# « A Lagrangian relaxation and mathematical programming framework for static analysis and verification »

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

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

LOPSTR & SAS 2004 — Verona, Italy — 28 Aug. 2004

# An impromptu[1] invited talk :-)

on summer work with Radhia Cousot.

[1] French for “extemporaneous”.

# 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 [2]:

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

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

[2] under Tarski's fixpoint theorem hypotheses

# Constraint-based static analysis

– Effectively solve a postfixpoint constraint:

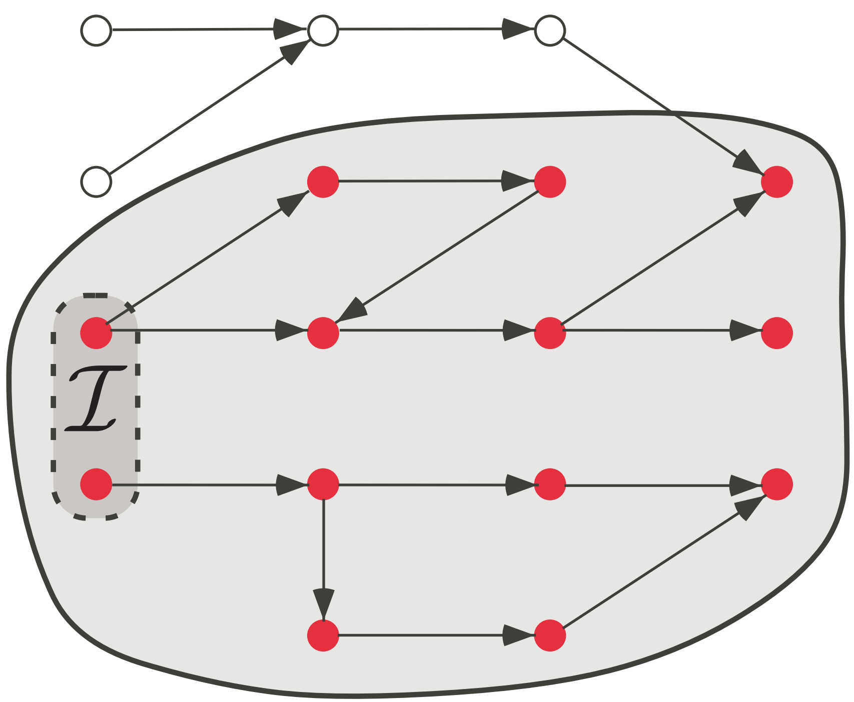
$$\mathsf{lfp}\, F = \bigsqcap\{X \mid F(X) \sqsubseteq X\}$$

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

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

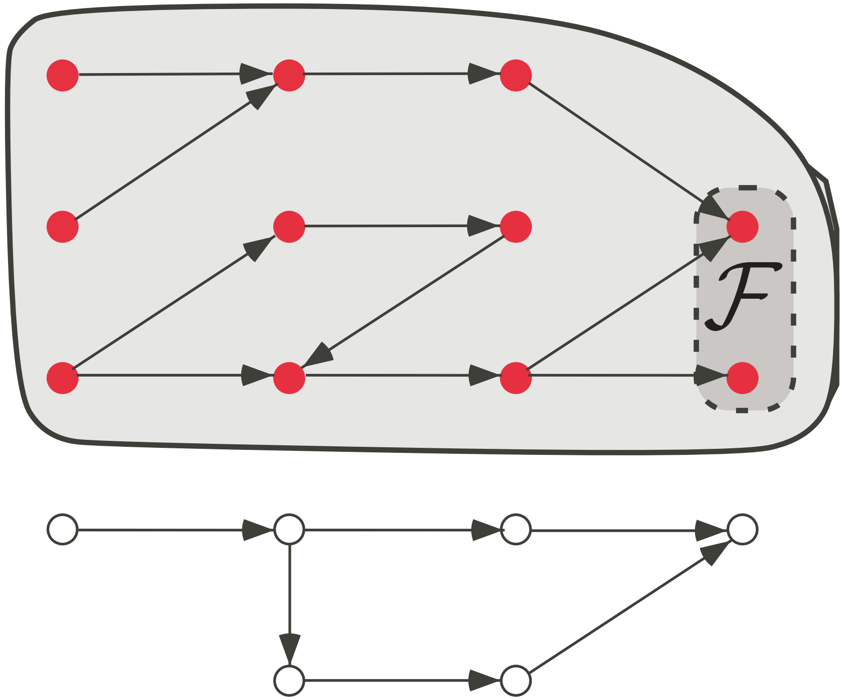
# Program properties

# Forward/reachability properties



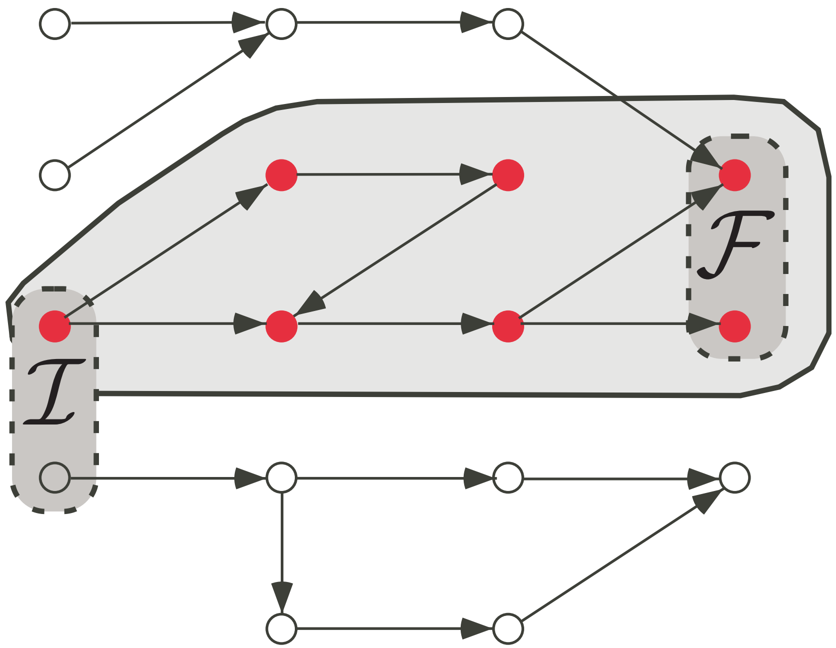

Example: partial correctness (must stay into safe states)

# Backward/ancestry properties




Example: termination (must reach final states)

# Forward/backward properties




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

# Floyd's total correctness proof method for while loops

$$\frac{\{I(\alpha) \wedge \alpha > 0\}\ B\,;C\ \{\exists \beta < \alpha : I(\beta)\},\ \ I(0) \Rightarrow \neg B}{\{\exists \epsilon : I(\epsilon)\}\ \texttt{while}\ B\ \texttt{do}\ C\ \texttt{od}\ \{I(0)\}}$$

To be incorporated in backward analysis. . .

