« Program termination proofs by convex optimization »

Patrick Cousot
École normale supérieure
45 rue d'Ulm, 75230 Paris cedex 05, France

Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

IBM Research Seminar — Thomas J. Watson Research Center, Hawthorne, NY — 5 January 2006

Abstract

Program termination is based on reasonings by induction (e.g. on program steps, program data) which involves the discovery of unknown inductive arguments (e.g. rank functions, invariants) satisfying universally quantified termination conditions. For static program analysis, the discovery of the inductive arguments must be automated, which consists in solving the constraints provided by the termination conditions. Several methods have been considered: recurrence/difference equation resolution; iteration, possibly with convergence acceleration through widening/narrowing; or direct methods (such as elimination). All these methods involve some form of simplification of the constraints formalized by abstract interpretation. In this talk, we explore parametric abstraction of rank function and invariants and direct resolution of Floyd/Naur/Hoare termination constraints by Lagrangian relaxation (to handle implication) and semidefinite programming relaxation (to handle universal implication). Finally the parameters are computed using numerical semidefinite programming solvers. This new approach exploits the recent progress in the numerical resolution of linear or bilinear matrix inequalities by semidefinite programming using efficient polynomial primal/dual interior point methods generalizing those well-known in linear programming to convex optimization. The framework is applied to invariance and termination proof of sequential, nondeterministic, concurrent, and fair parallel imperative polynomial programs and can easily be extended to other safety and liveness properties.

Reference

[1] P. Cousot. – Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. *In: Proc. Sixth Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI 2005)*, R. Cousot (Ed.), Paris, France, 17–19 Jan. 2005. pp. 1–24. – Lecture Notes In Computer Science 3385, Springer.

Static analysis

Principle of static analysis

– Define the most precise program property as a fixpoint $\mathsf{lfp}\ F$

– Effectively compute a fixpoint approximation:

  - iteration-based fixpoint approximation
  - constraint-based fixpoint approximation

Iteration-based static analysis

– Effectively overapproximate the iterative fixpoint definition [1]:

$$\mathsf{lfp}\ F = \bigsqcup_{\lambda \in \mathbb{O}} X^{\lambda}$$

$$X^0 = \bot$$
$$X^{\lambda} = \bigsqcup_{\eta < \lambda} F(X^{\eta})$$

[1] under Tarski's fixpoint theorem hypotheses

Constraint-based static analysis

– Effectively solve a postfixpoint constraint:

$$\mathsf{lfp}\ F = \bigsqcap \{X \mid F(X) \sqsubseteq X\}$$

since $F(X) \sqsubseteq X$ implies $\mathsf{lfp}\ F \sqsubseteq X$

– Sometimes, the constraint resolution algorithm is nothing but the iterative computation of $\mathsf{lfp}\ F$ [2]

– Constraint-based static analysis is the main subject of this talk.

[2] An example is *set-based analysis* as shown in Patrick Cousot & Radhia Cousot. *Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation.* In *Conference Record of FPCA '95 ACM Conference on Functional Programming and Computer Architecture*, pages 170–181, La Jolla, California, U.S.A., 25-28 June 1995.

Parametric abstraction

– Parametric abstract domain: $X \in \{f(a) \mid a \in \Delta\}$, a is an unknown parameter

– Verification condition: X satisfies $F(X) \sqsubseteq X$ if [and only if] $\exists a \in \Delta : F(f(a)) \sqsubseteq f(a)$ that is $\exists a : C_F(a)$ where $C_F \in \Delta \mapsto \mathbb{B}$ are constraints over the unknown parameter a

Fixpoint versus Constraint-based Approach for Termination Analysis

1. Termination can be expressed in fixpoint form [3]
2. However we know no effective fixpoint underapproximation method needed to overestimation the termination rank
3. So we consider a constraint-based approach abstracting Floyd's ranking function method

[3] See Sect. 11.2 of Patrick Cousot. *Constructive Design of a hierarchy of Semantics of a Transition System by Abstract Interpretation. Theoret. Comput. Sci.* 277(1—2):47—103, 2002. © Elsevier Science.

Overview of the Termination Analysis Method

Proving Termination of a Loop

```
while B do
   C
```

1. Termination precondition
2. Loop invariant
3. Loop operational semantics
4. Ranking function

The main point in this talk is (4).

Proving Termination of a Loop

1. Perform an iterated forward/backward relational static analysis of the loop with *termination hypothesis* to determine a *necessary* proper termination precondition
2. Assuming the *termination precondition*, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd's ranking function method to prove termination of the loop

Arithmetic Mean Example

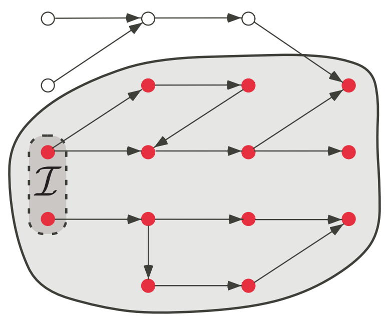
```
while (x <> y) do
    x := x - 1;
    y := y + 1
od
```

The polyhedral abstraction used for the static analysis of the examples is implemented using Bertrand Jeannet's NewPolka library.

