« Program Termination Proof by Parametric Abstraction and Semi-definite Programming »

Patrick Cousot
École normale supérieure, Paris
cousot@ens.fr  www.di.ens.fr/~cousot

Seminar, Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, September 6, 2006

Reference

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
### Iteration-based static analysis

- Effectively overapproximate the iterative fixpoint definition:\(^1\)

\[
\text{lfp } F = \bigcup_{\lambda \in \mathbb{O}} X^\lambda
\]

\[
X^0 = \perp
\]

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

---

\(^1\) under Tarski’s fixpoint theorem hypotheses

### 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 \( \text{lfp } F \subseteq X \)

- Sometimes, the constraint resolution algorithm is nothing but the iterative computation of \( \text{lfp } F \)\(^2\)

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

---


### Parametric abstraction

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

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

### Fixpoint versus Constraint-based Approach for Termination Analysis

1. Termination can be expressed in fixpoint form\(^3\)

2. However we know no effective fixpoint underapproximation method needed to overestimation the termination rank

3. So we consider a constraint-based approach abstracting Floyd’s ranking function method

---

Overview of the Termination Analysis Method

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

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)

Forward/reachability properties

Example: total correctness (must reach 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

\[ X^0 = \top \]

\[ \ldots \]

\[ X^{2n+1} = \text{lfp } \lambda Y . X^{2n} \cap F(Y) \]

\[ X^{2n+2} = \text{lfp } \lambda Y . X^{2n+1} \cap B(Y) \]

\[ \ldots \]

Idea 1

The auxiliary termination counter method

Arithmetic Mean Example: Termination Precondition (1)

\[ \{x > y\} \]

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

Arithmetic Mean Example: Termination Precondition (2)

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

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

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+2k) & (x>y));
{x=x+y+2k,x>y+y2};
while (x <> y) do
  {x=x+y+2k,x>=y+2k};
k := k - 1;
{x=x+y+2k+1,x>=y+2k+1};
x := x - 1;
y := y + 1
{x=x+2k,x>y};
{y=x=y+2k};
{y=x=y+y2};
```

Arithmetic Mean Example: Body Relational Semantics

```
Case x < y:
  assume (x=y+2k) & (x>y+2);
  {x=x+y+2k,x>y+y2};
  assume (x < y);
  empty(6);
Case x > y:
  assume (x=y+2k) & (x>y+2);
  {x=x+y+2k,x>y+y2};
  assume (x > y);
  empty(6);
  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 a $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unknown rank function $r$ such that:

- The rank is nonnegative:
  \[ \forall x_0, x : I(x_0) \land [B; C](x_0, x) \Rightarrow r(x_0) \geq 0 \]

- The rank is strictly decreasing:
  \[ \forall x_0, x : I(x_0) \land [B; C](x_0, x) \Rightarrow r(x) \leq r(x_0) - \eta \]

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

Problems

- How to get rid of the implication $\Rightarrow$?
  → Lagrangian relaxation
- How to get rid of the universal quantification $\forall$?
  → Quantifier elimination/mathematical programming & relaxation

Algorithmically interesting cases

- linear inequalities
  → linear programming
- linear matrix inequalities (LMI)/quadratic forms
  → semidefinite programming
- semialgebraic sets
  → polynomial quantifier elimination, or
  → relaxation with semidefinite programming
Arithmetic Mean Example:
Ranking Function with Semi-definite Programming Relaxation

Input the loop abstract semantics

Quantifier elimination (Tarski-Seidenberg)
- quantifier elimination for the first-order theory of real closed fields:
  - $F$ is a logical combination of polynomial equations and inequalities in the variables $x_1, \ldots, x_n$
  - Tarski-Seidenberg decision procedure
    transforms a formula
    $$\forall \exists x_1 : \ldots \forall \exists x_n : F(x_1, \ldots, x_n)$$
    into an equivalent quantifier free formula
- cannot be bound by any tower of exponentials [Heintz, Roy, Solerno 89]
Quantifier elimination (Collins)
- cylindrical algebraic decomposition method by Collins
- implemented in MATHEMATICA®
- worst-case time-complexity for real quantifier elimination is “only” doubly exponential in the number of
  quantifier blocks
- Various optimisations and heuristics can be used