# Iterated forward/backward iteration-based approximate static analysis

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

– Overapproximate

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

by overapproximations of the decreasing sequence

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

# Examples (with polyhedral[3] abstraction)

```
{x<=0}
while (x > 0) do
   {empty(1)}
      skip
   {empty(1)}
od
{x<=0}
```

3 using Bertand Jeannet's NewPolka library

# Bubble-sort example

```
{n>=0}
   i := n;
{n=i,n>=0}
   while (i <> 0 ) do
      {i>=1,n>=i}
         j := 0;
      {j=0,i>=1,n>=i}
         while (j <> i) do
            {j>=0,i>=j+1,n>=i}
               j := j + 1
            {j>=1,i>=j,n>=i}
      {i=j,i>=1,n>=i}
         i := i - 1
      {i+1=j,i>=0,n>=i+1}
   od
{i=0,n>=0}
```

# Arithmetic mean example

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

# Arithmetic mean example (cont'd)

Adding a backward loop counter:

```
{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}
```

# Operational semantics

# Small-step relational semantics of loops

`while B do C od`

- $x \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: 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
- $[\![\mathtt{B};\mathtt{C}]\!](x, x')$: small-step relational semantics of *one iteration of the loop body*
- $[\![\mathtt{B};\mathtt{C}]\!](x, x') = \bigwedge_{i=1}^{N} \sigma_i(x, x') \geqslant 0$ (where $\geqslant$ is $>$, $\geq$ or $=$)
- not a restriction for numerical programs

# Example of linear program (Arithmetic mean)

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

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

```
+1.x -1.y -1 >= 0
+1.x -1.y -2.k = 0
-1.k +1.k' +1 = 0
-1.x +1.x' +1 = 0
-1.y +1.y' +1 = 0
```

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

# Example of quadratic form program (factorial)

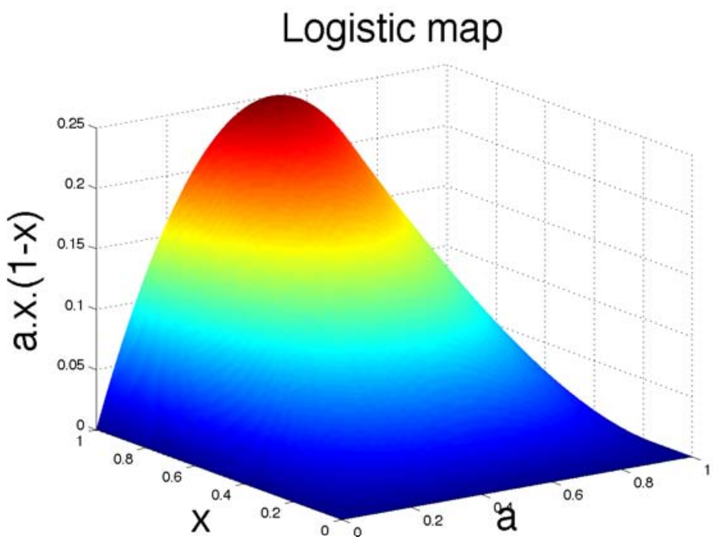
$$[x\,x']A[x\,x']^\top + 2[x\,x']\,q + r \geqslant 0$$

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

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

# 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
```



# Constraint-based static analysis

# Floyd's method for invariance

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

– The invariant is *initial*:

$$\forall\, x : P(x) \Rightarrow I(x)$$

– The invariant is *inductive*:

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

# Floyd's method for numerical programs

Given a loop precondition $P(x) \geqslant 0$, find an unkown loop invariant $I(x) \geqslant 0$ such that:

– The invariant is *initial*:

$$\forall\, x : P(x) \geqslant 0 \Rightarrow I(x) \geqslant 0$$

– The invariant is *inductive*:

$$\forall\, x, x' : \left( I(x) \geqslant 0 \wedge \bigwedge_{i=1}^{N} \sigma_i(x, x') \geqslant 0 \right) \Rightarrow I(x') \geqslant 0$$

# Floyd's method for termination

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

– The rank is *nonnegative*:

$$\forall\, x : I(x) \Rightarrow r(x) \geq 0$$

– The invariant is *inductive*:

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

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

# Solving the constraints

- Fix the form of the unkown $(I(x) \geqslant 0/r(x) \geqslant 0)$ using parameters $a$ in the form $Q(a, x) \geqslant 0$.
- The problem has the form:

$$
\begin{array}{l}
\exists\ a : \\
\quad \left( \bigwedge_{k=1}^{n} \forall\ x, x' : Q(a, x) \geqslant 0 \wedge C_k(x, x') \geqslant 0 \right) \\
\qquad \Longrightarrow \\
\qquad\quad Q(a, x') \geqslant 0
\end{array}
$$

- Find an algorithm to effectively compute $a$!

# Problems

In order to compute $a$:

– 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

  $\rightarrow$ linear programming

– linear matrix inequalities (LMI)/quadratic forms

  $\rightarrow$ semidefinite programming

– semialgebraic sets

  $\rightarrow$ polynomial quantifier elimination, or

  $\rightarrow$ relaxation with semidefinite programming

# 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

# Example: quadratic termination of logistic map

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

```
In[1]:= Clear All;
    Timing[LogicalExpand[Reduce[
            ForAll[ϵ, ϵ > 0, ForAll[a, (0 ≤ a) && (a ≤ 1 - ϵ),
              ForAll[x0, (ϵ ≤ x0) && (x0 ≤ 1),
                ForAll[x1, x1 == a * x0 * (1 - x0),
                  Exists[η, (η > 0) &&
                    (c * x0^2 + d * x0 + e ≥ 0) &&
                    (c * x0^2 + d * x0 - c * x1^2 - d * x1 ≥ η)]]]]],
            {c, d, e}, Reals]]]//TraditionalForm
```

No result after hours of computations!

# Example: linear termination of logistic map

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

```
In[1]:= Clear All;

    Timing[LogicalExpand[Reduce[
           ForAll[ε, ε > 0,
             ForAll[a, (0 ≤ a) && (a ≤ 1 - ε),
               ForAll[x0, (ε ≤ x0) && (x0 ≤ 1),
                 ForAll[x1, x1 == a * x0 * (1 - x0),
                   Exists[η, (η > 0) &&
                        (c * x0 + d ≥ 0) && (c * x0 - c * x1 ≥ η)]]]]],
           {c, d}, Reals]]]//TraditionalForm

