An impromptu\textsuperscript{1} invited talk :-) on summer work with Radhia Cousot.

\textbf{"A Lagrangian relaxation and mathematical programming framework for static analysis and verification"}

\textbf{Patrick Cousot}
École normale supérieure
45 rue d’Ulrm
75230 Paris cedex 05, France

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

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

\textsuperscript{1} French for “extemporaneous”.
Principle of static analysis

- Define the most precise program property as a fixpoint \( \text{lfp } F \)
- Effectively compute a fixpoint approximation:
  - iteration-based fixpoint approximation
  - constraint-based fixpoint approximation

Constraint-based static analysis

- Effectively solve a postfixpoint constraint:
  \[
  \text{lfp } F = \bigcap \{ X \mid F(X) \subseteq X \}
  \]

since \( F(X) \subseteq X \) implies \( \text{lfp } F \subseteq X \)

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

---

Program properties

---

\[ X^0 = \bot \]
\[ X^\lambda = \bigcup_{\eta < \lambda} F(X^\eta) \]
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

\[
\{I(\alpha) \land \alpha > 0\} \ B; C \ {\exists \beta < \alpha : I(\beta)} , \ I(0) \Rightarrow \neg B \\
\{\exists \epsilon : I(\epsilon)\} \ \text{while} \ B \ \text{do} \ C \ \text{od} \ \{I(0)\}
\]

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

\[ \text{lfp } F \cap \text{lfp } B \]

by overapproximations of the decreasing sequence

\[ X^0 = \top \]
\[ \cdots \]
\[ X^{2n+1} = \text{lfp } \lambda Y \cdot X^{2n} \cap F(Y) \]
\[ X^{2n+2} = \text{lfp } \lambda Y \cdot X^{2n+1} \cap B(Y) \]
\[ \cdots \]

Examples (with polyhedral\(^3\) abstraction)

\[
\{x \leq 0\} \\
\text{while } (x > 0) \text{ do} \\
\quad \{\text{empty(1)}\} \\
\quad \text{skip} \\
\quad \{\text{empty(1)}\} \\
\text{od} \\
\{x \leq 0\}
\]

\[
\{n \geq 0\} \\
i := n; \\
\{n=i, n \geq 0\} \\
\text{while } (i <> 0) \text{ do} \\
\quad \{i >= 1, n > i\} \\
\quad j := 0; \\
\quad \{j=0, i > i, n = i\} \\
\quad \text{while } (j <> i) \text{ do} \\
\quad \quad \{j > 0, i > j+1, n > i\} \\
\quad \quad j := j + 1 \\
\quad \quad \{j > 1, i > j, n = i\} \\
\quad \quad \{i=j, i > i, n = i\} \\
\quad \quad i := i - 1 \\
\quad \quad \{i+1 = j, i > 0, n > i+1\} \\
\text{od} \\
\{i=0, n > 0\}
\]

Bubble-sort example

\[
\{n \geq 0\} \\
i := n; \\
\{n=i, n \geq 0\} \\
\text{while } (i <> 0) \text{ do} \\
\quad \{i >= 1, n > i\} \\
\quad j := 0; \\
\quad \{j=0, i > i, n = i\} \\
\quad \text{while } (j <> i) \text{ do} \\
\quad \quad \{j > 0, i > j+1, n > i\} \\
\quad \quad j := j + 1 \\
\quad \quad \{j > 1, i > j, n = i\} \\
\quad \quad \{i=j, i > i, n = i\} \\
\quad \quad i := i - 1 \\
\quad \quad \{i+1 = j, i > 0, n > i+1\} \\
\text{od} \\
\{i=0, n > 0\}
\]

\(^3\) using Bertand Jeannet’s NewPolka library
Arithmetic mean example

{x\geq y}
while (x <> y) do
  {x\geq y+2}
  x := x - 1;
  {x\geq y+1}
  y := y + 1
  {x\geq y}
endwhile
{x=y}

Arithmetic mean example (cont’d)

Adding a backward loop counter:

{x=y+2k,x\geq y}
while (x <> y) do
  {x=y+2k,x\geq y+2}
  k := k - 1;
  {x=y+2k+2,x\geq y+2}
  x := x - 1;
  {x=y+2k+1,x\geq y+1}
  y := y + 1
  {x=y+2k,x\geq y}
endwhile
{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
- \(\llbracket B; C \rrbracket (x, x')\): small-step relational semantics of one iteration of the loop body
- \(\llbracket B; C \rrbracket (x, x') = \bigwedge_{t=1}^{N} \sigma_t(x, x') \geq 0\) (where \(\geq\) is \(>, \geq\) or \(=\))
- not a restriction for numerical programs
Example of linear program (Arithmetic mean)

\[
[AA'][\begin{array}{c}x \\ x'\end{array}]^T \geq b
\]

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

\[
\begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 \\
1 & -1 & -2 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 \\
-1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
x \\ y \\ k \\ x' \\ y' \\ k' \end{bmatrix}
\geq
\begin{bmatrix}
1 \\
0 \\
-1 \\
-1 \\
-1 \\
\end{bmatrix}
\]

Example of quadratic form program (factorial)

\[
[xx']A[xx']^T + 2[xx']q + r \geq 0
\]

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

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

Example of semialgebraic program (logistic map)

\[\epsilon = 1.0e-9;\]
while (0 <= a) & (a <= 1 - \epsilon) & (\epsilon <= 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 unknown loop invariant $I$ such that:

- The invariant is **initial**:
  $$\forall x : P(x) \Rightarrow I(x)$$

- The invariant is **inductive**:
  $$\forall x, x' : I(x) \land [B; C](x, x') \Rightarrow I(x')$$

Floyd’s method for numerical programs

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

- The invariant is **initial**:
  $$\forall x : P(x) \geq 0 \Rightarrow I(x) \geq 0$$

- The invariant is **inductive**:
  $$\forall x, x' : I(x) \geq 0 \land \bigwedge_{i=1}^{N} \sigma_i(x, x') \geq 0 \Rightarrow I(x') \geq 0$$

Floyd’s method for termination

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

- The rank is **nonnegative**:
  $$\forall x : I(x) \Rightarrow r(x) \geq 0$$

- The invariant is **inductive**:
  $$\forall x, x' : I(x) \land [B; C](x, x') \Rightarrow r(x') \leq r(x) - \eta$$

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

Solving the constraints

- Fix the form of the unknown $(I(x) \geq 0/r(x) \geq 0)$ using parameters $a$ in the form $Q(a, x) \geq 0$.
- The problem has the form:
  $$\exists a : \left( \bigwedge_{k=1}^{n} \forall x, x' : Q(a, x) \geq 0 \land C_k(x, x') \geq 0 \right) \Rightarrow Q(a, x') \geq 0$$
- Find an algorithm to effectively compute $a$!
Problems

In order to compute $a$:

- How to get rid of the implication $\Rightarrow$?
  $\rightarrow$ Lagrangian relaxation
- How to get rid of the universal quantification $\forall$?
  $\rightarrow$ 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

```math
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 * x0 + e >= 0) && 
                        (c * x0^2 + d * x0 - c * x1^2 - d + x1 >= η))]),],] /. 
    Reals]] // TraditionalForm
```

```
Out[1]= {0.16 Second, c > 0 \land d > 0}
```

Example: linear termination of logistic map

```math
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))]),],] /. 
    Reals]]//TraditionalForm
```

```
Out[1]= {0.16 Second, c > 0 \land d > 0}
```

Scaling up

- does not scale up beyond a few variables!
- too bad!

No result after hours of computations!
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 \( A \neq \emptyset \))
\[ \Leftarrow \text{(soundness)} \]
\[ \Rightarrow \text{(completeness)} \]
\[ \text{border of } A \text{ parallel to border of } B \]

### Lagrangian relaxation (linear case)
Lagrangian relaxation, formally

Let $\mathbb{V}$ be a finite dimensional linear vector space, $N > 0$ and $\forall k \in [1, N] : \sigma_k \in \mathbb{V} \mapsto \mathbb{R}$.