Arithmetic Mean Example

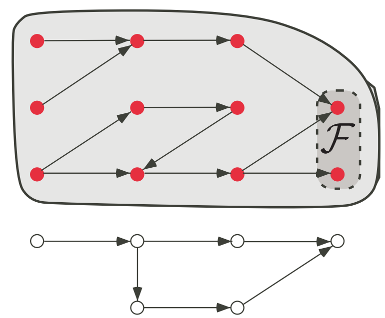
1. Perform an iterated forward/backward relational static analysis of the loop with *termination hypothesis* to determine a *necessary* proper termination precondition
2. Assuming the *termination precondition*, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd's ranking function method to prove termination of the loop

Forward/reachability properties



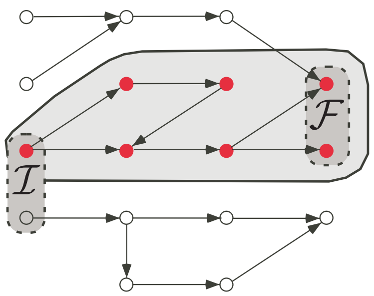
Example: partial correctness (must stay into safe states)

Backward/ancestry properties



Example: termination (must reach final states)

Forward/backward properties



Example: total correctness (stay safe while reaching final states)

Principle of the iterated forward/backward iteration-based approximate analysis

– Overapproximate

$$\mathsf{lfp}\, F \sqcap \mathsf{lfp}\, B$$

by overapproximations of the decreasing sequence

$$\begin{aligned} X^0 &= \top \\ &\ldots \\ X^{2n+1} &= \mathsf{lfp}\, \lambda Y \cdot X^{2n} \sqcap F(Y) \\ X^{2n+2} &= \mathsf{lfp}\, \lambda Y \cdot X^{2n+1} \sqcap B(Y) \\ &\ldots \end{aligned}$$

Arithmetic Mean Example: Termination Precondition (1)

```
{x>=y}
  while (x <> y) do
    {x>=y+2}
      x := x - 1;
    {x>=y+1}
      y := y + 1
    {x>=y}
  od
{x=y}
```


Idea 1

The auxiliary termination counter method

Arithmetic Mean Example: Termination Precondition (2)

```
{x=y+2k,x>=y}
  while (x <> y) do
    {x=y+2k,x>=y+2}
      k := k - 1;
    {x=y+2k+2,x>=y+2}
      x := x - 1;
    {x=y+2k+1,x>=y+1}
      y := y + 1
    {x=y+2k,x>=y}
  od
{x=y,k=0}
  assume (k = 0)
{x=y,k=0}
```

Add an auxiliary termination counter to enforce (bounded) termination in the backward analysis!

Arithmetic Mean Example

1. Perform an iterated forward/backward relational static analysis of the loop with *termination hypothesis* to determine a *necessary* proper termination precondition
2. Assuming the *termination precondition*, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd's ranking function method to prove termination of the loop

Arithmetic Mean Example: Loop Invariant

```
  assume ((x=y+2*k) & (x>=y));
{x=y+2k,x>=y}
  while (x <> y) do
    {x=y+2k,x>=y+2}
      k := k - 1;
    {x=y+2k+2,x>=y+2}
      x := x - 1;
    {x=y+2k+1,x>=y+1}
      y := y + 1
    {x=y+2k,x>=y}
  od
{k=0,x=y}
```


Arithmetic Mean Example

1. Perform an iterated forward/backward relational static analysis of the loop with *termination hypothesis* to determine a *necessary* proper termination precondition
2. Assuming the *termination precondition*, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd's ranking function method to prove termination of the loop

Arithmetic Mean Example: Body Relational Semantics

Case x < y:

```
assume (x=y+2*k)&(x>=y+2);
{x=y+2k,x>=y+2}
assume (x < y);
empty(6)
assume (x0=x)&(y0=y)&(k0=k);
empty(6)
k := k - 1;
x := x - 1;
y := y + 1
empty(6)
```

Case x > y:

```
assume (x=y+2*k)&(x>=y+2);
{x=y+2k,x>=y+2}
assume (x > y);
{x=y+2k,x>=y+2}
assume (x0=x)&(y0=y)&(k0=k);
{x=y+2k0,y=y0,x=x0,x=y+2k,
                    x>=y+2}
k := k - 1;
x := x - 1;
y := y + 1
{x+2=y+2k0,y=y0+1,x+1=x0,
              x=y+2k,x>=y}
```


Arithmetic Mean Example

1. Perform an iterated forward/backward relational static analysis of the loop with *termination hypothesis* to determine a *necessary* proper termination precondition
2. Assuming the *termination precondition*, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd's ranking function method to prove termination of the loop

Floyd's method for termination of `while B do C`

Given a loop invariant I, find an $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown rank function r such that:

– The rank is *nonnegative*:

$$\forall\, x_0, x : I(x_0) \wedge [\![\mathtt{B};\mathtt{C}]\!](x_0, x) \Rightarrow r(x_0) \geq 0$$

– The rank is *strictly decreasing*:

$$\forall\, x_0, x : I(x_0) \wedge [\![\mathtt{B};\mathtt{C}]\!](x_0, x) \Rightarrow r(x) \leq r(x_0) - \eta$$

$\eta \geq 1$ for $\mathbb{Z}$, $\eta > 0$ for $\mathbb{R}/\mathbb{Q}$ to avoid Zeno $\frac{1}{2}, \frac{1}{4}, \frac{1}{8} \ldots$

Problems

– How to get rid of the implication $\Rightarrow$?

  $\longrightarrow$ Lagrangian relaxation

– How to get rid of the universal quantification $\forall$?

  $\longrightarrow$ Quantifier elimination/mathematical programming & relaxation

Algorithmically interesting cases

- linear inequalities
  - → linear programming
- linear matrix inequalities (LMI)/quadratic forms
  - → semidefinite programming
- semialgebraic sets
  - → polynomial quantifier elimination, or
  - → relaxation with semidefinite programming

Arithmetic Mean Example: Ranking Function with Semidefinite Programming Relaxation

```
» clear all;
[v0,v] = variables('x','y','k')
% linear inequalities
%       x0 y0 k0
Ai =  [  0  0  0];
%        x  y  k
Ai_ = [  1 -1  0]; % x0 - y0 >= 0
bi = [0];
[N Mk(:,:,:)]=linToMk(Ai,Ai_,bi);
% linear equalities
%       x0 y0 k0
Ae =  [  0  0 -2;
         0 -1  0;
        -1  0  0;
         0  0  0];
%        x  y  k
Ae_ = [  1 -1  0;  % x - y - 2*k0 - 2 = 0
         0  1  0;  % y - y0 - 1 = 0
         1  0  0;  % x - x0 + 1 = 0
         1 -1 -2]; % x - y - 2*k = 0
be = [2; -1; 1; 0];
[M Mk(:,:,N+1:N+M)]=linToMk(Ae,Ae_,be);
```

