1 - ic_2 + 1$, where $\bar{x} = (x, y)$ and $\bar{u} = N$.

Since $1 \leq i \leq N$ and $i + 1 \leq j \leq (N - i)/2$, and since $x = j - 1$ and $y = N + i - j$, it comes that $1 \leq x \leq (N - 3)/2$ and $(N + 1)/2 \leq y \leq N - 1$.

Hence we have $\bar{x} = (x, y)$, $\bar{u} = N$ and $D_x = [1..(N-3)/2] \times [(N+1)/2..N-1]$. Transformation ϕ to map D_x on the unit box is defined by $\bar{y} = (\frac{2x-2}{N-5}, \frac{2y-N-1}{N-3})$ which is not valid for $N = 5$ and $N = 3$. Let us consider $N > 5$. Mapping ϕ transforms polynomial $p(\bar{x}, \bar{u})$ into $\tilde{p}(\bar{y}, \bar{u})$:

$$\tilde{p}(\bar{y}, \bar{u}) = (\tfrac{1}{4}N^2 - 2N + \tfrac{15}{4})x'y' + (-\tfrac{1}{4}N^2 + \tfrac{3}{2}N - \tfrac{5}{4})x' + (\tfrac{1}{8}N^2 - \tfrac{3}{4}N + \tfrac{9}{8})y'^2 \\ +(\tfrac{17}{4}N - 6 - \tfrac{3}{4}N^2)y' - \tfrac{5}{2}N + \tfrac{15}{8} + \tfrac{5}{8}N^2 + 1$$

with $\bar{y} = (x', y') \in [0, 1]^2$ and $N > 5$. We compute the Bernstein coefficients of $\tilde{p}(\bar{y}, \bar{u})$ which are: $\frac{5}{8}N^2 - \frac{5}{2}N + \frac{23}{8}$, $\frac{1}{4}N^2 - \frac{3}{8}N - \frac{1}{8}$, $N - 2$, $\frac{3}{8}N^2 - N + \frac{13}{8}$, $\frac{1}{8}N^2 + \frac{1}{8}N + \frac{1}{2}$, $\frac{1}{2}N + \frac{1}{2}$.

The largest coefficient is obviously $\frac{5}{8}N^2 - \frac{5}{2}N + \frac{23}{8}$. Hence the maximum number of simultaneously active array elements that are referenced through `A(j-1,N+i-j)` has been found, and the total number of simultaneously active array elements is $\frac{5}{8}N^2 - \frac{5}{2}N + \frac{23}{8} + N + 1 = \frac{5}{8}N^2 - \frac{3}{2}N + \frac{31}{8}$. Finally, array `A` can be contracted to $\frac{5}{8}N^2 - \frac{3}{2}N + \frac{31}{8}$ elements.

7 Limits, Further Developments, and Related Works

Two main problems may arise when applying our symbolic Bernstein method:

1. The program output can be a complex system of multivariate polynomial inequalities from which it is impossible to conclude immediately whether the initial polynomial inequation always holds or not, or what are the cases when it holds.
2. It can be impossible to conclude whether the initial polynomial is always positive or negative over the box D_x, since the output system of inequations has no solution.

In the first case, we propose to iterate our method on the most complex polynomials composing the output system, by setting a box for some parameters becoming the variables of a new problem. Solutions found by this way can then be propagated into the other polynomials forming the system. After several such iterations, solutions of the whole system should be found. We are currently investigating this approach.

In the second case, the initial box D_x needs to be cut in order to find solutions on some sub-boxes. Garloff presents in [12] a procedure to cut boxes in an opportune way. But since we consider parameterized polynomials, this procedure can not apply in our case. We will first try the approach of bisecting boxes into sub-boxes of equal volumes, and then think about a more elaborated strategy.

Another direction consists in elevating the degree of the polynomial $\tilde{p}(\bar{y}, \bar{u})$ over the unit box to make the Bernstein bounds tighter and get a system of inequations having some solutions.

Maslov and Pugh present in [15] a technique to simplify polynomial constraints. It is based on a decomposition of any polynomial constraint into a conjunction of affine constraints and 2-variable hyperbolic and elliptical inequalities and equalities that can later be linearized. Hence their approach is not general and can only handle those polynomials that can be decomposed in this way.

In [6], Blume and Eigenmann propose to test dependences between iterations of a loop, in the presence of nonlinear expressions, by verifying whether the ranges of such an expression does not overlap between successive iterations. However their technique is only valid if the expressions are monotonically increasing or decreasing with the loop index, and if symbolic lower and upper bounds can be easily determined. We are not concerned with such limitation.

