Kleiner:

$$[l_1, u_1] <^\sharp [l_2, u_2] = \begin{cases} [1,1] & \text{falls} \quad u_1 < l_2 \\ [0,0] & \text{falls} \quad u_2 \leq l_1 \\ [0,1] & \text{sonst} \end{cases}$$

Kleiner:

$$[l_1, u_1] <^\sharp [l_2, u_2] = \begin{cases} [1,1] & \text{falls} \quad u_1 < l_2 \\ [0,0] & \text{falls} \quad u_2 \leq l_1 \\ [0,1] & \text{sonst} \end{cases}$$

Beispiel:

$$\begin{aligned} [1,2] <^\sharp [9,42] &= [1,1] \\ [0,7] <^\sharp [0,7] &= [0,1] \\ [3,4] <^\sharp [1,2] &= [0,0] \end{aligned}$$

Mithilfe von $\mathbb{I}$ konstruieren wir den vollständigen Verband:

$$\mathbb{D}_{\mathbb{I}} = (\mathit{Vars} \to \mathbb{I})_{\perp}$$

Beschreibungsrelation:

$$\rho \ \Delta \ D \qquad \text{gdw.} \qquad D \neq \perp \quad \wedge \quad \forall\, x \in \mathit{Vars} : \ (\rho\, x) \ \Delta \ (D \ x)$$

Die abstrakte Ausdrucksauswertung definieren wir analog Konstantenpropagation. Wir finden:

$$([\![e]\!]\ \rho) \ \Delta \ ([\![e]\!]^{\sharp}\ D) \qquad \text{sofern} \qquad \rho \ \Delta \ D$$

Die Kanteneffekte:

$$
\begin{array}{lcl}
[\![;]\!]^\sharp\, D & = & D \\
[\![x = e;]\!]^\sharp\, D & = & D \oplus \{x \mapsto [\![e]\!]^\sharp\, D\} \\
[\![x = M[R];]\!]^\sharp\, D & = & D \oplus \{x \mapsto \top\} \\
[\![M[R_1] = R_2;]\!]^\sharp\, D & = & D \\
[\![\mathrm{Pos}\,(e)]\!]^\sharp\, D & = & \begin{cases} \bot & \text{falls} \quad [0,0] = [\![e]\!]^\sharp\, D \\ D & \text{sonst} \end{cases} \\
[\![\mathrm{Neg}\,(e)]\!]^\sharp\, D & = & \begin{cases} D & \text{falls} \quad [0,0] \sqsubseteq [\![e]\!]^\sharp\, D \\ \bot & \text{sonst} \end{cases}
\end{array}
$$

... sofern $D \neq \bot$:-)

Bessere Ausnutzung von Bedingungen:

$$[\![\text{Pos}\,(e)]\!]^\sharp\, D \;=\; \begin{cases} \bot & \text{falls} \quad [0,0] = [\![e]\!]^\sharp\, D \\ D_1 & \text{sonst} \end{cases}$$

wobei :

$$D_1 \;=\; \begin{cases} D \oplus \{x \mapsto (D\,x) \sqcap ([\![e_1]\!]^\sharp\, D)\} & \text{falls } e \equiv x == e_1 \\ D \oplus \{x \mapsto (D\,x) \sqcap [-\infty, u]\} & \text{falls } e \equiv x \leq e_1, [\![e_1]\!]^\sharp\, D = [\_, u] \\ D \oplus \{x \mapsto (D\,x) \sqcap [l, \infty]\} & \text{falls } e \equiv x \geq e_1, [\![e_1]\!]^\sharp\, D = [l, \_] \end{cases}$$