Input the loop abstract semantics


```
» display_Mk(Mk, N, v0, v);
...
 +1.x -1.y >= 0
 -2.k0 +1.x -1.y +2 = 0
 -1.y0 +1.y -1 = 0
 -1.x0 +1.x +1 = 0
 +1.x -1.y -2.k = 0
...
```

- Display the abstract semantics of the loop `while B do C`
- compute ranking function, if any

```
» [diagnostic,R] = termination(v0, v, Mk, N, 'integer', 'linear');
» disp(diagnostic)
   feasible (bnb)
» intrank(R, v)

r(x,y,k) = +4.k -2
```


Quantifier Elimination

Quantifier elimination (Tarski-Seidenberg)

– quantifier elimination for the first-order theory of real closed fields:
  - F is a logical combination of polynomial equations and inequalities in the variables $x_1, \ldots, x_n$
  - Tarski-Seidenberg decision procedure

    transforms a formula

$$\forall/\exists x_1 : \ldots \forall/\exists x_n : F(x_1, \ldots, x_n)$$

    into an equivalent quantifier free formula

– cannot be bound by any tower of exponentials [Heintz, Roy, Solerno 89]

Quantifier elimination (Collins)

– cylindrical algebraic decomposition method by Collins

– implemented in MATHEMATICA®

– worst-case time-complexity for real quantifier elimination is “only” doubly exponential in the number of quantifier blocks

– Various optimisations and heuristics can be used [4]

[4] See e.g. REDLOG http://www.fmi.uni-passau.de/~redlog/

Scaling up

However

– does not scale up beyond a few variables!

– too bad!

Proving Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming

Idea 2

> Express the loop invariant and relational semantics as numerical positivity constraints

Relational semantics of `while B do C od` loops

- $x_0 \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: values of the loop variables *before* a loop iteration
- $x \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: values of the loop variables *after* a loop iteration
- $I(x_0)$: loop invariant, $[\![\mathtt{B};\mathtt{C}]\!](x_0, x)$: relational semantics of *one iteration of the loop body*
- $I(x_0) \wedge [\![\mathtt{B};\mathtt{C}]\!](x_0, x) = \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \geqslant_i 0 \quad (\geqslant_i \in \{>, \geq, =\})$
- not a restriction for numerical programs

Example of linear program (Arithmetic mean)

$$[A\ A'][x_0\ x]^\top \geqslant b$$

```
{x=y+2k,x>=y}
while (x <> y) do
    k := k - 1;
    x := x - 1;
    y := y + 1
od
```

```
+1.x -1.y >= 0
-2.k0 +1.x -1.y +2 = 0
-1.y0 +1.y -1 = 0
-1.x0 +1.x +1 = 0
+1.x -1.y -2.k = 0
```

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -2 & 1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & -2 \end{array}\right] \begin{bmatrix} x_0 \\ y_0 \\ k_0 \\ x \\ y \\ k \end{bmatrix} \begin{array}{c} \geq \\ = \\ = \\ = \\ = \end{array} \begin{bmatrix} 0 \\ -2 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

Example of quadratic form program (factorial)

$$[x\ x']A[x\ x']^\top + 2[x\ x']\,q + r \geqslant 0$$

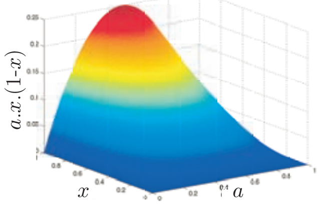
```
n := 0;
f := 1;
while (f <= N) do
    n := n + 1;
    f := n * f
od
```

```
-1.f0 +1.N0 >= 0
+1.n0 >= 0
+1.f0 -1 >= 0
-1.n0 +1.n -1 = 0
+1.N0 -1.N = 0
-1.f0.n +1.f = 0
```

$$[n_0 f_0 N_0 n\, f\, N] \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} n_0 \\ f_0 \\ N_0 \\ n \\ f \\ N \end{bmatrix} + 2[n_0 f_0 N_0 n\, f\, N] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix} + 0 = 0$$

Example of semialgebraic program (logistic map)

```
eps = 1.0e-9;
while (0 <= a) & (a <= 1 - eps)
       & (eps <= x) & (x <= 1) do
   x := a*x*(1-x)
od
```



Floyd's method for termination of `while B do C`

Find an $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown rank function r and $\eta > 0$ such that:

– The rank is *nonnegative*:

$$\forall\, x_0, x : \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \geqslant_i 0 \Rightarrow r(x_0) \geq 0$$

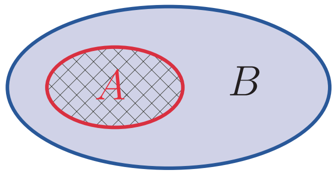
– The rank is *strictly decreasing*:

$$\forall\, x_0, x : \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \geqslant_i 0 \Rightarrow r(x_0) - r(x) - \eta \geq 0$$

Idea 3

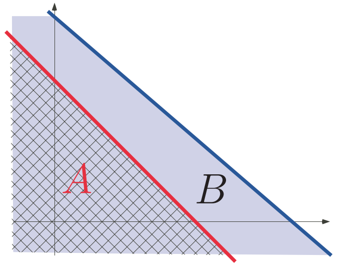
Eliminate the conjunction $\bigwedge$ and implication $\Rightarrow$ by Lagrangian relaxation

Implication (general case)



$$A \Rightarrow B$$
$$\Leftrightarrow$$
$$\forall x \in A : x \in B$$

Implication (linear case)



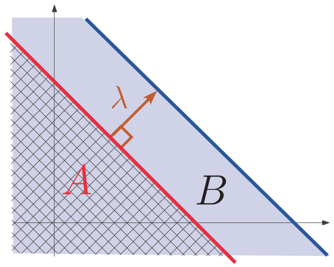
$A \Rightarrow B$ (assuming $A \neq \emptyset$)

$\Leftarrow$ (soundness)

$\Rightarrow$ (completeness)

border of A parallel to border of B

Lagrangian relaxation (linear case)