Out[1]= {0.16 Second, c > 0 ∧ d ≥ 0}
```

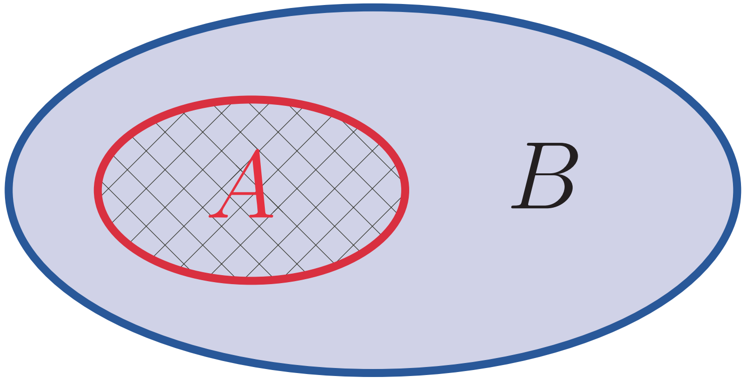
# Scaling up

– does not scale up beyond a few variables!

– too bad!

# Lagrangian relaxation for implication elimination

# Implication (general case)

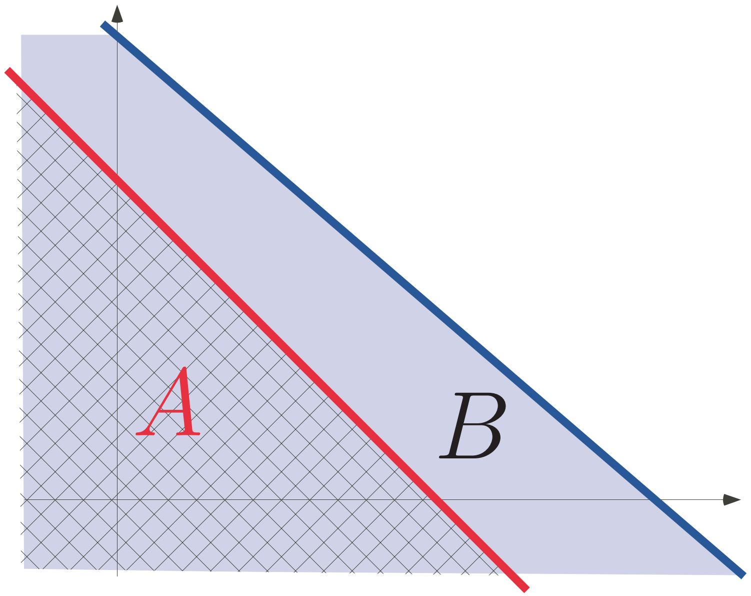



$$A \Rightarrow B$$

$$\Leftrightarrow$$

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

# Implication (linear case)



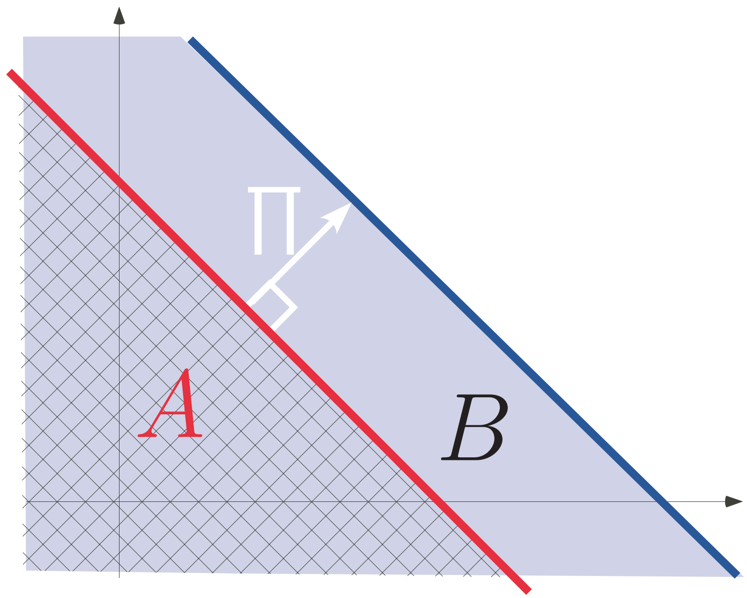

$A \Rightarrow B$ (assuming $A \neq \emptyset$)

$\Leftarrow$ (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 $\forall k \in [1, N] : \sigma_k \in \mathbb{V} \mapsto \mathbb{R}$.

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

$\Leftarrow$ 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, $\lambda_i$ = 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)

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

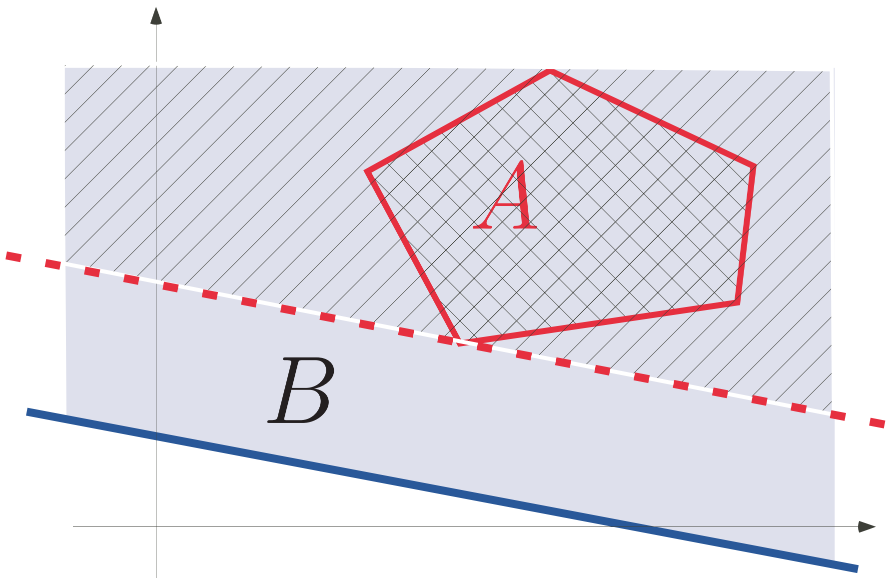
$$\wedge \ \exists \lambda' \in [1, N] \mapsto \mathbb{R}_* : \forall x \in \mathbb{V} : \sigma_0(x) + \sum_{k=1}^{N} \lambda'_k \sigma_k(x) \geq 0$$

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

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

# 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

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

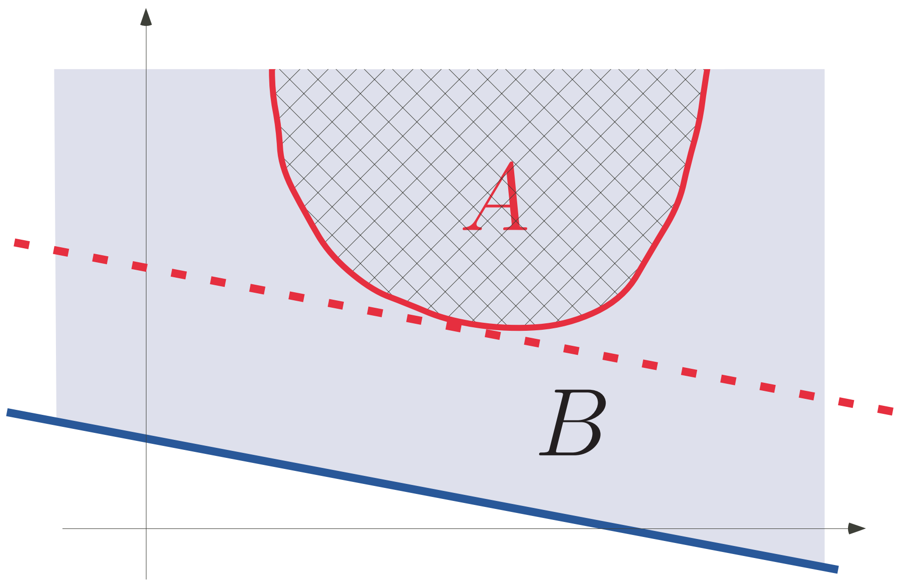
$$\Leftarrow \text{(soundness, Lagrange)}$$

$$\Rightarrow \text{(completeness, Farkas)}$$

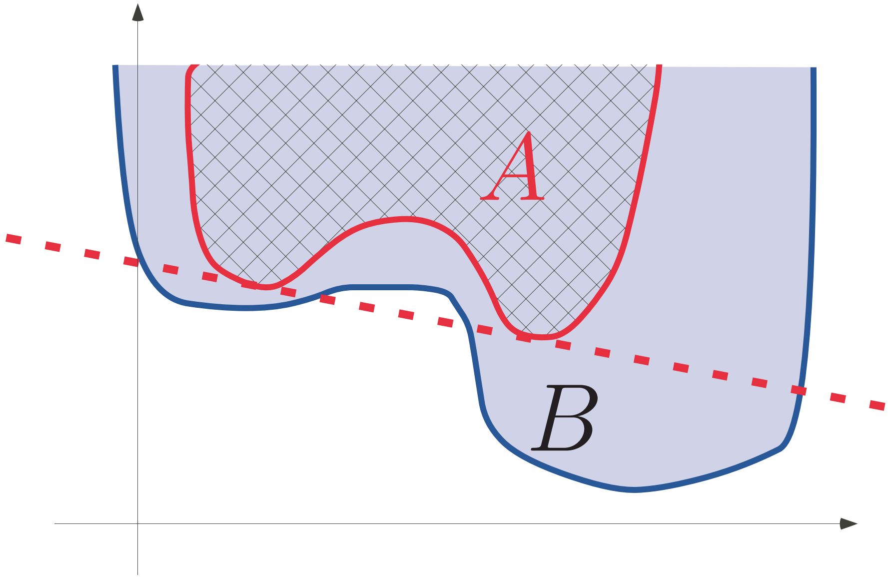
$$\exists \lambda \geq 0 : \forall x : cx + d - \lambda(Ax + b) \geq 0 \ .$$

# 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)

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

# Semidefinite programming for quantifier elimination

# Mathematical programming

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

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

**feasibility problem** : find a solution to the constraints

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

# 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

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

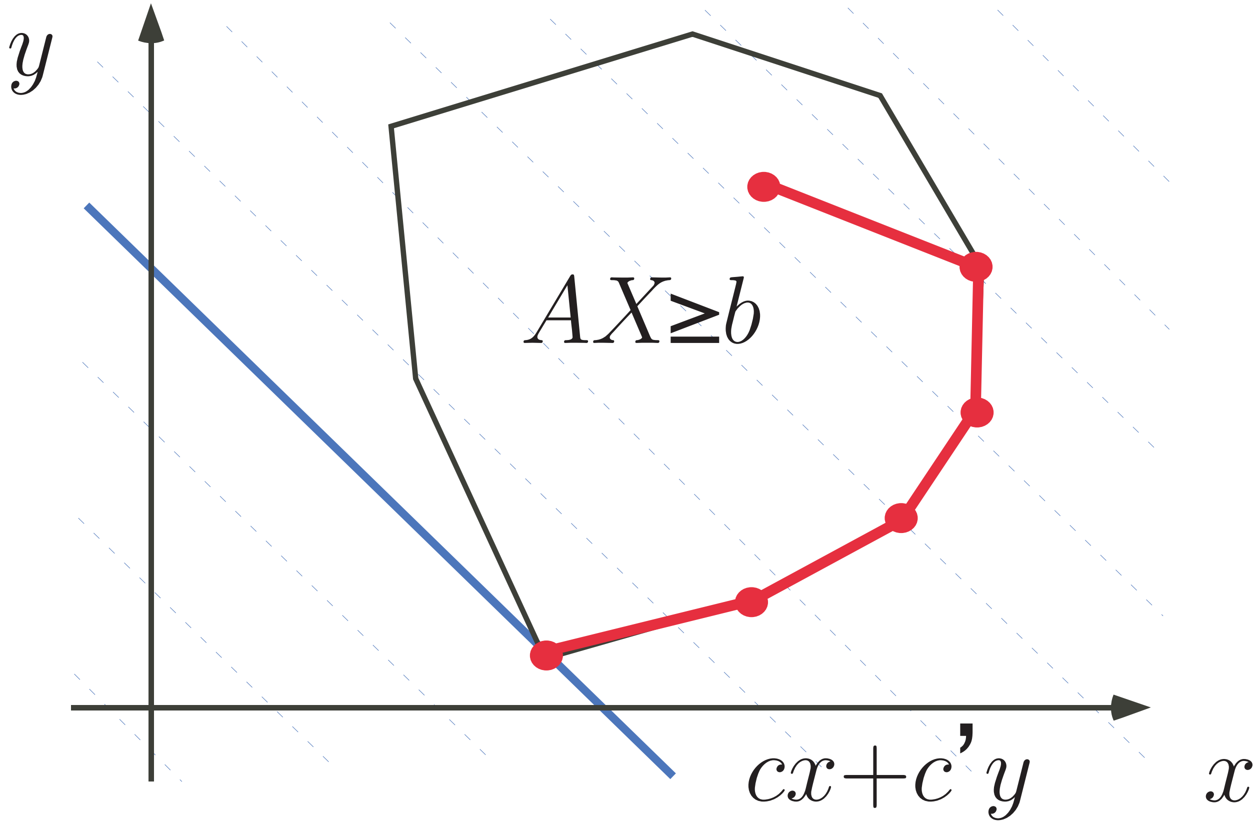
# Example: linear programming

# Example: linear programming

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

[Minimizing $cx$]

# The simplex



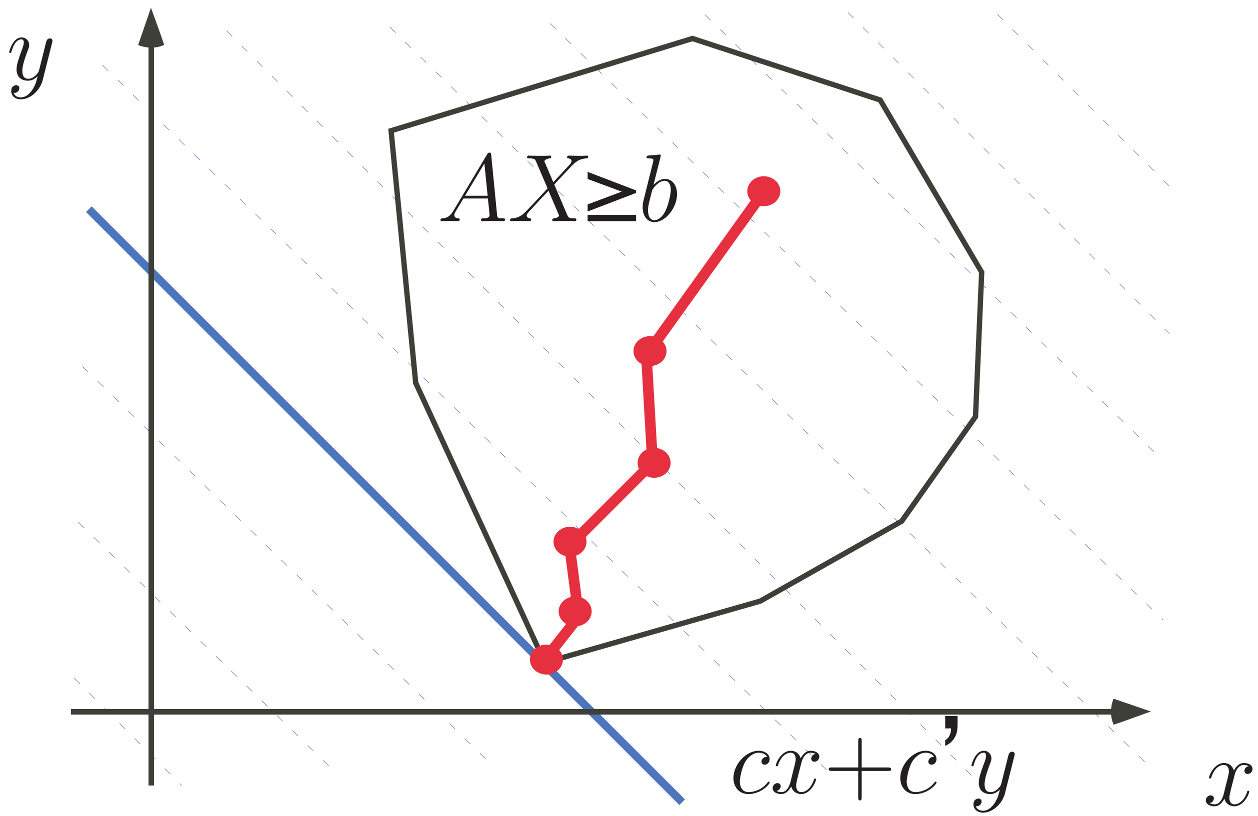

Dantzig 1948, exponential in worst case, good in practice

# Polynomial methods

**Ellipsoid method** : Khachian 1979, polynomial in worst case but not good in practice

**Interior point method** : Kamarkar 1984, polynomial in worst case and good in practice (hundreds of thousands of variables)

# The interior point method



# Example: semidefinite programming

# Semidefinite programming

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

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

Where the linear matrix inequality is

