Toward a Polynomial Model
with Application to the OpenStream Language

Paul Feautrier
with Albert Cohen and Alain Darte

June 13, 2016

Université Claude Bernard Lyon 1

Models in General

- A model is a mathematical (or computational) object that emulate a real world object (syntax).
- The "natural operations" of the model must emulate the behaviour of the real world object (semantics).
- An example: Newton's laws of motion:

$$F = m\gamma, \qquad F = G\frac{m.m'}{r^2}$$

  A system of ordinary differential equations. Semantics:
  - Proof of the existence of solutions
  - Computing the solution, either in closed form or numerically
  - If properly initialized the solution matches the trajectory of a rocket in the solar system.

NB. Modern programs and architectures are so complex they can be considered as real world objects.

Models of Programs

To reason about the behaviour of a program, one needs a notation for:

- its set of operations (*instances*, not statements)
- the execution order (a.k.a. the *Happens Before Relation* HB)
- a mapping from operations to memory *cells*

These sets are enormous: a 1 Gflops processor (big deal!) running for 1 second generates 10^9 operations.
The only possibility is to take advantage of regularities and represent these sets by symbolic constraints.
The HB relation can in some cases be represented by a *schedule*.
Program optimization or parallelization = changing the schedule.

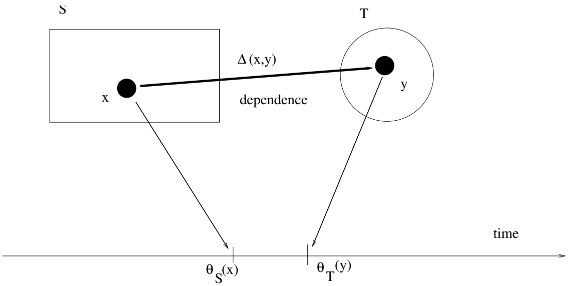
Requirements

A model must be faithfull and manageable.
The necessary operations must have efficient implementations:

- emptyness test
- intersection, union, complement
- projection, image
- optimization

Beware: do not confuse "efficient implementation" with decidability.

The Instancewise Approach



Constraints

- The schedules must be positive in the respective domains:

$$x \in S \Rightarrow \theta_S(x) \geq 0,$$

$$y \in T \Rightarrow \theta_T(y) \geq 0,$$

- The *delay* must be positive in the dependence relation:

$$\Delta(x,y) \Rightarrow \theta_T(y) - \theta_S(x) \geq \tau.$$

Problems

All such problems have the same form:

- Given a set S defined by constraints:

$$S = \{x | g_1(x) \geq 0, \cdots, g_n(x) \geq 0\},$$

- characterize all function f of a given *shape* such that $x \in S \Rightarrow f(x) \geq 0$.

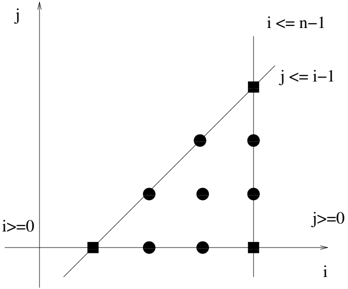
All solutions are in the form of a *certificate of positivity*

$$f(x) = \sum_k^N \lambda_k h_k(x), \lambda_k \geq 0,$$

where the $h_k(x) \geq 0$ are obvious consequences of the g_i constraints.

The polyhedral Model

Sets are represented by Z-polyhedra: the set of integral solutions of affine inequalities.
These sets are associated to *regular* loop nests.




```
for(i=0;i<n;i++)
  for(j=0;j<i;j++) ...
```

Tools

- Farkas lemma: construct an affine formula positive inside a polyhedron
- Linear programming : to solve the resulting constraints, emptiness test
- Fourier Motzkin elimination algorithm : emptiness test, projection

Linear programming has very efficient implementations: glpk, CPLEX, gurobi, and parametric extensions (PIP).

Evaluation

Most (large) programs do not fit into the polyhedral model. Some approaches:

- Extract small polyhedral kernels (*SCOPs*), optimize independently and plug the results back into the original program. A SCOP running time must represents a significant portion of the total running time (Amdhal law).
- Approximate: construct a polyhedral program with more operations, more dependences and more memory than the original. Optimizations valid for the approximations are valid for the original but the approximation may have no parallelism.
- Invent other models.

Other Models

The Tree model

- Represent sets by formal languages
- Regular languages for flat programs
- Context free languages for (recursive) procedures
- Many questions are undecidable by reduction to Post correspondance problem.

The Polynomial Model

- Represent sets by semi-algebraic sets.
- Problem 1: no projection algorithm in integers
- Problem 2: Hilbert 10th problem.

