Static Analysis by Policy Iteration on Relational Domains

S. Gaubert, E. Goubault, A. Taly, S. Zennou

29 march 2007

Static Analysis

Static Anlysis

(1) Parse C code, produce semantic equations

(2) Provide abstractions of operators and equations

(3) Solve the abstract equations (costly step!)

```
0    i = 150; j = 175;
1    while (j >= 100){
2      i++;
3      if (j<= i){
4          i = i - 1;
5          j = j - 2;
6      }
7  }
```

$$
\begin{aligned}
M_0 &= \mathit{context\_initialization} \\
M_1 &= \mathit{Assign}_{\,i \leftarrow 150,\ j \leftarrow 175}(M_0) \\
M_2 &= (M_1 \sqcup M_6) \sqcap (j \geq 100) \\
M_3 &= \mathit{Assign}_{\,i \leftarrow i+1}(M_2) \\
M_4 &= M_3 \sqcap (j \leq i) \\
M_5 &= \mathit{Assign}_{\,i \leftarrow i-1,\ j \leftarrow j-2}(M_4) \\
M_6 &= (M_5 \sqcap (j > i) \sqcup M_7 \\
M_7 &= (M_2 \sqcup M_8) \sqcap (j < 100)
\end{aligned}
$$

Static Analysis by Abstract Interpretation

- D_a and D_c are complete lattices
- F_c is the best abstraction for F_a

$$\begin{array}{lccc} \text{Concrete Domain} & (D_c, \sqsubseteq_c) & \xrightarrow{F_c} & (D_c, \sqsubseteq_c) \\ & \alpha\downarrow\uparrow\gamma & & \alpha\downarrow\uparrow\gamma \\ \text{Abstract Domain} & (D_a, \sqsubseteq_a) & \xrightarrow{F_a} & (D_a, \sqsubseteq_a) \end{array}$$

Galois connection [Cousot and Cousot:77]

$\alpha(x) \sqsubseteq_a y \iff x \sqsubseteq_c \gamma(y)$
$\implies \gamma(lfp(F_a))$ is an overapproximation of $lfp(F_c)$

Linear Relational Abstract Domains 1/2

Program variables $x_i \in \mathbb{R}$ $(1 \leq i \leq n)$

Zone Abstract Domain [Mine:01]

- Vector of intervals representing bounds for $x_i - x_j$, $\forall i, j$.
- $\gamma([0,100],[0,10],[-10,5]) = \{(i,j) \mid 0 \leq i \leq 100,$ $0 \leq j \leq 10,\ -10 \leq i - j \leq 5\}$

TCM Domain [Manna : 05]

- Parametrized by a set of vectors called $TCM = \{e_1, \ldots e_n\}$ which is fixed apriori
- Represented as a vector, each coordinate keeps track of the upper bound of $e_i.(x_1, \ldots, x_n)^T$
- For the TCM $\{(1,0),(0,-1),(2,1)\}$, $\gamma(5,10,3) = \{(i,j) \mid i \leq 5,\ j \geq 10,\ 2i + j \leq 3\}$

Closure

Two different elements may have the same concretization

Example

$\gamma(0 \leq i \leq 100, 0 \leq j \leq 10, -10 \leq i - j \leq \infty]) = \gamma(0 \leq i \leq 100, 0 \leq j \leq 10, -10 \leq i - j \leq 100)$

Closure

- $a \equiv b$ iff $\gamma(a) = \gamma(b)$.
- Closure$(a) = a^* = \sqsubseteq$-minimal element of eqv class of a
- Closure$([0, 100], [0, 10], [-10, \infty]) = ([0, 100], [0, 10], [-10, 100])$
- *Tightest bound* for each constraint.

Kleene Iteration technique

Principle

- Iteration 0 : $M_0^0 = M_1^0 = \ldots = M_p^0 = \bot$
- $k+1$ Iteration :
  $(M_0^{k+1}, \ldots, M_p^{k+1}) = (M_0^k, \ldots, M_p^k) \cup F(M_0^k, \ldots, M_p^k)$
- To ensure convergence, after some iterations replace $\cup$ by Widening operator ∇

Problems

- Closure and Widening are incompatible
- Widening $\Longrightarrow$ Loss of Relations !!!