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

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

Example of quadratic form program (factorial)

\[
l(x, x_0) > + 2l(x, x_0)q + r > 0
\]

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

Example of linear program (Arithmetic mean)

\[
\begin{align*}
\{x = y + 2k, x > y\} & \quad +1.x - 1.y >= 0 \\
\text{while } (x <> y) \text{ do } & \quad -2.k0 + 1.x - 1.y + 2 = 0 \\
  k := k - 1; & \quad -1.y0 + 1.y - 1 = 0 \\
  x := x - 1; & \quad -1.x0 + 1.x + 1 = 0 \\
  y := y + 1 & \quad +1.x - 1.y - 2.k = 0
\end{align*}
\]

Example of semialgebraic program (logistic map)

\[
\text{eps} = 1.0e-9;
\text{while } (0 <= a) \& (a <= 1 - \text{eps}) \& (\text{eps} <= x) \& (x <= 1) \text{ do } \\
  x := a*x*(1-x)
od
\]

Example of quadratic form program (factorial)

\[
n := 0;
f := 1;
while (f <= N) do 
  n := n + 1;
f := n * f
od
\]
**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**:
  \[
  \forall x_0, x : \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \gtrless 0 \implies r(x_0) \geq 0
  \]

- The rank is **strictly decreasing**:
  \[
  \forall x_0, x : \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \gtrless 0 \implies r(x_0) - r(x) - \eta \geq 0
  \]

**Idea 3**

Eliminate the conjunction \( \land \) and implication \( \Rightarrow \) by Lagrangian relaxation

**Implication (general case)**

\[
A \Rightarrow B
\]

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

**Implication (linear case)**

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

\[
A \Rightarrow B \iff \text{(soundness)} \implies \text{(completeness)}
\]

border of \( A \) parallel to border of \( B \)
Lagrangian relaxation (linear case)

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

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

\[ \begin{align*}
\text{soundness (Lagrange)} \\
\exists \lambda \in [0, N] \mapsto \mathbb{R}^+ : \forall x \in V : \sigma_0(x) - \sum_{k=1}^{N} \lambda_k \sigma_k(x) \geq 0
\end{align*} \]

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

\[ \begin{align*}
\Leftrightarrow (\lambda'' = \frac{\lambda' - \lambda}{2}) \\
\exists \lambda'' \in [0, N] \mapsto \mathbb{R} : \forall x \in V : \sigma_0(x) - \sum_{k=1}^{N} \lambda''_k \sigma_k(x) \geq 0
\end{align*} \]

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
  \[
  \forall x : Ax + b \geq 0 \Rightarrow cx + d \geq 0
  \]
- (soundness, Lagrange)
- (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 \( \exists \xi \in \mathbb{V} : \sigma(\xi) > 0 \).
- The S-procedure is lossless in the case of one regular quadratic constraint:
  \[
  \forall x \in \mathbb{R}^n : x^T P_1 x + 2q_1^T x + r_1 \geq 0 \Rightarrow
  x^T P_0 x + 2q_0^T x + r_0 \geq 0
  \]
  - (Lagrange)
  - (Yakubovich)
  \[
  \exists \lambda \geq 0 : \forall x \in \mathbb{R}^n : x^T \left( \begin{bmatrix} P_0 & q_0 \\ q_0^T & r_0 \end{bmatrix} - \lambda \begin{bmatrix} P_1 & q_1 \\ q_1^T & r_1 \end{bmatrix} \right) x \geq 0.
  \]
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**: \( \exists \lambda \in [1, N] \mapsto \mathbb{R}^+ \) :
  \[ \forall x_0, x : r(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0 \]

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

---

**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 unkown parameters \( a \)
  2. Compute the parameters \( a \) numerically