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, I

A semi-algebraic set (sas):

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

where x is a set of unknowns $x_1, \dots, x_p$ and the p_is are polynomials in x. A polyhedron is an sas such that all the p_is are of first degree. One usually include the trivial $1 \geq 0$ among the p_is.

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

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

The quantity $N = \sum_{i=1}^{n} e_i$ is the *order* of the product, not to be confused with its degree.

Notation, II

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 and only if it can be represented as a Schweighofer sum for some finite $Z \in \mathbb{N}^n$.

Comparisons

Notice the similarity between the *conclusion* of the two theorems, and the difference with Farkas lemma: since there is no useful bound on the size of Z, it is usually impossible to obtain a negative answer.
Another difference: those two theorems deals with *strictly* positive inequalities, while Farkas deals with non-strict inequalities.

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 order.
- 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.

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 $\theta_R : D_R \rightarrow \mathbb{N}$ 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.

An Example

Find a schedule for:

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

- This program has complexity $O(N^2)$, hence cannot have an affine schedule.
- One can find a bidimensional schedule, which can be converted into a quadratic schedule by counting.
- Can one find the quadratic schedule directly?

Answer: YES

```
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)
```

What is the use(s) of a schedule?

- Detecting programming errors (e.g. when an infinity of tasks are to be executed at the same time), or when the program needs unbounded memory.
- Proving the absence of deadlock.
- Parallel code generation, solved for polyhedra (CLooG, Cédric Bastoul), work in progress for polynomials.

But:

- Sequential programs do not have deadlocks. Applies only to parallel programs (e.g. OpenStream) or process networks (e.g. KPN) with infinite loops.
- Deadlocks are caused by cycles in the stream structure.

Introduction to the OpenStream Language

```
#pragma omp task output (x) // Task T1
x = ...;
for (i = 0; i < N; ++i) {
  int window_a[2], window_b[3];

  #pragma omp task output (x << window_a[2]) // Task T2
  window_a[0] = ...; window_a[1] = ...;
  if (i % 2) {
    #pragma omp task input (x >> window_b[2]) // Task T3
    use (window_b[0], window_b[1]);
  }
  #pragma omp task input (x) // Task T4
  use (x);
}
```

(Pop, Cohen, 2011)

- Sequential control program for task activations.
- Reservation for reads/writes in streams with burst and horizon.
- Single assignment in streams (by construction) + dataflow semantics.

- Instance of Pop-Cohen CDDF (Control-Driven Data Flow).
- Optimization in Erbium runtime system explored by Pop & Miranda.
- Unlike KPN, streams with multiple inputs/outputs (but deterministic).

Some properties of polyhedral OpenStream

- The order of task activations is $\prec$, the sequential order of the control program.
- Each stream s has a write index, I_s and a read index, J_s, which are functions of the iteration vector of the task instance.
- At each task *activation*, the relevant indices are incremented by the corresponding burst.

$$I_s(t, i) = \sum_{\text{t}'\text{writes s}} \sum_{<t', i'> \prec <t, i>} \mathrm{burst}(t', i')$$

- In the polyedral case, can be evaluated by tools for counting integer point inside a polyhedron.
- The degree of the result is the loop depth of the task activation.
- Dependence analysis amount to comparing indices and needs polynomial tools.

Deadlocks

Deadlock characterization iff there is, in the unrolled dependence graph,

- a statement T and an infinite sequence $(i_j)_{j \in N}$ such that $T(i_j)$ depends on $T(i_{j+1})$ and i_j is before i_{j+1} in the activation program;
- a cycle if the program is bounded or follows Kahnian semantics.

Consequence: There is no deadlock iff there is a schedule.

Scheduling OpenStream

- Compute the read and write indices using (for instance) the ISL library.
- Construct a polynomial template for each task.
- Construct a scheduling problem for each pair of tasks, one writing and the other reading the same stream.
- Use algorithm H to convert this problem into a system of affine constraints and solve.
- If this system is unfeasible, either the program has a deadlock or the order is not large enough.

One can use additional constraints for bounding the streams or construct *bounded delay schedules*.

Conclusion and Future Work, I

- 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 order of Schweighofer products. However, well within the capability of glpk or CPLEX.
- Can one use an oracle to guess which products are useful?

Conclusion and Future Work, II

We are still far from a polynomial model.
Other polynomial tools: CAD, Berntein, combine?
Very preliminary implementation using glpk and Z3.

Related Work

- Dependence Analysis
  - 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.
- Armin Größlinger: using Cylindrical Algebraic Decomposition for code generation.
- 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.

THE END – QUESTIONS