Script **generated by TTT**

Title: Seidl: Programmoptimierung (25.11.2013)

Date: Mon Nov 25 14:15:54 CET 2013

Duration: 91:13 min

Pages: 34

Problem:

→ The solution can be computed with RR-iteration — after about 42 rounds :-(

→ On some programs, iteration may never terminate :-((

Idea 1: Widening

- Accelerate the iteration — at the prize of imprecision :-)
- Allow only a bounded number of modifications of values !!!

... in the Example:

- dis-allow updates of interval bounds in $\mathbb{Z}$ ...

$\Longrightarrow$ a maximal chain:

$$[3, 17] \sqsubset [3, +\infty] \sqsubset [-\infty, +\infty]$$

Formalization of the Approach:

Let $x_i \sqsupseteq f_i\,(x_1, \ldots, x_n)\,, \quad i = 1, \ldots, n$ (1)

denote a system of constraints over $\mathbb{D}$ where the f_i are not necessarily monotonic.

Nonetheless, an accumulating iteration can be defined. Consider the system of equations:

$$x_i = x_i \sqcup f_i\,(x_1, \ldots, x_n)\,, \quad i = 1, \ldots, n \qquad (2)$$

We obviously have:

(a) $\underline{x}$ is a solution of (1) iff $\underline{x}$ is a solution of (2).

(b) The function $G : \mathbb{D}^n \to \mathbb{D}^n$ with $G\,(x_1, \ldots, x_n) = (y_1, \ldots, y_n)\,, \quad y_i = x_i \sqcup f_i\,(x_1, \ldots, x_n)$ is increasing, i.e., $\underline{x} \sqsubseteq G\,\underline{x}$ for all $\underline{x} \in \mathbb{D}^n$.

(c) The sequence $G^k\,\underline{\perp}\,,\quad k \geq 0,$ is an ascending chain:

$$\underline{\perp} \sqsubseteq G\,\underline{\perp} \sqsubseteq \ldots \sqsubseteq G^k\,\underline{\perp} \sqsubseteq \ldots$$

(d) If $G^k\,\underline{\perp} = G^{k+1}\,\underline{\perp} = \underline{y}$, then $\underline{y}$ is a solution of (1).

(e) If $\mathbb{D}$ has infinite strictly ascending chains, then (d) is not yet sufficient ...

but: we could consider the modified system of equations:

$$x_i = x_i \sqcup f_i(x_1, \ldots, x_n)\,, \quad i = 1, \ldots, n \qquad (3)$$

for a binary operation widening:

$$\sqcup : \mathbb{D}^2 \to \mathbb{D} \qquad \text{with} \qquad v_1 \sqcup v_2 \sqsubseteq v_1 \sqcup v_2$$

(RR)-iteration for (3) still will compute a solution of (1) :-)

... for Interval Analysis:

- The complete lattice is: $\mathbb{D}_{\mathbb{I}} = (Vars \to \mathbb{I})_\perp$
- the widening $\sqcup$ is defined by:

$$\perp \sqcup D = D \sqcup \perp = D \qquad \text{and for} \quad D_1 \neq \perp \neq D_2:$$

$$\begin{array}{rcll}
(D_1 \sqcup D_2)\, x & = & (D_1\, x) \sqcup (D_2\, x) & \text{where} \\
{[l_1, u_1]} \sqcup [l_2, u_2] & = & [l, u] & \text{with} \\
l & = & \begin{cases} l_1 & \text{if} \quad l_1 \leq l_2 \\ -\infty & \text{otherwise} \end{cases} & \\
u & = & \begin{cases} u_1 & \text{if} \quad u_1 \geq u_2 \\ +\infty & \text{otherwise} \end{cases} &
\end{array}$$

$\Longrightarrow$ $\sqcup$ is not commutative !!!

Example:

$$\begin{array}{rcl}
[0,2] \sqcup [1,2] & = & [0,2] \\
[1,2] \sqcup [0,2] & = & [-\infty, 2] \\
[1,5] \sqcup [3,7] & = & [1, +\infty]
\end{array}$$

→ Widening returns larger values more quickly.

