Program Verification by Parametric Abstraction and Semi-definite Programming

Patrick Cousot
École normale supérieure
45 rue d’Ulm, 75230 Paris cedex 05, France
Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot


Reference


Static analysis

Principle of static analysis

– Define the most precise program property as a fixpoint $\llbracket P \rrbracket$
– Effectively compute a fixpoint approximation:
  - iteration-based fixpoint approximation
  - constraint-based fixpoint approximation
Iteration-based static analysis

- Effectively overapproximate the iterative fixpoint definition:\n\[ \text{lfp } F = \bigcup_{\lambda \in \Omega} X^\lambda \]
\[ X^0 = \bot \]
\[ X^\lambda = \bigcup_{\eta < \lambda} F(X^\eta) \]

---

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 \( \text{lfp } F \)
- Constraint-based static analysis is the main subject of this talk.

---

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

---

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 \( a \)

---

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


Overview of the Termination Analysis Method

Constraints and Verification, INI, 8 May 2006 —9—

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 a *forward relational static analysis* of the loop to determine the loop invariant
3. Assuming the loop invariant, perform a *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.

The main point in this talk is (4).
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.

Backward/ancestry properties

Example: termination (must reach final states)

Forward/reachability properties

Example: partial correctness (must stay into safe 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

\[ X^0 = \top \]
\[ \ldots \]
\[ X^{2n+1} = \text{lfp } \lambda Y. X^{2n} \sqcap F(Y) \]
\[ X^{2n+2} = \text{lfp } \lambda Y. X^{2n+1} \sqcap B(Y) \]
\[ \ldots \]

Arithmetic Mean Example:
Termination Precondition (1)

\{x>=y\}
\{x>=y+2\}
\{x>=y+1\}
\{x>=y\}
\{x=y\}

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

Add an auxiliary termination counter to enforce (bounded) termination in the backward analysis!

Idea 1

The auxiliary termination counter method
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 a forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform a 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=x+y+2k,x>=y}
while (x <> y) do
    {x=x+y+2k,x>=y+2}
    k := k - 1;
    {x=x+y+2k+2,x>=y+2}
    x := x - 1;
    {x=x+y+2k+1,x>=y+1}
    y := y + 1
    {x=x+y+2k,x>=y}
od
{k=0,x=y}
```

Arithmetic Mean Example: Body Relational Semantics

Case $x < y$:

```
assume (x=y+2*k) & (x>=y+2);
{x=x+y+2k,x>=y+2}
assume (x < y);
empty(6)
assume (x0=x) & (y0=y) & (k0=k);
{k=0,x=x0,y=y0,x=x0,x=y+2k,x>=y+2}
k := k - 1;
{x=x+y+2k,x>=y+2}
x := x - 1;
{y=y+1,x=x+y+2k,x>=y+1}
empty(6)
```

Case $x > y$:

```
assume (x=y+2*k) & (x>=y+2);
{x=x+y+2k,x>=y+2}
assume (x > y);
empty(6)
assume (x0=x) & (y0=y) & (k0=k);
{k=0,x=x0,y=y0,x=x0,x=y+2k,x>=y+2}
k := k - 1;
{x=x+y+2k,x>=y+2}
x := x - 1;
{y=y+1,x=x+y+2k,x>=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 $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unknown rank function $r$ such that:

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

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

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

**Problems**

- How to get rid of the implication $\implies$?
  - $\rightarrow$ 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
Arithmetic Mean Example: Ranking Function with Semi-definite Programming Relaxation

```matlab
> 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 - 2*k0 - 2 = 0
Ae_ = [ 1 -1 0; % x - y0 - 1 = 0
 0 1 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);
```

Input the loop abstract semantics

```matlab
> 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, 'integer', 'linear');
> disp(diagnostic)
feasible (bnb)
intrank(R, v)
```

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

Quantifier Elimination

- 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

4 See e.g. Redlog http://www.fmi.uni-passau.de/~redlog/

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

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

Example of linear program (Arithmetic mean)

\begin{align*}
[A A']x &\geq b \\
\{x=y+2k, x>y\} &
\begin{cases}
+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 \\
\end{cases}
\end{align*}

while \(x <> y\) do

\begin{align*}
    &k := k - 1; \\
    &x := x - 1; \\
    &y := y + 1
\end{align*}

od

Example of quadratic form program (factorial)

\begin{align*}
[x x']A[x x']^T + 2[x x']q + r &\geq 0 \\
n &:= 0; \\
f &:= 1; \\
while (f <= N) do \\
    &n := n + 1; \\
    &f := n * f
od

\begin{align*}
    &-1.f0 +1.N0 > = 0 \\
    &+1.n0 >= 0 \\
    &-1.n0 +1.n -1 = 0 \\
    &+1.N0 -1.N = 0 \\
    &-1.f0.n +1.f = 0
\end{align*}