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

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:

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

of the form of the formulæ we are interested in!

# Bilinear/quadratic forms

Bilinear forms:

$$Y^{\top} M X$$

Quadratic forms:

$$X^{\top} M X$$

# Example of quadratic forms: linear inequalities

A line of $(A\ A')(x\ x')^\top + b$ is $(A_{k,:}\ A'_{k,:})(x\ x')^\top + b_k = (x\ x'\ 1)M_k(x\ x'\ 1)^\top$ where

$$M_k = \begin{bmatrix} \mathsf{0}^{(2n\times 2n)} & \begin{matrix} \frac{A_{k,:}^\top}{2} \\ \frac{{A'}_{k,:}^\top}{2} \end{matrix} \\ \begin{matrix} \frac{A_{k,:}}{2} & \frac{A'_{k,:}}{2} \end{matrix} & b_k \end{bmatrix}$$

$$[x\ x'\ 1] M_k [x\ x'\ 1]^\top$$

$$= (x\ x'\ 1) \begin{bmatrix} 0^{(2n \times 2n)} & \frac{A_{k,:}^\top}{2} \\ & \frac{{A'}_{k,:}^\top}{2} \\ \frac{A_{k,:}}{2} \quad \frac{A'_{k,:}}{2} & b_k \end{bmatrix} \begin{bmatrix} x^\top \\ {x'}^\top \\ 1 \end{bmatrix}$$

$$= (x\ x'\ 1) \begin{bmatrix} \frac{A_{k,:}^\top}{2} \\ \frac{{A'}_{k,:}^\top}{2} \\ \frac{A_{k,:}}{2} x^\top \frac{A'_{k,:}}{2} {x'}^\top + b_k \end{bmatrix}$$

$$= \quad x \frac{A_{k,:}^\top}{2} + x' \frac{{A'}_{k,:}^\top}{2} + \frac{A_{k,:}}{2} x^\top + \frac{A'_{k,:} {x'}^\top}{2} + b_k$$

$$= \quad (A_{k,:}\ A'_{k,:})(x\ x')^\top + b_k \qquad \wr \text{since } (AB)^\top = B^\top A^\top \int$$

# Example of quadratic forms: quadratic inequalities

$$(x\,x')P_k(x\,x')^\top + 2q_k^\top(x\,x')^\top + r_k \geq 0$$

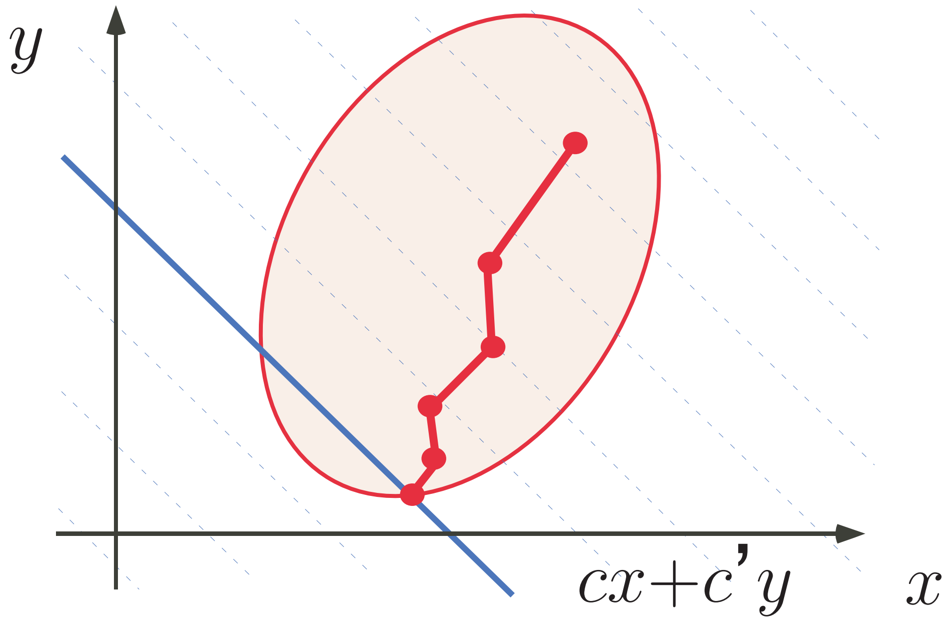
$$= (x\,x'\,1)M_k(x\,x'\,:\,1)^\top$$

where

$$M_k = \begin{bmatrix} P_k & q_k \\ q_k^\top & r_k \end{bmatrix}$$

# Interior point method for semidefinite programming

– Nesterov & Nemirovskii 1988, polynomial in worst case and good in practice (thousands of variables)




– Various path strategies e.g. "stay in the middle"

# Interior point algorithms for semidefinite programming

Interior point algorithms work because of appropriate generalizations from polyhedra:

– linear $\rightarrow$ convex

– partial ordering $\geq \rightarrow \succcurlyeq$

# Semidefinite programming solvers

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

– `lmilab`: P. Gahinet, A. Nemirovskii, A.J. Laub, M. Chilali

– `SeDuMi`: J. Sturm

– `bnb`: J. Löfberg (integer semidefinite programming)

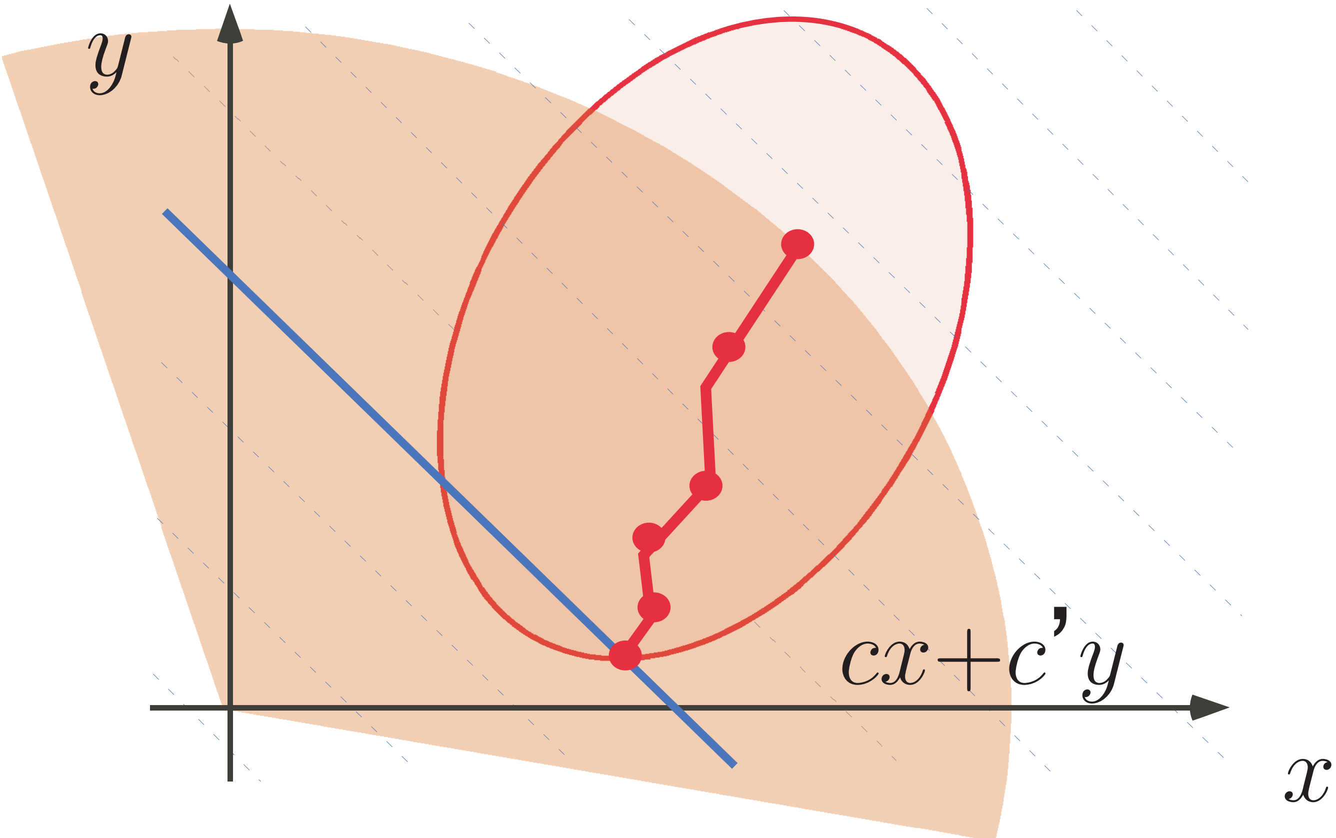
Common interfaces to these solvers, a.o.:

– `Yalmip`: J. Löfberg

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

Main application: nonlinear automatic control theory

# Imposing a feasibility radius



# Well-posedness problem

– Equality constraints may cause well-posedness problems with feasibility (solvers better handle strict inequalities)

– In this case, one can slightly relax the constraint by adding a negative shift

# Example with a variable shift

