

« Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming »

Patrick Cousot

École normale supérieure

45 rue d'Ulm, 75230 Paris cedex 05, France

Patrick.Cousot@ens.fr

www.di.ens.fr/~cousot

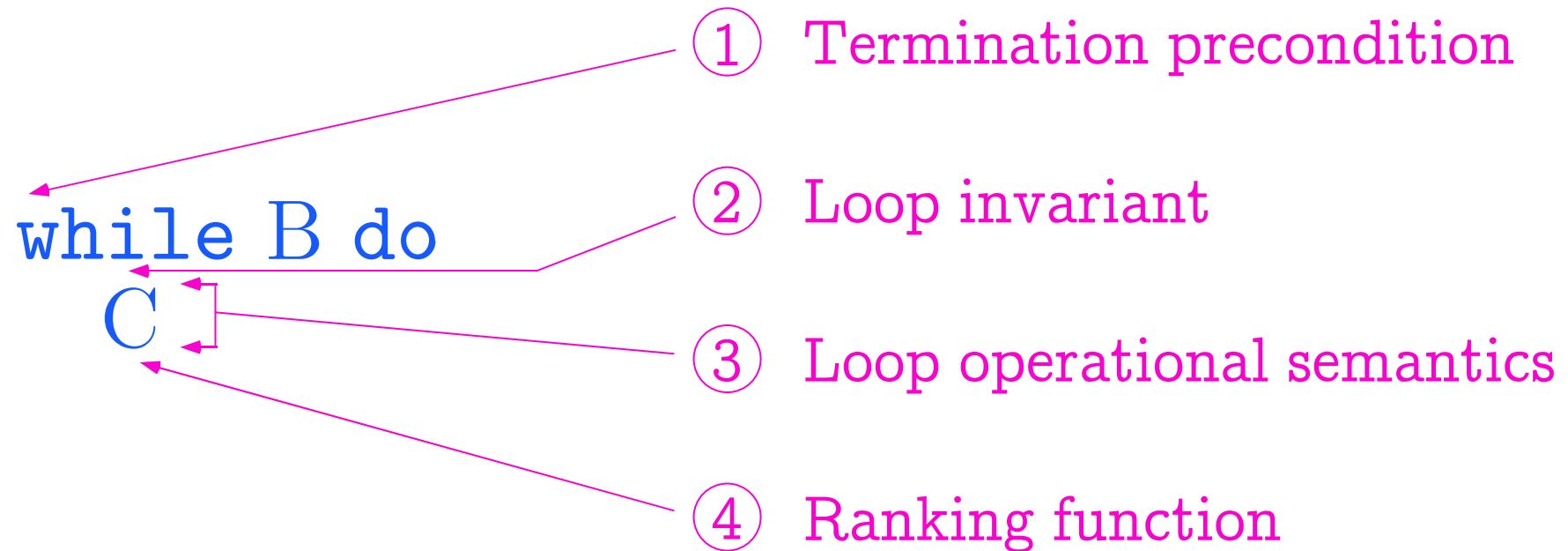
VMCAI'05 — Paris, France — 17 Jan. 2005



Overview of the Termination Analysis Method



Proving Termination of a Loop



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.

