Abstract

Program termination is based on reasonings by induction (e.g. on program steps, program data) which involves the discovery of unknown inductive arguments (e.g. rank functions, invariants) satisfying universally quantified termination conditions. For static program analysis, the discovery of the inductive arguments must be automated, which consists in solving the constraints provided by the termination conditions. Several methods have been considered: recurrence/difference equation resolution; iteration, possibly with convergence acceleration through widening/narrowing; or direct methods (such as elimination). All these methods involve some form of simplification of the constraints formalized by abstract interpretation.

In this talk, we explore parametric abstraction of rank function and invariants and direct resolution of Floyd/Naur/Hoare termination constraints by Lagrangian relaxation (to handle implication) and semidefinite programming relaxation (to handle universal implication). Finally the parameters are computed using numerical semidefinite programming solvers. This new approach exploits the recent progress in the numerical resolution of linear or bilinear matrix inequalities by semidefinite programming using efficient polynomial primal/dual interior point methods generalizing those well-known in linear programming to convex optimization. The framework is applied to invariance and termination proof of sequential, nondeterministic, concurrent, and fair parallel imperative polynomial programs and can easily be extended to other safety and liveness properties.

Reference

Principle of static analysis

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

Constraint-based static analysis

- Effectively solve a postfixpoint constraint:
  \[
  \text{ifp} \ F = \bigcap \{ X \mid F(X) \subseteq X \}
  \]
  since \( F(X) \subseteq X \) implies \( \text{ifp} \ F \subseteq X \)
- Sometimes, the constraint resolution algorithm is nothing but the iterative computation of \( \text{ifp} \ F \)
- Constraint-based static analysis is the main subject of this talk.

Iteration-based static analysis

- Effectively overapproximate the iterative fixpoint definition \(^1\):
  \[
  \text{ifp} \ F = \bigsqcup_{\lambda \in \mathbb{N}} X^\lambda
  \]
  \[
  X^0 = \bot
  \]
  \[
  X^\lambda = \bigsqcup_{\eta < \lambda} F(X^\eta)
  \]

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
Fixpoint versus Constraint-based Approach for Termination Analysis

1. Termination can be expressed in fixpoint form.
2. However, we know no effective fixpoint underapproximation method needed to overestimate 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.

---

Proving Termination of a Loop

1. Termination precondition
2. Loop invariant
3. Loop operational semantics
4. Ranking function

The main point in this talk is (4).

---

Arithmetic Mean Example

```c
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.

---

Forward/reachability properties

Example: partial correctness (must stay into safe states)

---

Backward/ancestry properties

Example: termination (must reach final states)

---

Arithmetic Mean Example

1. Perform an iterated forward/backward relational static analysis of the loop with termination hypothesis to determine a necessary proper termination precondition
2. Assuming the termination precondition, perform an forward relational static analysis of the loop to determine the loop invariant
3. Assuming the loop invariant, perform an forward relational static analysis of the loop body to determine the loop abstract operational semantics
4. Assuming the loop semantics, use an abstraction of Floyd’s ranking function method to prove termination of the loop
Forward/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

\[
\begin{align*}
X^0 & = \top \\
\cdots \\
X^{2n+1} & = \text{lfp } \lambda Y. X^{2n} \cap F(Y) \\
X^{2n+2} & = \text{lfp } \lambda Y. X^{2n+1} \cap B(Y) \\
\cdots
\end{align*}
\]

Arithmetic Mean Example: Termination Precondition (1)

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

Idea 1

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

\{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+1\}
x := x - 1;
\{y = y + 1\}
y := y + 1
\{x=y+2k, x \geq y\}
\od
\{k=0, x=y\}

IBM Research Seminar, Jan. 5th, 2007

Arithmetic Mean Example:
Loop Invariant

assume ((x=y+2*k) & (x\geq y));
\{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+1\}
x := x - 1;
\{y = y + 1\}
y := y + 1
\{x=y+2k, x\geq y\}
\od
\{k=0, x=y\}

IBM Research Seminar, Jan. 5th, 2007

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

IBM Research Seminar, Jan. 5th, 2007

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

IBM Research Seminar, Jan. 5th, 2007
Arithmetic Mean Example

Body Relational Semantics

Case $x < y$:
assume $(x=y+2k) & (x \geq y+2)$;
\{ $x=y+2k, x \geq y+2$ \}
assume $(x < y)$;
empty(6)

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

Floyd’s method for termination of while $B$ do $C$

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

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

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

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

Problems
- 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**
  → 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 Semidefinite Programming Relaxation**