- Examples:
  
  \[
  r_a(x) = a.x^T \quad \text{linear}
  
  r_a(x) = a.(x \ 1)^T \quad \text{affine}
  
  r_a(x) = (x \ 1).a.(x \ 1)^T \quad \text{quadratic}
  \]
Idea 5

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

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

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

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 $cx$]

Semidefinite programming

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

[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

$$M(x) \succ 0 = \forall X \in \mathbb{R}^N : X^\top M(x) X \geq 0$$
Semidefinite programming, once again

Feasibility is:

\[ \exists x \in \mathbb{R}^n : \forall X \in \mathbb{R}^N : X^T \begin{pmatrix} M_0 + \sum_{k=1}^{n} x_k M_k \end{pmatrix} X \geq 0 \]

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

\[ \bigwedge_{i=1}^{N} \sigma_i(x_0, x) \geq 0 = \bigwedge_{i=1}^{N} (x_0 \times 1)M_i(x_0 \times 1)^\top \geq 0 \]

Floyd’s method for termination of while B do C

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

- Nonnegative: \( \exists \lambda \in [1, N] \mapsto \mathbb{R}^+ \):

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

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

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

Idea 6

Solve the convex constraints by semidefinite programming

The simplex for linear programming

Dantzig 1948, exponential in worst case, good in practice
Polynomial Methods for Linear Programming

Ellipsoid method:
– Shor 1970 and Yudin & Nemirovskii 1975,
– polynomial in worst case Khachian 1979,
– but not good in practice

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

Semidefinite programming solvers
Numerous solvers available under MATLAB®, a.o.:
– Sdpdlr: S. Burer, R. Monteiro, C. Choi
– SeDuMi: J. Sturm
– bnlx: 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
A1 = [ 0 0 0; 0 0 0; 0 0 0];
% y q r
A1_ = [ 1 0 0; 0 1 0; 0 0 1];
% y q r
bi = [-1; -1; 0];
% linear equalities
% y0 q0 r0
Ae = [ 0 -1 0; -1 0 0; 0 0 -1; 0 -1 0; 0 0 1];
% y q r
Ae_ = [ 0 1 0; 1 0 0; 1 0 1];
% linear equalities
% y0 q0 r0
be = [-1; 0; 0];

Sep. 6, 2006 — 69 — ľ P. Cousot

Iterated forward/backward polyhedral analysis:

\{ y > 1 \}
q := 0;
\{ q=0, y > 1 \}
r := r - y;
\{ r > 0, q > 0 \}
q := q + 1
\{ r > 0, q > 1 \}
while (y <= r) do
\{ y <= r, q > 0 \}
r := r - y;
\{ r > 0, q > 0 \}
q := q + 1
\{ r > 0, q > 1 \}
end
\{ q > 0, y > r + 1 \}

Quadratic program: termination of factorial

Program:
\begin{align*}
n & := 0; \quad -1.0 + 1.0 N0 >= 0 \\
f & := 1; \quad +1.0 N0 >= 0 \\
\text{while } (f <= N) \text{ do} & \quad +1.0 f0 - 1.0 >= 0 \\
\quad n & := n + 1; \quad -1.0 N0 + 1.0 n - 1.0 = 0 \\
\quad f & := n * f \quad +1.0 N0 - 1.0 N0 = 0 \\
\end{align*}
\begin{align*}
\text{od} & \quad -1.0 f0 . n + 1.0 f0 = 0 \\
r(n, f, N) & = -9.993455e-01.0 + 4.346533e-04.0 f + 2.689218e+02.0 + 8.744670e+02.0 \\
\end{align*}

---

Imposing a feasibility radius

---

Floyd's proposal \( r(x, y, q, r) = x - q \) is more intuitive but requires to discover the nonlinear loop invariant \( x = r + q y \).
Idea 7

Convex abstraction of non-convex constraints

Semidefinite programming relaxation for polynomial programs

\[
\text{eps} = 1.0e-9; \\
\text{while } (0 \leq a) \& (a \leq 1 - \text{eps}) \& (\text{eps} \leq x) \& (x \leq 1) \text{ do} \\
\text{ x := a*x*(1-x) } \\
\text{ end}
\]