→ It should be constructed in such a way that termination of iteration is guaranteed :-)

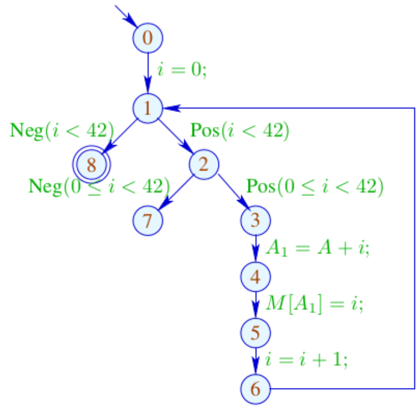
→ For interval analysis, widening bounds the number of iterations by:

$$\#points \cdot (1 + 2 \cdot \#Vars)$$

Conclusion:

- In order to determine a solution of (1) over a complete lattice with infinite ascending chains, we define a suitable widening and then solve (3) :-)
- Caveat: The construction of suitable widenings is a dark art !!!
  Often $\sqcup$ is chosen dynamically during iteration such that
  - → the abstract values do not get too complicated;
  - → the number of updates remains bounded ...

Our Example:

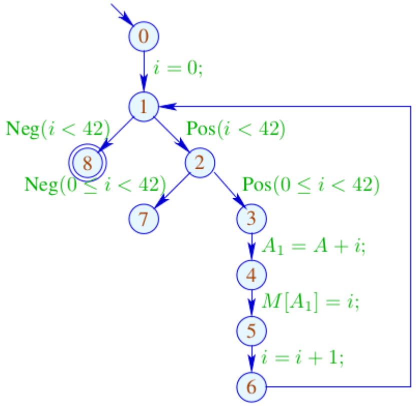


	1	
	l	u
0	$-\infty$	$+\infty$
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	1	1
7	$\perp$	
8	$\perp$	

Our Example:

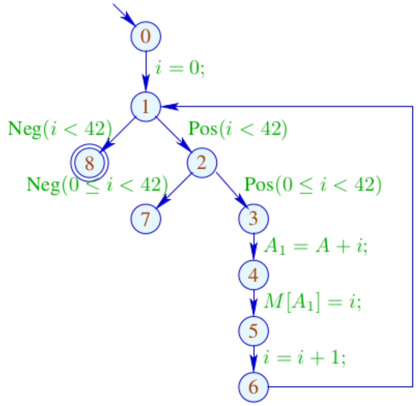
$$[0,0] \sqcup [0,1] = [0,\infty]$$

$$[0,0] \sqcup [0,1] = [0,\infty]$$



	1		2		3	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$		
1	0	0	0	$+\infty$		
2	0	0	0	$+\infty$		
3	0	0	0	$+\infty$		
4	0	0	0	$+\infty$	dito	
5	0	0	0	$+\infty$		
6	1	1	1	$+\infty$		
7	$\bot$		42	$+\infty$		
8	$\bot$		42	$+\infty$		

Our Example:



	1		2		3	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$		
1	0	0	0	$+\infty$		
2	0	0	0	$+\infty$		
3	0	0	0	$+\infty$		
4	0	0	0	$+\infty$	dito	
5	0	0	0	$+\infty$		
6	1	1	1	$+\infty$		
7	$\bot$		42	$+\infty$		
8	$\bot$		42	$+\infty$		

... obviously, the result is disappointing :-(

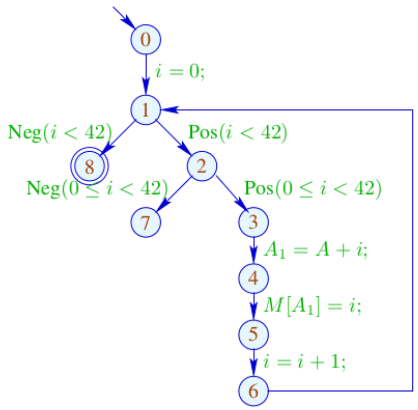
Idea 2:

In fact, acceleration with $\sqcup$ need only be applied at sufficiently many places!

A set I is a loop separator, if every loop contains at least one point from I :-)

If we apply widening only at program points from such a set I, then RR-iteration still terminates !!!