Same authors in [5] present a expression-comparison algorithm consisting in replacing each variable in expressions with their range. Their technique needs the use of rewrite rules for simplifying expressions containing ranges. Two variable replacing methods are described. A first one that can generate overly conservative lower and upper bounds, and a second one, giving more accurate bounds, but that only applies on expressions that are monotonic over the considered range. Moreover, monotony has first to be proved by use of the first method. Since overly conservative bounds can be generated, some monotonic expressions can improperly be evicted.

Van Engelen *et al.* in [9] propose to obtain accurate bounds of a polynomial over the parameterized box $[0, n-1]$ by computing its Newton series. However their method is limited to univariate and monotonically increasing or decreasing polynomials.

8 Conclusion

It has been shown that the symbolic use of Bernstein expansion for the analysis of multivariate polynomials has several advantages:

- it is generally more accurate than classic interval methods, and even exact in many cases ;
- it handles any multivariate polynomial without restriction ;
- complexity of the output system of inequations directly depends on the complexity of the parameterized coefficients. If these coefficients are linear combination of parameters, the output is a system of linear equations that can be solved easily.

The main drawback is its need of computing time and memory which grows exponentially with the number of variables. However, most problems arising in static analysis involves only a few variables. Moreover, some very recent developments have shown the opportunity of a linear complexity algorithm [8].

Our symbolic approach has been implemented for the field of coefficients in $\mathbb{Q}$. The software uses the GNU-MP library for arbitrary precision arithmetic. For the future we plan to integrate our tool with the polyhedral library *PolyLib* in order to be able to directly solve the involved linear equation systems and to extend *Polylib* to handle multi-variate polynomials.

When considering nonlinear expressions in program analysis, propositions of non-exhaustive techniques become more and more current: program tracing, heuristics, rough approximations, dynamic or run-time analysis, genetic algorithms, ... We argue that results obtained by these ways can never be as accurate as the ones obtained by static analysis using general mathematical frameworks. Moreover, correctness is becoming of paramount concern in more and more areas such as safety-critical systems or compilation into silicon.

References

1. The polyhedral library *polylib*. http://icps.u-strasbg.fr/PolyLib.
2. Jakob Berchtold and Adrian Bowyer. Robust arithmetic for multivariate bernstein-form polynomials. *Computer-aided Design*, 32:681–689, 2000.
3. S. Bernstein. *Collected Works*, volume 1. USSR Academy of Sciences, 1952.
4. S. Bernstein. *Collected Works*, volume 2. USSR Academy of Sciences, 1954.
5. W. Blume and R. Eigenmann. Symbolic range propagation. In *9th Int. Parallel Processing Symposium*, April 1995.
6. W. Blume and R. Eigenmann. Non-linear and symbolic data dependence testing. *IEEE Transactions on Parallel and Distributed Systems*, 9(12):1180–1194, December 1998.
7. Ph. Clauss and B. Meister. Automatic memory layout transformation to optimize spatial locality in parameterized loop nests. *ACM SIGARCH Computer Architecture News*, 28(1), March 2000.
8. J. Delgado and J.M. Peña. A linear complexity algorithm for the bernstein basis. In *IEEE Int. Conf. on Geometric Modeling and Graphics (GMAG'03)*, pages 162–167, July 2003.
9. R. Van Engelen, K. Gallivan, and B. Walsh. Tight timing estimation with the newton-gregory formulae. In *10th Workshop on Compilers for Parallel Computers, CPC 2003*, Jan. 2003.
10. G. Farin. *Curves and Surfaces in Computer Aided Geometric Design.* Academic Press, San Diego, 1993.
11. R.T. Farouki and V.T. Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4:191–216, 1987.
12. J. Garloff. Application of bernstein expansion to the solution of control problems. In J. Vehi and M. A. Sainz, editors, *Proceedings of MISC'99 - Workshop on Applications of Interval Analysis to Systems and Control*, pages 421–430. University of Girona, Girona (Spain), 1999.
13. J. Garloff and B. Graf. *The Use of Symbolic Methods in Control System Analysis and Design*, chapter Solving Strict Polynomial Inequalities by Bernstein Expansion, pages 339–352. Institution of Electrical Engineers (IEE), London, 1999.
14. R. Martin, H. Shou, I. Voiculescu, A. Bowyer, and G. Wang. Comparison of interval methods for plotting algebraic curves. *Computer Aided Geometric Design*, 19:553–587, 2002.
15. Vadim Maslov and William Pugh. Simplifying polynomial constraints over integers to make dependence analysis more precise. In *CONPAR 94 - VAPP VI, Int. Conf. on Parallel and Vector Processing*, Sept. 1994.
16. Sébastian Pop. Analysis of induction variables using chains of recurrences: extensions. Master's thesis, Université Louis Pasteur, Strasbourg, July 2003.
17. V. Stahl. *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations.* PhD thesis, Johannes Kepler University Linz, Austria, 1995.