```
» x = sdpvar(1,1);
» F = set(diag([x -x])>0);
» solvesdp(F, [] ,sdpsettings('solver', 'lmilab'))
...
ans = ...
          info: 'Infeasible problem (LMILAB)'
          ...
» t = sdpvar(1,1);
» solvesdp(F, -t, sdpsettings('solver', 'lmilab', 'shift',t))
...
ans = ...
          info: 'No problems detected (LMILAB)'
          ...
» disp(double(x))
     0
» disp(double(t))
  -2.0154e-11
```

# Lagrangian relaxation and semidefinite programming for static analysis (1) Examples

# Linear example: termination of decrementation

```
» [N Mk(:,:,:)]=linToMk([1 0; 0 1],[0 0; 0 0],[-1; -1]);
» [M Mk(:,:,N+1:N+M)]=linToMk([-1 1; 0 -1],[1 0; 0 1],[0; 0]);
» N
N = 2
» M
M = 2
» format rational; Mk
Mk(:,:,1) =                     Mk(:,:,2) =
   0    0    0    0   1/2          0    0    0    0    0
   0    0    0    0    0           0    0    0    0   1/2
   0    0    0    0    0           0    0    0    0    0
   0    0    0    0    0           0    0    0    0    0
  1/2   0    0    0   -1           0   1/2   0    0   -1
Mk(:,:,3) =                     Mk(:,:,4) =
   0    0    0    0 -1/2           0    0    0    0    0
   0    0    0    0  1/2           0    0    0    0 -1/2
   0    0    0    0  1/2           0    0    0    0    0
   0    0    0    0    0           0    0    0    0   1/2
 -1/2 1/2  1/2   0    0            0 -1/2   0   1/2   0
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
   x := x - y
od
```