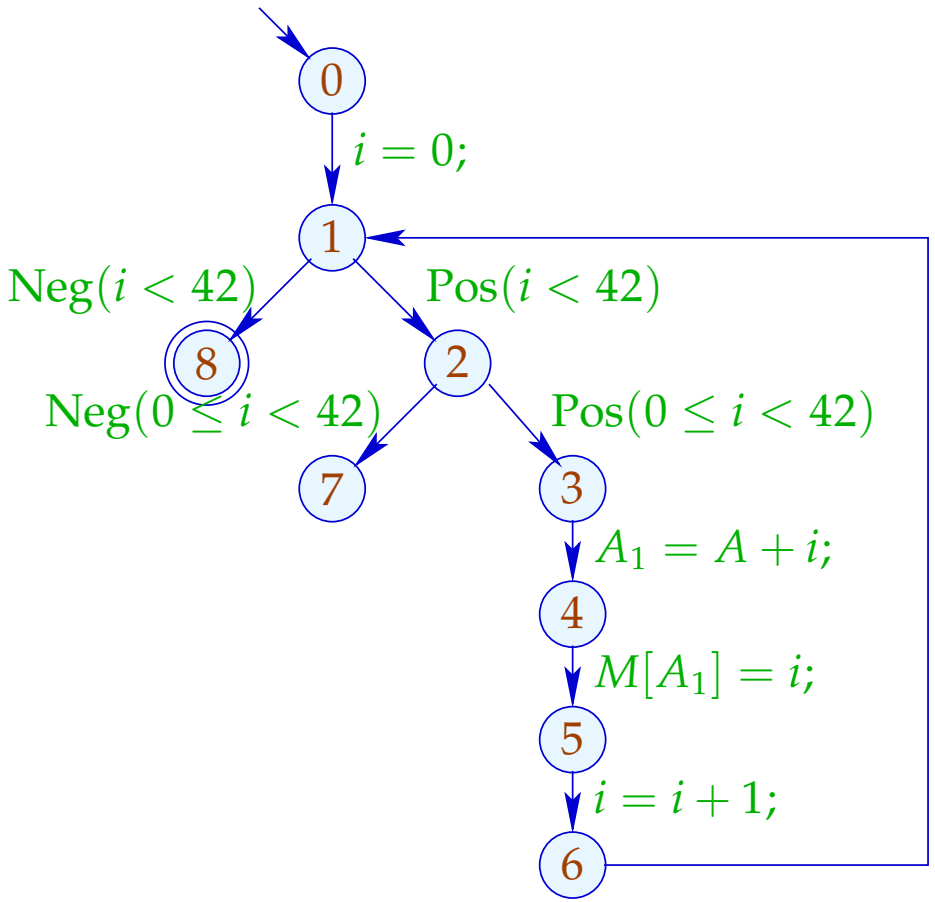
Bessere Ausnutzung von Bedingungen (Forts.):

$$[\![\mathrm{Neg}\,(e)]\!]^\sharp\, D \;=\; \begin{cases} \bot & \text{falls} \quad [0,0] \not\sqsubseteq [\![e]\!]^\sharp\, D \\ D_1 & \text{sonst} \end{cases}$$

wobei :

$$D_1 \;=\; \begin{cases} D \oplus \{x \mapsto (D\,x) \sqcap ([\![e_1]\!]^\sharp\, D)\} & \text{falls } e \equiv x \neq e_1 \\ D \oplus \{x \mapsto (D\,x) \sqcap [-\infty, u]\} & \text{falls } e \equiv x > e_1, [\![e_1]\!]^\sharp\, D = [\_, u] \\ D \oplus \{x \mapsto (D\,x) \sqcap [l, \infty]\} & \text{falls } e \equiv x < e_1, [\![e_1]\!]^\sharp\, D = [l, \_] \end{cases}$$

Beispiel:



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

Problem:

$\rightarrow$ Die Lösung lässt sich mit RR-Iteration berechnen —
nach ca. 42 Runden :-(

$\rightarrow$ Auf manchen Programmen terminiert die Iteration nie :-((

Idee 1: Widening

- Iteriere beschleunigt — unter Preisgabe von Präzision :-)
- Erlaube nur beschränkt oft die Modifikation eines Werts !!!

  ... im Beispiel:
- verbiete Updates von Intervall-Grenzen in $\mathbb{Z}$ ...