Example of semialgebraic program (logistic map)

\begin{align*}
\text{eps} &:= 1.0e-9; \\
\text{while } (0 <= a) \land (a <= 1 - \text{eps}) \land (\text{eps} <= x) \land (x <= 1) \text{ do} \\
    &x := a*x*(1-x)
\end{align*}

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:

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

- The rank is strictly decreasing:

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

Idea 3

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

Implication (general case)

\[
A \Rightarrow B
\]

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

Implication (linear case)

$A \Rightarrow B$

(assuming $A \neq 0$)

$\iff$ (soundness)

$\Rightarrow$ (completeness)

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

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

\( \iff \) soundness (Lagrange)

\( \Rightarrow \) completeness (lossless)

\( \not\Rightarrow \) incompleteness (lossy)

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

relaxation = approximation, \( \lambda \) = Lagrange coefficients

Lagrangian relaxation, equality constraints

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

\( \iff \) soundness (Lagrange)

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

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

\( \iff \left( \lambda'' = \frac{\lambda' - \lambda}{2} \right) \)

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

Example: affine Farkas’ lemma, informally

- An application of Lagrangian relaxation to the case when \( A \) is a polyhedron
Example: affine Farkas’ lemma, formally

- Formally, if the system $Ax + b \geq 0$ is feasible then
  \[ \forall x : Ax + b \geq 0 \Rightarrow cx + d \geq 0 \]
  \[ \Leftarrow \text{(soundness, Lagrange)} \]
  \[ \Rightarrow \text{(completeness, Farkas)} \]
  \[ \exists \lambda \geq 0 : \forall x : cx + d - \lambda (Ax + b) \geq 0. \]

Yakubovich’s S-procedure, informally

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

Incompleteness (convex case)

Yakubovich’s S-procedure, completeness cases

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

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

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 unknown parameters \( a \)
2. Compute the parameters \( a \) numerically

- **Examples**:
  \[
  r_a(x) = a \cdot x \quad \text{linear}
  \]
  \[
  r_a(x) = a \cdot (x_1)^\top \quad \text{affine}
  \]
  \[
  r_a(x) = (x_1) \cdot a \cdot (x_1)^\top \quad \text{quadratic}
  \]

Idea 4

Parametric abstraction of the ranking function \( r \)

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**: \( \exists \eta > 0 : \forall \lambda' \in [1, N] \mapsto \mathbb{R}^+ \):
  \[
  \forall x_0, x : (r_a(x_0) - r_a(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0
  \]
**Idea 5**

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

**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**

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

[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

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

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

Idea 6

Solve the convex constraints by semidefinite programming

Floyd’s method for termination of while B do C

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

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

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

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

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

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)

Interior point method for semidefinite programming
- Nesterov & Nemirovskii 1988, good in practice (thousands of variables)
  - Various path strategies e.g. “stay in the middle”

The interior point method

Semidefinite programming solvers
Numerous solvers available under MATLAB®, a.o.:
- lmilab: P. Gahinet, A. Nemirovskii, A.J. Laub, M. Chilali
- Scplr: S. Burer, R. Monteiro, C. Choi
- Scpt3: 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**

```plaintext
% clear all
% linear inequalities
A1 = [ 0 0 0; 0 0 0; 0 0 0 ];
% y q r
A1_ = [ 1 0 0; y - 1 >= 0
0 1 0; q - 1 >= 0
0 0 1; r >= 0
bi = [-1; -1; 0];
% linear equalities
% y q r
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:**

```plaintext
{y>=1}
q := 0;
{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}
```

**Quadratic program: termination of factorial**

**Program:**

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

**LMI semantics:**

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

```plaintext
r(n,f,N) = -9.993455e-01.n +4.346533e-04.f
+2.689218e+02.N +8.744670e+02
```

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

Convex abstraction of non-convex constraints

Constraints and Verification, INI, 8 May 2006 — 73 — @ P. Cousot

Semidefinite programming relaxation for polynomial programs

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

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

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

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

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

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

Constraints and Verification, INI, 8 May 2006 — 77 — © P. Cousot

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

Constraints and Verification, INI, 8 May 2006 — 78 — © P. Cousot

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

\[ r(n,i,j) = +7.024176e-04 . n^2 + 4.394909e-05 . n . i \ldots + 2.809222e-03 . n . j + 1.533829e-02 . n \ldots + 1.569773e-03 . i^2 + 7.077127e-05 . i . j \ldots + 3.093629e+01 . i - 7.021870e-04 . j^2 \ldots + 9.940151e-01 . j + 4.237694e+00 \]

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 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:

```
... +1.i' -1 > 0
+1.j' -1 > 0
+1.n0' -1.i' > 0
-i.j +1.j' -1 = 0
-i.i +1.i' - 0
-i.n +1.m0' = 0
+1.n0 -1.n0' = 0
+1.n0' -1.n' = 0
... 
```

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

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

Constraints and Verification, INI, 8 May 2006 — 81 — P. Cousot

Example of termination of nested loops:
Bubblesort outer loop

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