Kleene Iteration technique

Principle

- Iteration 0 : $M_0^0 = M_1^0 = \ldots = M_p^0 = \bot$
- $k+1$ Iteration :
  $(M_0^{k+1}, \ldots, M_p^{k+1}) = (M_0^k, \ldots, M_p^k) \cup F(M_0^k, \ldots, M_p^k)$
- To ensure convergence, after some iterations replace $\cup$ by Widening operator ∇

Problems

- Closure and Widening are incompatible
- Widening $\Longrightarrow$ Loss of Relations !!!

Plan

Policy Iteration

Express a monotone map F as an infimum of monotone maps $g_1, \ldots, g_n$ (called *policies*) over a complete lattice L.

Selection Property

F satisfies the selection property if

(1) $F(x) = (inf\{g_1(x), g_2(x), \ldots\ g_n(x)\})^*$

(2) $\forall x \exists g\ F(x) = g(x)$

Theorem

If F satisfies the selection property then the least fixed point of F (denoted by F^-) is given by $F^- = \inf\{(g_1^-)^*, (g_2^-)^*, \ldots (g_n^-)^*\}$

Policy Iteration

Express a monotone map F as an infimum of monotone maps $g_1, \ldots, g_n$ (called *policies*) over a complete lattice L.

Selection Property

F satisfies the selection property if

(1) $F(x) = (inf\{g_1(x), g_2(x), \ldots\ g_n(x)\})^*$

(2) $\forall x \exists g\ F(x) = g(x)$

Theorem

If F satisfies the selection property then the least fixed point of F (denoted by F^-) is given by $F^- = \inf\{(g_1^-)^*, (g_2^-)^*, \ldots (g_n^-)^*\}$

Algorithm

$k \leftarrow 1, g_1 \leftarrow$ *initial_policy*$(\mathcal{G})$
while true **do**
  $x_k \leftarrow (g_k^-)^*$
  if $x_k = F(x_k)$ **then**
    return x_k
  else
    find g such that $F(x_k) = g(x_k)$
    $k \leftarrow k + 1, g_k \leftarrow g$
  end if
end while

Theorem

- Algorithm returns a fixed point of F
- lfps of the successive policies decrease. $(g_{k+1}^-)^* \sqsubset (g_k^-)^*$
- If F is *NonExpansive* ($\| F(x) - F(y) \|_\infty \leq \| x - y \|_\infty$) then algorithm returns lfp of F (CAV:05)

Algorithm

$k \leftarrow 1, g_1 \leftarrow$ *initial_policy*$(\mathcal{G})$
while true **do**
  $x_k \leftarrow (g_k^-)^*$
  if $x_k = F(x_k)$ **then**
    return x_k
  else
    find g such that $F(x_k) = g(x_k)$
    $k \leftarrow k + 1, g_k \leftarrow g$
  end if
end while

Theorem

- Algorithm returns a fixed point of F
- lfps of the successive policies decrease. $(g_{k+1}^-)^* \sqsubset (g_k^-)^*$
- If F is *NonExpansive* ($\| F(x) - F(y) \|_\infty \leq \| x - y \|_\infty$) then algorithm returns lfp of F (CAV:05)

Finding a Policy Selection

Intersection Policy

- Intervals: $[a, b] \sqcap_{Pol} [c, d] = inf\{[a, b], [a, d], [c, b], [c, d]\}$
- For Zones, extend to vector of intervals.
- Finitely many policies and selection property is satisfied.