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

\[ r(x) = 1.222356e-13.x + 1.406392e+00 \]

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

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

→ case analysis:

\[ \begin{cases} 
  i \geq 0 \\
  i < 0 
\end{cases} \]

**Quadratic termination of linear loop**

\( \{n=0\} \)

\[ i := n; \ j := n; \]

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

← termination precondition determined by iterated forward/backward polyhedral analysis

**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

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

\[
\begin{align*}
  r(n,i,j) &= +7.024176e-04 \cdot n^2 + 4.394909e-05 \cdot n \cdot i \ldots \\
 &- 2.809222e-03 \cdot n \cdot j + 1.533829e-02 \cdot n \ldots \\
 &+ 1.569773e-03 \cdot i^2 + 7.077127e-05 \cdot i \cdot j \ldots \\
 &+ 3.093629e+01 \cdot i - 7.021870e-04 \cdot j^2 \ldots \\
 &+ 9.940151e-01 \cdot j + 4.237694e+00 
\end{align*}
\]

Successive values of \( r(n,i,j) \) for \( n = 10 \) on loop entry
Example of termination of nested loops: Bubblesort inner loop

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

```
assume (n0 = n & j >= 0 & i >= 1 & n0 >= i & j < i);
{no=n,i>=1,j>=0,n0>=i}
```

```
{j=1,i=1,n0=n1,n0=n01,n0=n,i>=1,j>=0,n0>=i}
```

```
j := j + 1
```

```
{j=1+1,i=1,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

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

```
assume (n0=n & i>=0 & n>=i & i <> 0);
{no=n,i>=0,n0=i}
```

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

termination (lmilab)

```
r(n0,n,i,j) = +243487866.n0 +16834142.n +100314562.i +65646865
```

Handling nondeterminacy

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

Example of termination of a concurrent program

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

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

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