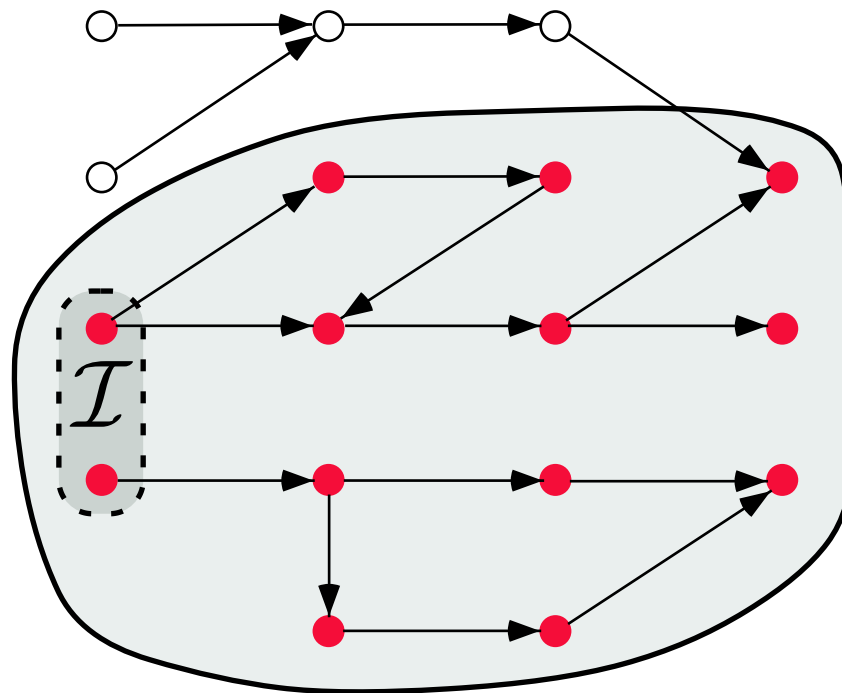


Arithmetic Mean Example

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*



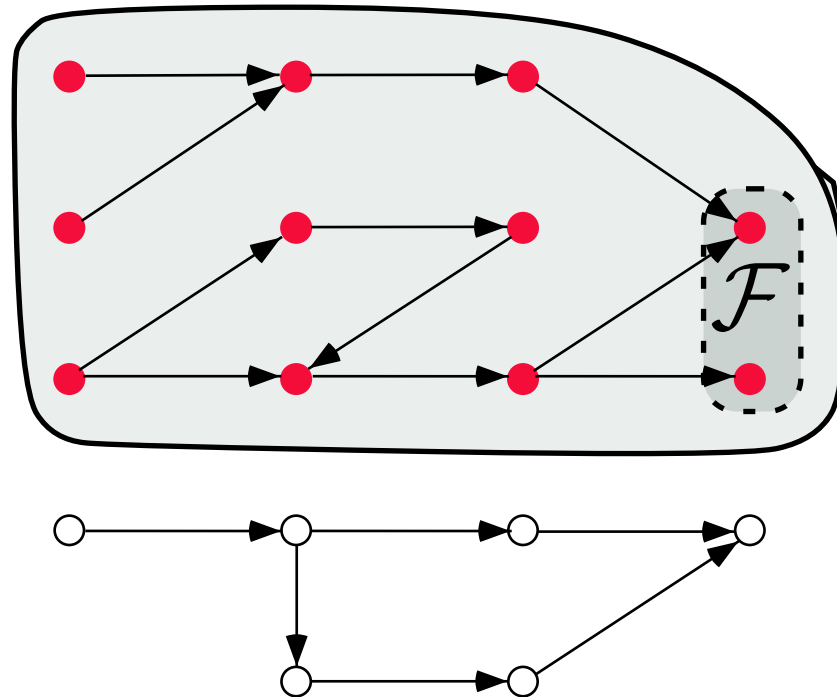
Forward/reachability properties



Example: **partial correctness** (must stay into safe states)



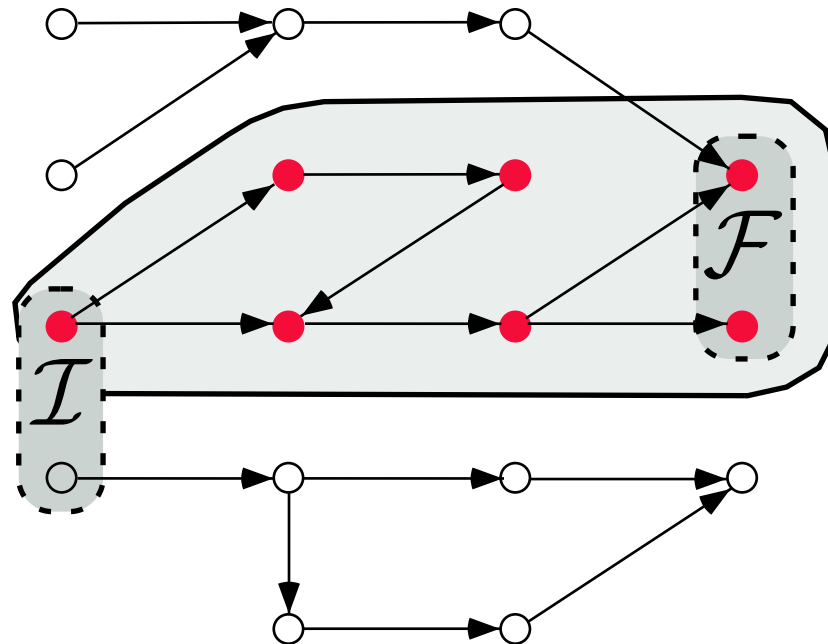
Backward/ancestry properties



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

$$\text{lfp } F \sqcap \text{lfp } B$$

by overapproximations of the decreasing sequence

$$\begin{aligned} X^0 &= \top \\ &\dots \\ X^{2n+1} &= \text{lfp } \lambda Y . X^{2n} \sqcap F(Y) \\ X^{2n+2} &= \text{lfp } \lambda Y . X^{2n+1} \sqcap B(Y) \\ &\dots \end{aligned}$$



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 $(x=y+2*k) \& (x \geq y+2)$;

$\{x=y+2k, x \geq y+2\}$

assume $(x < y)$;

empty(6)

assume $(x_0=x) \& (y_0=y) \& (k_0=k)$;

empty(6)

$k := k - 1$;

$x := x - 1$;

$y := y + 1$

empty(6)

Case $x > y$:

assume $(x=y+2*k) \& (x \geq y+2)$;

$\{x=y+2k, x \geq y+2\}$

assume $(x > y)$;

$\{x=y+2k, x \geq y+2\}$

assume $(x_0=x) \& (y_0=y) \& (k_0=k)$;

$\{x=y+2k_0, y=y_0, x=x_0, x=y+2k,$
 $x \geq y+2\}$

$k := k - 1$;

$x := x - 1$;

$y := y + 1$

$\{x+2=y+2k_0, y=y_0+1, x+1=x_0,$
 $x=y+2k, x \geq y\}$



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 unknown rank function r such that:

- The rank is *nonnegative*:

$$\forall x_0, x : I(x_0) \wedge \llbracket B; C \rrbracket(x_0, x) \Rightarrow r(x_0) \geq 0$$

- The rank is *strictly decreasing*:

$$\forall x_0, x : I(x_0) \wedge \llbracket B; C \rrbracket(x_0, x) \Rightarrow r(x) \leq r(x_0) - \eta$$

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



```

» 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);

```

Arithmetic Mean Example: Ranking Function

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
```

...

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

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

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



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, $\llbracket B; C \rrbracket(x_0, x)$: relational semantics of *one iteration of the loop body*
- $$I(x_0) \wedge \llbracket B; C \rrbracket(x_0, x) = \bigwedge_{i=1}^N \sigma_i(x_0, x) \geq_i 0 \quad (\geq_i \in \{>, \geq, =\})$$
- not a restriction for numerical programs



Example of linear program (Arithmetic mean)

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

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

$$\begin{aligned} +1.x \ -1.y &\geq 0 \\ -2.k_0 \ +1.x \ -1.y \ +2 &= 0 \\ -1.y_0 \ +1.y \ -1 &= 0 \\ -1.x_0 \ +1.x \ +1 &= 0 \\ +1.x \ -1.y \ -2.k &= 0 \end{aligned}$$

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



Example of quadratic form program (factorial)

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