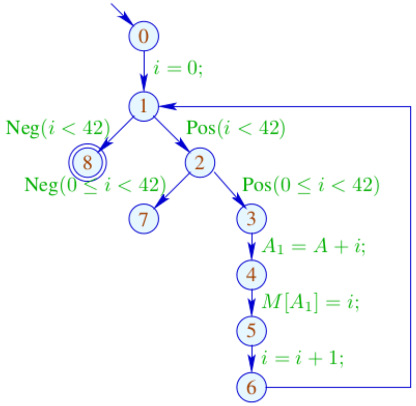
In our Example:



$$I_1 = \{1\} \quad \text{or:}$$
$$I_2 = \{2\} \quad \text{or:}$$
$$I_3 = \{3\}$$

$[0,0] \sqcup [0,1] = [0,\infty]$

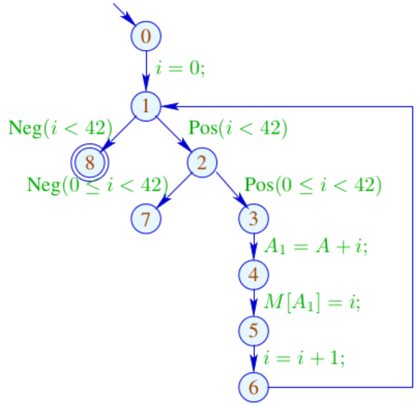
The Analysis with $I = \{1\}$:



	1		2		3	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$		
1	0	0	0	$+\infty$		
2	0	0	0	41		
3	0	0	0	41		
4	0	0	0	41	dito	
5	0	0	0	41		
6	1	1	1	42		
7	$\perp$		$\perp$			
8	$\perp$		42	$+\infty$		

$[0,0] \sqcup [1,42] = [0,42]$

The Analysis with $I = \{2\}$:



	1		2		3		4
	l	u	l	u	l	u	
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$	
1	0	0	0	1	0	42	
2	0	0	0	$+\infty$	0	$+\infty$	
3	0	0	0	41	0	41	
4	0	0	0	41	0	41	dito
5	0	0	0	41	0	41	
6	1	1	1	42	1	42	
7	$\perp$		42	$+\infty$	42	$+\infty$	
8	$\perp$		$\perp$		42	42	

Discussion:

- Both runs of the analysis determine interesting information :-)
- The run with $I = \{2\}$ proves that always $i = 42$ after leaving the loop.
- Only the run with $I = \{1\}$ finds, however, that the outer check makes the inner check superfluous :-(

How can we find a suitable loop separator I ???

Idea 3: Narrowing

Let $\underline{x}$ denote any solution of (1), i.e.,

$$x_i \sqsupseteq f_i\,\underline{x}\,, \qquad i = 1, \ldots, n$$

Then for monotonic f_i ,

$$\underline{x} \sqsupseteq F\,\underline{x} \sqsupseteq F^2\,\underline{x} \sqsupseteq \ldots \sqsupseteq F^k\,\underline{x} \sqsupseteq \ldots$$

// Narrowing Iteration

Idea 3: Narrowing

Let $\underline{x}$ denote any solution of (1) , i.e.,

$$x_i \sqsupseteq f_i \, \underline{x} \ , \qquad i = 1, \ldots, n$$

Then for monotonic f_i ,

$$\underline{x} \sqsupseteq F \, \underline{x} \sqsupseteq F^2 \, \underline{x} \sqsupseteq \ldots \sqsupseteq F^k \, \underline{x} \sqsupseteq \ldots$$

// Narrowing Iteration

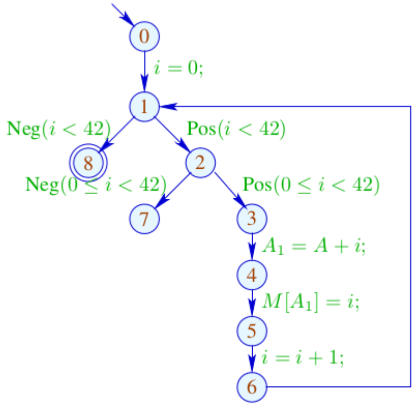
Every tuple $F^k \, \underline{x}$ is a solution of (1) :-)