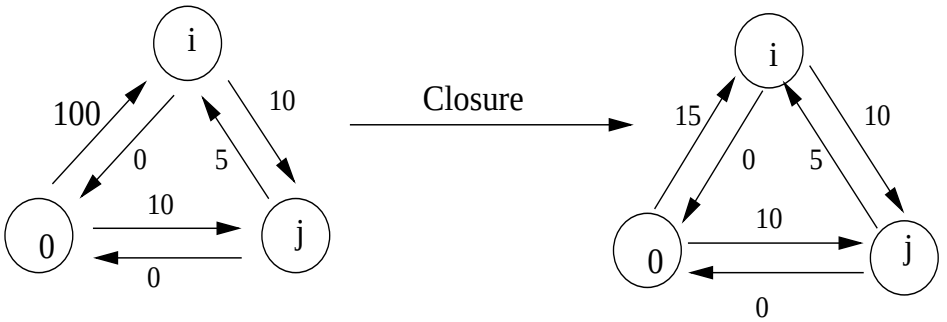
Example

$$([1, 2], [a, b]) \sqcap ([c, d], [3, 4]) = inf\{ \begin{array}{l} ([1, 2], [a, b]), \\ ([c, 2], [a, b]), \\ \ldots \\ ([1, 2], [3, 4]), \\ \ldots \\ ([c, d], [3, 4])\} \end{array}$$

Closure Policy

Zones $\Longleftrightarrow$ Weighted directed graph

- $x_1, \ldots, x_n$ form the nodes
- $x_j \xrightarrow{c} x_i$ iff $x_i - x_j \leq c$



Closure

- $c^* =$ shortest path from x_j to x_i in the graph.
- $c^* = \inf(\mathcal{P}_{ij})$ where
  $\mathcal{P}_{ij} =$ Set of weights of all paths from i to j
- Policy = Weight of particular path from i to j
- Extend to all edges

Policy Selection for TCMS

- **Intersection**
  $(e.X^T \leq c_1) \wedge (e.X^T \leq c_2) = (e.X^T \leq inf(c_1, c_2))$.
- **Closure**
  - Finding *tightest bound* for every vector e in the TCM.
  - Amounts to solving the Linear Program :
    Sup $e.X^T$ subject to $\forall\ i\ e_i.X^T \leq c_i$
  - Dual: Inf $\lambda^T c$ subject to $e_1.\lambda_1 +, \ldots, +e_n\lambda_n = e$
  - Solution at extreme point
  - Closure = Inf(Set of values at all extreme points)

Algorithmic Issues

lfp of a policy

- Apply policy selection to meet and closure operations of F.
- Each policy g is now convex
- $g^- = \inf\{x | g(x) \sqsubseteq x\}$
- Express $g(x) \sqsubseteq x$ as a set of linear constraints
- Solve using Linear Programming.

Initial Policy Heuristic

- Intersection :
  - Choose constant as opposed to ∞ or variable.
  - Intial Policy for $[a, 10] \sqcap [4, \infty]$ is $[4, 10]$
- Closure :
  - Choose Identity policy ie length 1 path.

Plan

Benchmarks

```
/* test1 */
i = 1; j = 10;
while (i <= j){
  i = i + 2; j = j - 1;
}
```

```
/* test2 */
i = 150 ; j = 175;
while (j >= 100){
  i++;
  if( j - i <= 0 ){
      j = j - 2; i = i - 1;
  }
}
```

```
/* test3 */
i = 150; j = 175;
while ( 100 <= j && j <= 300){
  i++;
  if( j <= i ){
      j = j - 2; i = i - 1;
  }
}
```

```
/* test4 */
i = 0 ; j = -100;
while (i <= 100){
    i = i + 1;
    while ( j <= 19 ){
      j = i + j;
    }
  }
```

```
/* ex5 */
i = 0; j = 100;
k = 1000; l = 10000;
m = 100000;
while (i < 1000)
  i = i+1;
while (j  < 1000)
  j = j+k;
while (k > 100)
  k = k-j;
while (l > 1000)
  l = l+k;
while (m > 1)
  m = m-l;
```