```

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

```

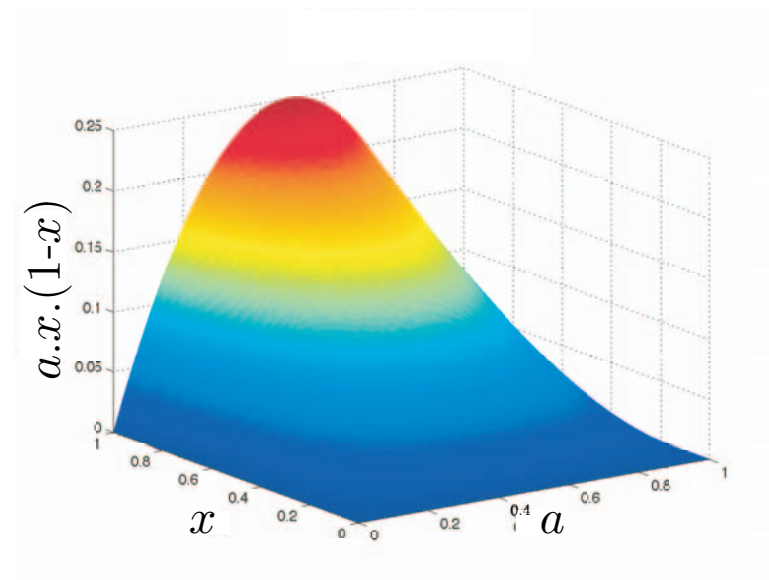
-1.f0 +1.N0 >= 0
+1.n0 >= 0
+1.f0 -1 >= 0
-1.n0 +1.n -1 = 0
+1.N0 -1.N = 0
-1.f0.n +1.f = 0
    
```

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



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 unknown rank function r and $\eta > 0$ such that:

- The rank is *nonnegative*:

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

- The rank is *strictly decreasing*:

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

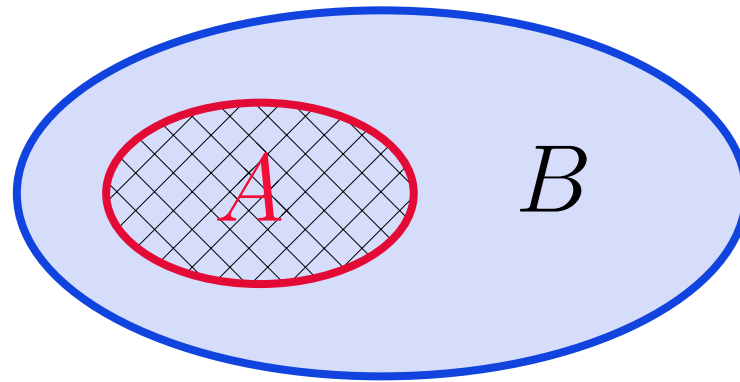


Idea 3

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



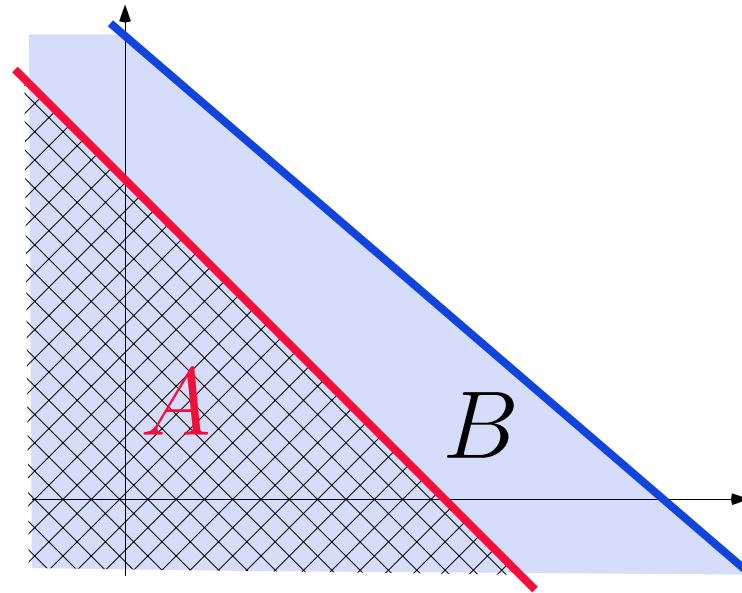
Implication (general case)



$$\begin{aligned} & A \Rightarrow B \\ \Leftrightarrow & \\ & \forall x \in A : x \in B \end{aligned}$$



Implication (linear case)



