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 \( \mathbb{Z} \).
  a maximal chain:
  \[ [3, \mathit{17}] \subset [3, +\infty] \subset [-\infty, +\infty] \]

Formalization of the Approach:

Let \( x_i \equiv f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \) 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:

\[ x_i = x_i \cup f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]

We obviously have:

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

(b) The function \( G : \mathbb{D}^n \rightarrow \mathbb{D}^n \) with

\[ G(x_1, \ldots, x_n) = (y_1, \ldots, y_n), \quad y_i = x_i \cup f_i(x_1, \ldots, x_n) \]

is increasing, i.e., \( \bar{x} \sqsubseteq G\bar{x} \) for all \( \bar{x} \in \mathbb{D}^n \).
The sequence $G^k \perp$, $k \geq 0$, is an ascending chain:
\[ \perp \subseteq G \perp \subseteq \ldots \subseteq G^k \perp \subseteq \ldots \]

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

If $D$ has infinite strictly ascending chains, then (d) is not yet sufficient ...

but: we could consider the modified system of equations:
\[ x_i = x_i \cup f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]  

for a binary operation widening:
\[ \cup : D^2 \to D \quad \text{with} \quad v_1 \cup v_2 \subseteq v_1 \cup v_2 \]

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

---

**Formalization of the Approach:**

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

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

Nonetheless, an accumulating iteration can be defined. Consider the system of equations:
\[ x_i = x_i \cup f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \]  

We obviously have:

(a) $x$ is a solution of (1) iff $x$ is a solution of (2).

(b) The function $G : D^n \to D^n$ with $G(x_1, \ldots, x_n) = (y_1, \ldots, y_n), \quad y_i = x_i \cup f_i(x_1, \ldots, x_n)$
is increasing, i.e., $x \supseteq y \Rightarrow x \supseteq y$ for all $x, y \in D^n$.

---

**... for Interval Analysis:**

- The complete lattice is: $D_i = (\text{Vars} \to \mathbb{I})_{\perp}$
- the widening $\cup$ is defined by:
\[ \perp \cup D = D \quad \text{and for} \quad D_1 \neq \perp \neq D_2: \]
\[ (D_1 \cup D_2) x = (D_1 x) \cup (D_2 x) \]

where
\[ [l_1, u_1] \cup [l_2, u_2] = [l, u] \]

with
\[ l = \begin{cases} l_1 & \text{if } l_1 \leq l_2 \\ -\infty & \text{otherwise} \end{cases} \]

\[ u = \begin{cases} u_1 & \text{if } u_1 \geq u_2 \\ +\infty & \text{otherwise} \end{cases} \]

\[ \Rightarrow \quad \cup \quad \text{is not commutative} \Box \]

\[ \Rightarrow \quad \cup \quad \text{is not commutative} \Box \]
Example:

\[
[0, 2] \cup [1, 2] = [0, 2] \\
[1, 2] \cup [0, 2] = [-\infty, 2] \\
[1, 5] \cup [3, 7] = [1, +\infty]
\]

→ 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 \( \cup \) is chosen dynamically during iteration such that

→ the abstract values do not get too complicated;
→ the number of updates remains bounded …

Our Example:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( i = 0 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neg(( i &lt; 42 ))</td>
<td>Pos(( i &lt; 42 ))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neg(( 0 \leq i &lt; 42 ))</td>
<td>Pos(( 0 \leq i &lt; 42 ))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A_i = A + i )</td>
<td>( M[A_i] = i )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( i = i + 1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( l )</th>
<th>( u )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-( \infty )</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>( \bot )</td>
<td>( \bot )</td>
</tr>
</tbody>
</table>

Our Example:

\([1, \infty] \cup [1, \infty]\)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l )</td>
<td>( \infty )</td>
<td>+( \infty )</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( u )</td>
<td>0</td>
<td>0</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( l )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( u )</td>
<td>0</td>
<td>0</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( l )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( u )</td>
<td>0</td>
<td>0</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( l )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( u )</td>
<td>( \bot )</td>
<td>42</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( l )</td>
<td>( \bot )</td>
<td>42</td>
<td>+( \infty )</td>
</tr>
<tr>
<td>( u )</td>
<td>( \bot )</td>
<td>42</td>
<td>+( \infty )</td>
</tr>
</tbody>
</table>
... obviously, the result is disappointing  

