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) :-)

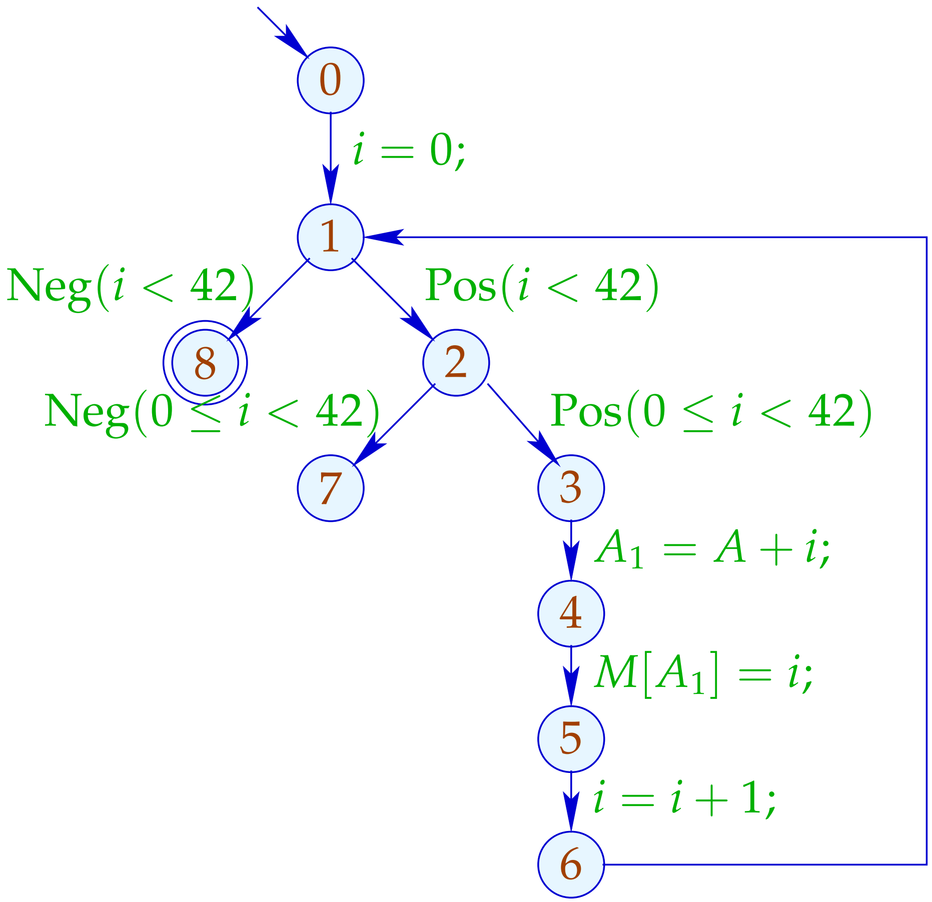
- Warning: The construction of suitable widenings is a dark art !!!

  Often $\sqcup$ is chosen dynamically during iteration such that

  $\longrightarrow$ the abstract values do not get too complicated;