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

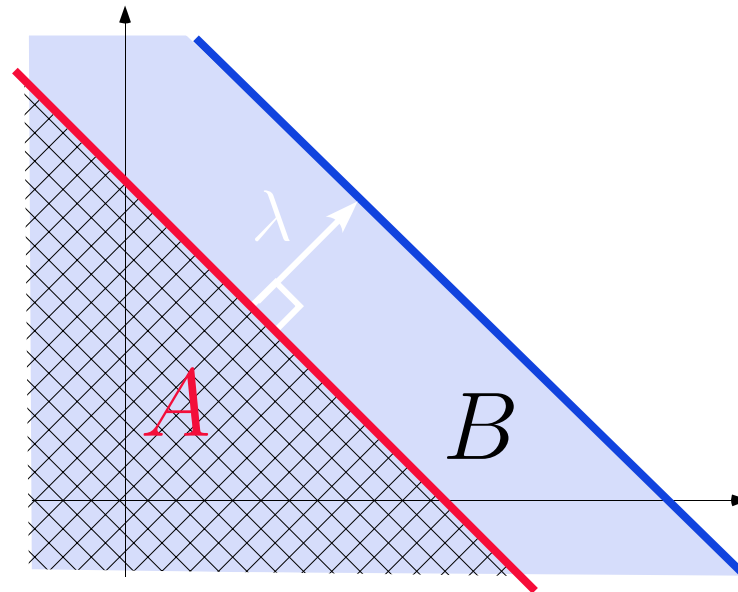
$\Leftarrow$ (soundness)

$\Rightarrow$ (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*)
 $\nRightarrow$ 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



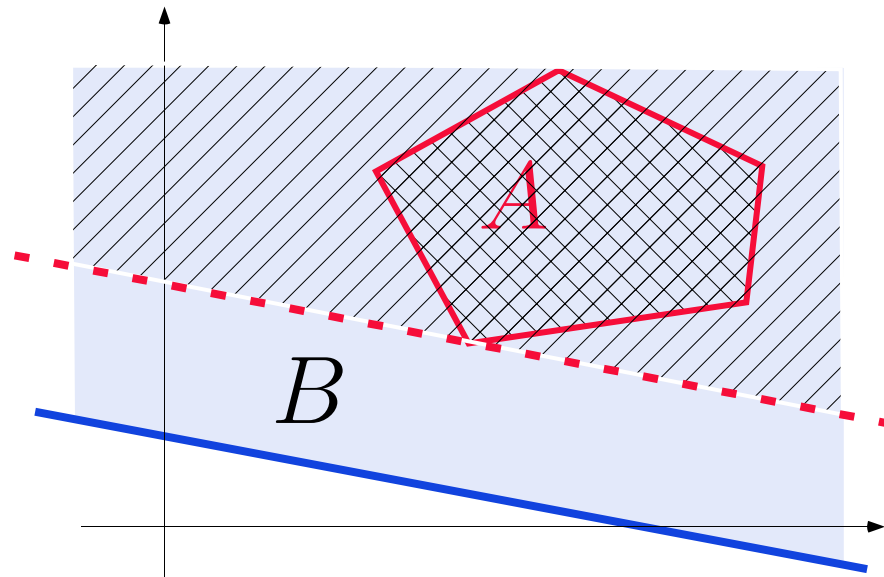
Lagrangian relaxation, equality constraints

$$\begin{aligned}
 & \forall x \in \mathbb{V} : \left(\bigwedge_{k=1}^N \sigma_k(x) = 0 \right) \Rightarrow (\sigma_0(x) \geq 0) \\
 \Leftarrow & \text{soundness (Lagrange)} \\
 & \exists \lambda \in [1, N] \mapsto \mathbb{R}^+ : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0 \\
 \wedge & \exists \lambda' \in [1, N] \mapsto \mathbb{R}^+ : \forall x \in \mathbb{V} : \sigma_0(x) + \sum_{k=1}^N \lambda'_k \sigma_k(x) \geq 0 \\
 \Leftrightarrow & \left(\lambda'' = \frac{\lambda' - \lambda}{2} \right) \\
 & \exists \lambda'' \in [1, N] \mapsto \mathbb{R} : \forall x \in \mathbb{V} : \sigma_0(x) - \sum_{k=1}^N \lambda''_k \sigma_k(x) \geq 0
 \end{aligned}$$



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)

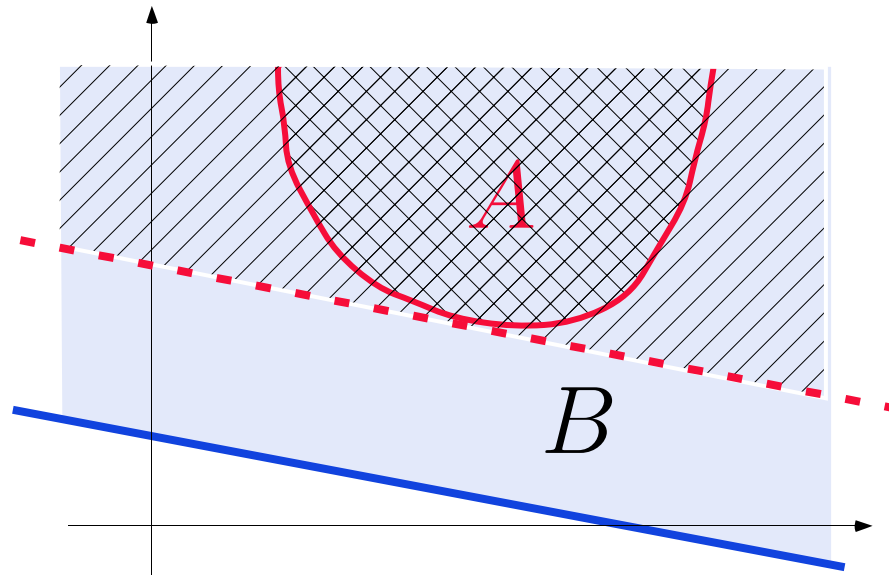
$\Rightarrow$ (completeness, Farkas)