```matlab
% Display the abstract semantics of the loop while B do C
» display_Mk(Mk, N, v0, v);
...
+1.x -1.y >= 0
-2.x0 +1.x -1.y +2 = 0
-1.y0 +1.y -1 = 0
-1.x0 +1.x +1 = 0
+1.x -1.y -2.k = 0
...
» [diagnostic,R] = termination(v0, v, Mk, N, 'integer', 'linear');
» disp(diagnostic)
fractional (bnb)
» intrank(R, v)
```

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

---

**Quantifier Elimination**

```matlab
% clear all;
[v0,v] = variables('x','y','k')% linear inequalities% x0 y0 kAi = [ 0 0 0];% x y ka_i = [ 1 -1 0]; % x0 - y0 >= 0
bi = [0];
[N Mk(:,:,:)] = linToMk(Ai, a_i, bi);
% linear equalities% x0 y0 kAe = [ 0 0 -2;
0 -1 0;
-1 0 0;
0 0 0];
% x y kAe_ = [ 1 -1 0; % x - y - 2*k0 - 2 = 0
0 1 0; % y - y0 - 1 = 0
1 0 0; % x - x0 + 1 = 0
1 -1 -2]; % x - y - 2*k = 0
be = [2; -1; 1; 0];
[M Mk(:,:,N+1:N+M)] = linToMk(Ae, a_e, be);
```

---

**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 \textsc{Mathematica}^\textsuperscript{4}
- 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\(^4\)

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

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

Example of linear program (Arithmetic mean)

$[A A^t|x_0 x]^\top \geq b$

\[
\begin{align*}
\{x=y+2k,x\geqslant y\} & \quad +1.x -1.y \geq 0 \\
\text{while } (x \not< y) \text{ do} & \quad -2.k0 +1.x -1.y +2 = 0 \\
& \quad -1.y0 +1.y -1 = 0 \\
& \quad -1.x0 +1.x +1 = 0 \\
& \quad +1.x -1.y -2.k = 0 \\
\text{od}
\end{align*}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & -2 & 1 & -1 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & -2
\end{bmatrix}
\begin{bmatrix}
x_0 \\
y_0 \\
k_0 \\
x \\
y
\end{bmatrix} \geq \begin{bmatrix} 0 \\ -2 \\ 1 \\ -1 \\ 0 \end{bmatrix}
\]

Example of quadratic form program (factorial)

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

\[
\begin{align*}
n := 0; & \quad -1.f0 +1.n0 >= 0 \\
f := 1; & \quad +1.n0 >= 0 \\
\text{while } (f <= N) \text{ do} & \quad +1.f0 -1 >= 0 \\
& \quad -1.n0 +1.n -1 = 0 \\
& \quad +1.N0 -1.N = 0 \\
& \quad -1.f0.n +1.f = 0 \\
\text{od}
\end{align*}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
n_0 \\
f_0 \\
n0 \\
f \\
N
\end{bmatrix} + 2[n_0f_0n0fN] + 0 = 0
\]

\[
\begin{bmatrix}
0 \\
1 \\
0 \\
\frac{1}{2}
\end{bmatrix}
\begin{bmatrix}
0 \\
N
\end{bmatrix} + 0 = 0
\]
Example of semialgebraic program
(logistic map)