Lagrangian relaxation, formally

Let $\mathbb{V}$ be a finite dimensional linear vector space, $N > 0$ and $\forall k \in [0, N] : \sigma_k \in \mathbb{V} \mapsto \mathbb{R}$.

$$\forall x \in \mathbb{V} : \left(\bigwedge_{k=1}^{N} \sigma_k(x) \geq 0 \right) \Rightarrow (\sigma_0(x) \geq 0)$$

$\Leftarrow$ soundness (Lagrange)

$\Rightarrow$ completeness (*lossless*)

$\not\Rightarrow$ incompleteness (*lossy*)

$$\exists \lambda \in [1, N] \mapsto \mathbb{R}^+ : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^{N} \lambda_k \sigma_k(x) \geq 0$$

relaxation = approximation, λ_i = Lagrange coefficients

Lagrangian relaxation, equality constraints

$$\forall x \in \mathbb{V} : \left(\bigwedge_{k=1}^{N} \sigma_k(x) = 0 \right) \Rightarrow (\sigma_0(x) \geq 0)$$

$\Leftarrow$ soundness (Lagrange)

$$\exists \lambda \in [1, N] \mapsto \mathbb{R}^+ : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^{N} \lambda_k \sigma_k(x) \geq 0$$

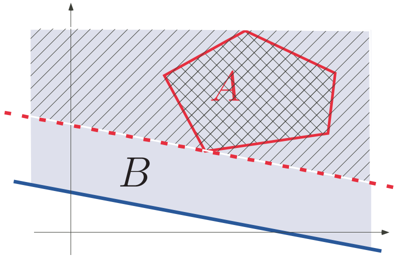
$$\wedge \ \exists \lambda' \in [1, N] \mapsto \mathbb{R}^+ : \forall x \in \mathbb{V} : \sigma_0(x) + \sum_{k=1}^{N} \lambda'_k \sigma_k(x) \geq 0$$

$$\Leftrightarrow \left(\lambda'' = \frac{\lambda' - \lambda}{2}\right)$$

$$\exists \lambda'' \in [1, N] \mapsto \mathbb{R} : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^{N} \lambda''_k \sigma_k(x) \geq 0$$

Example: affine Farkas' lemma, informally

– An application of Lagrangian relaxation to the case when A is a polyhedron



Example: affine Farkas' lemma, formally

– Formally, if the system $Ax + b \geq 0$ is feasible then

$$\forall x : Ax + b \geq 0 \Rightarrow cx + d \geq 0$$

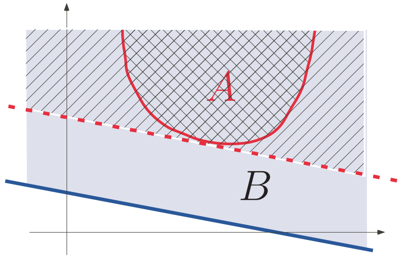
$\Leftarrow$ (soundness, Lagrange)

$\Rightarrow$ (completeness, Farkas)

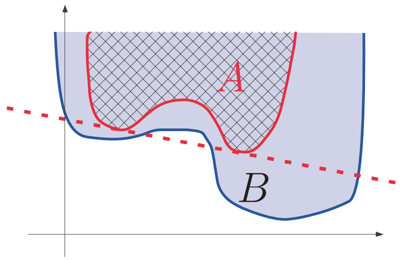
$$\exists \lambda \geq 0 : \forall x : cx + d - \lambda(Ax + b) \geq 0 \ .$$

Yakubovich's S-procedure, informally

– An application of Lagrangian relaxation to the case when A is a quadratic form



Incompleteness (convex case)



Yakubovich's S-procedure, completeness cases

- The constraint $\sigma(x) \geq 0$ is *regular* if and only if $\exists \xi \in \mathbb{V} : \sigma(\xi) > 0$.
- The S-procedure is lossless in the case of one regular quadratic constraint:

$$\forall x \in \mathbb{R}^n : x^\top P_1 x + 2q_1^\top x + r_1 \geq 0 \Rightarrow x^\top P_0 x + 2q_0^\top x + r_0 \geq 0$$

$\Leftarrow$ (Lagrange)
$\Rightarrow$ (Yakubovich)

$$\exists \lambda \geq 0 : \forall x \in \mathbb{R}^n : x^\top \left(\begin{bmatrix} P_0 & q_0 \\ q_0^\top & r_0 \end{bmatrix} - \lambda \begin{bmatrix} P_1 & q_1 \\ q_1^\top & r_1 \end{bmatrix} \right) x \geq 0.$$

Floyd's method for termination of `while B do C`

Find an $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown rank function r which is:

- *Nonnegative*: $\exists \lambda \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : r(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0$$

- *Strictly decreasing*: $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : (r(x_0) - r(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0$$

Idea 4

Parametric abstraction of the ranking function r

Parametric abstraction

- How can we compute the ranking function r?

$\rightarrow$ parametric abstraction:

1. Fix the form r_a of the function r a priori, in term of unkown parameters a
2. Compute the parameters a numerically

- Examples:

$r_a(x)$	$= a.x^\top$	linear
$r_a(x)$	$= a.(x\ 1)^\top$	affine
$r_a(x)$	$= (x\ 1).a.(x\ 1)^\top$	quadratic

Floyd's method for termination of `while B do C`

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown parameters a, such that:

– *Nonnegative*: $\exists \lambda \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : r_a(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0$$

– *Strictly decreasing*: $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : (r_a(x_0) - r_a(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0$$

Idea 5

Eliminate the universal quantification $\forall$ using linear matrix inequalities (LMIs)

Mathematical programming

$$\exists x \in \mathbb{R}^n : \quad \bigwedge_{i=1}^{N} g_i(x) \geqslant 0$$

$$[\text{Minimizing} \quad f(x)]$$

feasibility problem : find a solution to the constraints

optimization problem : find a solution, minimizing $f(x)$

Example: Linear programming

$$\exists x \in \mathbb{R}^n : \quad Ax \geqslant b$$

$$[\text{Minimizing} \quad cx]$$

Feasibility

– feasibility problem: find a solution $s \in \mathbb{R}^n$ to the optimization program, such that $\bigwedge_{i=1}^{N} g_i(s) \geq 0$, or to determine that the problem is *infeasible*

