Relational semantics of loops

while B do C od

- $x \in \mathbb{R}/\mathbb{Q}/\mathbb{Z}$: 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
- $[B; C](x, x')$: relational semantics of one loop iteration
- $[B; C](x, x') = \bigwedge_{i=1}^{N} \sigma_{i}(x, x') \geq 0$ (where $\geq$ is $>$, $\geq$ or $=$)

- not a restriction for numerical programs
Example of quadratic form program (factorial)

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

Invariance proof

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

---

Appetiser:
Floyd/Hoare/Naur correctness proof method

\[
\begin{bmatrix}
[|n f N n'| f' N'] & 0
\end{bmatrix}
\begin{bmatrix}
[n f N n'] & +2|n f N n'| f' N' & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 1
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0
0
0
0
-1
1
0
0
0
0
\end{bmatrix}
\begin{bmatrix}
0
0
0
0
0
0
0
0
0
0
\end{bmatrix}
+ 0 = 0
\]
Termination proof

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

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

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

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

Conditional termination

- In general a loop does not terminate for all initial values of the variables
- In that case we can find no rank function!
- We must automatically determine a necessary loop precondition
- We use a iterated forward/backward static analysis . . . with an auxiliary counter counting the number of remaining iterations down to zero

Wine service:
Iterated forward/backward static analysis for conditional termination

Arithmetic mean example, polyhedral abstraction without auxiliary counter

\[
{x\geq y} \\
\text{while } (x <> y) \text{ do} \\
\{x\geq y+2\} \\
\quad x := x - 1; \\
\{x\geq y+1\} \\
\quad y := y + 1 \\
\{x\geq y\} \\
\text{od} \\
\{x=y\}
\]
Arithmetic mean example, polyhedral abstraction with auxiliary counter

\begin{verbatim}
{x=y+2k,x>=y}  
while (x <> y) do  
  {x=y+2k,x>=y+2}  
    k := k - 1;  
  {x=y+2k+2,x>=y+2}  
    x := x - 1;  
  {x=y+2k+1,x>=y+1}  
    y := y + 1  
  {x=y+2k,x>=y}  
od  
{x=y,k=0}  
assume (k = 0)  
{x=y,k=0}
\end{verbatim}

Parametric constraints

- Fix the form of the unknown \((I(x) \geq 0/r(x) \geq 0)\) using parameters \(a\) in the form \(Q(a, x) \geq 0\)
- This is an abstraction
- Examples:
  - \(r(x, y) = a.x + b.y + c\)
  - \(I(x, x') = a.x^2 + b.x.x' + c.x'^2 + d.x + e.x' + f\)

Entrée:
Abstraction to parametric constraints

Solving the constraints

- The invariance [termination] problems have the form:

\[ \exists a : \forall x, x' : \left( [Q(a, x) \geq 0 \land \bigwedge_{k=1}^{n} C_k(x, x') \geq 0] \implies Q'(a, x, x') \geq 0 \right) \]

- Find an algorithm to effectively compute \(a\)!
Problems

In order to compute $a$:
- How to handle $\land$?
- How to get rid of the implication $\Rightarrow$?
  $\rightarrow$ Lagrangian relaxation
- How to get rid of the universal quantification $\forall$?
- How to handle $\land$?
  $\rightarrow$ quantifier elimination (does not scale up)
  $\rightarrow$ mathematical programming

Algorithmically interesting cases

- linear inequalities
  $\rightarrow$ linear programming$^1$
- linear matrix inequalities (LMI)/quadratic forms
- bilinear matrix inequalities (BMI)
  $\rightarrow$ semidefinite programming
- semialgebraic sets
  $\rightarrow$ polynomial quantifier elimination, or
  $\rightarrow$ relaxation with semidefinite programming

$^1$ Already explored for invariants by Sankaranarayanan, Spira, Manna (CAV'03, SAS'04, heuristic solver) and for termination by Podelski & Rybalchenko (VMCAI'03, Lagrange coefficients eliminated by hand to reduce to linear programming so no disjunctions, no tests, etc).

First main course:
Lagrangian relaxation for implication elimination

Example of linear Lagrangian relaxation

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

$\Leftrightarrow$ (soundness)

$\Rightarrow$ (completeness)

border of $A$ parallel to border of $B$
Lagrangian relaxation, formally

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

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

$\Leftarrow$ soundness (Lagrange)

$\Rightarrow$ completeness (lossless)

$\Rightarrow$ incompleteness (lossy)

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

relaxation = approximation, $\lambda_i =$ Lagrange coefficients

Lagrangian relaxation of the constraints

$\exists a : \forall x, x' : [Q(a, x) \geq 0 \land \bigwedge_{k=1}^{n} C_k(x, x') \geq 0]$

$\Rightarrow Q'(a, x, x') \geq 0$

$\Leftarrow$ (is relaxed into)

$\exists a : [\exists \lambda \geq 0] : \exists \lambda_k \geq 0 : \forall x, x'$:

$Q'(a, x, x')[-\lambda.Q(a, x)] - \sum_{k=1}^{n} \lambda_k.C_k(x, x') \geq 0$

$\uparrow$ linear in $a$

$\uparrow$ linear in the $\lambda_k$

$\uparrow$ bilinear in $a \& \lambda$

Lagrangian relaxation, completeness cases

- **Linear case**
  (affine Farkas’ lemma)

- **Linear case with at most 2 quadratic constraints**
  (Yakubovich’s S-procedure)

---

Second main course:
Mathematical programming for quantifier elimination
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 \( f(x) \)

Semidefinite programming

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

[Minimizing \( cx \)]

Where the linear matrix inequality 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 (linear) formulæ we are interested in for programs with linear matricial semantics.