```
eps = 1.0e-9;
while (0 <= a) & (a <= 1 - eps)
    & (eps <= x) & (x <= 1) do
    x := a*x*(1-x)
od
```

Floyd’s method for termination of while B do C

Find an ℝ/ℚ/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 \( \wedge \) and implication \( \Rightarrow \) by Lagrangian relaxation

Implication (general case)

\[
A \Rightarrow B \\
\Leftrightarrow \forall x \in A : x \in B
\]
Implication (linear case)

\[ A \Rightarrow B \quad \text{(assuming } A \neq 0) \]

\[ \Leftarrow \text{ (soundness)} \]

\[ \Rightarrow \text{ (completeness)} \]

border of \( A \) parallel to border of \( B \)

Lagrangian relaxation, formally

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

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

\[ \Leftarrow \text{ soundness (Lagrange)} \]

\[ \Rightarrow \text{ completeness (lossless)} \]

\[ \neq \text{ incompleteness (lossy)} \]

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

relaxation = approximation, \( \lambda_i = \text{Lagrange coefficients} \)

Lagrangian relaxation, equality constraints

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

\[ \Leftarrow \text{ soundness (Lagrange)} \]

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

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

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

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

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

\[ B \]

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 $\exists \xi \in V : \sigma(\xi) > 0$.
- The S-procedure is lossless in the case of one regular quadratic constraint:
  $$\forall x \in \mathbb{R}^n : x^\top P_1 x + 2q_1^\top x + r_1 \geq 0 \Rightarrow x^\top P_0 x + 2q_0^\top x + r_0 \geq 0$$
  $$(\text{Lagrange})$$
  $$(\text{Yakubovich})$$
  $$\exists \lambda \geq 0 : \forall x \in \mathbb{R}^n : x^\top \left( \begin{bmatrix} P_0 & q_0 \\ q_0^\top & r_0 \end{bmatrix} - \lambda \begin{bmatrix} P_1 & q_1 \\ q_1^\top & r_1 \end{bmatrix} \right) x \geq 0.$$  

Floyd’s method for termination of while B do C
Find an $\mathbb{R}/\mathbb{Q}/\mathbb{Z}$-valued unkown rank function $r$ which is:
- Nonnegative: $\exists \lambda \in [1, N] \mapsto \mathbb{R}^{+i}$:
  $$\forall x_0, x : r(x_0) - \sum_{i=1}^{N} \lambda_i \sigma_i(x_0, x) \geq 0$$
- Strictly decreasing: $\exists \eta > 0 : \exists \lambda' \in [1, N] \mapsto \mathbb{R}^{+i}$:
  $$\forall x_0, x : (r(x_0) - r(x) - \eta) - \sum_{i=1}^{N} \lambda'_i \sigma_i(x_0, x) \geq 0$$

Idea 4
Parametric abstraction of the ranking function $r$

Parametric abstraction
- How can we compute the ranking function $r$?
  $\rightarrow$ 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:
  $$r_a(x) = a.x^\top$$  linear
  $$r_a(x) = a.(x 1)^\top$$  affine
  $$r_a(x) = (x 1).a.(x 1)^\top$$  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

$\exists x \in \mathbb{R}^N : \begin{bmatrix} A \end{bmatrix} x \geq b$  

[Minimizing $cx$]

Feasibility

- feasibility problem: find a solution $s \in \mathbb{R}^n$ to the optimization program, such that $\bigwedge_{i=1}^{N} g_i(s) \geq 0$, or to determine that the problem is infeasible

- feasible set: $\{x | \bigwedge_{i=1}^{N} g_i(x) \geq 0\}$

- a feasibility problem can be converted into the optimization program

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

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

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

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

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

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

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

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

Semidefinite programming, once again

Feasibility is:

\[ \exists x \in \mathbb{R}^n : \forall X \in \mathbb{R}^N : X^\top \left( M_0 + \sum_{k=1}^{n} x_k M_k \right) X \geq 0 \]

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

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

Idea 6

Solve the convex constraints by semidefinite programming
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”
Semidefinite programming solvers
Numerous solvers available under Mathlab, a.o.:
- lmilab: P. Gahinet, A. Nemirovskii, A.J. Laub, M. Chilali
- Sdplr: S. Burer, R. Monteiro, C. Choi
- Sdpt3: R. Tütüncü, K. Toh, M. Todd
- SeDuMi: J. Sturm
- bnb: J. Löfberg (integer semidefinite programming)
Common interfaces to these solvers, a.o.:
- Yalmip: J. Löfberg

Sometime need some help (feasibility radius, shift,...)

Linear program: termination of Euclidean division

```
> clear all
% linear inequalities
% y0 q0 r0
A = [ 0 0 0; 0 0 0; 0 0 0];
% y q r
A_ = [ 1 0 0; % y - 1 >= 0
      0 1 0; % q - 1 >= 0
      0 0 1]; % r >= 0
bi = [-1; -1; 0];
% linear equalities
% y0 q0 r0
Ae = [ 0 -1 0; % -q0 + q -1 = 0
       -1 0 0; % -y0 + y = 0
       0 0 -1]; % -r0 + y + r = 0
% y q r
Ae_ = [ 0 1 0; 1 0 1];
be = [-1; 0; 0];
```

Iterated forward/backward polyhedral analysis:
- \{y\geq 1\}
  - \{y=0, q\geq 0\}
  - \{q=0, r\geq 0\}
  - \{x=r, q=0, y\geq 1\}

while (y \leq r) do
  - \{y \leq r, q \geq 0\}
  - \{r \geq r - y, q \geq 0\}
  - \{q \geq 0, y \geq r + 1\}
```

