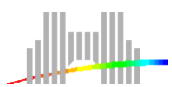
Polynomials, Season Two

Paul Feautrier
with Albert Cohen and Alain Darte

October 4, 2016

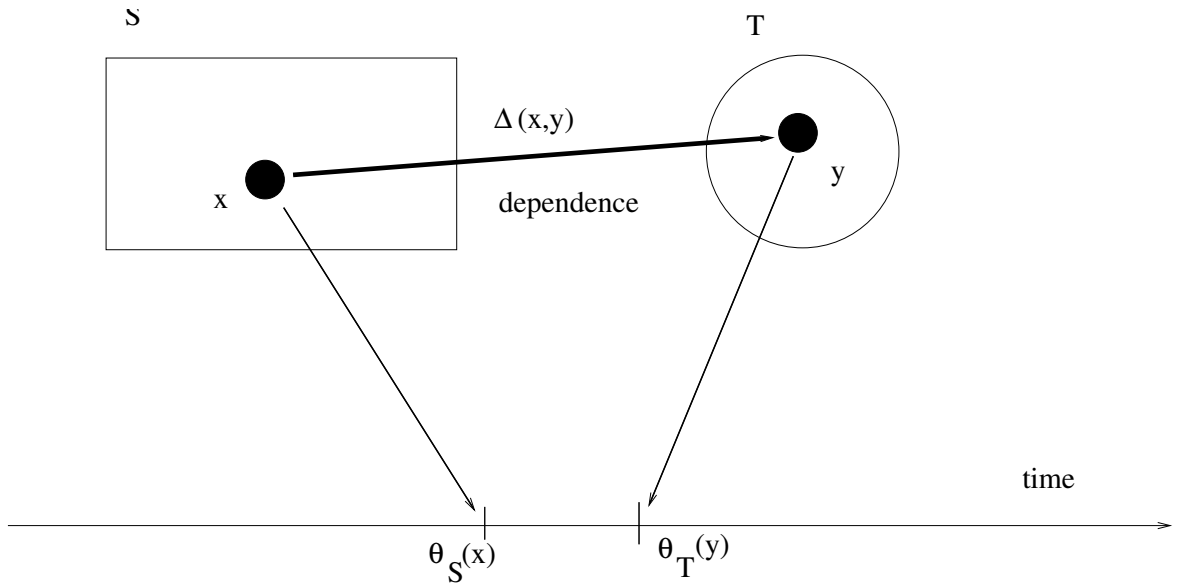


Introduction

A follow-up on my previous talk.
What's new?

- A complete implementation in Java, using ISCC (the command line version of the Integer Set Library) and glpk (a CPLEX like linear solver).
- Needed the solution of several minor problems.
- Extension to the OpenStream language: array of streams.

Scheduling



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.

Extensions

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

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

- Extension II: K is a semi-algebraic set and f is a polynomial.

Notations, I

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. One usually include the trivial $1 \geq 0$ among the p_is.

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

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

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 $Z \in \mathbb{N}^n$.

Application: Algorithm H

- Equate the unknown function to a Schweighofer sum with unknown coefficients.
- Expand in the monomial basis.
- Result: a system of linear equation to be solved in positive unknowns.

Warnings

- Since there is no useful bound on the size of Z, it is usually impossible to obtain a negative answer.
- Both theorems deals with *strictly* positive polynomials, while Farkas deals with non-strict afine forms.
- The reason: Schweighofer sum are never zero *inside* an sas unless they are identically zero.
- There is no extension to integer sets.

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

What is the use 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 (Armin Größlinger);

But:
Sequential programs do not have deadlocks. Applies only to parallel programs (e.g. OpenStream) or process networks (e.g. KPN) with infinite loops.

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.

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

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

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, or its streams are not bounded (see later).