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

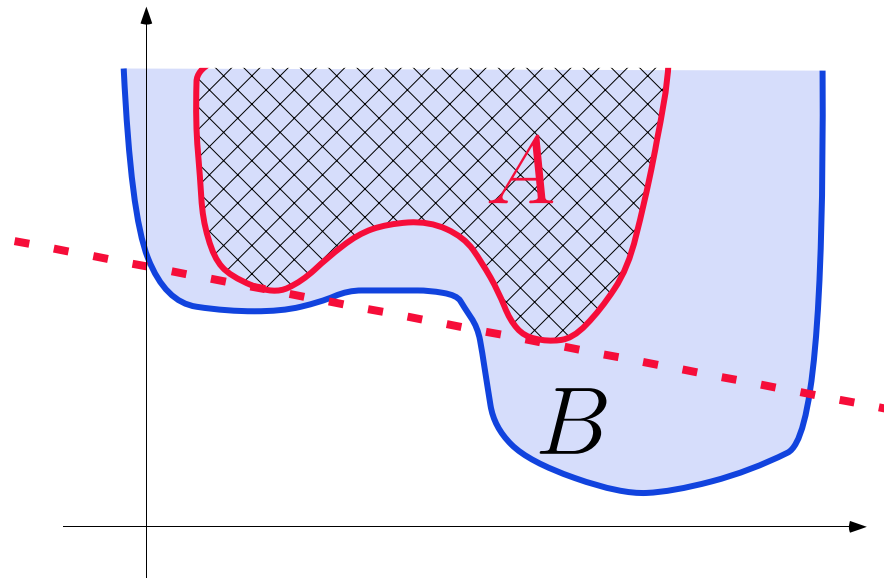


Yakubovich's S-procedure, informally

- 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 unknown 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 ?
- parametric abstraction:
 1. Fix the form r_a of the function r a priori, in term of unknown parameters a
 2. Compute the parameters a numerically
- Examples:

$$\begin{array}{ll} r_a(x) = a.x^\top & \text{linear} \\ r_a(x) = a.(x \ 1)^\top & \text{affine} \\ r_a(x) = (x \ 1).a.(x \ 1)^\top & \text{quadratic} \end{array}$$



Floyd's method for termination of while B do C

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$ -valued unknown 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

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

[Minimizing $f(x)$]

feasibility problem : find a solution to the constraints

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

Example: Linear programming

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

[Minimizing 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

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

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

Where the linear matrix inequality (LMI) is

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

with symmetric 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*:

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



Floyd's method for termination of while B do C

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$ -valued unknown 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*: $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^{+i}$:

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

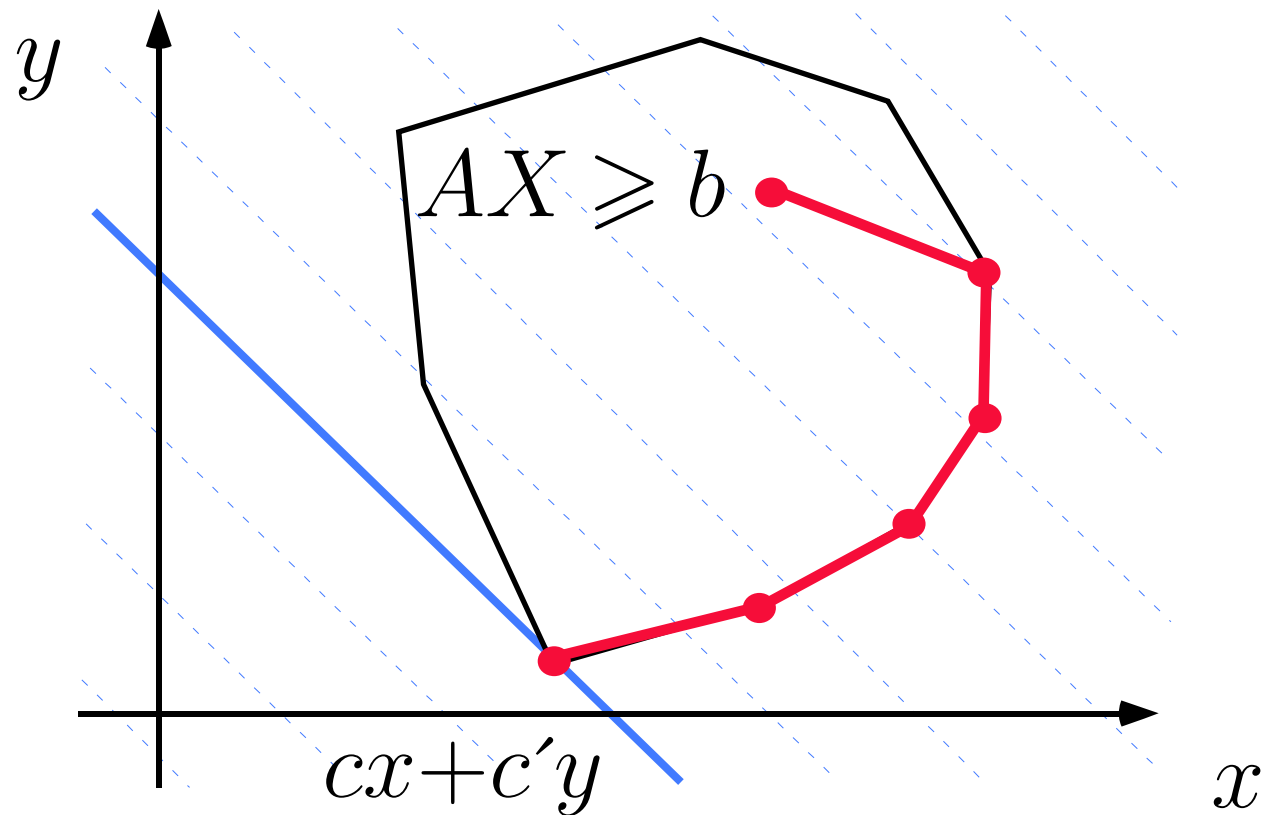


Idea 6

Solve the convex constraints by semidefinite programming



The simplex for linear programming



Dantzig 1948, exponential in worst case, good in practice



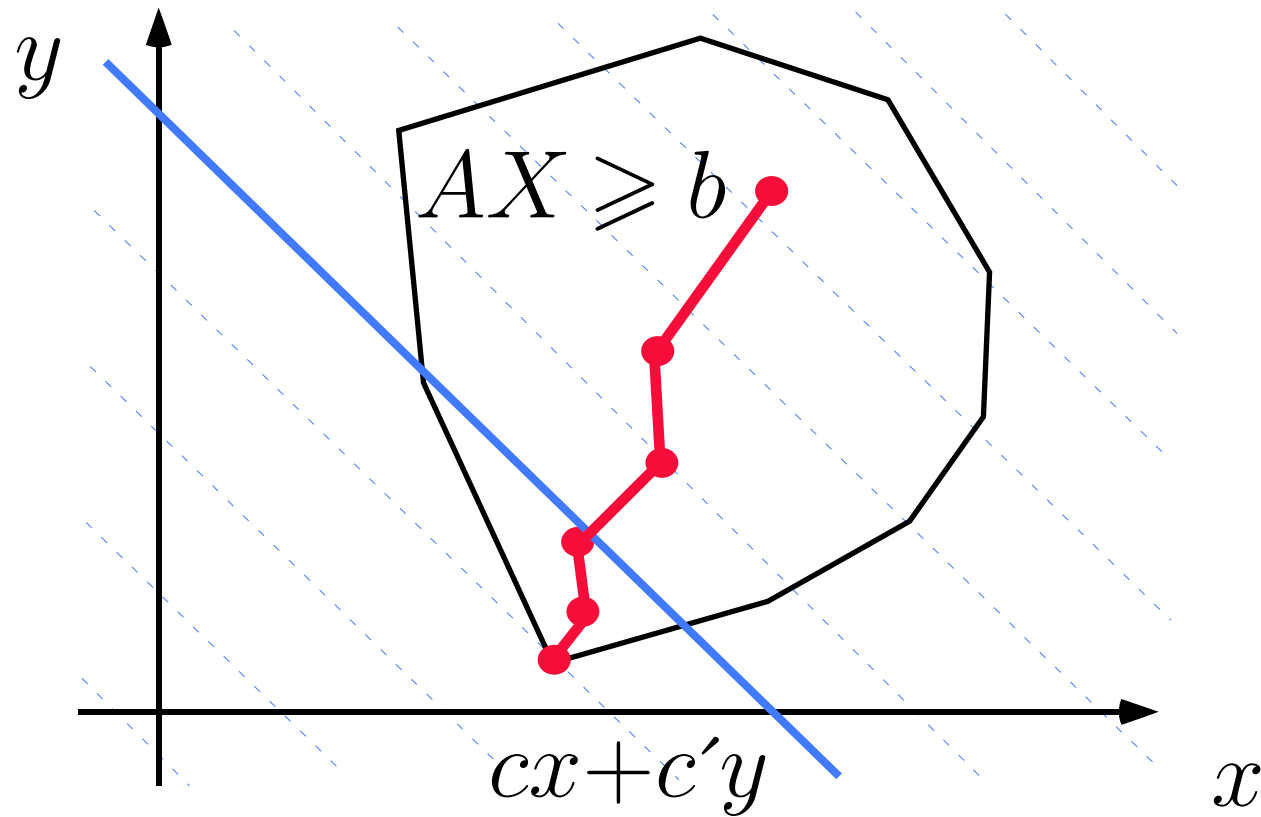
Polynomial methods

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)

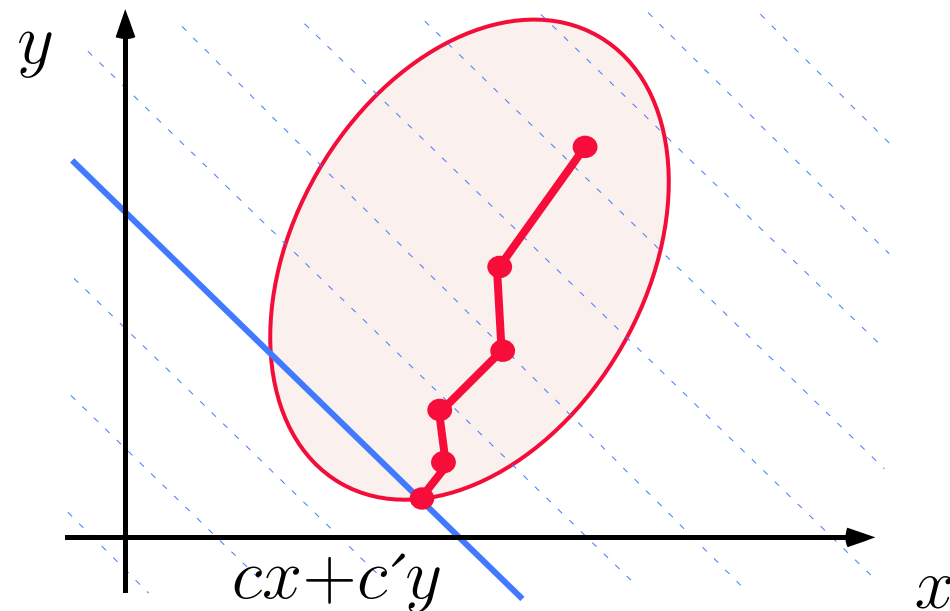


The interior point method



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”