$\Longrightarrow$

Termination is no problem anymore:
we stop whenever we want :-))

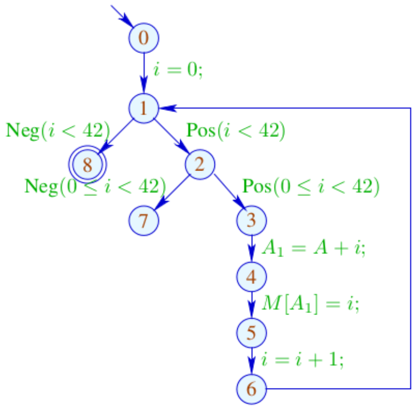
// The same also holds for RR-iteration.

Narrowing Iteration in the Example:



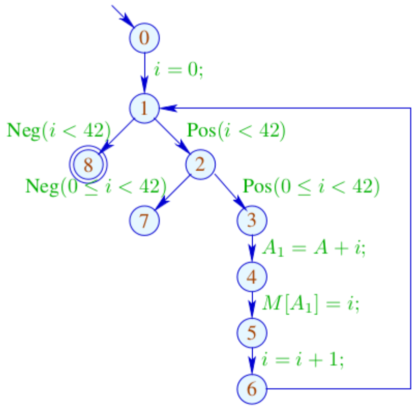
	0	
	l	u
0	$-\infty$	$+\infty$
1	0	$+\infty$
2	0	$+\infty$
3	0	$+\infty$
4	0	$+\infty$
5	0	$+\infty$
6	1	$+\infty$
7	42	$+\infty$
8	42	$+\infty$

Narrowing Iteration in the Example:



	0		1	
	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$
2	0	$+\infty$	0	41
3	0	$+\infty$	0	41
4	0	$+\infty$	0	41
5	0	$+\infty$	0	41
6	1	$+\infty$	1	42
7	42	$+\infty$	$\perp$	
8	42	$+\infty$	42	$+\infty$

Narrowing Iteration in the Example:



	0		1		2	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$	0	42
2	0	$+\infty$	0	41	0	41
3	0	$+\infty$	0	41	0	41
4	0	$+\infty$	0	41	0	41
5	0	$+\infty$	0	41	0	41
6	1	$+\infty$	1	42	1	42
7	42	$+\infty$	$\perp$		$\perp$	
8	42	$+\infty$	42	$+\infty$	42	42

Discussion:

- $\rightarrow$ We start with a safe approximation.
- $\rightarrow$ We find that the inner check is redundant :-)
- $\rightarrow$ We find that at exit from the loop, always $i = 42$:-))
- $\rightarrow$ It was not necessary to construct an optimal loop separator :-)))

Last Question:

Do we have to accept that narrowing may not terminate ???

4. Idea: Accelerated Narrowing

Assume that we have a solution $\underline{x} = (x_1, \ldots, x_n)$ of the system of constraints:

$$x_i \sqsupseteq f_i\,(x_1, \ldots, x_n)\,, \quad i = 1, \ldots, n \qquad (1)$$

Then consider the system of equations:

$$x_i \;=\; x_i \sqcap f_i\,(x_1, \ldots, x_n)\,, \quad i = 1, \ldots, n \qquad (4)$$

Obviously, we have for monotonic f_i : $H^k\,\underline{x} \;=\; F^k\,\underline{x}$:-)

where $H\,(x_1, \ldots, x_n) = (y_1, \ldots, y_n)\,, \quad y_i = x_i \sqcap f_i\,(x_1, \ldots, x_n)$.