```
» display_Mk(Mk, N, {'x' 'y'});
...
 +1.x -1 >= 0
 +1.y -1 >= 0
 -1.x +1.y +1.x' = 0
 -1.y +1.y' = 0
...
» [diagnostic,R] = termination(Mk, N,...
       'float', 'linear');
» disp(diagnostic)
  termination (lmilab)
» fltrank(R, {'x' 'y'})
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
   x := x - y
od
```

```
r(x,y) = +2.178955e+12.x +1.453116e+12.y -1.451513e+12
```

one possible ranking function amongst infinitely many others

# Fixing the radius:

```
clear all;
[N Mk(:,:,:)]=linToMk([1 0; 0 1],...
                   [0 0; 0 0],[-1; -1]);
[M Mk(:,:,N+1:N+M)]=linToMk([-1 1; 0 -1],...
                               [1 0; 0 1],[0; 0]);
[diagnostic,R] = termination(Mk, N, 'float',...
                      'linear', 1.0e4);
disp(diagnostic)
fltrank(R, {'x' 'y'})
...
   f-radius saturation:  85.927% of R =  1.00e+04
...
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
    x := x - y
od
```

```
termination (lmilab)
r(x,y) = +4.074723e+03.x +2.786715e+03.y +1.549410e+03
```

# Changing the solver:

```
\begin{verbatim}
[N Mk(:,:,:)]=linToMk([1 0; 0 1],...
                  [0 0; 0 0],[-1; -1]);
[M Mk(:,:,N+1:N+M)]=linToMk([-1 1; 0 -1],...
                              [1 0; 0 1],[0; 0]);
[diagnostic,R] = termination(Mk, N, 'float',...
                        'linear', 1.0e4, 'sedumi');
disp(diagnostic)
fltrank(R, {'x' 'y'})
...
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
   x := x - y
od
```

```
termination (sedumi)
r(x,y) = +2.271450e+03.x +1.810903e+03.y -3.623997e+03
```

# Enforcing an integer ranking function:

```
clear all;
[N Mk(:,:,:)]=linToMk([1 0; 0 1],...
                   [0 0; 0 0],[-1; -1]);
[M Mk(:,:,N+1:N+M)]=linToMk([-1 1; 0 -1],...
                               [1 0; 0 1],[0; 0]);
[diagnostic,R] = termination(Mk, N,...
                      'integer', 'linear');
disp(diagnostic)
intrank(R, {'x' 'y'} )
...
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
    x := x - y
od
```

```
termination (bnb)
r(x,y) = +2.x +2.y -3
```

(integer semidefinite programming still in infancy)

# Linear example: termination of arithmetic mean

```
» clear all;
% linear inequalities
%        x0 y0 k0
Ai =  [  1 -1  0]; % x0 - y0 - 1 >= 0
%         x  y  k
Ai_ = [  0  0  0];
bi = [-1];
% linear equalities
%        x0 y0 k0
Ae =  [  1 -1 -2;  % x0 - y0 - 2*k0 = 0
         0  0 -1;
        -1  0  0;
         0 -1  0];
%         x  y  k
Ae_ = [  0  0  0;
         0  0  1;  % k - k0 + 1 = 0
         1  0  0;  % x - x0 + 1 = 0
         0  1  0]; % y - y0 + 1 = 0
be = [0; 1; 1; 1];
```

Iterated forward/backward polyhedral analysis:

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

```
» N Mk(:,:,:)]=linToMk(Ai,Ai_,bi);
» [M Mk(:,:,N+1:N+M)]=linToMk(Ae,Ae_,be);
» display_Mk(Mk, N,{'x' 'y' 'k'});
...
 +1.x -1.y -1 >= 0
 +1.x -1.y -2.k = 0
 -1.k +1.k' +1 = 0
 -1.x +1.x' +1 = 0
 -1.y +1.y' +1 = 0
...
» [diagnostic,R] = termination(Mk, N, 'integer', 'linear');
» disp(diagnostic)
   termination (lmilab)
» fltrank(R, {'x' 'y' 'k'})
```

```
r(x,y,k) = +1.382113e+03.x -1.382113e+03.y +4.978695e+03.k
                                                +2.711732e+03
```

# Linear example: 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:

```
1: {y>=1}
  q := 0;
2: {q=0,y>=1}
  r := x;
3: {x=r,q=0,y>=1}
  loop invariant: {q>=0}
  while (y <= r) do
    4: {y<=r,q>=0}
      r :=  - y + r;
    5: {r>=0,q>=0}
      q := q + 1
    6: {r>=0,q>=1}
  od {y - r - 1 >= 0}
7: {q>=0,y>=r+1}
```

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

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

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

# Quadratic example: termination of factorial

```
» clear all
Ai  = [0 -1  1;  % inequality constraints
       1  0  0; 0  1  0]
Ai_ = [0  0  0;
       0  0  0; 0  0  0]
bi  = [0; 0; -1]
[N Mk(:,:,:)]=linToMk(Ai,Ai_,bi);
Ae  = [-1  0  0; % equality constraints
        0  0  1]
Ae_ = [ 1  0  0; 0  0 -1]
be  = [-1; 0]
[M Mk(:,:,N+1:N+M)]=linToMk(Ae,Ae_ ,be);
P(:,:,1)=[0 0    0 0    0 0;  0 0    0 -1/2 0 0; % quadratic equality
          0 0    0 0    0 0;  0 -1/2 0 0    0 0;
          0 0    0 0    0 0;  0 0    0 0    0 0]
q(:,1)=[0; 0; 0; 0; 1/2; 0]
r(:,1)=0
```

Iterated forward/backward polyhedral analysis:

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

```
» [m Mk(:,:,N+M+1:N+M+m)]=quaToMk(P,q,r);
» M = M + m;
» display_Mk(Mk, N, {'n' 'f' 'N'});
...
 -1.f +1.N >= 0
 +1.n >= 0
 +1.f -1 >= 0
 -1.n +1.n' -1 = 0
 +1.N -1.N' = 0
 -1.f.n' +1.f' = 0
...
» [diagnostic R] = termination(Mk, N, 'float', 'linear', 1.0e+3, 'sedumi');
» disp(diagnostic)
» fltrank(R, {'n' 'f' 'N'} )
```

```
termination (sedumi)
r(n,f,N) = -9.993462e-01.n +1.617225e-04.f +2.688476e+02.N
                                                    +8.745232e+02
```

# Lagrangian relaxation and semidefinite programming for static analysis (2) Foundations

# Main steps in a typical soundness/completeness proof

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

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

$\Longleftarrow$ ≀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)≀

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

$$\Longleftrightarrow \quad \wr \text{Choose form of } r(x, x') = (x\, x'\, 1) M_0 (x\, x'\, 1)^\top \wr$$

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

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

$\Longleftrightarrow$ $\wr$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$$\wr$

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

$\wr$LMI solver provides $M_0$ (and $\lambda$)$\wr$

# Example: LMI constraints for decrementation

```
» [N Mk(:,:,:)]=linToMk([1 0; 0 1],[0 0; 0 0],[-1; -1]);
» [M Mk(:,:,N+1:N+M)]=linToMk([-1 1; 0 -1],[1 0; 0 1],[0; 0]);
» N
N = 2
» M
M = 2
» format rational; Mk
Mk(:,:,1) =                    Mk(:,:,2) =
  0    0    0    0   1/2         0    0    0    0    0
  0    0    0    0    0          0    0    0    0   1/2
  0    0    0    0    0          0    0    0    0    0
  0    0    0    0    0          0    0    0    0    0
 1/2   0    0    0   -1          0   1/2   0    0   -1
Mk(:,:,3) =                    Mk(:,:,4) =
  0    0    0    0  -1/2         0    0    0    0    0
  0    0    0    0   1/2         0    0    0    0  -1/2
  0    0    0    0   1/2         0    0    0    0    0
  0    0    0    0    0          0    0    0    0   1/2
-1/2 1/2  1/2    0    0          0  -1/2   0   1/2   0
```

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
   x := x - y
od
```

We look for a linear termination function $r(x, y) = c_1 x + c_2 y + d$

in matrix form $X = \begin{bmatrix} 0 & 0 & \frac{c_1}{2} \\ 0 & 0 & \frac{c_2}{2} \\ \frac{c_1}{2} & \frac{c_2}{2} & d \end{bmatrix}$

The semidefinite constraints are

Iterated forward/backward polyhedral analysis:

```
{y >= 1}
while (x >= 1) do
   x := x - y
od
```

```
M0 = [X(1:n,1:n) zeros(n,n) X(1:n,n+1);
      zeros(n,n) zeros(n,n) zeros(n,1);
      X(n+1,1:n) zeros(1,n) X(n+1,n+1)];
one = [zeros(2*n,2*n) zeros(2*n,1);
       zeros(1,2*n) 1];
```

```
M_0 = [zeros(n,n) zeros(n,n+1);
       zeros(n+1,n) X];
```

```
M0-l(1,1)*Mk(:,:,1)-l(2,1)*Mk(:,:,2)-l(3,1)*Mk(:,:,3)-l(4,1)*Mk(:,:,4)>0
M0-M_0-one-l_(1,1)*Mk(:,:,1)-l_(2,1)*Mk(:,:,2)-l_(3,1)*Mk(:,:,3)-l_(4,1)*Mk(:,:,4)>0
l(1,1)>0            l_(1,1)>0
l(2,1)>0            l_(2,1)>0
```

# When constraint resolution fails...

Infeasibility of the constraints does not mean "non termination" but simply failure:

- There can be a ranking of a different form (e.g. quadratic while looking for a linear one),
- The solver may have failed (e.g. add a shift).