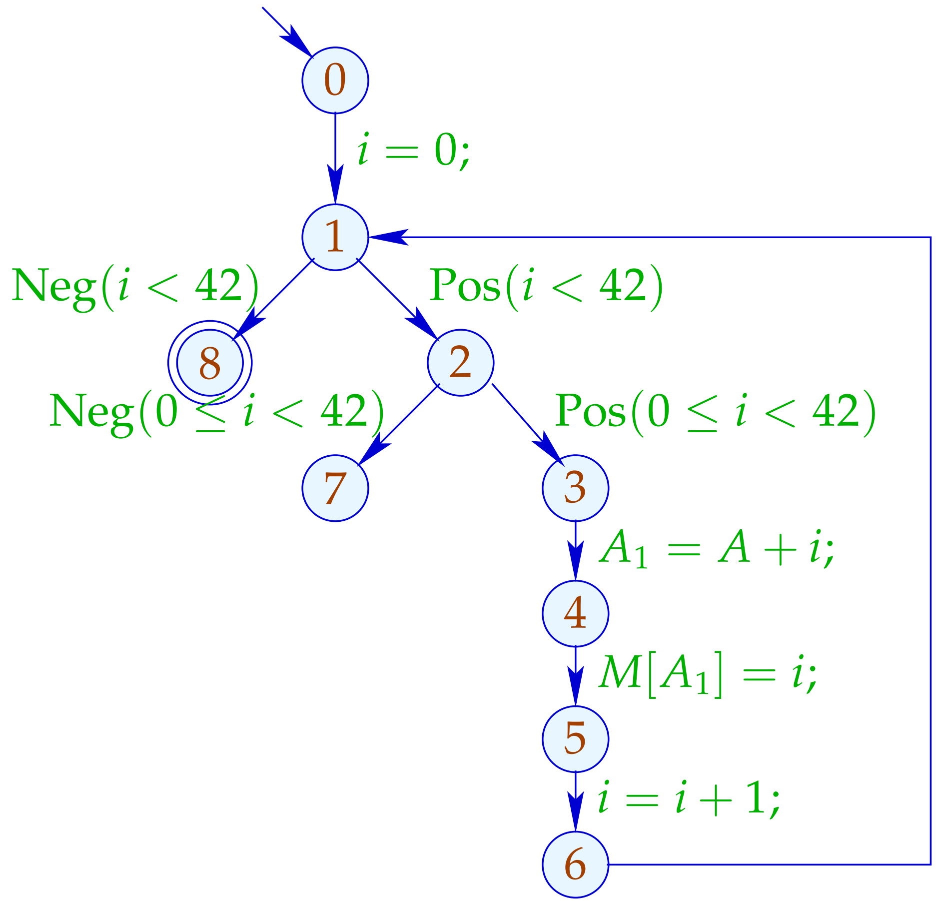
  $\longrightarrow$ the number of updates remains bounded ...

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:



	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	$\perp$		42	$+\infty$		
8	$\perp$		42	$+\infty$		

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

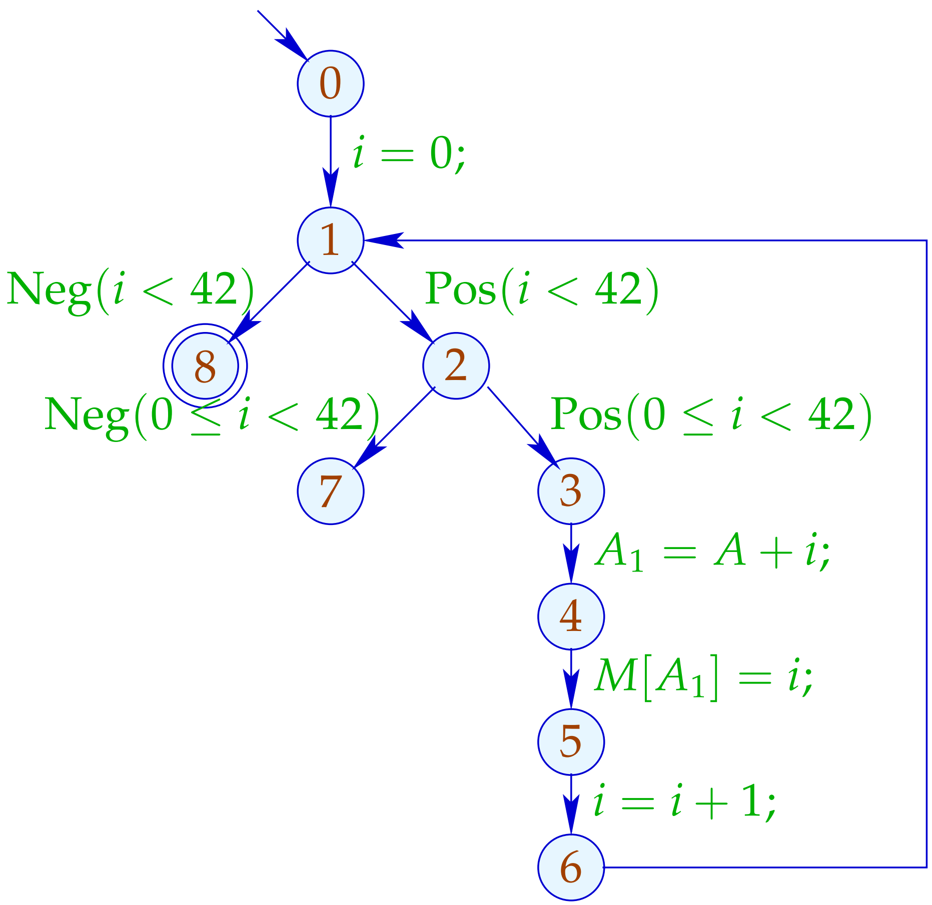
Idea 2:

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 :-)

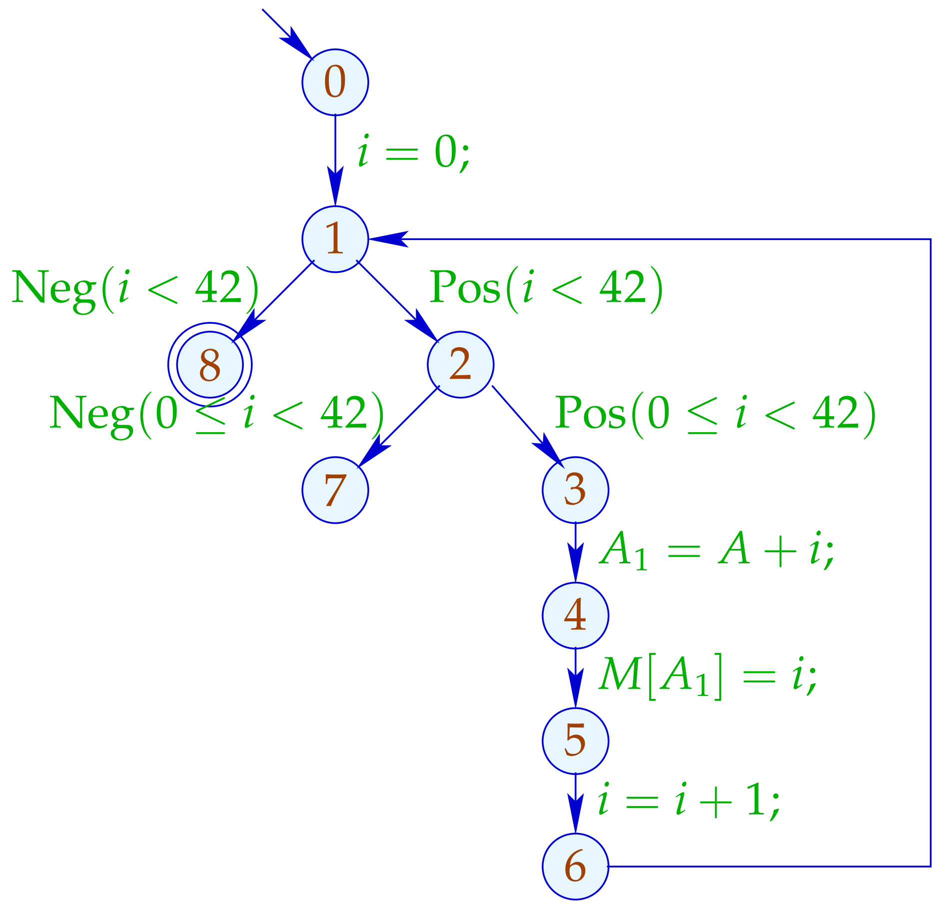
If we apply widening only at program points from such a set I, then RR-iteration still terminates !!!

In our Example:



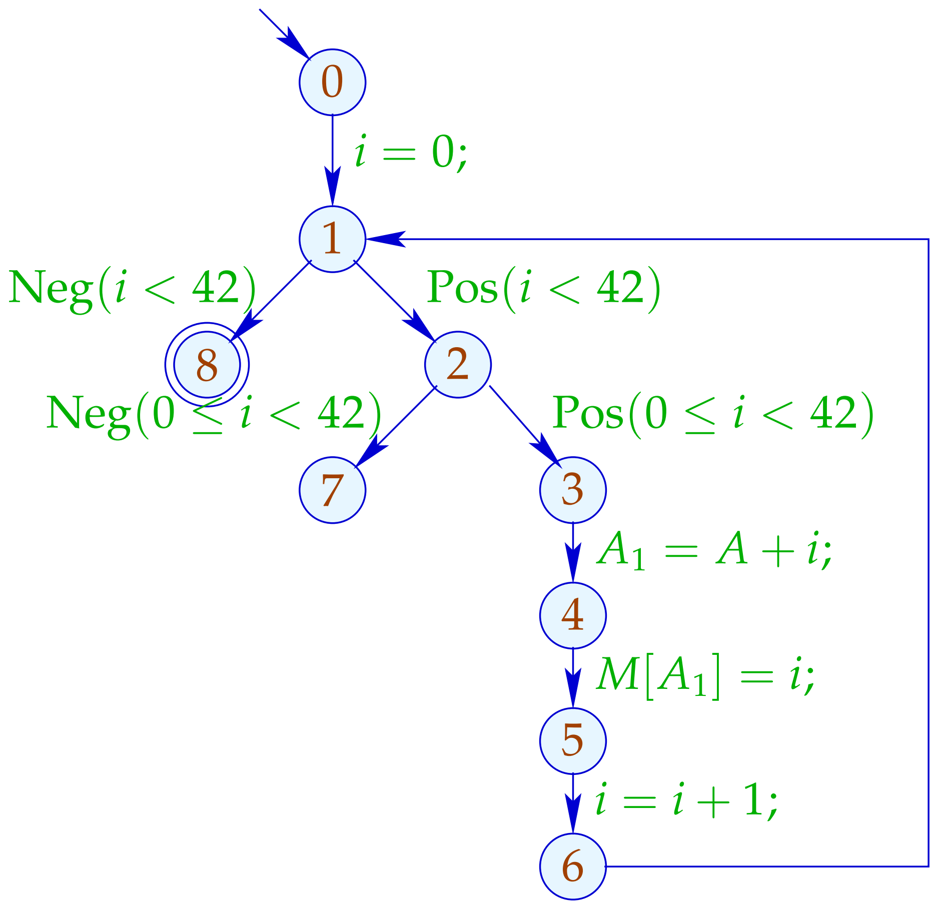
$$I_1 = \{1\} \quad \text{or:}$$
$$I_2 = \{2\} \quad \text{or:}$$
$$I_3 = \{3\}$$

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	$+\infty$		

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



	1		2		3		4
	l	u	l	u	l	u	
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$	dito
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	
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

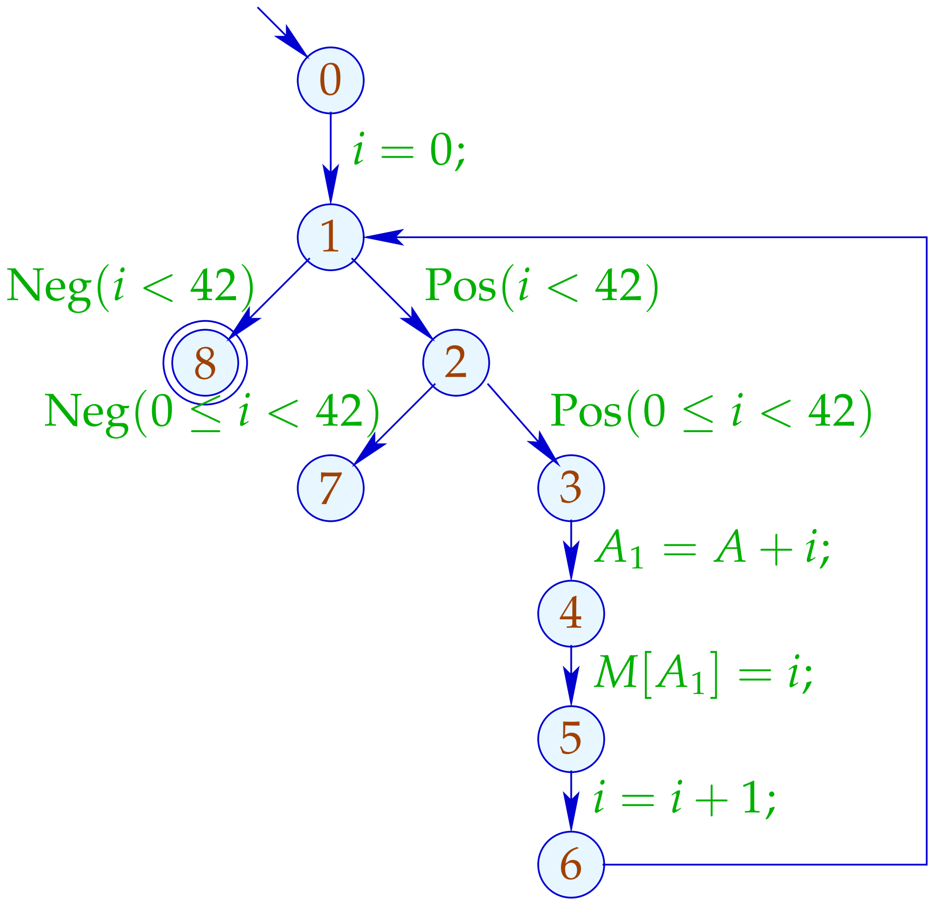
Every tuple $F^k\,\underline{x}$ is a solution of (1) :-)