In (4) , we replace $\sqcap$ durch by the novel operator $\overline{\sqcap}$ where:

$$a_1 \sqcap a_2 \;\sqsubseteq\; a_1 \,\overline{\sqcap}\, a_2 \;\sqsubseteq\; a_1$$

... for Interval Analysis:

We preserve finite interval bounds :-)

Therefore, $\bot \,\overline{\sqcap}\, D \;=\; D \,\overline{\sqcap}\, \bot \;=\; \bot$ and for $D_1 \neq \bot \neq D_2$:

$$\begin{array}{rcl}
(D_1 \,\overline{\sqcap}\, D_2)\, x & = & (D_1\, x) \,\overline{\sqcap}\, (D_2\, x) \qquad \text{where} \\
{[l_1, u_1]} \,\overline{\sqcap}\, [l_2, u_2] & = & [l, u] \qquad \text{with} \\
l & = & \begin{cases} l_2 & \text{if} \quad l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases} \\
u & = & \begin{cases} u_2 & \text{if} \quad u_1 = \infty \\ u_1 & \text{otherwise} \end{cases}
\end{array}$$

$\Longrightarrow$ $\overline{\sqcap}$ is not commutative !!!

$[0,\infty] \,\overline{\sqcap}\, [5,20] =$

$[0,20]$

... for Interval Analysis:

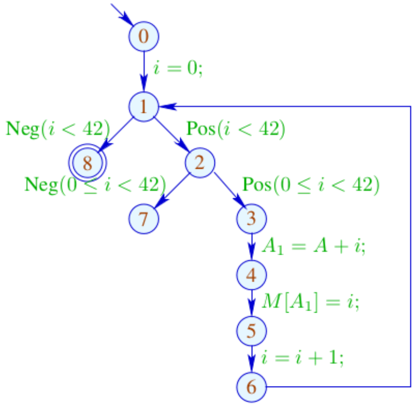
We preserve finite interval bounds :-)

Therefore, $\bot \,\overline{\sqcap}\, D \;=\; D \,\overline{\sqcap}\, \bot \;=\; \bot$ and for $D_1 \neq \bot \neq D_2$:

$$\begin{array}{rcl}
(D_1 \,\overline{\sqcap}\, D_2)\, x & = & (D_1\, x) \,\overline{\sqcap}\, (D_2\, x) \qquad \text{where} \\
{[l_1, u_1]} \,\overline{\sqcap}\, [l_2, u_2] & = & [l, u] \qquad \text{with} \\
l & = & \begin{cases} l_2 & \text{if} \quad l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases} \\
u & = & \begin{cases} u_2 & \text{if} \quad u_1 = \infty \\ u_1 & \text{otherwise} \end{cases}
\end{array}$$

$\Longrightarrow$ $\overline{\sqcap}$ is not commutative !!!

Accelerated Narrowing in the Example:



	0		1		2	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$	0	42
2	0	$+\infty$	0	41	0	41
3	0	$+\infty$	0	41	0	41
4	0	$+\infty$	0	41	0	41
5	0	$+\infty$	0	41	0	41
6	1	$+\infty$	1	42	1	42
7	42	$+\infty$	$\perp$		$\perp$	
8	42	$+\infty$	42	$+\infty$	42	42

Discussion:

→ Caveat: Widening also returns for non-monotonic f_i a solution. Narrowing is only applicable to monotonic f_i !!

→ In the example, accelerated narrowing already returns the optimal result :-)

→ If the operator $\sqcap$ only allows for finitely many improvements of values, we may execute narrowing until stabilization.

→ In case of interval analysis these are at most:

$$\#points \cdot (1 + 2 \cdot \#Vars)$$

Discussion:

→ Caveat: Widening also returns for non-monotonic f_i a solution. Narrowing is only applicable to monotonic f_i !!

→ In the example, accelerated narrowing already returns the optimal result :-)

→ If the operator $\sqcap$ only allows for finitely many improvements of values, we may execute narrowing until stabilization.

→ In case of interval analysis these are at most:

$$\#points \cdot (1 + 2 \cdot \#Vars)$$

1.6 Pointer Analysis

Questions:

→ Are two addresses possibly equal?

→ Are two addresses definitively equal?