Idea 2:
In fact, acceleration with $\square$ 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:

$I_1 = \{1\}$  or:

$I_2 = \{2\}$  or:

$I_3 = \{3\}$

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

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

\[ \{0, \bar{0}\} \subseteq [A, \bar{A}] \subseteq \{0, \bar{0}\} \]
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 $I$ ???

---

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>n</td>
<td>l</td>
<td>n</td>
<td>l</td>
</tr>
<tr>
<td>0</td>
<td>$-\infty$</td>
<td>$+\infty$</td>
<td>$-\infty$</td>
<td>$+\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$+\infty$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>$\perp$</td>
<td>42</td>
<td>$+\infty$</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td>42</td>
<td>42</td>
</tr>
</tbody>
</table>

---

Idea 3: Narrowing

Let $x$ denote any solution of (1), i.e.,

$$x_i \supseteq f_i x,$$

$i = 1, \ldots, n$

Then for monotonic $f_i$,

$$x \supseteq F x \supseteq F^2 x \supseteq \ldots \supseteq F^k x \supseteq \ldots$$

// Narrowing Iteration

---

Every tuple $F^k x$ is a solution of (1) :-)

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

// The same also holds for RR-Iteration.
Narrowing Iteration in the Example:

\[ \begin{array}{c|c|c}
0 & 1 & 2 \\
\hline
l & u & l & u & l & u \\
\hline
0 & -\infty & +\infty & -\infty & +\infty & -\infty & +\infty \\
1 & 0 & +\infty & 0 & +\infty & 0 & 42 \\
2 & 0 & +\infty & 0 & 41 & 0 & 41 \\
3 & 5 & +\infty & 0 & 41 & 0 & 41 \\
4 & 0 & +\infty & 0 & 41 & 0 & 41 \\
5 & 0 & +\infty & 0 & 41 & 0 & 41 \\
6 & 1 & +\infty & 1 & 42 & 1 & 42 \\
7 & 42 & +\infty & 1 & 42 & 1 & 42 \\
8 & 42 & +\infty & 42 & +\infty & 42 & 42 \\
\end{array} \]

Discussion:

\[ [0, \delta] \subseteq [1, \eta_2] = (0, \eta_2] \]

We start with a safe approximation.

We find that the inner check is redundant.

We find that at exit from the loop, always \( i = 42 \) :-)

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 \( \vec{x} = (x_1, \ldots, x_n) \) of the system of constraints:
\[
x_i \supseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n
\]
(1)

Then consider the system of equations:
\[
x_i = x_i \cap f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n
\]
(4)

Obviously, we have for monotonic \( f_i \): \( H^k \vec{x} = F^k \vec{x} \implies \)
where \( H(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \), \( y_i = x_i \cap f_i(x_1, \ldots, x_n) \).

In (4), we replace \( \cap \) by the novel operator \( \bowtie \) where:
\[
a_1 \bowtie a_2 \subseteq a_1 \cap a_2 \subseteq a_1
\]

... for Interval Analysis:

We preserve finite interval bounds \( \vdash \)

Therefore, \( \bot \bowtie D = D \bowtie \bot = \bot \) and for \( D_1 \neq \bot \neq D_2 \):
\[
(D_1 \bowtie D_2) \bowtie x = (D_1 \bowtie x) \cap (D_2 \bowtie x)
\]
where
\[
[l_1, u_1] \cap [l_2, u_2] = [l, u]
\]
\[
l = \begin{cases} 
  l_2 & \text{if } l_1 = -\infty \\
  l_1 & \text{otherwise}
\end{cases}
\]
\[
u = \begin{cases} 
  u_2 & \text{if } u_1 = \infty \\
  u_1 & \text{otherwise}
\end{cases}
\]