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

$\leftarrow$ soundness (Lagrange)

$\Rightarrow$ completeness (lossless)

$\not\Rightarrow$ ncompleteness (lossy)

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

relaxation = approximation, $\lambda_i = $ Lagrange coefficients

Lagrangian relaxation, equality constraints

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

$\leftarrow$ soundness (Lagrange)

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

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

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

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

Example: affine Farkas’ lemma, formally

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

- 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

Yakubovich’s S-procedure, completeness cases

- The constraint $\sigma(x) \geq 0$ is regular if and only if $\exists \xi \in \forall \sigma(\xi) > 0$.
- 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 ch)

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

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

[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**: \( \{ x | \bigwedge_{i=1}^{N} g_i(x) \geq 0 \} \)
- a feasibility problem can be converted into the optimization program

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

[Minimizing \( cx \)]

**Example: linear programming**

\[ \exists x \in \mathbb{R}^n : \quad Ax \geq b \]

[Minimizing \( cx \)]
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)

**Example: semidefinite programming**
Semidefinite programming

\[ \exists x \in \mathbb{R}^n: \quad M(x) \succeq 0 \]

[Minimizing \( cx \)]

Where the linear matrix inequality is

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

with symmetric matrices \( M_k = M_k^\top \) and the positive semidefiniteness is

\[ M(x) \succeq 0 = \forall X \in \mathbb{R}^N : X^\top M(x) X \geq 0 \]

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 \( (A_{k;i}; A'_{k;i})(x \ x')^\top + b_k = (x \ x' 1)M_k(x \ x' 1)^\top \) where

\[ M_k = \begin{bmatrix} 0_{(2n \times 2n)} & \frac{A_k^\top}{2} \\ \frac{A_k}{2} & \frac{A'_k}{2} & b_k \end{bmatrix} \]
\[ [x \ x'] M_k [x \ x']^\top \]
\[ = (x' 1) \begin{bmatrix} 0_{(2n \times 2n)} & \frac{A_{k, i}}{2} & \frac{A'_{k, i}}{2} & b_k \\ \frac{A_{k, i}}{2} & \frac{A'_{k, i}}{2} & 0 & 0 \\ \frac{A_{k, i}}{2} & 0 & \frac{A'_{k, i}}{2} & 0 \\ 0 & b_k & 0 & 1 \end{bmatrix} \begin{bmatrix} x^\top \\ x'^\top \\ x \\ 1 \end{bmatrix} \]
\[ = (x' 1) \begin{bmatrix} \frac{A_{k, i}}{2} & \frac{A'_{k, i}}{2} \\ \frac{A_{k, i}}{2} & \frac{A'_{k, i}}{2} \\ \frac{A_{k, i}}{2} & 0 \\ 0 \\ b_k \end{bmatrix} + \frac{A_{k, i}}{2} x^\top x' + \frac{A'_{k, i}}{2} x'^\top x + b_k \]
\[ = (A_{k, i} A'_{k, i}) (x x')^\top + b_k \]
\[ \text{since } (AB)^\top = B^\top A^\top \]

**Example of quadratic forms: quadratic inequalities**

\[(x x') P_k (x x')^\top + 2 q_k^\top (x x')^\top + r_k \geq 0 \]
\[= (x' 1) M_k (x x' : 1)^\top \]

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 \(\geq\) \(\gg\)
Semidefinite programming solvers

Numerous solvers available under MATLAB®, a.o.:
- **lmlab**: 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 needs some help (feasibility radius, shift, …)

Main application: nonlinear automatic control theory

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

```matlab
> x = sdpvar(1,1);
> F = set(diag([x -x])>0);
> solvesdp(F, [], sdpsettings('solver', 'lmlab'))
... ans = ...
    info: 'Infeasible problem (LMILAB)'
... > t = sdpvar(1,1);
> solvesdp(F, -t, sdpsettings('solver', 'lmlab', '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(:,:,1,:,:)]=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 = 2
> M = 2
> format rational; Mk
Mk(:,:,1,:) =
 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 0 0
 1/2 0 0 0 -1 0 1/2 0 0
Mk(:,:,2,:) =
 0 0 0 0 0 0 0 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
Mk(:,:,3,:) =
 0 0 0 0 0 0 0 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
Mk(:,:,4,:) =
 0 0 0 0 0 0 0 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
```

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(:,:,1,:,:)]=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'})
```

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

```
Iterated forward/backward polyhedral analysis:
{y >= 1}
while (x >= 1) do
  x := x - y
od
```

```
iteration (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],[1; 0; 0; 0]);
[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);
intrank(R, {'x' 'y'});
...