1.6 Pointer Analysis

Questions:

→ Are two addresses possibly equal? — May Alias

→ Are two addresses definitively equal? — Must Alias

⟹ Alias Analysis

The analyses so far without alias information:

(1) Available Expressions:

- Extend the set *Expr* of expressions by occurring loads $M[e]$.
- Extend the Effects of Edges:

$$
\begin{array}{lll}
[\![x = e;]\!]^\sharp\, A & = & (A \cup \{e\}) \backslash \mathit{Expr}_x \\
[\![x = M[e];]\!]^\sharp\, A & = & (A \cup \{e, M[e]\}) \backslash \mathit{Expr}_x \\
[\![M[e_1] = e_2;]\!]^\sharp\, A & = & (A \cup \{e_1, e_2\}) \backslash \mathit{Loads}
\end{array}
$$

(2) Values of Variables:

- Extend the set *Expr* of expressions by occurring loads $M[e]$.
- Extend the Effects of Edges:

$$
[\![x = M[e];]\!]^\sharp\, V\, e' \;=\; \begin{cases} \{x\} & \text{if} \quad e' = M[e] \\ \emptyset & \text{if} \quad e' = e \\ V\, e' \backslash \{x\} & \text{otherwise} \end{cases}
$$

$$
[\![M[e_1] = e_2;]\!]^\sharp\, V\, e' \;=\; \begin{cases} \emptyset & \text{if} \quad e' \in \{e_1, e_2\} \\ V\, e' & \text{otherwise} \end{cases}
$$

(3) Constant Propagation:

- Extend the abstract state by an abstract store M
- Execute accesses to known memory locations!

$$
[\![x = M[e];]\!]^\sharp\, (D, M) \;=\; \begin{cases} (D \oplus \{x \mapsto M\, a\}, M) & \text{if} \\ & [\![e]\!]^\sharp\, D = a \sqsubset \top \\ (D \oplus \{x \mapsto \top\}, M) & \text{otherwise} \end{cases}
$$

$$
[\![M[e_1] = e_2;]\!]^\sharp\, (D, M) \;=\; \begin{cases} (D, M \oplus \{a \mapsto [\![e_2]\!]^\sharp D\}) & \text{if} \\ & [\![e_1]\!]^\sharp\, D = a \sqsubset \top \\ (D, \underline{\top}) & \text{otherwise} \end{cases} \quad \text{where}
$$

$$
\underline{\top}\, a \;=\; \top \qquad (a \in \mathbb{N})
$$

$\{a_1 \mapsto 5, a_2 \mapsto 10, a_3 \mapsto 30\}$

(3) Constant Propagation:

- Extend the abstract state by an abstract store M
- Execute accesses to known memory locations!

$$[\![x = M[e];]\!]^\sharp\,(D, M) = \begin{cases} (D \oplus \{x \mapsto M\,a\}, M) & \text{if } [\![e]\!]^\sharp\,D = a \sqsubset \top \\ (D \oplus \{x \mapsto \top\}, M) & \text{otherwise} \end{cases}$$

$$[\![M[e_1] = e_2;]\!]^\sharp\,(D, M) = \begin{cases} (D, M \oplus \{a \mapsto [\![e_2]\!]^\sharp D\}) & \text{if } [\![e_1]\!]^\sharp\,D = a \sqsubset \top \\ (D, \underline{\top}) & \text{otherwise} \quad \text{where} \end{cases}$$

$$\underline{\top}\,a = \top \qquad (a \in \mathbb{N})$$

Problems:

- Addresses are from $\mathbb{N}$:-(
  There are no infinite strictly ascending chains, but ...
- Exact addresses at compile-time are rarely known :-(
- At the same program point, typically different addresses are accessed ...
- Storing at an unknown address destroys all information M :-(

$\Longrightarrow$ constant propagation fails :-(

$\Longrightarrow$ memory accesses/pointers kill precision :-(

$M[5], M[6]$

Simplification: $R[\ldots], S[\ldots],$

- We consider pointers to the beginning of blocks A which allow indexed accesses $A[i]$:-)
- We ignore well-typedness of the blocks.
- New statements:

$x = \mathsf{new}();$ // allocation of a new block
$x = y[e];$ // indexed read access to a block
$y[e_1] = e_2;$ // indexed write access to a block

- Blocks are possibly infinite :-)
- For simplicity, all pointers point to the beginning of a block.