Semidefinite programming solvers

Numerous solvers available under MATLAB[®], 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)

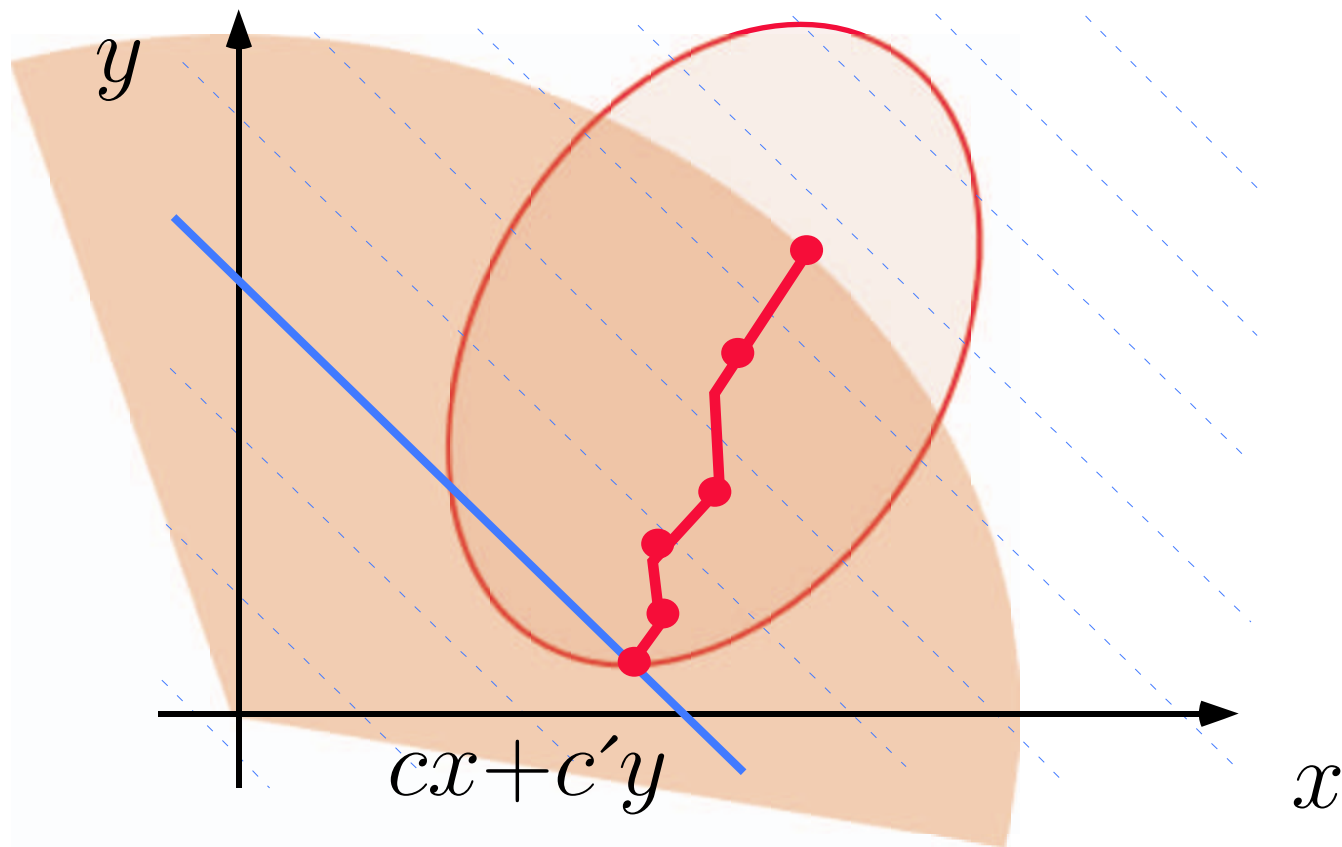
```

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

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
```



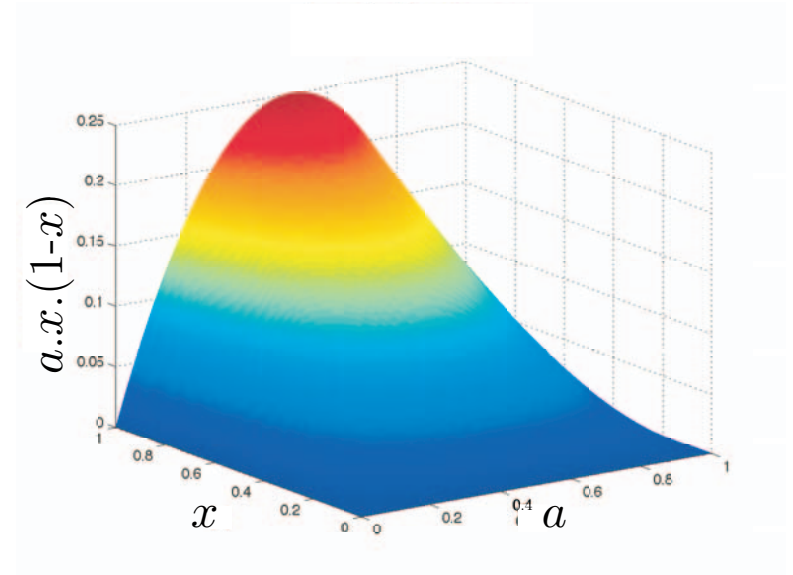
Idea 7

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:

$$r(x) = 1.222356e-13 \cdot x + 1.406392e+00$$



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
```

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

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

$\{n \geq 0\}$

$i := n; j := n;$

while ($i \neq 0$) do

 if ($j > 0$) then

$j := j - 1$

 else

$j := n; i := i - 1$

 fi

od

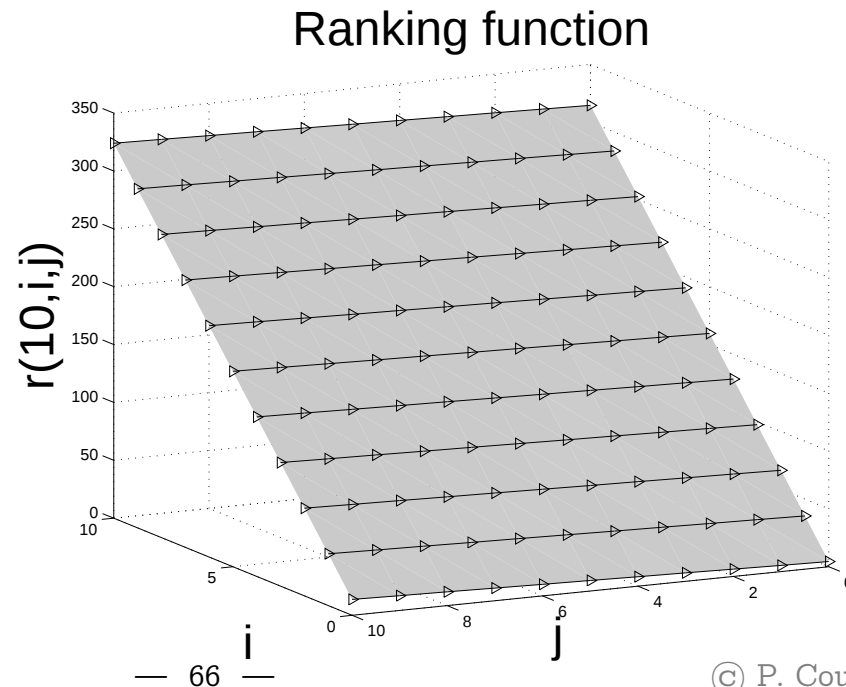
← termination precondition
determined by iterated forward/backward polyhedral analysis



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

$$\begin{aligned} r(n,i,j) = & +7.024176e-04.n^2 +4.394909e-05.n.i \dots \\ & -2.809222e-03.n.j +1.533829e-02.n \dots \\ & +1.569773e-03.i^2 +7.077127e-05.i.j \dots \\ & +3.093629e+01.i -7.021870e-04.j^2 \dots \\ & +9.940151e-01.j +4.237694e+00 \end{aligned}$$

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 scheduler



Termination of a concurrent program

<pre>[1: while [x+2 < y] do 2: [x := x + 1] od 3: 1: while [x+2 < y] do 2: [y := y - 1] od 3:]</pre>	interleaving →	<pre>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</pre>
---	------------------------------	--

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
→

$\{m \geq 1\}$ ← 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 unknown loop **invariant** I such that:

- The invariant is *initial*:

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

- The invariant is *inductive*:

$$\forall x, x' : I(x) \wedge \llbracket B; C \rrbracket(x, x') \Rightarrow I(x')$$

↑
↑

???



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 unknown invariant by **parametric abstraction**

... we get ...



Floyd's method for numerical programs

Find $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$ -valued unknown 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*: $\exists \lambda \in [0, N] \longrightarrow \mathbb{R}^+ :$

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

$\uparrow \quad \uparrow$

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 MATLAB[®]:

- [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:

$$+2.14678e-12*i - 3.12793e-10*j + 0.486712 \geq 0$$

– 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



Conclusion



Constraint resolution failure

- infeasibility of the constraints does not mean “non termination” or “non invariance” but simply **failure**
- inherent to **abstraction!**



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 ???)



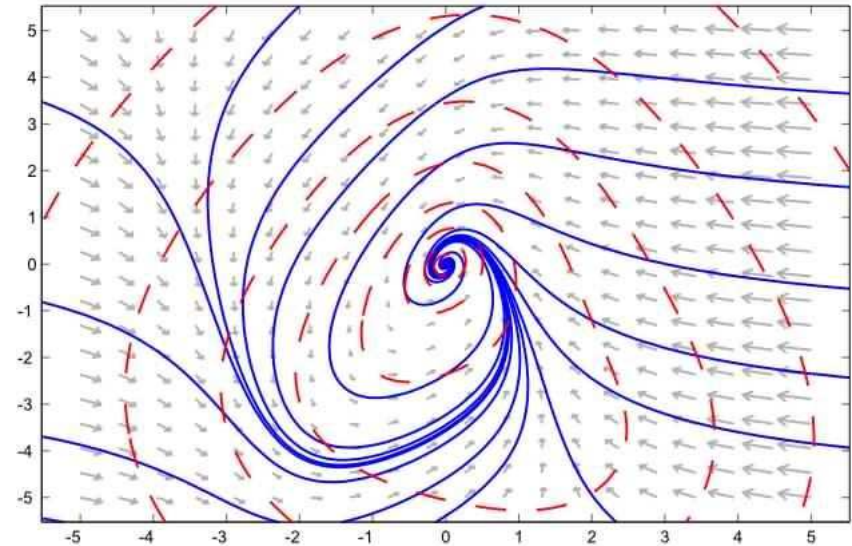
Related work

- 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



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