```
... +1.i' +1 >= 0
+1.n0' +1.i' -1 >= 0
+1.i -1.i' +1 = 0
-l.n +1.m0' = 0
+1.n0 -1.n0' = 0
+1.n0' +1.n' = 0
+1.n0' -1.n' = 0
... 
```

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

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

j := 0;
while (j <= i) do
  j := j + 1
od;
i := i - 1
{i+1=j,i+1=i1,n0=n01,n0=n01,n0>=i,i+1>=0,n0>=i+1}
```

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

Constraints and Verification, INI, 8 May 2006 — 82 — P. Cousot

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)

Termination of a concurrent program
```
[1] while [x+2 < y] do
  x := x + 1
od
```
```
[2] 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
```
```
[3] od
```
```
penbmi: r(x,y) = 2.537395e+00.x+-2.537395e+00.y-
-2.046610e-01
```

Constraints and Verification, INI, 8 May 2006 — 84 — P. Cousot
Termination of a fair parallel program

\[
\begin{align*}
\text{while } (x > 0) \text{ or } (y > 0) \text{ do } & x := x - 1 \text{ od} || \\
\text{while } (x > 0) \text{ or } (y > 0) \text{ do } & y := y - 1 \text{ od} \\
\end{align*}
\]

\text{interleaving + scheduler}

\(m>1\) ← termination precondition determined by iterated forward/backward polyhedral analysis

\[t := ?;\]
\[s := ?;\]

\text{assume } (0 \leq t \& \& t \leq 1);\]
\text{assume } (1 \leq s \& \& s \leq m);\]

\text{while } (x > 0) \text{ or } (y > 0) \text{ do}

\[\text{if } (t = 1) \text{ then } x := x - 1 \text{ else } y := y - 1 \text{ fi;}\]
\[s := s - 1;\]

\text{if } (s = 0) \text{ then}

\[t := 0;\]
\text{else}

\[t := 1;\]
\text{assume } (1 \leq s \& \& s \leq m);\]
\text{fi;}\]

\[\text{fi;}\]
\[s := s - 1;\]
\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

Floyd's method for invariance

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

– The invariant is \textit{initial}:

\[\forall x : P(x) \Rightarrow I(x)\]

– The invariant is \textit{inductive}:

\[\forall x, x' : I(x) \land [B; C][x, x'] \Rightarrow I(x')\]

Relaxed Parametric Invariance Proof Method

– Express loop semantics as a conjunction of LMI \textit{constraints} (by relaxation for polynomial semantics)

– Eliminate the conjunction and implication by \textit{Lagrangian relaxation}

– Fix the form of the unknown invariant by \textit{parametric abstraction}

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

Bilinear matrix inequality (BMI) solvers

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

[Minimizing $x^\top Qx + cx$]

Two solvers available under MATLAB®:
- PenBMI: M. Kočvara, M. Stingl
- bmbibn: J. Löfberg

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

Example: linear invariant

Program:

```
 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 anterior 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
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”.

THE END, THANK YOU

More details and references in the VMCAI'05 paper.

ANNEX

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

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

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

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

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

\[ \iff \{ \text{Semantics abstracted in LMI form (\iff if exact abstraction)} \} \]

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

\[ \iff \{ \text{Choose form of } r(x, x') = (x \ x') 1 \mathbb{M}_0(x \ x')^\top \} \]

\[ \iff \exists \mathbb{M}_0 : \exists \lambda \in [1, N] : \mathbb{R}_* : \forall x, x' : \mathbb{D}^n : (x \ x') 1 \mathbb{M}_0(x \ x')^\top \geq 0 \]

\[ \iff \{ \text{LMI solver provides } \mathbb{M}_0 \text{ (and } \lambda) \} \]

\[ \exists \mathbb{M}_0 : \exists \lambda \in [1, N] : \mathbb{R}_* : \left( \mathbb{M}_0 - \sum_{k=1}^{N} \lambda_k \mathbb{M}_k \right) \geq 0 \]
SOS Relaxation Principle

- Show $\forall x : p(x) \geq 0$ by $\forall x : p(x) = \sum_{i=1}^{k} q_i(x)^2$
- Hibert’s 17th problem (sum of squares)
- Undecidable (but for monovariable or low degrees)
- Look for an approximation (relaxation) by semidefinite programming

General relaxation/approximation idea

- Write the polynomials in quadratic form with monomials as variables: $p(x, y, \ldots) = z^T Q z$ where $Q \succeq 0$ 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
- The equality $p(x, y, \ldots) = z^T Q z$ yields LMI contrains on the unknown $Q$: $z^T M(Q) z \succeq 0$

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 $Q \succeq 0$, check for semidefinite positive-ness $\exists Q : \forall X : X^T M(Q) X \succeq 0$ i.e. $\exists Q : M(Q) \succeq 0$ with LMI solver
- Implement with SOSTools under MATLAB 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