# 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 disjunctive loop tests and tests in loop body

– By case analysis

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

# Example of tests in loop body

```
...
test true:
 -1.x +1.y -1 >= 0
 +1.i >= 0
 -1.i -1.x +1.x' -1 = 0
 -1.y +1.y' = 0
 -1.i +1.i' = 0
test false:
 -1.x +1.y -1 >= 0
 -1.i -1 >= 0
 -1.i -1.y +1.y' = 0
 -1.x +1.x' = 0
 -1.i +1.i' = 0
...
```

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

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

# Handling nondeterminacy

– Same for concurrency by interleaving
– Same with fairness by nondeterministic interleaving with encoding of an explicit scheduler scheduler

# Semidefinite programming relaxation for polynomial quantifier elimination (1) Examples

# Semialgebraic example: logistic map

```
» clear all;
pvar a x0 x1 c0 d0 e0 l1 l2 l3 l4 l5 m1 m2 m3 m4 m5;
eps=1.0e-10;
iv = [a;x0;x1];
uv = [c0;d0;l1;l2;l3;l4;l5;m1;m2;m3;m4;m5];
pb=sosprogram(iv,uv);
pb=sosineq(pb,l1);
pb=sosineq(pb,l2);
pb=sosineq(pb,l3);
pb=sosineq(pb,l4);
pb=sosineq(pb,c0*x0+d0-l1*a-l2*(1-eps-a)-l3*(x0-eps)-l4*(1-x0)-l5*(x1-a*x0*(1-x0)));
pb=sosineq(pb,m1);
pb=sosineq(pb,m2);
pb=sosineq(pb,m3);
pb=sosineq(pb,m4);
pb=sosineq(pb,c0*x0-c0*x1-eps^2-m1*a-m2*(1-eps-a)-m3*(x0-eps)...
-m4*(1-x0)-m5*(x1-a*x0*(1-x0)));
spb=sossolve(pb);
```

```
c=sosgetsol(spb,c0);
d=sosgetsol(spb,d0);
disp(sprintf('r(x) = %i.x + %i',double(c),double(d)));
Size: 28  22

SeDuMi 1.05R5 by Jos F. Sturm, 1998, 2001-2003.
Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
eqs m = 22, order n = 37, dim = 41, blocks = 11
nnz(A) = 78 + 0, nnz(ADA) = 84, nnz(L) = 53
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg
  0 :            6.76E-01 0.000
  1 :   1.08E-20 1.87E-01 0.000 0.2771 0.9000 0.9000   1.00  1  0
  2 :   1.53E-20 6.85E-03 0.000 0.0366 0.9900 0.9900   1.00  1  1
  3 :   1.54E-20 2.20E-05 0.000 0.0032 0.9990 0.9990   1.00  1  1
  4 :   1.54E-20 2.22E-06 0.023 0.1006 0.9450 0.9450   1.00  1  1
  5 :   1.54E-20 1.20E-07 0.293 0.0542 0.9675 0.9675   1.00  1  2
  6 :   1.54E-20 6.23E-10 0.026 0.0052 0.9990 0.9990   1.00  2  8
  7 :   1.54E-20 1.63E-11 0.389 0.0261 0.9900 0.9900   1.00  2 13
```

```
iter seconds digits       c*x                 b*y
  7      1.1   Inf  0.0000000000e+00  1.5417832245e-20
|Ax-b| =   7.0e-11, [Ay-c]_+ =   1.3E-11, |x|=  5.7e+00, |y|=  3.1e+00
Max-norms: ||b||=1.000000e-20, ||c|| = 0,
Cholesky |add|=1, |skip| = 5, ||L.L|| = 500000.

Residual norm: 7.0272e-11

       cpusec: 1.0900
         iter: 7
    feasratio: 1.0000
         pinf: 0
         dinf: 0
       numerr: 0
```

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

»

# Semidefinite programming relaxation for polynomial quantifier elimination
# (2) Foundations

# Principle

– Show $\forall x : p(x) \geq 0$ by $\forall x : p(x) = \Sigma_{i=1}^{k} q_i(x)^2$

– 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 [4]

– The equality $p(x, y, \ldots) = z^\top Q z$ yields LMI contrains on the unkown $Q$: $z^\top M(Q) z \succcurlyeq 0$

[4] 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 $p(x) = z^\top Q z = z^\top L^\top L z = (Lz)^\top L z = [L_{i,:} \cdot z]^\top [L_{i,:} \cdot z] = \sum_i (L_{i,:} \cdot z)^2$ (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 $\begin{pmatrix} n+m \\ m \end{pmatrix}$ for multivariate polynomials of degree $n$ with $m$ variables

# Data structures

– Use norms (size, height,...) mapping data structures to $\mathbb{R}$ and then Lagrangian relaxation with semidefinite programming [relaxation]
– One of the first uses of polyhedral analysis
– Studied since 20 years in the logic programming community
– But can now go beyond linear norms

# Conclusion

# Numerical errors

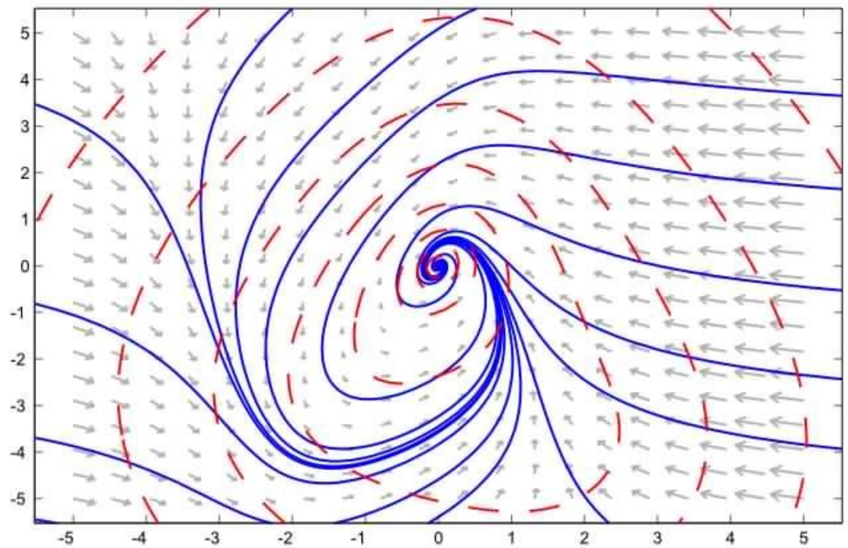
- LMI solvers do numerical computations with rounding errors, shifts, etc
- ranking function is subject to numerical errors
- the hard point is to discover a candidate for the ranking function
- much less difficult, when it is known, to re-check for satisfaction (e.g. by static analysis)

# Related work

- Linear case (Farkas):
  - 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

# 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