PseudOpen, how and why

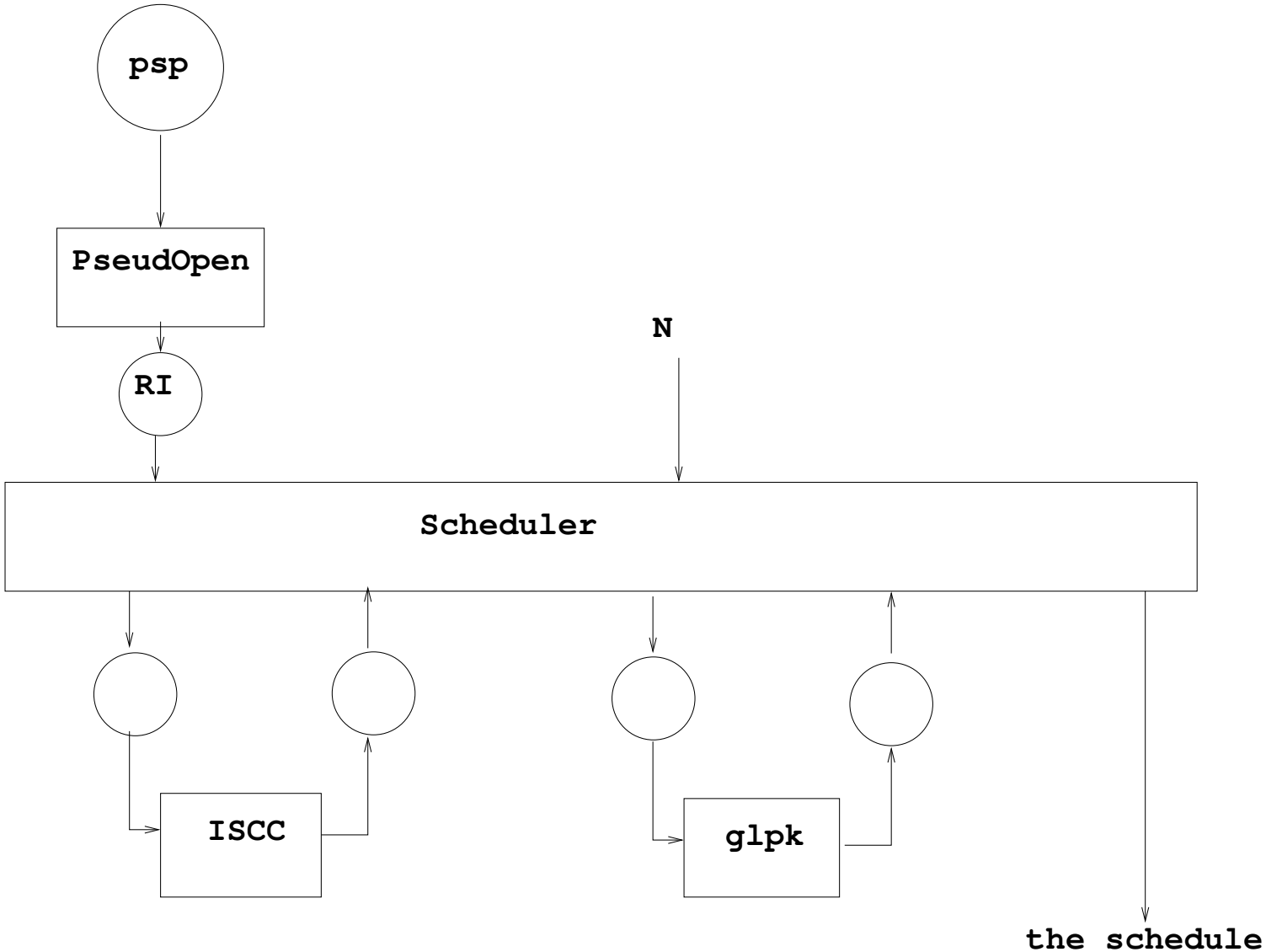
- The needed information for OpenStream scheduling should have been generated by the standard OpenStream compiler. But ..
- ... this compiler has been developped from gcc, which is *not* polyhedral friendly.
- Hence the use of a pseudocode (introduced by Alain Darte in IMPACT16):

A delay line

Notice the use of a stream array.

```
parameter N;
stream s;
task starter {write once in s[0];
              }
for(i=0;i<N;i++)
  task a {read once from s[i];
          write once in s[i+1];
         }
```

The Scheduler



The objective function

A linear solver needs an objective function.

- Minimizing the latency is not suitable for infinite streaming programs.
- Minimize a delay bound:

$$\Delta(x, y) \Rightarrow \theta_T(y) - \theta_S(x) \leq B$$

- Minimize B, which must be a polynomial in the program parameters.

Bounding the memory

A realistic program must run in bounded memory, i.e. must have bounded parallelism and bounded streams.

- First solution: for each stream, assume a bounded buffer of size L used in round robin mode, and express that a write at position I destroys an item at position $I - L$.
- To stay linear, L must be a number.
- Second solution: *Miracle theorem, Alain Darte* A program with bounded delays and bounded parallelism has bounded buffers, whose size may be parametric.

Minor problems

- The cardinal of the empty set is 0, but 0 is never written by ISCC.
- Solution: insure that all sets are non empty, by adding a dummy element.
- glpk uses floating point arithmetics, and may generate numbers with large denominators, which cause overflows in subsequent rational calculations.
- Solution I: replace rational arithmetics by floating point arithmetics (dirty).
- Solution II: Scale-up the schedule by inserting stream propagation delays.

Conclusion and Future Work, II

We are still far from a polynomial model.
Other polynomial tools: CAD, Bernstein, combine?
Very preliminary implementation using glpk and ISL.
Hunting for interesting examples (systolic arrays?).

THE END – QUESTIONS

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).