$\Longrightarrow$

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

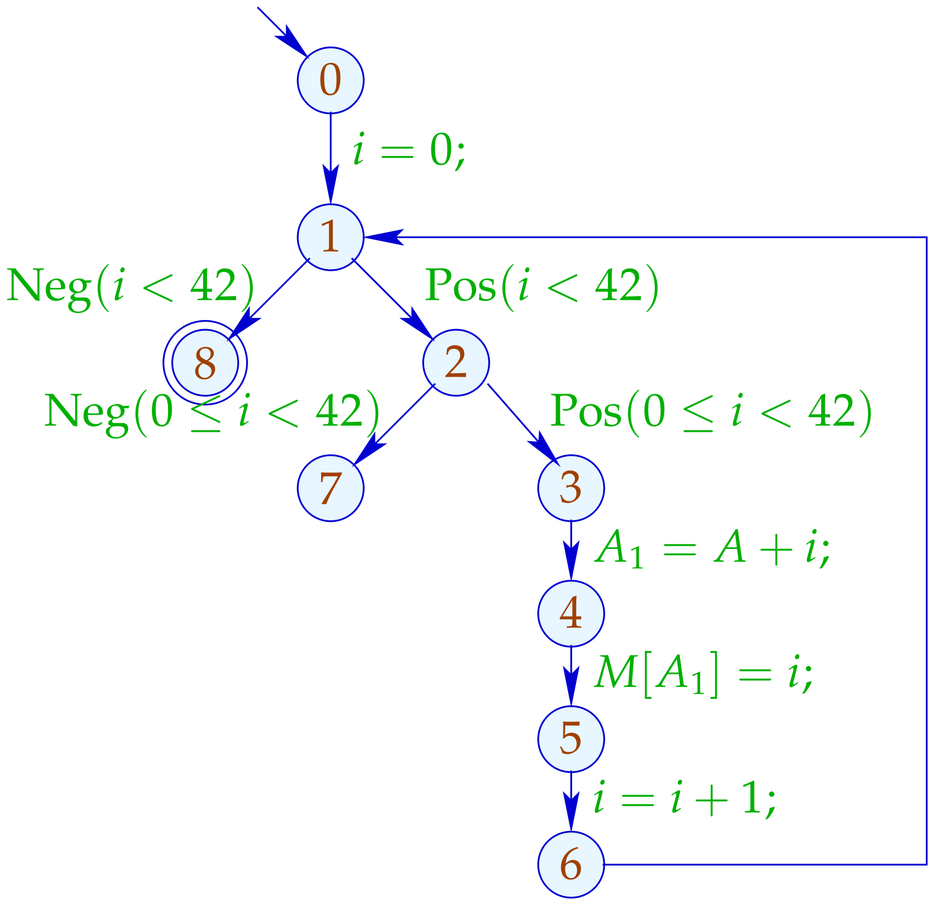
// The same also holds for RR-iteration.

Narrowing Iteration in the Example:



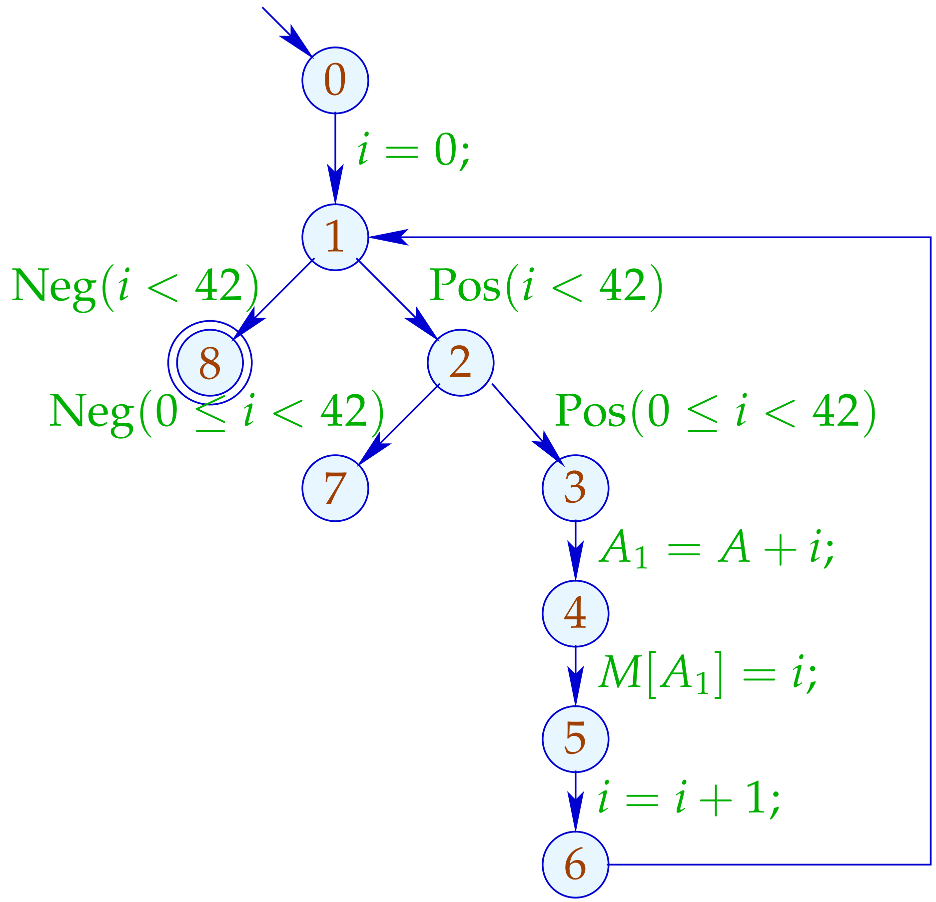
	0	
	l	u
0	$-\infty$	$+\infty$
1	0	$+\infty$
2	0	$+\infty$
3	0	$+\infty$
4	0	$+\infty$
5	0	$+\infty$
6	1	$+\infty$
7	42	$+\infty$
8	42	$+\infty$

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:

- $\longrightarrow$ We start with a safe approximation.
- $\longrightarrow$ We find that the inner check is redundant :-)
- $\longrightarrow$ We find that at exit from the loop, always $i = 42$:-))
- $\longrightarrow$ 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:

$$x_i \sqsupseteq f_i\,(x_1, \ldots, x_n)\;, \quad i = 1, \ldots, n \tag{1}$$

Then consider the system of equations:

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

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:

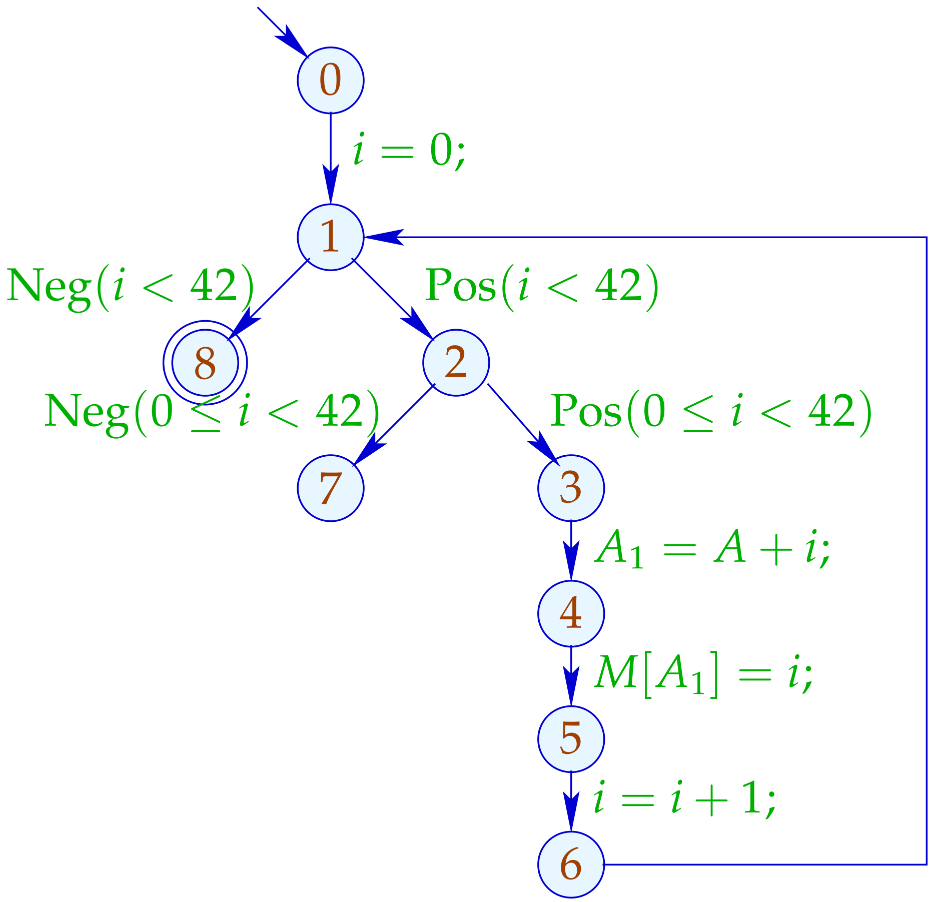
We preserve finite interval bounds :-)