  $\Longrightarrow$ eine maximale Kette:

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

Formalisierung dieses Vorgehens:

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

ein Ungleichungsssystem über $\mathbb{D}$, wobei die f_i nicht notwendigerweise monoton sind.

Trotzdem können wir eine akkumulierende Iteration definieren.
Betrachte das Gleichungssystem:

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

Offenbar gilt:

(a) $\underline{x}$ ist Lösung von (1) gdw. $\underline{x}$ Lösung von (2) ist.

(b) Die Funktion $G : \mathbb{D}^n \to \mathbb{D}^n$ mit
$G(x_1, \ldots, x_n) = (y_1, \ldots, y_n), \quad y_i = x_i \sqcup f_i(x_1, \ldots, x_n)$
ist vergrößernd, d.h. $\underline{x} \sqsubseteq G\,\underline{x}$ für alle $\underline{x} \in \mathbb{D}^n$.

(c) Die Folge $G^k \underline{\bot}\,, \quad k \geq 0,$ ist eine aufsteigende Kette:

$$\underline{\bot} \sqsubseteq G\underline{\bot} \sqsubseteq \ldots \sqsubseteq G^k\underline{\bot} \sqsubseteq \ldots$$

(d) Gilt $G^k \underline{\bot} = G^{k+1}\underline{\bot} = \underline{y}$ ist $\underline{y}$ eine Lösung von (1).

(e) Hat $\mathbb{D}$ unendliche aufsteigende Ketten, ist uns mit (d) noch nicht viel gedient ...

aber: wir könnten statt Gleichungssystem (2) ein Gleichungssystem:

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

betrachten für eine binäre Operation Widening:

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

Dann berechnet (RR)-Iteration für (3) immer noch eine Lösung von (1) :-)

... für die Intervall-Analyse:

- Der vollständige Verband ist: $\mathbb{D}_{\mathbb{I}} = (\mathit{Vars} \to \mathbb{I})_\perp$
- Das Widening $\sqcup$ definieren wir als:

$$\perp \sqcup D = D \sqcup \perp = D \qquad \text{und für} \quad D_1 \neq \perp \neq D_2\text{:}$$

$$\begin{aligned}
(D_1 \sqcup D_2)\, x &= (D_1\, x) \sqcup (D_2\, x) \qquad \text{wobei} \\
[l_1, u_1] \sqcup [l_2, u_2] &= [l, u] \qquad \text{mit} \\
l &= \begin{cases} l_1 & \text{falls} \quad l_1 \leq l_2 \\ -\infty & \text{sonst} \end{cases} \\
u &= \begin{cases} u_1 & \text{falls} \quad u_1 \geq u_2 \\ +\infty & \text{sonst} \end{cases}
\end{aligned}$$

$\Longrightarrow$ $\sqcup$ ist nicht kommutativ !!!

Beispiel:

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

$\rightarrow$ Widening liefert schneller größere Werte.

$\rightarrow$ Es sollte so gewählt werden, dass es die Terminierung der Iteration garantiert :-)

$\rightarrow$ Bei Intervall-Analyse begrenzt es die Anzahl der Iterationen auf:

$$\#\mathit{Punkte} \cdot (1 + 2 \cdot \#\mathit{Vars})$$

Fazit:

- Um eine Lösung von (1) über einem vollständigen Verband mit unendlichen aufsteigenden Ketten zu bestimmen, definieren wir ein geeignetes Widening und lösen dann (3) :-)