\( \Rightarrow \) \( \bowtie \) is not commutative \( \vdash \)
... for Interval Analysis:

We preserve finite interval bounds :-(

Therefore, \( \bot \sqcap D = D \sqcap \bot = \bot \) and for \( D_1 \neq \bot \neq D_2 \):

\[
(D_1 \sqcap D_2) x = (D_1 x) \sqcap (D_2 x)
\]

where

\[
[l_1, u_1] \cap [l_2, u_2] = [l, u] \quad \text{with}
\]

\[
\begin{align*}
    l &= \begin{cases} 
        l_2 & \text{if } l_1 = -\infty \\
        l_1 & \text{otherwise}
    \end{cases} \\
    u &= \begin{cases} 
        u_2 & \text{if } u_1 = \infty \\
        u_1 & \text{otherwise}
    \end{cases}
\end{align*}
\]

\[\implies \sqcap \text{ is not commutative !!!}\]

---

Discussion:

\[\rightarrow\] Caveat: Widening also returns for non-monotonic \( f_i \) a solution

\[\rightarrow\] Narrowing is only applicable to monotonic \( f_i \)

\[\rightarrow\] In the example, accelerated narrowing already returns the optimal result :-(

\[\rightarrow\] If the operator \( \sqcap \) only allows for finitely many improvements of values, we may execute narrowing until stabilization.

\[\rightarrow\] In case of interval analysis these are at most:

\[\#points \cdot (1 + 2 \cdot \#Vars)\]

---

1.6 Pointer Analysis

Questions:

\[\rightarrow\] Are two addresses possibly equal?

\[\rightarrow\] 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:
     \[
     [x = e]^E A = (A \cup \{ e \}) \setminus Expr
     \]
     \[
     [x = M[e];] A = (A \cup \{ e, M[e] \}) \setminus Expr
     \]
     \[
     [M[e_1] = e_2]^E A = (A \cup \{ e_1, e_2 \}) \setminus Loads
     \]

(2) Values of Variables:
   
   • Extend the set  Expr  of expressions by occurring loads  M[e].

   • Extend the Effects of Edges:
     \[
     [x = M[e];] V e' = \begin{cases} 
     \{ x \} & \text{if } e' = M[e] \\
     \emptyset & \text{if } e' = e \\
     V e' \setminus \{ x \} & \text{otherwise}
     \end{cases}
     \]
     \[
     [M[e_1] = e_2] V e' = \begin{cases} 
     \emptyset & \text{if } e' \notin \{ e_1, e_2 \} \\
     V e' & \text{otherwise}
     \end{cases}
     \]

(3) Constant Propagation:
   
   • Extend the abstract state by an abstract store  M

   • Execute accesses to known memory locations!
     \[
     [x = M[e];] (D, M) = \begin{cases} 
     (D \oplus \{ x \mapsto M a \}, M) & \text{if } [e]^E D = a \sqsubseteq \top \\
     (D \oplus \{ x \mapsto \top \}, M) & \text{otherwise}
     \end{cases}
     \]
     \[
     (D, M \oplus \{ a \mapsto [e_2]^E D \}) & \text{if } [e_1]^E D = a \sqsubseteq \top
     \]

\[\top a = \top \quad (a \in \mathbb{N})\]
(3) Constant Propagation:

- Extend the abstract state by an abstract store $M$

- Execute accesses to known memory locations!

\[
[x = M[e_1]]^*(D, M) = \begin{cases} 
(D \oplus \{x \mapsto M\ a\}, M) & \text{if} \ [e_1]^* D = a \sqsubseteq \top \\
(D \oplus \{x \mapsto \top\}, M) & \text{otherwise}
\end{cases}
\]

\[
[M[e_1] = e_2]^*(D, M) = \begin{cases} 
(D, M \oplus \{a \mapsto [e_2]^* D\}) & \text{if} \ [e_2]^* D = a \sqsubseteq \top \\
(D, \top) & \text{otherwise}
\end{cases}
\]

\[
\bot a = \top \quad (a \in \mathbb{N})
\]