Overview of the Termination Analysis Method

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

The main point in this talk is (4).
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

Example: partial correctness (must stay into safe states)

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

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

by overapproximations of the decreasing sequence

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

Arithmetic Mean Example:

Termination Precondition (1)

\{x>y\}
while (x <> y) do
{\{x>y+2\}}
\begin{align*}
x & := x - 1; \\
y & := y + 1
\end{align*}
\{x>y\}
\text{od}
\{x=y\}

Idea 1

The auxiliary termination counter method
Arithmetic Mean Example: Termination Precondition (2)

{\(x=y+2k,x\geq y\)}
while (\(x \neq y\)) 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
{\(x=y+2k,x\geq y\)}
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: Body Relational Semantics

Case $x < y$:
assume $(x=y+2k) \land (x\geq y+2)$
{x=y+2k, x\geq y+2}
assume $(x < y)$
empty(6)
assume $(x0=x) \land (y0=y) \land (k0=2k)$
empty(6)
k := k - 1;
x := x - 1;
y := y + 1
empty(6)

Case $x > y$:
assume $(x=y+2k) \land (x\geq y+2)$
{x=y+2k, x\geq y+2}
assume $(x > y)$
{x=y+2k0, y=y0+1, x+1=x0, x=y+2k, x\geq y+2}
k := k - 1;
x := x - 1;
y := y + 1
empty(6)

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 $\mathbb{R/Q/Z}$-valued unknown rank function $r$ such that:

- The rank is nonnegative:
  $$ \forall x, x : I(x0) \land [B;C](x0, x) \Rightarrow r(x0) \geq 0 $$

- The rank is strictly decreasing:
  $$ \forall x, x : I(x0) \land [B;C](x0, x) \Rightarrow r(x) \leq r(x0) - \eta $$

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

```
% clear all:
[v0,v] = variables('x','y','k')
% linear inequalities
% x0 y0 k0
A1 = [ 0 0 0 ];
% x y k
A1_ = [ 1 -1 0 ]; % x0 - y0 => 0
b1 = [0];
[Mk(:,:,N+1:N+M)=linToMk(A1,A1_,b1);
% 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
b1 = [2 -1 1];
[Mk(:,:,N+1:N+M)=linToMk(Ae,Ae_,b1);
```

Arithmetic Mean Example: Ranking Function

Input the loop abstract semantics
Display the abstract semantics of the loop while \( B \) do \( C \)

- compute ranking function, if any

\[
\begin{align*}
+1.x - 1.y &\geq 0 \\
-2.k0 +1.x - 1.y +2 &\geq 0 \\
-1.y0 +1.y - 1 &\geq 0 \\
-1.x0 +1.x +1 &\geq 0 \\
+1.x - 1.y - 2.k &\geq 0
\end{align*}
\]

\[ [\text{diagnostic}, R] = \text{termination}(v0, v, Mk, N, \text{'integer'}, \text{'linear'}); \]
\[ \text{disp(diagnostic)} \]
\[ \text{feasible (bnb)} \]
\[ \text{intrank}(R, v) \]

\[ r(x, y, k) = +4.k - 2 \]
Example of linear program (Arithmetic mean)

\[ AA^T [x_0 x]^T \geq b \]

\{x = y + 2k, x > y\}

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

\frac{f}{n} \cdot f := n \cdot f
\frac{n}{k} := n + 1
\frac{x}{y} := x - 1
\frac{k}{y} := k - 1

Example of quadratic form program (factorial)

\[ \frac{f}{n} \cdot f := n \cdot f \]
\frac{n}{f} := n + 1
\frac{x}{y} := x - 1
\frac{k}{y} := k - 1

Example of semialgebraic program (logistic map)

\[ \text{eps} = 1.0e-9; \]

while (0 <= a) & (a <= 1 - \text{eps})
  & (\text{eps} <= x) & (x <= 1) do
  x := a*x*(1-x)
od

Floyd’s method for termination of while B do C

Find an \( \mathbb{R}/\mathbb{Q}/\mathbb{Z} \)-valued unkown \( r \) 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 \( \land \) and implication \( \Rightarrow \) by Lagrangian relaxation

Implication (general case)

\[
A \Rightarrow B \\
\Leftrightarrow \\
\forall x \in A : x \in B
\]