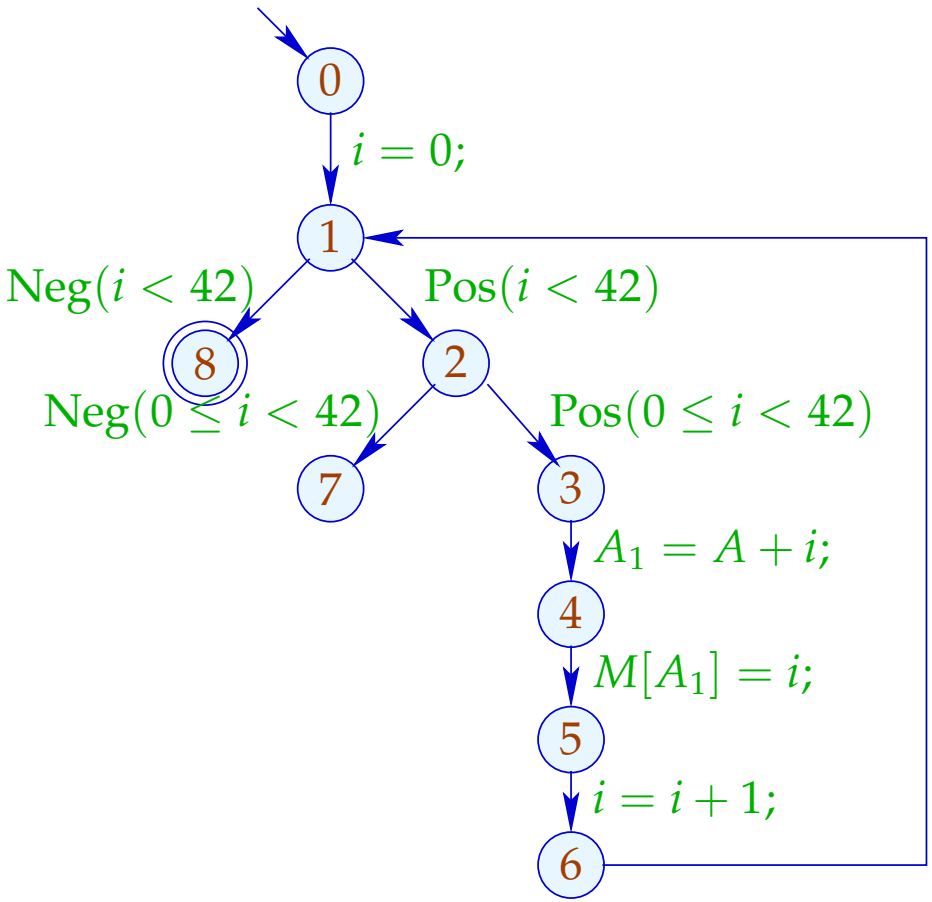
- Achtung: Die Konstruktion geeigneter Widenings ist eine schwarze Kunst !!!

  Oft wählt man $\sqcup$ ganz pragmatisch dynamisch während der Iteration, so dass

  $\rightarrow$ die abstrakten Werte nicht zu kompliziert werden;