– feasible set: $\{x \mid \bigwedge_{i=1}^{N} g_i(x) \geq 0\}$

– a feasibility problem can be converted into the optimization program

$$\min\{-y \in \mathbb{R} \mid \bigwedge_{i=1}^{N} g_i(x) - y \geq 0\}$$

Semidefinite programming

$$\exists x \in \mathbb{R}^n: \qquad M(x) \succcurlyeq 0$$

$$[\text{Minimizing} \quad cx]$$

Where the linear matrix inequality (LMI) is

$$M(x) = M_0 + \sum_{k=1}^{n} x_k M_k$$

with symetric matrices ($M_k = M_k{}^\top$) and the positive semidefiniteness is

$$M(x) \succcurlyeq 0 = \forall X \in \mathbb{R}^N : X^\top M(x) X \geq 0$$

Semidefinite programming, once again

Feasibility is:

$$\exists x \in \mathbb{R}^n: \forall X \in \mathbb{R}^N : X^\top \left(M_0 + \sum_{k=1}^{n} x_k M_k \right) X \geq 0$$

of the form of the formulæ we are interested in for programs which semantics can be expressed as *LMIs*:

$$\bigwedge_{i=1}^{N} \sigma_i(x_0, x) \geqslant_i 0 = \bigwedge_{i=1}^{N} (x_0 \ x \ 1) M_i (x_0 \ x \ 1)^\top \geqslant_i 0$$

Floyd's method for termination of `while B do C`

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown parameters a, such that:

– *Nonnegative*: $\exists \lambda \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : r_a(x_0) - \sum_{i=1}^{N} \lambda_i (x_0 \ x \ 1) M_i (x_0 \ x \ 1)^\top \geq 0$$

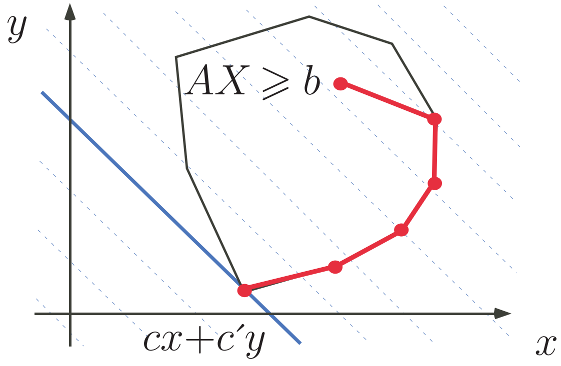
– *Strictly decreasing*: $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^{+i}$:

$$\forall\ x_0, x : (r_a(x_0) - r_a(x) - \eta) - \sum_{i=1}^{N} \lambda'_i (x_0 \ x \ 1) M_i (x_0 \ x \ 1)^\top \geq 0$$

Idea 6

Solve the convex constraints by semidefinite programming

The simplex for linear programming



Dantzig 1948, exponential in worst case, good in practice

Polynomial Methods for Linear Porgramming

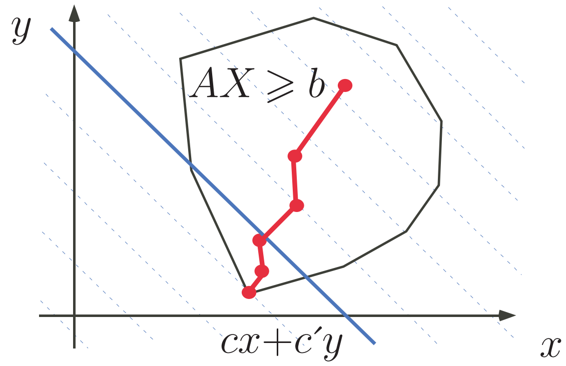
Ellipsoid method :

- Shor 1970 and Yudin & Nemirovskii 1975,
- polynomial in worst case Khachian 1979,
- but not good in practice

Interior point method :

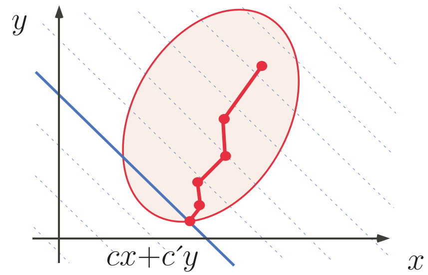
- Kamarkar 1984,
- polynomial for both average and worst case, and
- good in practice (hundreds of thousands of variables)

The interior point method



Interior point method for semidefinite programming

- Nesterov & Nemirovskii 1988, good in practice (thousands of variables)



- Various path strategies e.g. “stay in the middle”

Semidefinite programming solvers

Numerous solvers available under MATHLAB®, a.o.:

- `lmilab`: P. Gahinet, A. Nemirovskii, A.J. Laub, M. Chilali
- `Sdplr`: S. Burer, R. Monteiro, C. Choi
- `Sdpt3`: R. Tütüncü, K. Toh, M. Todd
- `SeDuMi`: J. Sturm
- `bnb`: J. Löfberg (integer semidefinite programming)

Common interfaces to these solvers, a.o.:

- `Yalmip`: J. Löfberg

Sometime need some help (feasibility radius, shift,...)

