The Power of Polynomials

Paul Feautrier

ENS de Lyon
Paul.Feautrier@ens-lyon.fr

January 8, 2015

Motivation: Polynomials Everywhere, I

```
k =0;
for(i=0; i<N; i++)
  for(j=0; j<N; j++)
    a[k++] = 0.;
```

→

```
for(i=0; i<N; i++)
  for(j=0; j<N; j++)
    a[N*i+j] =0.;
```

Are the loops parallel? Are there loop-carried dependences?

Can be solved by *delinearization*, or by the SMT solver Z3, or by ISL using Bernstein polynomials. Other approaches?

Polynomials Everywhere: Scheduling

Find a schedule for:

```
s = 0.;
for(i=1; i<N; i++)
  for(j=0; j<i; j++)
    s += a[i][j];
```

Since the program runs in time $O(N^2)$ whatever the number of processors, it has no affine schedule. It has a two-dimensional schedule, which is equivalent to a quadratic schedule.

Can one find the quadratic schedule directly?

Polynomials Everywhere: Transitive Closure

What is the *exact* transive closure of:

$$(x' = x + y, y' = y, i' = i + 1)?$$

Answer:

$$(x' - i'.y' = x - i.y, y' = y, i' \geq i).$$

a polynomial relation.

The Basic Problem

Given: a set K and a function f, is f positive in K:

$$\forall x \in K : f(x) > 0?$$

Extension: f is a *template* depending on a vector of parameters μ. Find μ such that:

$$\forall x \in K : f_\mu(x) > 0.$$

Farkas lemma is the case where K is a polyhedron $K = \{x \mid Ax + b \geq 0\}$ and f is affine. The solution is:

$$f(x) = \lambda_0 + \lambda.(Ax + b) \ , \lambda \geq 0$$

Notations

A semi-algebraic set (sas):

$$K = \{x \mid p_1(x) \geq 0, \ldots, p_n(x) \geq 0\}$$

where x is a set of unknowns $x_1, \ldots, x_p$ and the p_is are polynomials in x. A polyhedron is an sas such that all the p_is are of first degree.

Schweighofer products: for each $\vec{e} \in \mathbb{N}^n$:

$$S_{\vec{e}}(x) = p_1^{e_1}(x) \ldots p_n^{e_n}(x) = \prod_{i=1}^{n} p_i^{e_i}(x).$$

Given a finite subset $Z \subset \mathbb{N}^n$ the associated Schweighofer sum is:

$$S_Z(x) = \sum_{\vec{e} \in Z} \lambda_{\vec{e}}.S_{\vec{e}}(x), \ \lambda_{\vec{e}} > 0.$$

Clearly, all Schweighofer sums are positive in K.

Theorems

Theorem (Handelman, 1988)

If K is a compact polyhedron, then a polynomial p is strictly positive in K if and only if it can be represented as a Schweighofer sum for some finite $Z \in \mathbb{N}^n$.

Theorem (Schweighofer, 2002)

If K is the intersection of a compact polyhedron and a semi-algebraic set, then a polynomial p is strictly positive in K if it can be represented as a Schweighofer sum for some finite $Z \in \mathbb{N}^n$.

Notice the similarity between the *conclusion* of the two theorems, and the difference with Farkas lemma: since there is no known bound on the size of Z, it is usually impossible to obtain a negative answer.

Algorithm H

The aim of this algorithm is to collect a set $\mathcal{C}$ of constraints on the unknowns λ and μ.

- $\mathcal{C} = \emptyset$.
- Given: a set of Schweighofer products $\{S_{\vec{e}}(x) \mid \vec{e} \in Z \subset \mathbb{N}^n\}$ and a polynomial (template) $p_\mu(x)$,
- Result: A system of constraints on the λ and μ.
- Completely expand the *master equation*:

$$E = p_\mu(x) - \sum_{\vec{e} \in Z} \lambda_{\vec{e}}.S_{\vec{e}}(x).$$

- For each monomial $x_1^{f_1} \dots x_p^{f_p}$, collect its coefficient c and add $c = 0$ to $\mathcal{C}$. c is an affine form in the λ and μ.

Comments

- Algorithm H works equally well in the Handelman or Schweighofer case, provided one use a uniform representation of polynomials, whatever their degree.
- The main difficulty is the selection of the products. One may use an oracle(!), or all products of a given degree, or all products of a given number of antecedents.
- The resulting system of constraints may be used in many ways: it may be solved by itself, or may be combined with other constraints before solving.
- If a solution for the λ and μ is found, this solution can be certified, independently of Handleman or Schweighofer, by straightforward algebraic evaluation.

Dependence Tests

A dependence set D is defined by a system of constraints:

- The iteration domains of its source and destination,
- A set of subscript equations,
- An order predicate.

