We conclude: The assertion \((*)\) is true \(\therefore\)

The MOP-Solution

\[ D^*[v] = \bigcup \{ [[\pi]]^2 D_\pi \mid \pi : \text{start} \rightarrow^* v \} \]

where \( D_\pi x = \top \quad (x \in \text{Vars}) \).

By \((*)\), we have for all initial states \(s\) and all program executions \(\pi\) which reach \(v:\)

\(\quad ([\pi]\ s) \Delta (D^*[v])\)

In order to approximate the MOP, we use our constraint system \(\therefore\)

---

Example:

```
0  x = 10;
1  y = 1;

2  Neg(x > 1)
    +------------------+
    |                   |
    | 6  M[R] = y      |
    |                   |
    +------------------+

3  Pos(x > 1)
    +------------------+
    |                   |
    | 3  y = x * y;    |
    |                   |
    +------------------+

4  x = x - 1;
    +------------------+
    |                   |
    | 4  y = x * y;    |
    |                   |
    +------------------+

5  Neg(x > 1)
    +------------------+
    |                   |
    | 7  y = x * y;    |
    |                   |
    +------------------+
```

300
Example:

\[(\text{Neg}(x > 1) \land \text{Pos}(x > 1))\]

\[M[R] = y;\]

\[y = x \cdot y;\]

\[x = x - 1;\]

Example:

\[(\text{Neg}(x > 1) \lor (x = 10)];\]

\[M[R] = y;\]

\[y = x \cdot y;\]

\[x = x - 1;\]

Conclusion:

Although we compute with concrete values, we fail to compute everything 😞

The fixpoint iteration, at least, is guaranteed to terminate:

For \(n\) program points and \(m\) variables, we maximally need:

\[n \cdot (m + 1)\] rounds 😞

Caveat:

The effects of edge are not distributive 😞
Example:

\[
\begin{array}{c}
\text{0} & x = 10; \\
\text{1} & y = 1; \\
\text{2} & \neg(x > 1) \Rightarrow x = y; \\
\text{6} & M[R] = y; \\
\text{7} & x = x - 1;
\end{array}
\]

\[
\begin{array}{c|ccc}
& 1 & 2 & 3 \\
\hline
x & T & T & T \\
y & T & T & T \\
\text{ditto}
\end{array}
\]

Conclusion:

Although we compute with concrete values, we fail to compute everything :-(

The fixpoint iteration, at least, is guaranteed to terminate:

For \( n \) program points and \( m \) variables, we maximally need:

\( n \cdot (m + 1) \) rounds :-)

Caveat:

The effects of edge are not distributive !!!

Counter Example: \( f = [x = x + y]^2 \)

Let \( D_1 = \{ x \mapsto 2, y \mapsto 3 \} \)

\( D_2 = \{ x \mapsto 3, y \mapsto 2 \} \)

\[
\begin{align*}
\text{Dann} \quad f D_1 \cup f D_2 &= \{ x \mapsto 5, y \mapsto 3 \} \cup \{ x \mapsto 5, y \mapsto 2 \} \\
&= \{ x \mapsto 5, y \mapsto T \} \\
&\neq \{ x \mapsto T, y \mapsto T \} \\
&= f \{ x \mapsto T, y \mapsto T \} \\
&= f (D_1 \cup D_2) \\
\end{align*}
\]

:::-)}

We conclude:

The least solution \( D \) of the constraint system in general yields only an upper approximation of the MOP, i.e.,

\( D^*[v] \subseteq D[v] \)
We conclude:

The least solution $\mathcal{D}$ of the constraint system in general yields only an upper approximation of the MOP, i.e.,

$$\mathcal{D}^*[v] \subseteq \mathcal{D}[v]$$

As an upper approximation, $\mathcal{D}[v]$ nonetheless describes the result of every program execution $\pi$ which reaches $v$:

$$([\pi] (\rho, \mu)) \Delta (\mathcal{D}[v])$$

whenever $[\pi] (\rho, \mu)$ is defined.

---

Transformation 4: Removal of Dead Code

$$\mathcal{D}[u] = \bot$$

Transformation 4 (cont.): Removal of Dead Code

$$\bot \neq \mathcal{D}[u] = D$$

Neg ($e$)

$$[e] D = 0$$

Pos ($e$)

$$[e] D \notin \{0, \top\}$$

Transformation 4 (cont.): Simplified Expressions

$$x = c;$$

$$\bot \neq \mathcal{D}[u] = D$$

$$[e] D = c$$

$$x = c;$$
Transformation 4 (cont.): Simplified Expressions

\[ x = c \]

\[ \downarrow \neq D[u] = D \]

\[ [\text{if}] D = c \]

\[ x = k \]

Extensions:

- Instead of complete right-hand sides, also subexpressions could be simplified:

\[ x + (3 \ast y) \quad \overset{\{x \Rightarrow 3, y \Rightarrow 5\}}{\longrightarrow} \quad x + 15 \]

... and further simplifications be applied, e.g.:

\[ x \ast 0 \quad \Rightarrow \quad 0 \]

\[ x \ast 1 \quad \Rightarrow \quad x \]

\[ x + 0 \quad \Rightarrow \quad x \]

\[ x - 0 \quad \Rightarrow \quad x \]

\[ \ldots \]

So far, the information of conditions has not yet be optimally exploited:

\[ \exists \neg \exists \Downarrow \]

if \( (x = 7) \)

\[ y 
= \begin{cases} 
3; & \text{if } [x = 7] \text{ holds} \\
\end{cases} \]

Even if the value of \( x \) before the if statement is unknown, we at least know that \( x \) definitely has the value 7 — whenever the then-part is entered :-)

Therefore, we can define:

\[ [\text{Pos} (x = e)]^f D = \begin{cases} 
D & \text{if } [x = e]^f D = 1 \\
\bot & \text{if } [x = e]^f D = 0 \\
D_1 & \text{otherwise} 
\end{cases} \]

where

\[ D_1 = D \oplus \{ x \Rightarrow (D \sqcap [e]^f D) \} \]

The effect of an edge labeled \( \text{Neg} (x \neq e) \) is analogous :-)

Our Example:

\[ \begin{array}{c}
\text{Neg} (x = 7) \\
\text{Pos} (x = 7)
\end{array} \]

\[ \begin{array}{c}
0 \\
1 \\
2 \\
3 \\
y = x + 3;
\end{array} \]
1.5 Interval Analysis

Observation:

- Programmers often use global constants for switching debugging code on/off.
  
  Constant propagation is useful :-)

- In general, precise values of variables will be unknown — perhaps, however, a tight interval !!!
Idea 1:

Determine for every variable $x$ an (as tight as possible :-)) interval of possible values:

$$I = \{[l, u] : l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{+\infty\} : l \leq u\}$$

Partial Ordering:

$$[l_1, u_1] \subseteq [l_2, u_2] \iff l_2 \leq l_1 \land u_1 \leq u_2$$

Thus:

$$[l_1, u_1] \cup [l_2, u_2] = [l_1 \cap l_2, u_1 \cup u_2]$$

Caveat:

- $I$ is not a complete lattice :-)
- $I$ has infinite ascending chains, e.g.,

$$[0, 0] \subset [0, 1] \subset [-1, 1] \subset [-1, 2] \subset \ldots$$
Caveat:
→ \( \mathbb{I} \) is not a complete lattice ☺)
→ \( \mathbb{I} \) has infinite ascending chains, e.g.,
\[
[0, 0] \subset [0, 1] \subset [-1, 1] \subset [-1, 2] \subset \ldots
\]

Description Relation:
\[
\Sigma \subseteq [l, u] \quad \text{iff} \quad l \leq z \leq u
\]

Concretization:
\[
\gamma [l, u] = \{ z \in \mathbb{Z} \mid l \leq z \leq u \}
\]

Example:
\[
\gamma [0, 7] = \{ 0, \ldots, 7 \}
\]
\[
\gamma [0, \infty) = \{ 0, 1, 2, \ldots, \}
\]

Computing with intervals:
Interval Arithmetic ☺)

Addition:
\[
[l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]
\]
where
\[
-\infty + -\infty = -\infty
\]
\[
+\infty + -\infty = +\infty
\]
\[
// -\infty + +\infty \text{ cannot occur ☺)}
\]

Negation:
\[
-\lbrack l, u \rbrack = [-u, -l]
\]

Multiplication:
\[
[l_1, u_1] \times [l_2, u_2] = [a, b]
\]
where
\[
a = l_1 l_2 \cap l_1 u_2 \cap u_1 l_2 \cap u_1 u_2
\]
\[
b = l_1 l_2 \cup l_1 u_2 \cup u_1 l_2 \cup u_1 u_2
\]

Example:
\[
[0, 2] \times [3, 4] = [0, 8]
\]
\[
[-1, 2] \times [-2, 4] = [-4, 8]
\]
\[
[-1, 2] \times [3, 4] = [-6, 8]
\]
\[
[-1, 2] \times [-4, -3] = [-8, 4]
\]

Division:
\[
[l_1, u_1] / [l_2, u_2] = [a, b]
\]

- **If** \( 0 \) is **not** contained in the interval of the denominator, then:
\[
a = l_1 / l_2 \cap l_1 / u_2 \cap u_1 / l_2 \cap u_1 / u_2
\]
\[
b = l_1 / l_2 \cup l_1 / u_2 \cup u_1 / l_2 \cup u_1 / u_2
\]