Linear program: termination of Euclidean division

```
» clear all
% linear inequalities
%       y0 q0 r0
Ai = [ 0  0  0; 0  0  0;
       0  0  0];
%        y  q  r
Ai_ = [ 1  0  0;  % y - 1 >= 0
        0  1  0;  % q - 1 >= 0
        0  0  1]; % r >= 0
bi = [-1; -1; 0];
% linear equalities
%       y0 q0 r0
Ae =  [ 0 -1  0;  % -q0 + q -1 = 0
       -1  0  0;  % -y0 + y = 0
        0  0 -1]; % -r0 + y + r = 0
%        y  q  r
Ae_ = [ 0  1  0; 1  0  0;
        1  0  1];
be = [-1; 0; 0];
```

Iterated forward/backward polyhedral analysis:

```
{y>=1}
q := 0;
{q=0,y>=1}
r := x;
{x=r,q=0,y>=1}
while (y <= r) do
   {y<=r,q>=0}
   r := r - y;
   {r>=0,q>=0}
   q := q + 1
   {r>=0,q>=1}
od
{q>=0,y>=r+1}
```

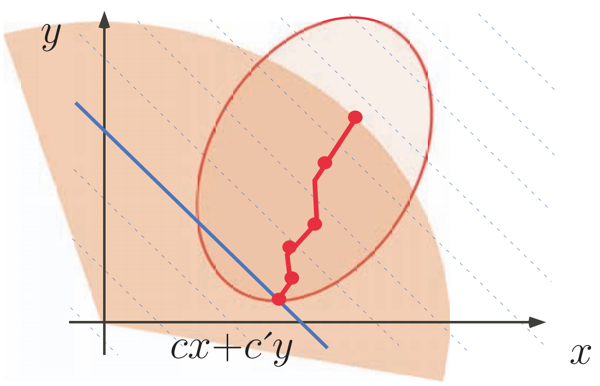


```
» [N Mk(:,:,:)]=linToMk(Ai, Ai_, bi);
» [M Mk(:,:,N+1:N+M)]=linToMk(Ae, Ae_, be);
» [v0,v]=variables('y','q','r');
» display_Mk(Mk, N, v0, v);
 +1.y -1 >= 0
 +1.q -1 >= 0
 +1.r >= 0
 -1.q0 +1.q -1 = 0
 -1.y0 +1.y = 0
 -1.r0 +1.y +1.r = 0
» [diagnostic,R] = termination(v0, v, Mk, N, 'integer', 'quadratic');
» disp(diagnostic)
   termination (bnb)
» intrank(R, v)
```

`r(y,q,r) = -2.y +2.q +6.r`

Floyd's proposal $r(x, y, q, r) = x - q$ is more intuitive but requires to discover the nonlinear loop invariant $x = r + qy$.

Imposing a feasibility radius



Quadratic program: termination of factorial

Program:

```
n := 0;
f := 1;
while (f <= N) do
    n := n + 1;
    f := n * f
od
```

LMI semantics:

```
-1.f0 +1.N0 >= 0
+1.n0 >= 0
+1.f0 -1 >= 0
-1.n0 +1.n -1 = 0
+1.N0 -1.N = 0
-1.f0.n +1.f = 0
```

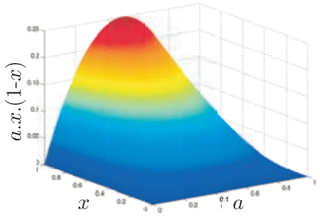
```
r(n,f,N) = -9.993455e-01.n +4.346533e-04.f
           +2.689218e+02.N +8.744670e+02
```


Idea 7

Convex abstraction of non-convex constraints

Semidefinite programming relaxation for polynomial programs

```
eps = 1.0e-9;
while (0 <= a) & (a <= 1 - eps)
      & (eps <= x) & (x <= 1) do
   x := a*x*(1-x)
od
```



Write the verification conditions in polynomial form, use SOS solver to relax in semidefinite programming form.

SOStool+SeDuMi:

```
r(x) = 1.222356e-13.x + 1.406392e+00
```


Considering More General Forms of Programs

Handling disjunctive loop tests and tests in loop body

– By case analysis
– and "conditional Lagrangian relaxation" (Lagrangian relaxation in each of the cases)

Loop body with tests

```
while (x < y) do
  if (i >= 0) then
     x := x+i+1
  else
     y := y+i
  fi
od
```

$\longrightarrow$ case analysis: $\begin{cases} \mathtt{i} \geq 0 \\ \mathtt{i} < 0 \end{cases}$

lmilab:

```
r(i,x,y) = -2.252791e-09.i -4.355697e+07.x +4.355697e+07.y
                                            +5.502903e+08
```


Quadratic termination of linear loop

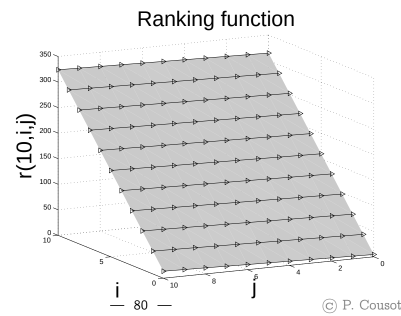
```
{n>=0}
i := n; j := n;
while (i <> 0) do
  if (j > 0) then
    j := j - 1
  else
    j := n; i := i - 1
  fi
od
```

$\longleftarrow$ termination precondition determined by iterated forward/backward polyhedral analysis

sdplr (with feasibility radius of 1.0e+3):

```
r(n,i,j) = +7.024176e-04.n^2 +4.394909e-05.n.i ...
           -2.809222e-03.n.j +1.533829e-02.n ...
           +1.569773e-03.i^2 +7.077127e-05.i.j   ...
           +3.093629e+01.i -7.021870e-04.j^2 ...
           +9.940151e-01.j +4.237694e+00
```

Successive values of r(n,i,j) for n = 10 on loop entry



Handling nested loops

– by induction on the loop depth
– use an iterated forward/backward symbolic analysis to get a necessary termination precondition
– use a forward symbolic symbolic analysis to get the semantics of a loop body
– use Lagrangian relaxation and semidefinite programming to get the ranking function

Example of termination of nested loops: Bubblesort inner loop

```
...
+1.i' -1 >= 0
+1.j' -1 >= 0
+1.n0' -1.i' >= 0
-1.j +1.j' -1 = 0
-1.i +1.i' = 0
-1.n +1.n0' = 0
+1.n0 -1.n0' = 0
+1.n0' -1.n' = 0
...
```