```
interleaving
```

```
penbmi: r(x,y) = 2.537395e+00.x+-2.537395e+00.y+-2.046610e-01
```
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:
  \[
  \forall x, x' : I(x) \land [B;C](x, x') \Rightarrow I(x')
  \]

Relaxed Parametric Invariance Proof Method

Termination of a fair parallel program

\[
\text{[[ while } \left[[x>0] \land [y>0] \right] \text{ do } x := x - 1 \text{ od } \left| \right. \text{ interleaving} \Rightarrow \\
\text{while } \left[[x>0] \land [y>0] \right] \text{ do } y := y - 1 \text{ od } \right]
\]

\[
\{an=1\} \leftarrow \text{ termination precondition determined by iterated forward/backward polyhedral analysis}
\]

\[
t := 7;
\]

\[
\text{assume } (0 \leq t \land t < 1);
\]

\[
s := 7;
\]

\[
\text{assume } (1 <= s) \land (s <= m);
\]

\[
\text{while } ((x > 0) \land (y > 0)) \text{ do }
\]

\[
\text{if } (t = 1) \text{ then }
\]

\[
x := x - 1
\]

\[
\text{else }
\]

\[
y := y - 1
\]

\[
\text{fi}
\]

\[
s := s - 1;
\]

\[
\text{od};
\]

\[
\text{penbmi: } r(x, y, m, s, t) = +1.000468e+00.x +1.000611e+00.y +2.855769e-02.m -3.929197e-07.s +6.588027e-06.t +9.998392e+03
\]

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

Bilinear matrix inequality (BMI) solvers

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

[Minimizing $x^T Q x + c^T x$]

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

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

Idea 8

Solve the bilinear matrix inequality (BMI) by semidefinite programming

Example: linear invariant

Program:
- Invariant:
  \[
  i := 2; j := 0;
  \text{while } (??) \text{ do}
  \text{if } (??) \text{ then}
  i := i + 4
  \text{else}
  i := i + 2;
  j := j + 1
  \text{end if}
  \]
- 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
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 ???)

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

Related anterior work
– Linear case (Farkas lemma):
  - Invariants: Sankaranarayanan, Spina, Manna (CAV’03, SAS’04, heuristic solver)
  - Termination: Podelski & Rybalchenko (VMCAI’03, Lagrange coefficients eliminated by hand to reduce to linear programming so no disjunctions, no tests, etc)
  - Parallelization & scheduling: Feautrier, easily generalizable to nonlinear case
Related posterior work
– Termination using Lyapunov functions: Roozbehani, Feron & Megrestki (HSCC 2005)

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

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

\[ \exists \rho : \forall x, x' : \left[ B \mathcal{C} \right] (x, x') \Rightarrow \rho (x, x') \geq 0 \]

\[ \iff \exists \rho : \forall x, x' : \sum_{k=1}^{n} \sigma_k (x, x') \geq 0 \Rightarrow \rho (x, x') \geq 0 \]

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

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

\[ \iff \text{Choose form of } \rho (x, x') = (x, x')M_0 (x, x')^\top \]

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

\[ (x, x')M_0 (x, x')^\top - \sum_{k=1}^{n} \lambda_k (x, x')M_k (x, x')^\top \geq 0 \]

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

\[ \iff \exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}^* : \left( M_0 - \sum_{k=1}^{n} \lambda_k M_k \right) \gg 0 \]
SOS Relaxation Principle

- Show $\forall x : p(x) \geq 0$ by $\forall x : p(x) = \sum_{i=1}^{k} q_i(x)^2$
- Hert'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 $Q \succeq 0$ is a semidefinite positive matrix of unknowns and $z = \{x^2, xy, y^2, \ldots x, y, 1\}$ is a monomial basis
- If such a $Q$ does exist then $p(x, y, \ldots)$ is a sum of squares$^5$
- The equality $p(x, y, \ldots) = z^T Q z$ yields LMI contrains on the unknow $Q$: $z^T M(Q) z \succeq 0$

---

$^5$ Since $Q \succeq 0$, $Q$ has a Cholesky decomposition $L$ which is an upper triangular matrix $L$ such that $Q=L^T L$. It follows that $p(x) = z^T Q z = z^T L^T L z = (L z)^T (L z) = [L_1 z ] [L_1 z ] = \sum_i (L_i z_i)^2$ (where $z$ is the vector dot product $x y = \sum_i x_i y_i$), 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 $Q \succeq 0$, check for semidefinite positiveness $\exists Q : \forall X : X^T M(Q) X \geq 0$ i.e. $\exists Q : M(Q) \succeq 0$ with LMI solver
- Implement with SOS tools under MATLAB$^*$ of Prajna, Papachristodoulo, Seiler and Parrilo
- Nonlinear cost since the monomial basis has size $\binom{n + m}{m}$ for multivariate polynomials of degree $n$ with $m$ variables

---

5 Sep. 6, 2006 — 105 — ľ P. Cousot

General relaxation/approximation idea

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

---

$^5$ Since $Q \succeq 0$, $Q$ has a Cholesky decomposition $L$ which is an upper triangular matrix $L$ such that $Q=L^T L$. It follows that $p(x) = z^T Q z = z^T L^T L z = (L z)^T (L z) = [L_1 z ] [L_1 z ] = \sum_i (L_i z_i)^2$ (where $z$ is the vector dot product $x y = \sum_i x_i y_i$), 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 $Q \succeq 0$, check for semidefinite positiveness $\exists Q : \forall X : X^T M(Q) X \geq 0$ i.e. $\exists Q : M(Q) \succeq 0$ with LMI solver
- Implement with SOS tools under MATLAB$^*$ of Prajna, Papachristodoulo, Seiler and Parrilo
- Nonlinear cost since the monomial basis has size $\binom{n + m}{m}$ for multivariate polynomials of degree $n$ with $m$ variables

---

5 Sep. 6, 2006 — 105 — ľ P. Cousot