Imposing a feasibility radius

```
> [N Mk(:,:,:)] = linToMk(Ai, Ai_, bi);
> [M Mk(:,:,N+1:N+M)] = linToMk(Ae, Ae_, be);
> [v0,v] = variables('y','q','r');
> display_Mk(Mk, N, v0, v);
+1.y -1 >= 0
+1.q -1 >= 0
+1.r >= 0
-1.q0 +1.q -1 = 0
-1.y0 +1.y = 0
-1.r0 +1.y +1.r = 0
> [diagnostic, R] = termination(v0, v, Mk, N, 'integer', 'quadratic');
> disp(diagnostic)
termination (bnb)
> intrank(R, v)
```

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

\[ r(n,f,N) = -9.993455e-01.n +4.346533e-04.f +2.689218e+02.N +8.744670e+02 \]

Semidefinite programming relaxation for polynomial programs

```plaintext
eps = 1.0e-9;
while (0 <= a) & (a <= 1 - eps) & (eps <= x) & (x <= 1) do
  x := a*x*(1-x)
od
```

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

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

Idea 7

Convex abstraction of non-convex constraints

Considering More General Forms of Programs
Handling disjunctive loop tests and tests in loop body

- By case analysis
- and “conditional Lagrangian relaxation” (Lagrangian relaxation in each of the cases)

Loop body with tests:

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

```plaintext
lmilab:
r(i,x,y) = -2.252791e-09.i -4.355697e+07.x +4.355697e+0 7.y +5.502903e+08
```

Quadratic termination of linear loop

\{n>=0\}

```plaintext
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

```plaintext
sdplr (with feasibility radius of 1.0e+3):
r(n,i,j) = +7.024176e-04.n^2 +4.394909e-05.n.i ... -2.809222e-03.n.j +1.533829e-02.n ... +1.569773e-03.i^2 +7.077127e-05.i.j ... +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:

\[ +1 \cdot i' +1 >= 0 \]
\[ +1 \cdot n0' -1 \cdot i' -1 >= 0 \]
\[ i = i +1 \]
\[ n0 = n0 +1 \]
\[ n = n -1 \]
\[ i0' = i0 -1 \]
\[ n0' = n0 -1 \]
\[ ... \]

termination (lmlab)

\[ r(n0, n, i, j) = +24348786 \cdot n0 +16834142 \cdot n +100314562 \cdot i +65646865 \]

Example of termination of nested loops: Bubblesort outer loop

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

\[ +1 \cdot i' +1 >= 0 \]
\[ +1 \cdot n0' -1 \cdot i' -1 >= 0 \]
\[ i = i +1 \]
\[ n0 = n0 +1 \]
\[ n = n -1 \]
\[ i0' = i0 -1 \]
\[ n0' = n0 -1 \]
\[ ... \]

termination (lmlab)

\[ r(n0, n, i, j) = +24348786 \cdot n0 +16834142 \cdot n +100314562 \cdot i +65646865 \]

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)
Termination of a concurrent program

\[
\begin{align*}
1: & \text{while } [x+2 < y] \text{ do} \\
2: & \quad [x := x + 1] \\
3: & \quad \text{interleaving} \\
1: & \text{while } [x+2 < y] \text{ do} \\
2: & \quad [y := y - 1] \\
3: & \quad \text{od}
\end{align*}
\]

\[\text{interleaving} \quad \begin{align*}
& \quad \begin{cases}
& \quad \text{if } ?=0 \text{ then} \\
& \quad \quad x := x + 1 \\
& \quad \text{else if } ?=0 \text{ then} \\
& \quad \quad y := y - 1 \\
& \quad \text{else} \\
& \quad \quad x := x + 1; \\
& \quad \quad y := y - 1 \\
& \quad \end{cases}
\end{align*}\]

\[\text{penbmi: } r(x,y) = 2.537395e+00 \cdot x + (-2.537395e+00 \cdot y + -2.046610e-01)\]

Termination of a fair parallel program

\[
\text{interleaving + scheduler} \\
\begin{align*}
1: & \text{while } [x>0] \text{ do } x := x - 1 \text{ od} \\
2: & \text{while } [y>0] \text{ do } y := y - 1 \text{ od}
\end{align*}
\]