Iterated forward/backward polyhedral analysis followed by forward analysis of the body:

```
assume (n0 = n & j >= 0 & i >= 1 & n0 >= i & j <> i);
{n0=n,i>=1,j>=0,n0>=i}
assume (n01 = n0 & n1 = n & i1 = i & j1 = j);
{j=j1,i=i1,n0=n1,n0=n01,n0=n,i>=1,j>=0,n0>=i}
j := j + 1
{j=j1+1,i=i1,n0=n1,n0=n01,n0=n,i>=1,j>=1,n0>=i}
```

```
termination (lmilab)
r(n0,n,i,j) = +434297566.n0 +226687644.n -72551842.i
                                          -2.j +2147483647
```


Example of termination of nested loops: Bubblesort outer loop

```
...
+1.i' +1 >= 0
+1.n0' -1.i' -1 >= 0
+1.i' -1.j' +1 = 0
-1.i +1.i' +1 = 0
-1.n +1.n0' = 0
+1.n0 -1.n0' = 0
+1.n0' -1.n' = 0
...
```

Iterated forward/backward polyhedral analysis followed by forward analysis of the body:

```
  assume (n0=n & i>=0 & n>=i & i <> 0);
{n0=n,i>=0,n0>=i}
  assume (n01=n0 & n1=n & i1=i & j1=j);
{j1=j,i=i1,n0=n1,n0=n01,n0=n,i>=0,n0>=i}
  j := 0;
  while (j <> i) do
      j := j + 1
  od;
  i := i - 1
{i+1=j,i+1=i1,n0=n1,n0=n01,n0=n,i+1>=0,n0>=i+1}
```

```
termination (lmilab)
r(n0,n,i,j) = +24348786.n0 +16834142.n +100314562.i +65646865
```


Handling nondeterminacy

– By case analysis
– Same for concurrency by interleaving
– Same with fairness by nondeterministic interleaving with encoding of an explicit bounded round-robin scheduler (with unknown bound)

Termination of a concurrent program

```
[|  1: while [x+2 < y] do
    2:     [x := x + 1]
       od
    3:
||
    1: while [x+2 < y] do
    2:     [y := y - 1]
       od
    3:
|]
```

interleaving $\longrightarrow$

```
while (x+2 < y) do
   if ?=0 then
     x := x + 1
   else if ?=0 then
     y := y - 1
   else
     x := x + 1;
     y := y - 1
   fi fi
od
```

penbmi: `r(x,y) = 2.537395e+00.x+-2.537395e+00.y+-2.046610e-01`

Termination of a fair parallel program

```
[[ while [(x>0)|(y>0) do x := x - 1] od ||
   while [(x>0)|(y>0) do y := y - 1] od ]]
```

interleaving + scheduler $\longrightarrow$

`{m>=1}` $\longleftarrow$ termination precondition determined by iterated forward/backward polyhedral analysis

```
t := ?;
assume (0 <= t & t <= 1);
s := ?;
assume ((1 <= s) & (s <= m));
while ((x > 0) | (y > 0)) do
  if (t = 1) then
     x := x - 1
  else
     y := y - 1
  fi;
  s := s - 1;
  if (s = 0) then
    if (t = 1) then
       t := 0
    else
       t := 1
    fi;
    s := ?;
    assume ((1 <= s) & (s <= m))
  else
    skip
  fi
od;;
```

penbmi: `r(x,y,m,s,t) = +1.000468e+00.x +1.000611e+00.y +2.855769e-02.m -3.929197e-07.s +6.588027e-06.t +9.998392e+03`

Relaxed Parametric Invariance Proof Method

Floyd's method for invariance

Given a loop precondition P, find an unkown loop invariant I such that:

– The invariant is *initial*:

$$\forall\, x : P(x) \Rightarrow \underset{\uparrow}{I}\,(x)$$

– The invariant is *inductive*:

$$\forall\, x, x' : \underset{\substack{\uparrow \\ ???}}{I}\,(x) \wedge [\![\mathtt{B};\mathtt{C}]\!](x, x') \Rightarrow \underset{\uparrow}{I}\,(x')$$

Abstraction

– Express loop semantics as a conjunction of LMI constraints (by relaxation for polynomial semantics)
– Eliminate the conjunction and implication by Lagrangian relaxation
– Fix the form of the unkown invariant by parametric abstraction

... we get ...

Floyd's method for numerical programs

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown parameters a, such that:

– The invariant is *initial*: $\exists \mu \in \mathbb{R}^+$:

$$\forall\ x : I_a(x) - \mu.P(x) \geq 0$$

– The invariant is *inductive*: $\exists \lambda \in [0, N] \longrightarrow \mathbb{R}^+$:

$$\forall\ x, x' : I_a(x') - \underset{\uparrow}{\lambda_0}.\underset{\uparrow}{I_a}(x) - \sum_{k=1}^{N} \lambda_k.\sigma_k(x, x') \geq 0$$

bilinear in λ_0 and a

Idea 8

Solve the bilinear matrix inequality (BMI) by semidefinite programming

Bilinear matrix inequality (BMI) solvers

$$\exists x \in \mathbb{R}^n : \bigwedge_{i=1}^{m} \left(M_0^i + \sum_{k=1}^{n} x_k M_k^i + \sum_{k=1}^{n} \sum_{\ell=1}^{n} x_k x_\ell N_{k\ell}^i \succcurlyeq 0 \right)$$

[Minimizing $x^\top Qx + cx$]

Two solvers available under MATHLAB®:

– PenBMI: M. Kočvara, M. Stingl
– bmibnb: J. Löfberg

Common interfaces to these solvers:

– Yalmip: J. Löfberg

Example: linear invariant

Program:

```
i := 2; j := 0;
while (??) do
  if (??) then
    i := i + 4
  else
    i := i + 2;
    j := j + 1
  fi
od;
```

– Invariant:

```
+2.14678e-12*i -3.12793e-10*j +0.486712 >= 0
```

– Less natural than $i - 2j - 2 \geq 0$

– Alternative:

- Determine parameters (a) by other methods (e.g. random interpretation)
- Use BMI solvers to *check* for invariance

Constraint resolution failure

– infeasibility of the constraints does not mean "non termination" or "non invariance" but simply failure

– inherent to abstraction!