Interior point method for semidefinite programming

– Nesterov & Nemirovskii 1988, polynomial in worst case and good in practice (thousands of variables)

– Various path strategies e.g. “stay in the middle”
Semidefinite programming solvers

Numerous solvers available under 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

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

Recent generalization to bilinear matrix inequalities

- penbmi: M. Kočvara, M. Stingl

Feasibility is:

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

of the form of the (bilinear) formulae we are interested in!

Not enough time for ...
Desert
Invariance and Termination
Examples

Termination of a linear program

\{ y \geq 1 \} \quad \Leftarrow \quad \text{termination precondition determined by iterated forward/backward polyhedral analysis}

while (x y = 2k, x = y)
\text{while (x y = 2k, x = y)
}\begin{align*}
k &:= k - 1; \\
x &:= x - 1; \\
y &:= y + 1
\end{align*}
\text{od}
{assert (k = 0)}

termination precondition determined by iterated forward/backward polyhedral analysis

Termination of the arithmetic mean

\{ x = y + 2k, x = y \}
while (x y = 2k, x = y)
\text{while (x y = 2k, x = y)
}\begin{align*}
k &:= k - 1; \\
x &:= x - 1; \\
y &:= y + 1
\end{align*}
\text{od}
{assert (k = 0)}

Floyd’s proposal \( r(x, y, q, r) = x - q \) is more intuitive but requires to discover the nonlinear loop invariant \( x = r + qr \).
Termination of a quadratic program: factorial

\{true\} ← termination precondition determined by iterated forward/backward polyhedral analysis

\begin{verbatim}
n := 0;
f := 1;
while (f <= N) do
  n := n + 1;
f := n * f
od
\end{verbatim}

\texttt{sedumi} (with feasibility radius of 1.0e+3):

\begin{verbatim}
r(n,f,N) = -9.993462e-01.n +1.617225e-04.f +2.688476e+02.N +8.745232e+02
\end{verbatim}

Loop body with tests

\begin{verbatim}
while (x < y) do
  if (i >= 0) then
    x := x+i+1
  else
    y := y+i
  fi
od
\end{verbatim}

\texttt{lmilab}:

\begin{verbatim}
r(i,x,y) = -2.252791e-09.i -4.355697e+07.x +4.355697e+07.y +5.502903e+08
\end{verbatim}

Quadratic termination of linear loop

\{n>=0\} ← termination precondition determined by iterated forward/backward polyhedral analysis

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

\texttt{sdplr} (with feasibility radius of 1.0e+3):

\begin{verbatim}
r(n,i,j) = +7.024176e-04.n^2 +4.394909e-05.n.i ...
+2.809222e-03.n.j -2.809222e-03.n.j ...
+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
\end{verbatim}

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

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

penbmi: r(x,y) = 2.537395e+00.x+2.537395e+00.y+-2.046610e-01
```

Termination of a fair parallel program

```
[[ while [(x>0)&(y>0) do x := x - 1] od ]] interleaving
[[ while [(x>0)&(y>0) do y := y - 1] od ]] 

{n>=1} ← termination precondition determined by iterated forward/backward polyhedral analysis
assume (0 <= t & t <= 1);

s := ?;
assume ((1 <= s) & (s <= m));
while ((x > 0) | (y > 0)) do
    if (t = 1) then
        x := x - 1
    else
        y := y - 1
    fi;
    s := s - 1;
fi;

penbmi: r(x,y,m,s,t) = 1.000611e+00.x +1.000468e+00.y +2.855769e-02.m -3.929197e-07.s +6.588027e-06.t +9.998392e+03
```

Semidefinite programming relaxation for polynomial programs

```
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 SOStool to relax in semidefinite programming form.

SOSTool+SeDuMi:

r(x) = 1.222356e-13.x + 1.406392e+00
```

When constraint resolution fails...

Infeasibility of the constraints does not mean “non-termination” but simply failure:
- There can be a rank function of a different form (e.g. quadratic while looking for a linear one),
- The solver may have failed (e.g. add a shift).
**Coffee:**

**Conclusion**

**Numerical errors**

- LMI solvers do numerical computations with rounding errors, shifts, etc.
- Rank function is subject to numerical errors.
- The hard point is to discover a candidate for the rank function.
- Much less difficult, when it is known, to re-check for satisfaction (e.g. by static analysis).

**Invariance for Euclidian division**

```plaintext
assume (y > 0);
q := 0;
r := x;
while (y <= r) do
  r := -y + r;
  q := q + 1
od

yalmip bmi:
1.337645e-04*x+2.484973e-04*q*y+1.588933e-03*r >= 0
which is not false!
```

---

**Digestif:**

**Questions**

---
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

I hope you had a good and relaxed semantics lunch