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

\[t := ?; \text{assume } (0 \leq t & t \leq 1); \]
\[s := ?; \text{assume } ((1 \leq s) \& (s \leq m)); \]
\[\text{while } ((x > 0) \text{ | } (y > 0)) \text{ do} \\
\quad \text{if } (t = 1) \text{ then} \\
\quad \quad x := x - 1 \\
\quad \text{else} \\
\quad \quad y := y - 1 \\
\quad \text{fi}; \\
\quad s := s - 1; \\
\quad \text{if } (s = 0) \text{ then} \\
\quad \quad t := 0 \\
\quad \text{else} \\
\quad \quad t := 1 \\
\quad \text{fi}; \\
\quad s := ?; \text{assume } ((1 \leq s) \& (s \leq m)) \\
\quad \text{else} \\
\quad \text{skip} \\
\quad \text{fi}; \\
\text{od};
\]

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

Floyd’s method for invariance

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

- The invariant is initial:
  \[ \forall x : P(x) \Rightarrow I(x) \]

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

???
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

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

Bilinear matrix inequality (BMI) solvers

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

Minimizing \( x^\top Qx + cx \)

Two solvers available under MathLab*:
- PenBMI: M. Kočvara, M. Stingl
- bminbn: J. Löfberg

Common interfaces to these solvers:
- Yalmip: J. Löfberg
Example: linear invariant

Program:

\[
i := 2; j := 0; \text{while } (??) \text{ do}
\]
\[
\begin{align*}
&\text{if } (??) \text{ then } \\
&i := i + 4 \\
&\text{else } \\
&i := i + 2; \\
&j := j + 1 \\
&f_i \\
&\text{od;}
\end{align*}
\]

- Invariant:

\[+2.14678\times 10^{-12} i - 3.12793\times 10^{-10} j + 0.486712 \geq 0\]

- Less natural than \( i^2 - 2j^2 - 2 \geq 0 \)

- Alternative:

  - Determine parameters (\( a \)) by other methods (e.g. random interpretation)
  - Use BMI solvers to check for invariance

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 ??)

Conclusion
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

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 posterior work

- Termination using Lyapunov functions: Roozbehani, Feron & Megrestki (HSCC 2005)

THE END, THANK YOU
ANNEX

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

\[ r(x,x') \leq 0 \quad \forall x,x' \in \mathbb{D}^n : \begin{align*}
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k M_k(x,x')^T \geq 0 \\
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0 \\
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0 \quad \text{Lagrangian relaxation (if lossless)} \\
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0 \\
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0 \\
\exists \lambda \in [1,N] \Rightarrow \mathbb{R}^n : & \sum_{k=1}^N \lambda_k \sigma_k(x) \geq 0
\end{align*} \]

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

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

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

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

\[ \{\text{if (x1)A(x1)^T} \geq 0 \text{ for all } x, \text{ this is the same as (y t)A(y t)^T} \geq 0 \text{ for all } y \text{ and all } t \neq 0 \}
\text{ multiply the original inequality by } t^2 \text{ and call } xt = y \}
\text{ Since the latter inequality holds true for all } x \text{ and all } t \neq 0, \text{ by continuity it holds true for all } x, t, \text{ that is, the original inequality is equivalent to positive semidefiniteness of } A \]
\[ \exists M_0 : \exists \lambda \in [1, N] \mapsto \mathbb{R}_+^* : \left( M_0 - \sum_{k=1}^{N} \lambda_k M_k \right) \succcurlyeq 0 \]

LMI solver provides \( M_0 \) (and \( \lambda \)).

**SOS Relaxation 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: \( p(x, y, \ldots) = z^\top Q z \) 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 \( p(x, y, \ldots) = z^\top Q z \) yields LMI contains on the unknown \( Q : z^\top M(Q) z \succeq 0 \)

---

Since \( Q \succeq 0 \), \( Q \) has a Cholesky decomposition \( L \) which is an upper triangular matrix \( L \) such that \( Q = L^\top L \). It follows that \( p(x) = z^\top Q z = z^\top L^\top L z = (Lz)^\top (Lz) = \sum_{i,j} (L_{ij} z_i z_j) \) (where \( \cdot \) is the vector dot product \( x \cdot y = \sum x_i y_i \)), proving that \( p(x) \) is a sum of squares whence \( \forall z : p(z) \geq 0 \), which eliminates the universal quantification on \( z \).

- 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 \( \exists Q : \forall X : X^\top M(Q) X \succeq 0 \) i.e. \( \exists Q : M(Q) \succeq 0 \) with LMI solver
- Implement with SOStools under MATHLAB* 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