Comparing the invariants computed

Programme	*Pol + zones*	*Oct. (A.Miné)*	*TCMs oct (LPInv)*
`test1`	$\begin{cases} 1 \leq i \leq 12 \\ 0 \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$	$\begin{cases} 6 \leq i \leq 12 \\ \frac{9}{2} \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$	$\begin{cases} 6 \leq i \leq 13 \\ 4 \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$
`test2`	$\begin{cases} 150 \leq i \leq 174 \\ 98 \leq j \leq 99 \\ -76 \leq j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ 98 \leq j \leq 99 \\ j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ j \leq 99 \end{cases}$
`test3`	$\begin{cases} 150 \leq i \leq 174 \\ 98 \leq j \leq 99 \\ -76 \leq j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ 98 \leq j \leq 99 \\ j - i \leq -51 \end{cases}$	$\begin{cases} j \leq 99 \\ j - i \leq -51 \end{cases}$
`test4`	$\begin{cases} i = 101 \\ 1 \leq j \\ -100 \leq j - i \end{cases}$	$\begin{cases} 101 \leq i \leq 120 \\ 1 \leq j \leq 139 \\ -100 \leq j - i \leq 19 \end{cases}$	$\{ \; 101 \leq i$
`ex5`	$\begin{cases} i = 1000 \\ 1000 \leq j \\ k \leq 100 \\ l \leq 1000 \\ m \leq 1 \\ k - i \leq -900 \\ l - i \leq 0 \\ m - i \leq -999 \\ k - j \leq -99 \\ l - j \leq 0 \\ m - j \leq -999 \end{cases}$	$\begin{cases} i = 1000 \\ 1000 \leq j \leq 1999 \\ -1898 \leq k \leq -1 \\ -897 \leq l \leq 1000 \\ m \leq 1 \\ 0 \leq j - i \leq 999 \\ -2898 \leq k - i \leq -1001 \\ -3897 \leq k - j \leq -1001 \\ -1897 \leq l - i \leq 0 \\ -2896 \leq l - j \leq 0 \\ 1001 \leq l - k \leq 2898 \\ m - i \leq -999 \\ m - j \leq -999 \\ m - k \leq 1899 \\ m - l \leq 898 \end{cases}$	$\begin{cases} i = 1000 \\ 1000 \leq j \leq 1999 \\ k \leq -1 \\ l \leq 1000 \\ m \leq 1 \\ 1001 \leq l - k \end{cases}$

Experimental Results

Elementary Opertions and Iterations

Program	Elem op Pol	Elem op Oct	Elem op TCM	Iter Pol	Iter Oct	Iter TCM
test1	20	1132	14014	2	7	6
test3	34	1364	62348	1	12	16
test4	68	906	50940	3	4	16
ex5	392	49370	3.10^7	1	23	20

GLPK calls

Program	no GLPK Pol	Closure oct.(asgn)	no GLPK TCM
test1	113	138(14)	88(11.14/1.28)
test3	96	333(16)	282(14/1.13)
test4	124	220(13)	302(11.75/1.20)
ex5	659	394(24)	3.10^7

Conclusion

- Novel method for computing the fixed point in Linear relational domains
  - No widening used
  - lfp of each policy obtained using Linear Programming.
- Experimental Results
  - Fewer iterations and elementary operations required
  - Precise fixed points even when compared to more informative domains.
  - Hope for better results even with Policy Iteration on TCMs and large examples
- Future Work
  - Gaurenteedly finding the lfp by policy iteration.
  - Initial Policy Heuristic
  - Incremental Analysis

Conclusion

- Novel method for computing the fixed point in Linear relational domains
  - No widening used
  - lfp of each policy obtained using Linear Programming.
- Experimental Results
  - Fewer iterations and elementary operations required
  - Precise fixed points even when compared to more informative domains.
  - Hope for better results even with Policy Iteration on TCMs and large examples
- Future Work
  - Gaurenteedly finding the lfp by policy iteration.
  - Initial Policy Heuristic
  - Incremental Analysis