Conclusion

Numerical errors

– LMI/BMI solvers do numerical computations with rounding errors, shifts, etc

– ranking function is subject to numerical errors

– the hard point is to discover a candidate for the ranking function

– much less difficult, when the ranking function is known, to re-check for satisfaction (e.g. by static analysis)

– not very satisfactory for invariance (checking only ???)

Related anterior work

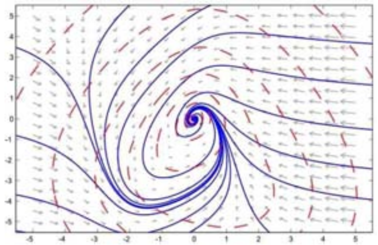
– Linear case (Farkas lemma):
  - Invariants: Sankaranarayanan, Spima, Manna (CAV'03, SAS'04, heuristic solver)
  - Termination: Podelski & Rybalchenko (VMCAI'03, Lagrange coefficients eliminated by hand to reduce to linear programming so no disjunctions, no tests, etc)
  - Parallelization & scheduling: Feautrier, easily generalizable to nonlinear case

Related posterior work

– Termination using Lyapunov functions: Roozbehani, Feron & Megrestki (HSCC 2005)

Seminal work

– LMI case, Lyapunov 1890, *"an invariant set of a differential equation is stable in the sense that it attracts all solutions if one can find a function that is bounded from below and decreases along all solutions outside the invariant set"*.



THE END, THANK YOU

More details and references in the VMCAI'05 paper.

ANNEX

- Main steps in a typical soundness/completeness proof
- SOS relaxation principle

Main steps in a typical soundness/completeness proof

$\exists r : \forall x, x' : [\![B;C]\!](x\, x') \Rightarrow r(x, x') \geq 0$

$\iff \exists r : \forall x, x' : \bigwedge_{k=1}^{N} \sigma_k(x, x') \geq 0 \Rightarrow r(x, x') \geq 0$

$\Longleftarrow$ ≀Lagrangian relaxation ($\Longrightarrow$ if lossless)≀

$\exists r : \exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \forall x, x' \in \mathbb{D}^n : r(x, x') - \sum_{k=1}^{N} \lambda_k \sigma_k(x\, x') \geq 0$

$\Longleftarrow$ ≀Semantics abstracted in LMI form ($\Longrightarrow$ if exact abstraction)≀

$\exists r : \exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \forall x, x' \in \mathbb{D}^n : r(x, x') - \sum_{k=1}^{N} \lambda_k (x\, x'\, 1) M_k (x\, x'\, 1)^\top \geq 0$

$\iff$ ≀Choose form of $r(x, x') = (x\, x'\, 1) M_0 (x\, x'\, 1)^\top$≀

$\iff \exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \forall x, x' \in \mathbb{D}^n : (x\, x'\, 1) M_0 (x\, x'\, 1)^\top - \sum_{k=1}^{N} \lambda_k (x\, x'\, 1) M_k (x\, x'\, 1)^\top \geq 0$

$\iff \exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \forall x, x' \in \mathbb{D}^{(n \times 1)} :$

$$\begin{bmatrix} x \\ x' \\ 1 \end{bmatrix}^\top \left(M_0 - \sum_{k=1}^{N} \lambda_k M_k \right) \begin{bmatrix} x \\ x' \\ 1 \end{bmatrix} \geq 0$$

$\iff$ ≀if $(x\ 1)A(x\ 1)^\top \geq 0$ for all x, this is the same as $(y\ t)A(y\ t)^\top \geq 0$ for all y and all $t \neq 0$ (multiply the original inequality by t^2 and call $xt = y$). Since the latter inequality holds true for all x and all $t \neq 0$, by continuity it holds true for all x, t, that is, the original inequality is equivalent to positive semidefiniteness of A≀

$$\exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}_* : \left(M_0 - \sum_{k=1}^{N} \lambda_k M_k\right) \succcurlyeq 0$$

≀LMI solver provides M_0 (and λ)≀

SOS Relaxation Principle

– Show $\forall x : p(x) \geq 0$ by $\forall x : p(x) = \sum_{i=1}^{k} q_i(x)^2$
– Hibert's 17th problem (sum of squares)
– Undecidable (but for monovariable or low degrees)
– Look for an approximation (relaxation) by semidefinite programming

General relaxation/approximation idea

– Write the polynomials in quadratic form with monomials as variables: $p(x, y, \ldots) = z^\top Q z$ where $Q \succcurlyeq 0$ is a semidefinite positive matrix of unknowns and $z = [\ldots x^2, xy, y^2, \ldots x, y, \ldots 1]$ is a monomial basis
– If such a Q does exist then $p(x, y, \ldots)$ is a sum of squares [5]
– The equality $p(x, y, \ldots) = z^\top Q z$ yields LMI contrains on the unkown Q: $z^\top M(Q) z \succcurlyeq 0$

[5] Since $Q \succcurlyeq 0$, Q has a Cholesky decomposition L which is an upper triangular matrix L such that $Q = L^\top L$. It follows that $p(x) = z^\top Q z = z^\top L^\top L z = (Lz)^\top Lz = [L_{i,:} \cdot z]^\top [L_{i,:} \cdot z] = \sum_i (L_{i,:} \cdot z)^2$ (where $\cdot$ is the vector dot product $x \cdot y = \sum_i x_i y_i$), proving that $p(x)$ is a sum of squares whence $\forall x : p(x) \geq 0$, which eliminates the universal quantification on x.

– Instead of quantifying over monomials values x, y, replace the monomial basis z by auxiliary variables X (loosing relationships between values of monomials)
– To find such a $Q \succcurlyeq 0$, check for semidefinite positiveness $\exists Q : \forall X : X^\top M(Q) X \geq 0$ i.e. $\exists Q : M(Q) \succcurlyeq 0$ with LMI solver
– Implement with `SOStools` under MATHLAB® of Prajna, Papachristodoulou, Seiler and Parrilo
– Nonlinear cost since the monomial basis has size $\binom{n+m}{m}$ for multivariate polynomials of degree n with m variables