Therefore, $\bot \sqcap\!\!\!\!\sqcap\, D \;=\; D \sqcap\!\!\!\!\sqcap\, \bot \;=\; \bot$ and for $D_1 \neq \bot \neq D_2$:

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

$\Longrightarrow$ $\sqcap\!\!\!\!\sqcap$ 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:

- $\longrightarrow$ Warning: Widening also returns for non-monotonic f_i a solution. Narrowing is only applicable to monotonic f_i !!
- $\longrightarrow$ In the example, accelerated narrowing already returns the optimal result :-)
- $\longrightarrow$ If the operator $\sqcap\!\!\!\!\sqcap$ only allows for finitely many improvements of values, we may execute narrowing until stabilization.
- $\longrightarrow$ 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:

- $\longrightarrow$ Are two addresses possibly equal? May Alias
- $\longrightarrow$ Are two addresses definitively equal? Must Alias

$\Longrightarrow$ 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:

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

(2) Values of Variables:

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

- Extend the Effects of Edges:

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

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

(3) Constant Propagation:

- Extend the abstract state by an abstract store M

- Execute accesses to known memory locations!

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

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

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

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 :-(

Simplification:

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

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

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

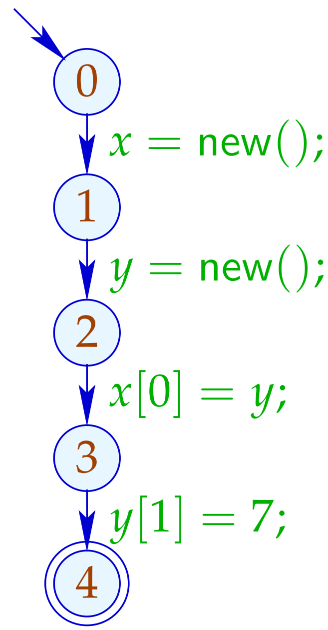
Simple Example:

$x = \mathsf{new}();$

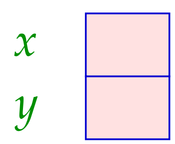
$y = \mathsf{new}();$

$x[0] = y;$

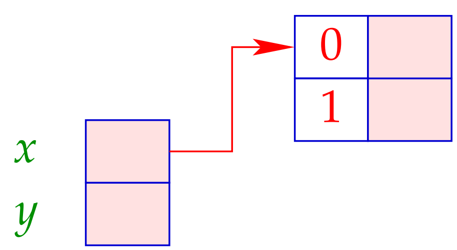
$y[1] = 7;$



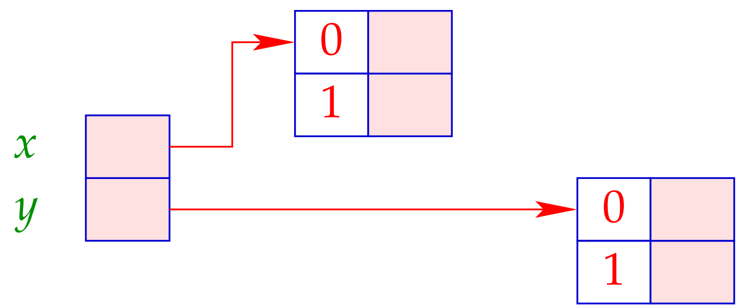
The Semantics:



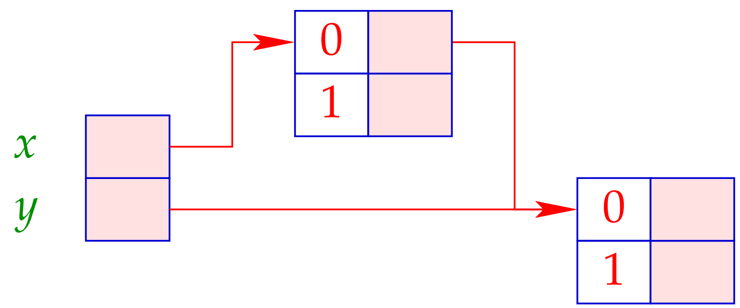
The Semantics:



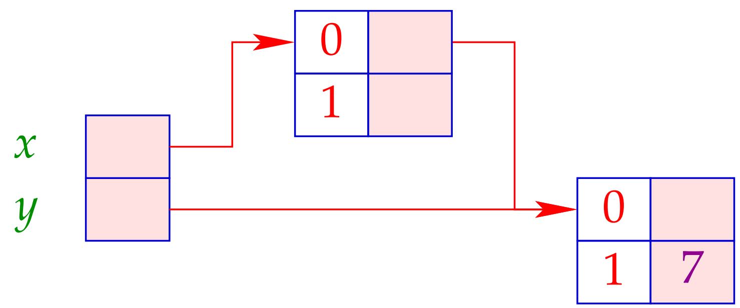
The Semantics:



The Semantics:



The Semantics:



More Complex Example:

$r = \text{Null};$

$\text{while } (t \neq \text{Null}) \ \{$

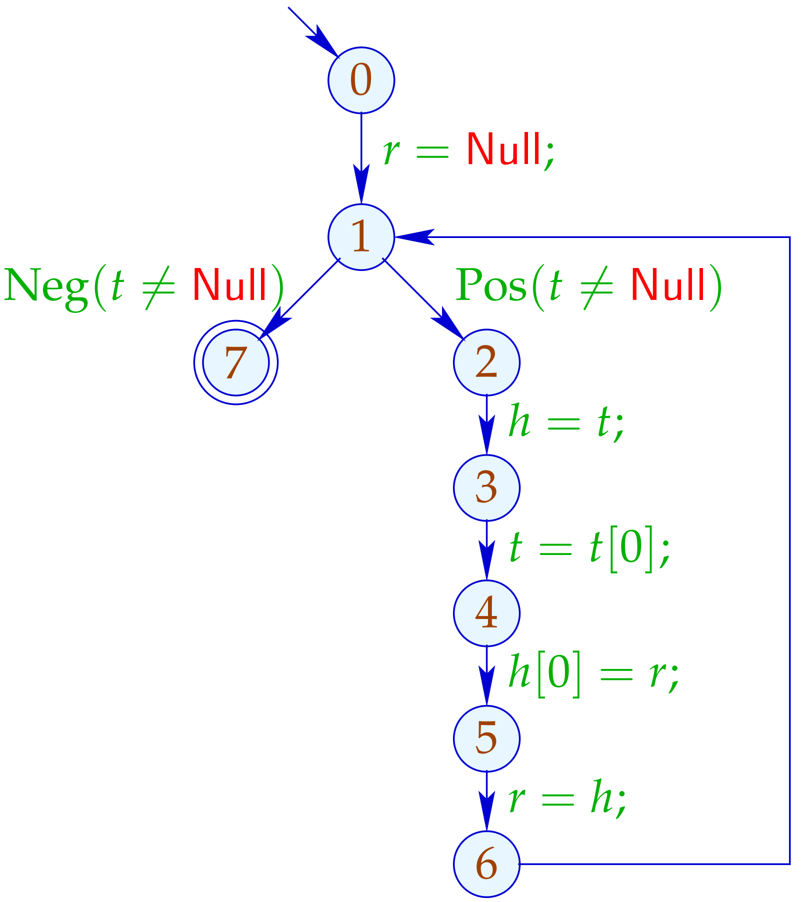
$\quad h = t;$

$\quad t = t[0];$

$\quad h[0] = r;$

$\quad r = h;$

$\}$



Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$
\begin{array}{lcll}
Addr_h & = & \{\mathsf{ref}\ a \mid 0 \leq a < h\} & // \quad \text{addresses} \\
Val_h & = & Addr_h \cup \mathbb{Z} & // \quad \text{values} \\
Store_h & = & (Addr_h \times \mathbb{N}_0) \to Val_h & // \quad \text{store} \\
State_h & = & (Vars \to Val_h) \times Store_h & // \quad \text{states}
\end{array}
$$

For simplicity, we set: $0 = \mathsf{Null}$