Implication (linear case)

\[
A \Rightarrow B \quad \text{(assuming } A \neq 0)\\n\Leftrightarrow \text{(soundness)}\\n\Rightarrow \text{(completeness)}\\n\text{border of } A \text{ parallel to border of } B
\]

Lagrangian relaxation (linear case)
Lagrangian relaxation, formally

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

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

- soundness (Lagrange)
- completeness (lossless)
- incompleteness (lossy)

Example: affine Farkas’ lemma, formally

- Formally, if the system $Ax + b \geq 0$ is feasible then
  $$\forall x : Ax + b \geq 0 \Rightarrow cx + d \geq 0$$
  $$\Leftrightarrow (\text{soundness, Lagrange})$$
  $$\Rightarrow (\text{completeness, Farkas})$$
  $$\exists \lambda \geq 0 : \forall x : cx + d - \lambda(Ax + b) \geq 0.$$
Yakubovich’s S-procedure, informally

- An application of Lagrangian relaxation to the case when \( A \) is a quadratic form

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:
  \[
  \forall x \in \mathbb{R}^n : x^\top P_1 x + 2q_1^\top x + r_1 \geq 0 \implies x^\top P_0 x + 2q_0^\top x + r_0 \geq 0
  \]
  \[
  \iff \quad (\text{Lagrange})
  \]
  \[
  \Rightarrow \quad (\text{Yakubovich})
  \]
  \[
  \exists \lambda \geq 0 : \forall x \in \mathbb{R}^n : x^\top \begin{bmatrix} P_0 & q_0 \\ q_0^\top & r_0 \end{bmatrix} - \lambda \begin{bmatrix} P_1 & q_1 \\ q_1^\top & r_1 \end{bmatrix} x \geq 0.
  \]

Floyd’s method for termination of \( \textbf{while} \ B \ \textbf{do} \ C \)

Find an \( \mathbb{R}/\mathbb{Q}/\mathbb{Z} \)-valued unknown rank function \( r \) which is:

- Nonnegative: \( \exists \lambda \in [1, N] \mapsto \mathbb{R}^+ : \)
  \[
  \forall x_0, x : r(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0
  \]
- Strictly decreasing: \( \exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^+ : \)
  \[
  \forall x_0, x : (r(x_0) - r(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0
  \]
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 terms of unknown parameters $a$
  2. Compute the parameters $a$ numerically
- Examples:
  
  $r_a(x) = a.x^\top$ \hspace{1cm} linear
  $r_a(x) = a.(x\ 1)^\top$ \hspace{1cm} affine
  $r_a(x) = (x\ 1).a.(x\ 1)^\top$ \hspace{1cm} quadratic

Floyd’s method for termination of while B do C

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

- **Nonnegative:** $\exists \lambda \in [1, N] \mapsto \mathbb{R}^+:$
  
  $\forall \ x_0, x : r_a(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0$

- **Strictly decreasing:** $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^+:$
  
  $\forall \ x_0, x : (r_a(x_0) - r_a(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0$

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

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

[Minimizing \( cx \)]

---

Semidefinite programming

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

[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

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

---

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, 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 formulae we are interested in for programs which semantics can be expressed as **LMIs**:

\[ \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \cong_i 0 = \bigwedge_{i=1}^{N} \left( x_0 x 1 \right) M_i(x_0 x 1)^\top \cong_i 0 \]
Floyd’s method for termination of \( \text{while } B \land C \)

Find \( \mathbb{R}/\mathbb{Q}/\mathbb{Z} \)-valued unknown parameters \( a \), such that:

- **Nonnegative**: \( \exists \lambda \in [1, N] \mapsto \mathbb{R}^+ \):
  \[
  \forall x_0, x : r_a(x_0) - \sum_{i=1}^{N} \lambda_i(x_0, x) M_i(x_0, x)^\top \geq 0
  \]

- **Strictly decreasing**: \( \exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^+ \):
  \[
  \forall x_0, x : (r_a(x_0) - r_a(x) - \eta) - \sum_{i=1}^{N} \lambda'_i(x_0, x) M_i(x_0, x)^\top \geq 0
  \]

The simplex for linear programming

\[
\begin{align*}
\text{minimize} & \quad c x + c'y \\
\text{subject to} & \quad A x \geq b \\
& \quad c x + c'y \\
& \quad x \geq 0, c \geq 0
\end{align*}
\]