termination (sedumi)
\end{verbatim}

Iterated forward/backward polyhedral analysis:

\begin{verbatim}
{y >= 1}
while (x >= 1) do
  x := x - y
od
\end{verbatim}

\begin{equation}
r(x, y) = +2.271450e+03.x +1.810903e+03.y -3.623997e+03
\end{equation}

Enforcing an integer ranking function:

\begin{verbatim}
clear all;
[N Mk(:,:,:)] = linToMk([1 0; 0 1],...
[0 0; 0 0],[1; -1],[1; 0; 0; 0]);
[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'});
...

termination (bnb)
\end{verbatim}

\begin{equation}
r(x, y) = +2.x +2.y -3
\end{equation}

Linear example: termination of arithmetic mean

\begin{verbatim}
> clear all;
% linear inequalities
% x0 y0 k0
A1 = [ 1 -1 0; % x0 - y0 - 1 >= 0
 1 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];
be = [0; 1; 1; 1];

display_Mk(Mk, N,{'x' 'y' 'k'});
...

> [diagnostic, R] = termination(Mk, N, 'integer', 'linear');
disp(diagnostic);
intrank(R, {'x' 'y' 'k'});

r(x,y,k) = +1.382113e+03.x -1.382113e+03.y +4.978695e+03.k
\end{verbatim}

Iterated forward/backward polyhedral analysis:

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

\begin{equation}
r(x, y, k) = +1.382113e+03.x -1.382113e+03.y +4.978695e+03.k +2.711732e+03
\end{equation}
Linear example: termination of Euclidean division

```matlab
> clear all
% linear inequalities
y 0 q 0 r
Ai = [ 0 0 0; 0 0 0; 0 0 0];
y q r
Ai_ = [ 1 0 0; 0 1 0; 0 0 1]; r 0 0
bi = [-1; -1; 0];
% linear equalities
y 0 q 0 r
Ae = [ 0 -1 0; -q +1 0; -r +1 0]
y q r
Ae_ = [ 0 0 1; 1 0 0; 1 0 1];
be = [-1; 0; 0];
[n Mk(:,:,:)] = linToMk(Ai, Ai_, bi);
[m Mk(:,:,N+1:M+M)] = linToMk(Ae, Ae_, be);
```

Iterated forward/backward polyhedral analysis:

```matlab
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}
end {y - r - 1 >= 0}
```

```matlab
[n Mk(:,:,:)] = linToMk(Ai, Ai_, bi);
```

```matlab
[m Mk(:,:,N+1:M+M)] = linToMk(Ae, Ae_, be);
```

Floyd’s proposal

```
r(y, q, r) = -2.0 y + 2.0 q + 4.0 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

```matlab
> clear all
A1 = [ 0 -1 1; 0 0 0; 0 0 0] % inequality constraints
A1_ = [ 1 0 0; 0 1 0; 0 0 1] % equality constraints
bi = [0; 0; -1]
[n Mk(:,:,:)] = linToMk(A1, A1_, bi);
```

Iterated forward/backward polyhedral analysis:

```matlab
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}
end {y - r - 1 >= 0}
```

```matlab
[n Mk(:,:,:)] = linToMk(A1, A1_, bi);
```

```matlab
[m Mk(:,:,N+1:M+M)] = linToMk(Ae, Ae_, be);
```

```
P(:,:,1) = [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; 0 0 0 0 0 0]
```

```matlab
q(:,1) = [0; 0; 0; 0; 1/2; 0]
r(:,1) = 0
```

```
[n Mk(:,:,:)] = linToMk(A1, A1_, bi);
```

```matlab
[m Mk(:,:,N+1:M+M)] = linToMk(Ae, Ae_, be);
```

```
P(:,:,1) = [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; 0 0 0 0 0 0]
```