- **If** \( l_2 \leq 0 \leq u_2 \), we define:
\[
[a, b] = [-\infty, +\infty]
\]
Equality:

\[
[l_1, u_1] = \equiv^= [l_2, u_2] = \begin{cases} 
[1, 1] & \text{if } l_1 = u_1 = l_2 = u_2 \\
[0, 0] & \text{if } u_1 < l_2 \lor u_2 < l_1 \\
[0, 1] & \text{otherwise}
\end{cases}
\]

Example:

\[
[42, 42] = \equiv^= [42, 42] = [1, 1] \\
[0, 7] = \equiv^= [0, 7] = [0, 1] \\
[1, 2] = \equiv^= [3, 4] = [0, 0]
\]

Less:

\[
[l_1, u_1] <^< [l_2, u_2] = \begin{cases} 
[1, 1] & \text{if } u_1 < l_2 \\
[0, 0] & \text{if } u_2 \leq l_1 \\
[0, 1] & \text{otherwise}
\end{cases}
\]

Example:

\[
[1, 2] <^< [9, 42] = [1, 1] \\
[0, 7] <^< [0, 7] = [0, 1] \\
[3, 4] <^< [1, 2] = [0, 0]
\]

327

328

329

330
By means of \( \mathbb{I} \) we construct the complete lattice:

\[
\mathbb{D}_1 = (\text{Vars} \mapsto \mathbb{I})_ot
\]

**Description Relation:**

\[ \rho \not\Delta D \iff D \neq \bot \land \forall x \in \text{Vars} : (\rho x) \not\Delta (D x) \]

The abstract evaluation of expressions is defined analogously to constant propagation. We have:

\[ ([e]\rho) \Delta ([e]^2 D) \quad \text{whenever} \quad \rho \not\Delta D \]

**The Effects of Edges:**

\[
\begin{align*}
[x = e] & D \\
D + \{x \mapsto [e]D\} & = D \\
[x = M[e]] & D \\
D + \{x \mapsto \top\} & = D \\
[M[e]] & D \\
D & = D \\
[\text{Pos}(e)] & D \\
\left\{ \begin{array}{ll}
\bot & \text{if } [0, 0] \subseteq [e]D \\
D & \text{otherwise}
\end{array} \right. \\
[\text{Neg}(e)] & D \\
\left\{ \begin{array}{ll}
\bot & \text{if } [0, 0] \subseteq [e]D \\
D & \text{otherwise}
\end{array} \right.
\]

... given that \( D \neq \bot \):

**Better Exploitation of Conditions:**

\[
[\text{Pos}(e)] D = \left\{ \begin{array}{ll}
\bot & \text{if } [0, 0] \subseteq [e]D \\
D_1 & \text{otherwise}
\end{array} \right.
\]

where:

\[
D_1 = \left\{ \begin{array}{ll}
D + \{x \mapsto (D x) \cap ([e] D)\} & \text{if } e \equiv x = e_1 \\
D + \{x \mapsto (D x) \cap [-\infty, u]\} & \text{if } e \equiv x \leq e_1, [e] D = [-, u] \\
D + \{x \mapsto (D x) \cap [l, \infty]\} & \text{if } e \equiv x \geq e_1, [e] D = [l, \infty]
\end{array} \right.
\]

**Better Exploitation of Conditions (cont.):**

\[
[\text{Neg}(e)] D = \left\{ \begin{array}{ll}
\bot & \text{if } [0, 0] \subseteq [e]D \\
D_1 & \text{otherwise}
\end{array} \right.
\]

where:

\[
D_1 = \left\{ \begin{array}{ll}
D + \{x \mapsto (D x) \cap ([e] D)\} & \text{if } e \equiv x \neq e_1 \\
D + \{x \mapsto (D x) \cap [-\infty, u]\} & \text{if } e \equiv x > e_1, [e] D = [-, u] \\
D + \{x \mapsto (D x) \cap [l, \infty]\} & \text{if } e \equiv x < e_1, [e] D = [l, \infty]
\end{array} \right.
\]
Example:

```
0                      0
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
1  0  42
2  0  41
3  0  41
4  0  41
5  0  41
6  1  42
7  \bot
8  42  42
```

Example:

```
0                      0
<table>
<thead>
<tr>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
1  0  42
2  0  41
3  0  41
4  0  41
5  0  41
6  1  42
7  \bot
8  42  42
```

Problem:

→ The solution can be computed with RR-iteration — after about 42 rounds :-O
→ 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 \( \mathbb{Z} \) ...
  \[ [3, 17] \sqsubseteq [3, +\infty] \sqsubseteq [-\infty, +\infty] \]