Some or all of these constraints may involve polynomials. The problem is to decide whether this set is empty or not.
A possible solution is to prove, using algorithm H, that -1 is a positive combination of Schweighofer products of D!
Since -1 can never be positive, it follows that the constraints defining D cannot all be satisfied at the same time, i.e. that D is empty. Compare to the familiar Fourier-Motzkin algorithm.

An Example

The dependence set:

```
for(i=0; i<n; i++)
  for(j=0; j<n; j++)
    a[N*i+j] = 0.;
```

$$\begin{aligned} 0 \le i \le N-1 \quad &, \quad 0 \le i' \le N-1 \\ 0 \le j \le N-1 \quad &, \quad 0 \le j' \le N-1 \\ Ni + j \quad &= \quad Ni' + j' \\ i + 1 \quad &\le \quad i' \end{aligned}$$

Algorithm H finds the following solution:

$$\begin{aligned} -1 \quad = \quad & (N - i - 1)(i' - i - 1) + i(i' - i - 1) + (i' - i - 1) \\ + \quad & j' + (N - j - 1) + (Ni + j - Ni' - j') \end{aligned}$$

Hence, the dependence set is empty.

Scheduling

Notations

- $R, S, \ldots$ a set of *instructions*
- D_R the iteration domain of R, usually a polyhedron, sometimes an sas
- $\Delta_{RS} \subseteq D_R \times D_S$, a dependence set from R to S.

Problem For each statement R find a function $\theta_R : D_R \rightarrow \mathbb{Z}$ such that:

$$x \in D_R \Rightarrow \theta_R(x) \geq 0$$

$$\begin{pmatrix} x \\ y \end{pmatrix} \in \Delta_{RS} \Rightarrow \theta_R(x) + 1 \leq \theta_S(y)$$

Method

- For each statement R, build a template schedule θ_R by applying the first part of algorithm H to D_R
- For each dependence, build a master equation for the *delay* $\theta_S(y) - \theta_R(x) - 1$ by applying algorithm H to Δ_{RS}
- Collect the constraints and solve for the λ and μs using a linear programming tool.

DEMONSTRATION

Result

```
table((__node,S) = [[i,j],{(N >= i+1),(i >= j+1),(i >= 1),
   (j >= 0)}],(__nodes) = [S],(__transition,T0) = [S,S,table(i = i',j = j'),
     {(i' >= i+1)}],(__transition,T1) = [S,S,table(i = i',j = j'),{(i = i'),
       (j' >= j+1)}],(__transitions) = [T0,T1])

(N * N)*mu_6+N*i*mu_11+N*i*mu_8+N*j*mu_15+N*mu_5+(i * i)*mu_12+
...
(j * j)*mu_16-j*mu_15-j*mu_16-j*mu_17-j*mu_7-mu_10-mu_5-mu_7

dependence polyhedron [(N >= i+1),(N >= i'+1),(i' >= i+1),(i >= j+1),
  (i >= 1),(i' >= j'+1),(i' >= 1),(j >= 0),(j' >= 0)]

dependence polyhedron [(N >= i+1),(N >= i'+1),(i = i'),(i >= j+1),(i >= 1),
  (i' >= j'+1),(i' >= 1),(j' >= j+1),(j >= 0),(j' >= 0)]

table(mu = 0,mu_10 = 1/2,mu_11 = 0,mu_12 = 0,mu_13 = 1/2,mu_14 = 1,mu_15 = 0,
   mu_16 = 0,mu_17 = 0,mu_18 = 0,mu_5 = 0,mu_6 = 0,mu_7 = 0,mu_8 = 0,mu_9 = 0
   )

theta[S] = [1/2*(i * i)+j-1/2*i] == (j) + 1/2 . (i-1)*(i-1) + 1/2 . (i-1)

delay [T0] = 1/2*i+1/2*(i' * i')+j'-1/2*(i * i)-1/2*i'-j-1
=== (j') + 1/2 . (i'-i-1)*(i'-1) + 1/2 . (i'-i-1)*(i-1) + (i-j-1) + (i'-i-1)
```

Related Work

- Early work by B. Pugh et. al. using uninterpreted functions, and by van Engelen et. al. using interval analysis
- Polynomial minimization using a Bernstein expansion, implemented in ISL, can be applied to dependence testing
- Work in progress by A. Maréchal and M. Périn (Verimag) on linearization (i.e. getting rid of polynomials) using Handelman theorem and an oracle to control complexity.

Conclusion and Future Work

- The method works well and give interesting results in acceptable time, at least for small problems
- Other applications: transitive closure, program termination, (perhaps) invariant construction, ressource allocation, ...
- Complexity, very high, exponential in the degree of Schweighofer products
- Can one use an oracle to guess which products are useful?

THE END – QUESTIONS