Dantzig 1948, exponential in worst case, good in practice

---

**Idea 6**

Solve the convex constraints by semidefinite programming

---

**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

\[ AX \geq b \]

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

```matlab
% 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 \geq 0
 0 1 0; % q - 1 \geq 0
 0 0 1]; % r \geq 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\geq1\} 
q := 0;
\{q=0,y\geq1\} 
r := x;
\{x=r,q=0,y\geq1\} 
while (y \leq r) do
\{y<r,q\geq0\} 
r := r - y;
\{r>0,q\geq0\} 
q := q + 1
\{r>0,q\geq1\} 
end
\{q\geq0,y\geq r+1\}
```

© P. Cousot
P. Cousot
Semidefinite programming relaxation for polynomial programs

\[ \text{eps} = 1.0 \times 10^{-9}; \]
\[ \text{while} (0 \leq a) \land (a \leq 1 - \text{eps}) \land (\text{eps} \leq x) \land (x \leq 1) \text{ do} \]
\[ x := a \times x \times (1 - x) \]
\[ \text{od} \]

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

SOSTool+SeDuMi:

\[ r(x) = 1.222356 \times 10^{-13} \times x + 1.406392 \times 10^{00} \]

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

\[ \text{while} (x < y) \text{ do} \]
\[ \text{if} (i \geq 0) \text{ then} \]
\[ x := x + i + 1 \]
\[ \text{else} \]
\[ y := y + i \]
\[ \text{fi} \]
\[ \text{od} \]

\[ \text{lmilab:} \]
\[ r(i,x,y) = -2.252791 \times 10^{-09} \times i - 4.355697 \times 10^{07} \times x + 4.355697 \times 10^{07} \times y + 5.502903 \times 10^{08} \]

Considering More General Forms of Programs
Quadratic termination of linear loop

\{n\geq 0\}
i := n; j := n;
while (i <> 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):

\[ r(n,i,j) = +7.024176e-04.n^2 +4.394909e-05.n.i +3.093629e+01.i -7.021870e-04.j^2 +9.940151e-01.j +4.237694e+00 \]

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

Handling nested loops

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

Example of termination of nested loops: Bubblesort inner loop

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

\begin{align*}
+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 \\
\end{align*}

termination (lmlab)

\[ r(n0,n,i,j) = +434297566.n0 +226687644.n -72551842.i -2.j +2147483647 \]
Example of termination of nested loops:

Bubblesort outer loop

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

```
assume (n0=m & i>=0 & n>=i & i <> 0);
{n0=i>=0,n0>=i}
```

```
assume (n01=n0 & n1=n & i1=i & j1=j);
{j1=j,i=i1,n0=n1,n01=n01,n0>=i0>=i}
```

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

```
{+1.i' +1 >= 0
+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
...}
```

```
termination (lmilab)
```

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

Termination of a concurrent program

```
while (x+2 < y) do
  x := x + 1
od
```

```
while (x+2 < y) do
  y := y - 1
```

```
interleaving
```

```
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
```

```
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>=1}
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 := ?;
  else
    if (t = 1) then
      x := x - 1
    else
      y := y - 1
    fi;
    fi
  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
```

Handling nondeterminacy

- By **case analysis**
- Same for **concurrency** by **interleaving**
- Same with **fairness** by nondeterministic interleaving with encoding of an explicit **scheduler**
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:
  \[ \forall x, x' : I(x) \land [B; C](x, x') \Rightarrow I(x') \]

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

\[ \text{... we get ...} \]

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: \( \exists \lambda \in [0, N] \rightarrow \mathbb{R}^+ : \)
  \[ \forall x, x' : I_a(x') - \lambda_0.I_a(x) - \sum_{k=1}^{N} \lambda_k.\sigma_k(x, x') \geq 0 \]
  \[ \text{bilinear in } \lambda_0 \text{ and } a \]
Idea 8

Solve the bilinear matrix inequality (BMI) by semidefinite programming

Bilinear matrix inequality (BMI) solvers

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

[Minimizing \( x^T Q x + cx \)]

Two solvers available under Mathlab*:
- PenBMI: M. Kočvara, M. Stingl
- bmibnb: J. Löfberg

Common interfaces to these solvers:
- Yalmip: J. Löfberg

Example: linear invariant

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

Invariant:
\[ +2.14678e-12 \cdot i - 3.12793e-10 \cdot j + 0.486712 \geq 0 \]

- 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