Conclusion

- Novel method for computing the fixed point in Linear relational domains
  - No widening used
  - lfp of each policy obtained using Linear Programming.
- Experimental Results
  - Fewer iterations and elementary operations required
  - Precise fixed points even when compared to more informative domains.
  - Hope for better results even with Policy Iteration on TCMs and large examples
- Future Work
  - Gaurenteedly finding the lfp by policy iteration.
  - Initial Policy Heuristic
  - Incremental Analysis

Thank You

Benchmarks

```
/* test1 */
i = 1; j = 10;
while (i <= j){
  i = i + 2; j = j - 1;
}
```

```
/* test2 */
i = 150 ; j = 175;
while (j >= 100){
  i++;
  if( j - i <= 0 ){
      j = j - 2; i = i - 1;
  }
}
```

```
/* test3 */
i = 150; j = 175;
while ( 100 <= j && j <= 300){
  i++;
  if( j <= i ){
      j = j - 2; i = i - 1;
  }
}
```

```
/* test4 */
i = 0 ; j = -100;
while (i <= 100){
    i = i + 1;
    while ( j <= 19 ){
      j = i + j;
    }
  }
```

```
/* ex5 */
i = 0; j = 100;
k = 1000; l = 10000;
m = 100000;
while (i < 1000)
  i = i+1;
while (j  < 1000)
  j = j+k;
while (k > 100)
  k = k-j;
while (l > 1000)
  l = l+k;
while (m > 1)
  m = m-l;
```

Comparing the invariants computed

Programme	*Pol + zones*	*Oct. (A.Miné)*	*TCMs oct (LPInv)*
`test1`	$\begin{cases} 1 \leq i \leq 12 \\ 0 \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$	$\begin{cases} 6 \leq i \leq 12 \\ \frac{9}{2} \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$	$\begin{cases} 6 \leq i \leq 13 \\ 4 \leq j \leq 10 \\ -3 \leq j - i \leq -1 \end{cases}$
`test2`	$\begin{cases} 150 \leq i \leq 174 \\ 98 \leq j \leq 99 \\ -76 \leq j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ 98 \leq j \leq 99 \\ j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ j \leq 99 \end{cases}$
`test3`	$\begin{cases} 150 \leq i \leq 174 \\ 98 \leq j \leq 99 \\ -76 \leq j - i \leq -51 \end{cases}$	$\begin{cases} 150 \leq i \\ 98 \leq j \leq 99 \\ j - i \leq -51 \end{cases}$	$\begin{cases} j \leq 99 \\ j - i \leq -51 \end{cases}$
`test4`	$\begin{cases} i = 101 \\ 1 \leq j \\ -100 \leq j - ia \end{cases}$	$\begin{cases} 101 \leq i \leq 120 \\ 1 \leq j \leq 139 \\ -100 \leq j - i \leq 19 \end{cases}$	$\{ \; 101 \leq i$
`ex5`	$\begin{cases} i = 1000 \\ 1000 \leq j \\ k \leq 100 \\ l \leq 1000 \\ m \leq 1 \\ k - i \leq -900 \\ l - i \leq 0 \\ m - i \leq -999 \\ k - j \leq -99 \\ l - j \leq 0 \\ m - j \leq -999 \end{cases}$	$\begin{cases} i = 1000 \\ 1000 \leq j \leq 1999 \\ -1898 \leq k \leq -1 \\ -897 \leq l \leq 1000 \\ m \leq 1 \\ 0 \leq j - i \leq 999 \\ -2898 \leq k - i \leq -1001 \\ -3897 \leq k - j \leq -1001 \\ -1897 \leq l - i \leq 0 \\ -2896 \leq l - j \leq 0 \\ 1001 \leq l - k \leq 2898 \\ m - i \leq -999 \\ m - j \leq -999 \\ m - k \leq 1899 \\ m - l \leq 898 \end{cases}$	$\begin{cases} i = 1000 \\ 1000 \leq j \leq 1999 \\ k \leq -1 \\ l \leq 1000 \\ m \leq 1 \\ 1001 \leq l - k \end{cases}$