```
r(1,:) = [0; 0; 0; 1/2; 0]
```

```
r(y, q, r) = -2.0 y + 2.0 q + 4.0 r
```

```
r(y, q, r) = -2.0 y + 2.0 q + 4.0 r
```

Floyd’s proposal \( r(x, y, q, r) = x - q \) is more intuitive but requires to discover the nonlinear loop invariant \( x = r + qy \).
Lagrangian relaxation and semidefinite programming for static analysis

(2) Foundations

Main steps in a typical soundness/completeness proof

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

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

\( \iff \{ \text{Lagrangian relaxation (\( \iff \) if lossless)} \} \)

\( \exists r : \exists \lambda \in [1, N] \mapsto \mathbb{R}^*_+ : \forall x, x' \in \mathbb{D}^n : r(x, x') - \sum_{k=1}^{N} \lambda_k(x x' 1)M_k(x x' 1)^T \geq 0 \)

\( \iff \{ \text{Choose form of } r(x, x') = (x x' 1)M_0(x x' 1)^T \} \)

\( \exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}^*_+ : \forall x, x' \in \mathbb{D}^n : (x x' 1)M_0(x x' 1)^T - \sum_{k=1}^{N} \lambda_k(x x' 1)M_k(x x' 1)^T \geq 0 \)

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

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

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

\( \iff \{ \text{LMI solver provides } M_0 (\text{and } \lambda) \} \)

Example: LMI constraints for decrementation
Iterated forward/backward polyhedral analysis:

\[ \text{Mk}(\cdot, : , 1) \]
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 & -1 \\
\end{bmatrix}
\]
\[ \text{Mk}(\cdot, : , 2) = \]
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1/2 & 0 & 0 & -1 \\
\end{bmatrix}
\]
\[ \text{Mk}(\cdot, : , 3) = \]
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & -1/2 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
-1/2 & 1/2 & 0 & 0 & 0 \\
\end{bmatrix}
\]
\[ \text{Mk}(\cdot, : , 4) = \]
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
-1/2 & 0 & 1/2 & 0 & 0 \\
\end{bmatrix}
\]

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

in matrix form
\[
X = \begin{bmatrix}
0 & 0 & c_1 \\
0 & 0 & c_2 \\
\end{bmatrix}
\]

Iterated forward/backward polyhedral analysis:

\[ \{ y >= 1 \} \]

while \( x >= 1 \) do
  \[ x := x - y \]
  od

The semidefinite constraints are

\[
M_0 = \begin{bmatrix}
X(1:n,1:n) & \text{zeros}(n,n) & X(1:n,n+1); \\
\text{zeros}(n,n) & \text{zeros}(n,n) & \text{zeros}(n,1); \\
\text{zeros}(n,n) & \text{zeros}(n,n) & \text{zeros}(n+1,n+1); \\
\text{zeros}(2*n,2*n) & \text{zeros}(2*n,1); \\
\text{zeros}(1,2*n) & 1
\end{bmatrix}
\]

\[
M_0-M_0-one-l_{(1,1)}>0 \\
l_{(1,1)}>0 \\
l_{(2,1)}>0 \\
\]

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 outer loop

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

```plaintext
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=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)
```

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

Example of termination of nested loops: Bubblesort inner loop

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

```plaintext
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)
```

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

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

... 

```plaintext
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
```

termination (lmilab)

```plaintext
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

```plaintext
Semialgebraic example: logistic map
```

```plaintext
> 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);
```
Semidefinite programming relaxation for polynomial quantifier elimination

(2) Foundations

Principle

- Show $\forall x : p(x) \geq 0$ by $\forall x : p(x) = \sum_{i=1}^{k} q_i(x)^2$
- Hilbert’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: \( p(x, y, \ldots) = z^T Q z \) where \( Q \succ 0 \) 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 \( p(x, y, \ldots) = z^T Q z \) yields LMI contrains on the unknown \( Q: z^T M(Q) z \succ 0 \)
- 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 \( Q \succ 0 \), check for semidefinite positiveness \( \exists Q: \forall X : X^T M(Q) X \succeq 0 \) i.e. \( \exists Q: M(Q) \succ 0 \) with LMI solver
- Implement with SOSTools under MATLAB\(^a\) 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

\(^4\) Since \( Q \succ 0 \), \( Q \) has a Cholesky decomposition \( L \) which is an upper triangular matrix \( L \) such that \( Q = L^T L \). It follows that \( p(x) = z^T Q z = z^T L^T L z = (Lx)^T (Lx) = \sum_i x_i y_i \) (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 \( x \).

Data structures

- Use norms (size, height, \ldots) 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
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)

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”.

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