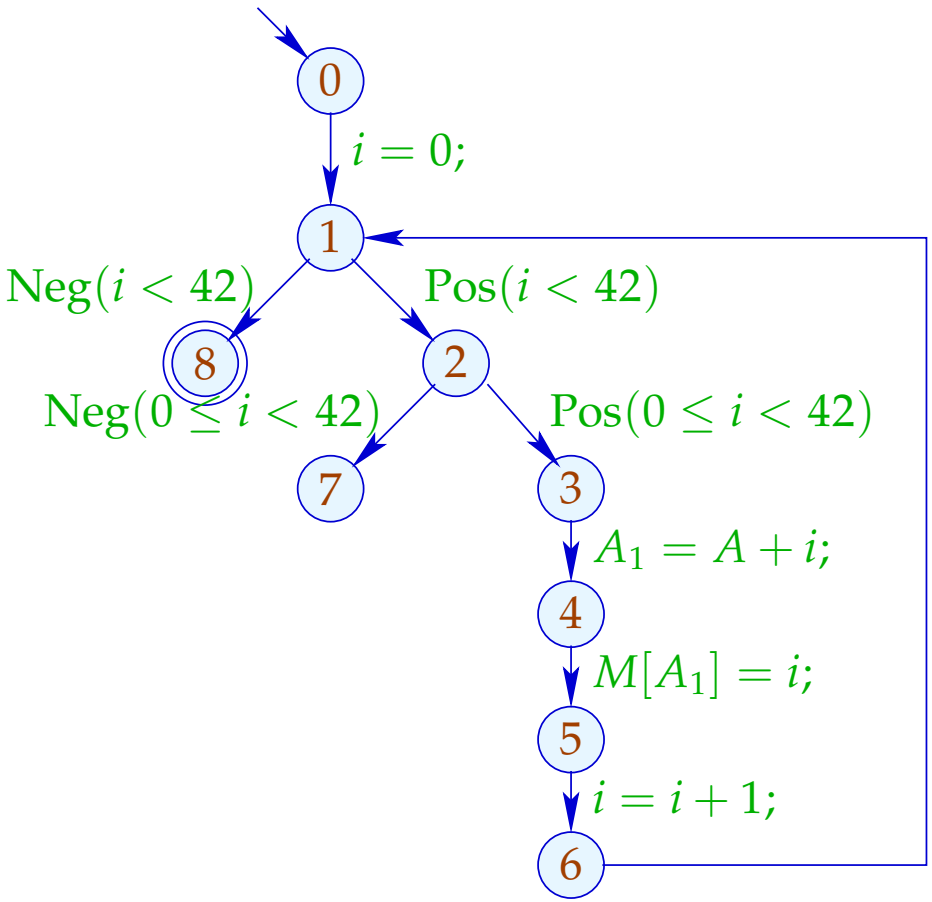
  $\rightarrow$ die Anzahl der Updates fest beschränkt bleibt ...

Unser Beispiel:



	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	$\bot$	
8	$\bot$	

Unser Beispiel:



	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	$+\infty$		

... offenbar ist das Ergebnis enttäuschend :-(

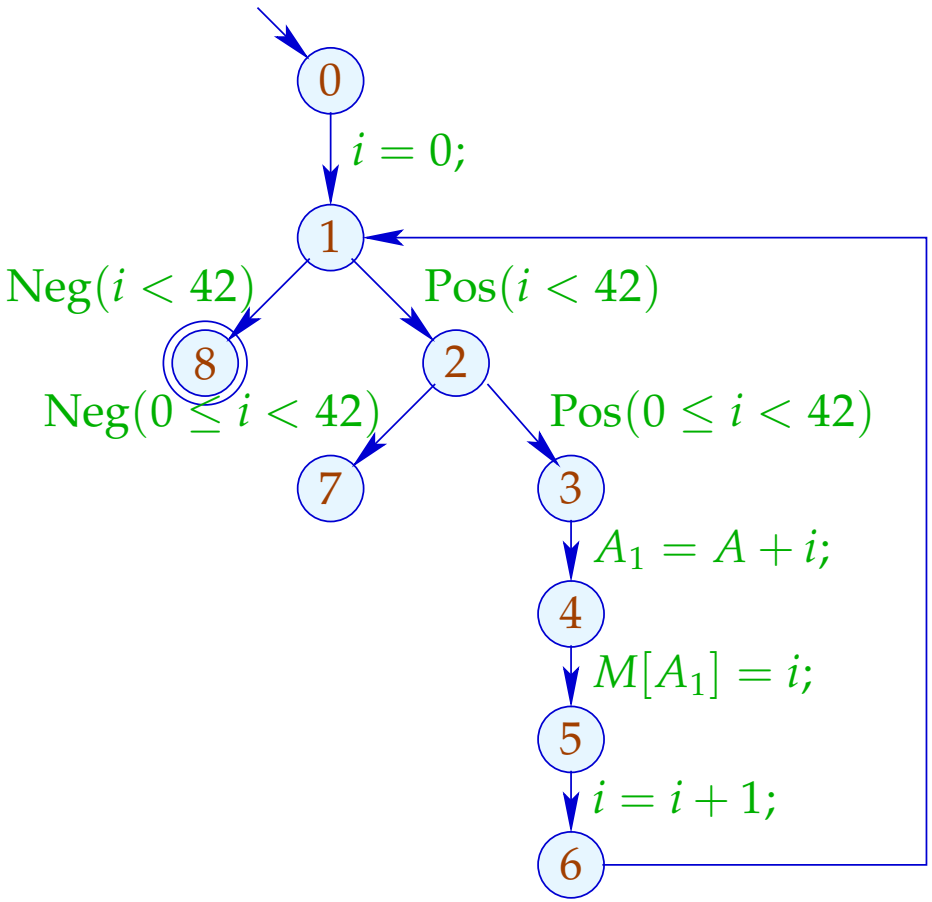
Idee 2:

Eigentlich reicht es, die Beschleunigung mittels $\sqcup$ nur an genügend vielen Stellen anzuwenden!

Eine Menge I heißt Loop Separator (Kreis-Trenner), falls jeder Kreis mindestens einen Punkt aus I enthält :-)

Wenden wir Widening nur an den Punkten aus einer solchen Menge I, terminiert RR-Iteration immer noch !!!

In unserem Beispiel:

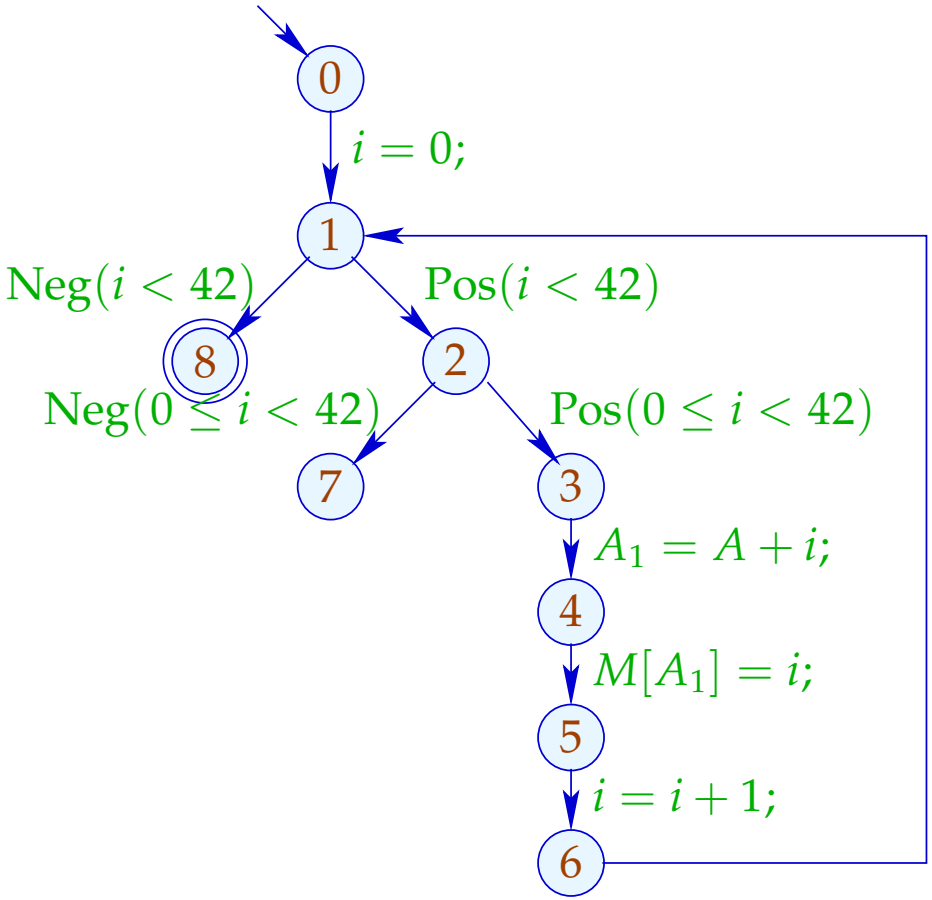


$I_1 = \{1\}$ oder auch:

$I_2 = \{2\}$ oder auch:

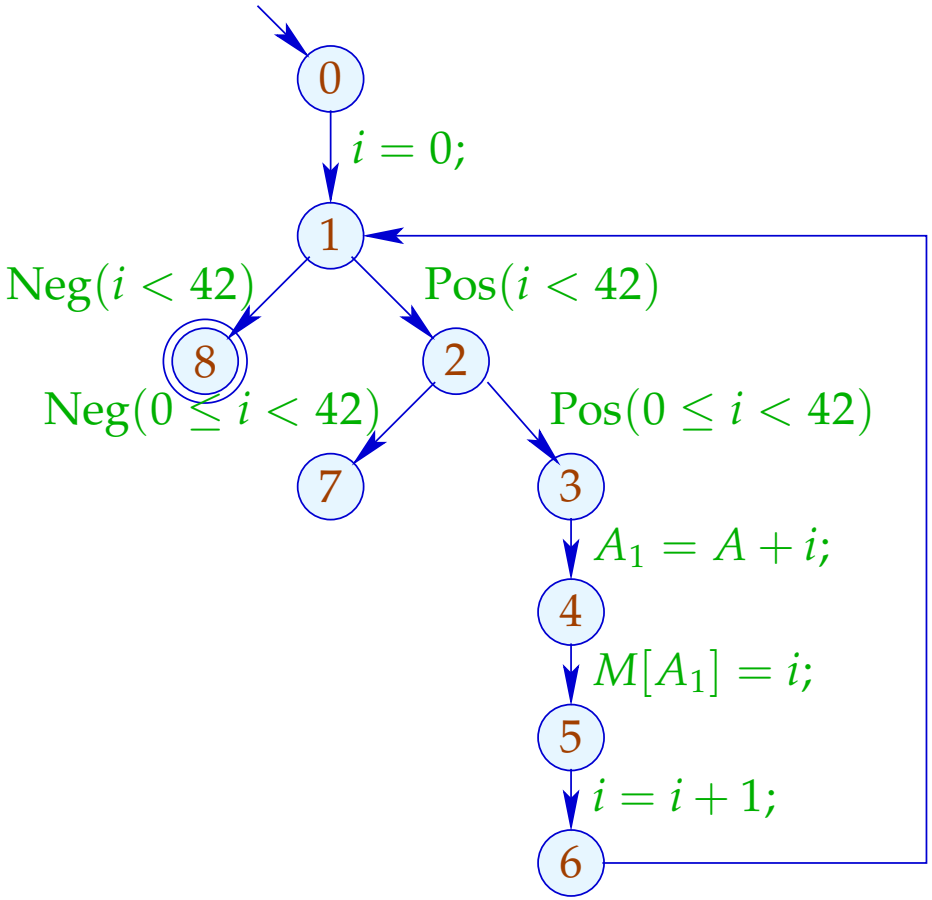
$I_3 = \{3\}$

Die Analyse mit $I = \{1\}$:



	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	$\bot$		$\bot$			
8	$\bot$		42	$+\infty$		

Die Analyse mit $I = \{2\}$:



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

Diskussion:

- Beide Analysen-Läufe berechnen interessante Informationen :-)
- Der Lauf mit $I = \{2\}$ belegt, dass nach Verlassen der Schleife stets $i = 42$ gilt.
- Nur der Lauf mit $I = \{1\}$ belegt aber, dass der äußere Test den inneren überflüssig macht :-(

Wie findet man einen geeigneten Loop Separator I ???

Idee 3: Narrowing

Sei $\underline{x}$ irgend eine Lösung von (1), d.h.

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

Dann gilt für monotone f_i,

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

// Narrowing Iteration

Idee 3: Narrowing

Sei $\underline{x}$ irgend eine Lösung von (1), d.h.

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

Dann gilt für monotone f_i,

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

// Narrowing Iteration

Jeder der Tupel $F^k\,\underline{x}$ ist eine Lösung von (1) :-)

$\Longrightarrow$

Terminierung ist kein Problem mehr:
wir stoppen, wenn wir keine Lust mehr haben :-))

// Analoges gilt für RR-Iteration.