Computer aided verification

lecture 9

Abstract interpretation

Sławomir Lasota
University of Warsaw

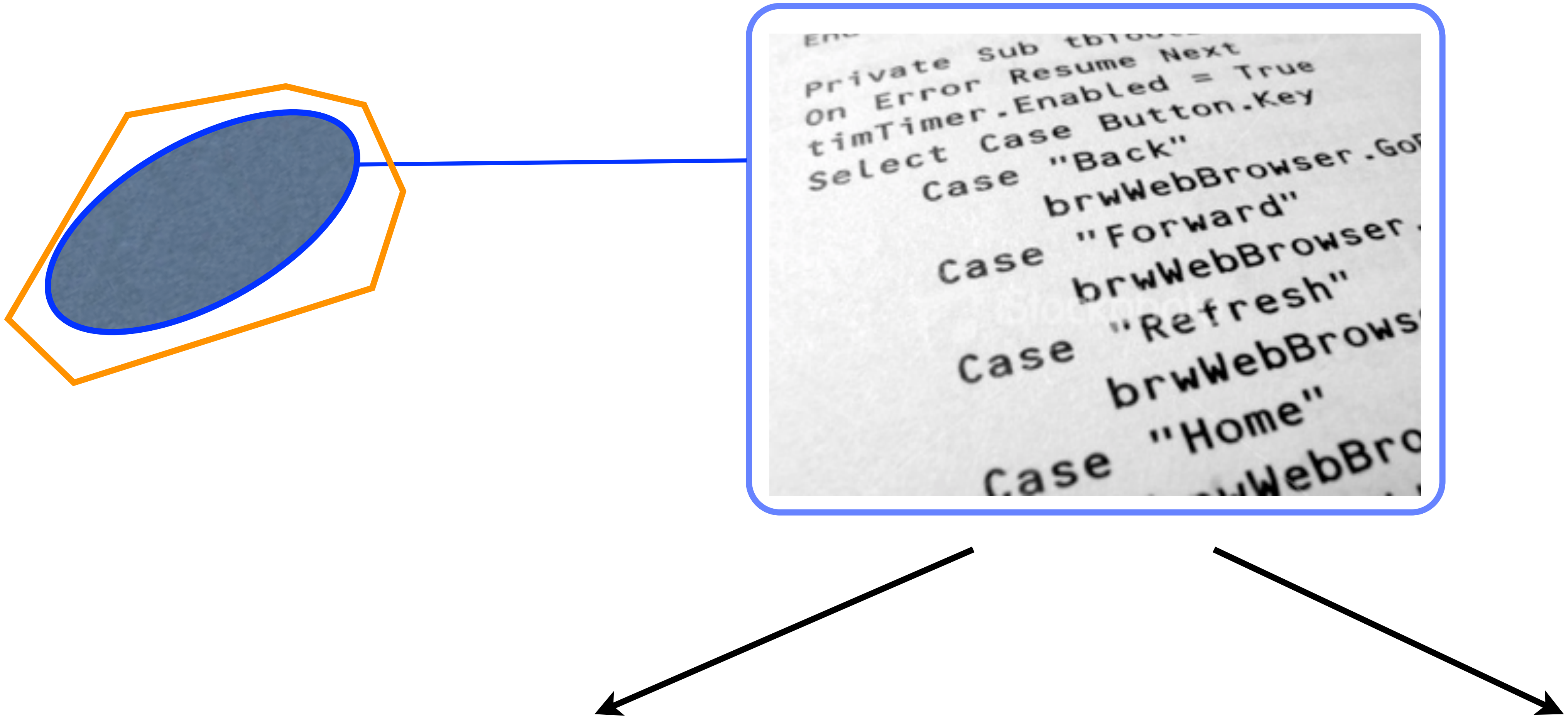
Literature

- F. Nielson, H.R. Nielson, C. Hankin, Principles of Program Analysis, Springer, 2005.
- http://www.imm.dtu.dk/~riis/PPA/slides2.pdf
- V. D'Silva, D. Kroening, G. Weissenbacher, A Survey of Automated Techniques for Formal Software Verification. IEEE Trans. on CAD of Integrated Circuits and Systems 27(7):1165-1178, 2008.

Pionieers

- P. Naur 1965
- P. Cousot, R. Cousot 1977

Approximate analysis



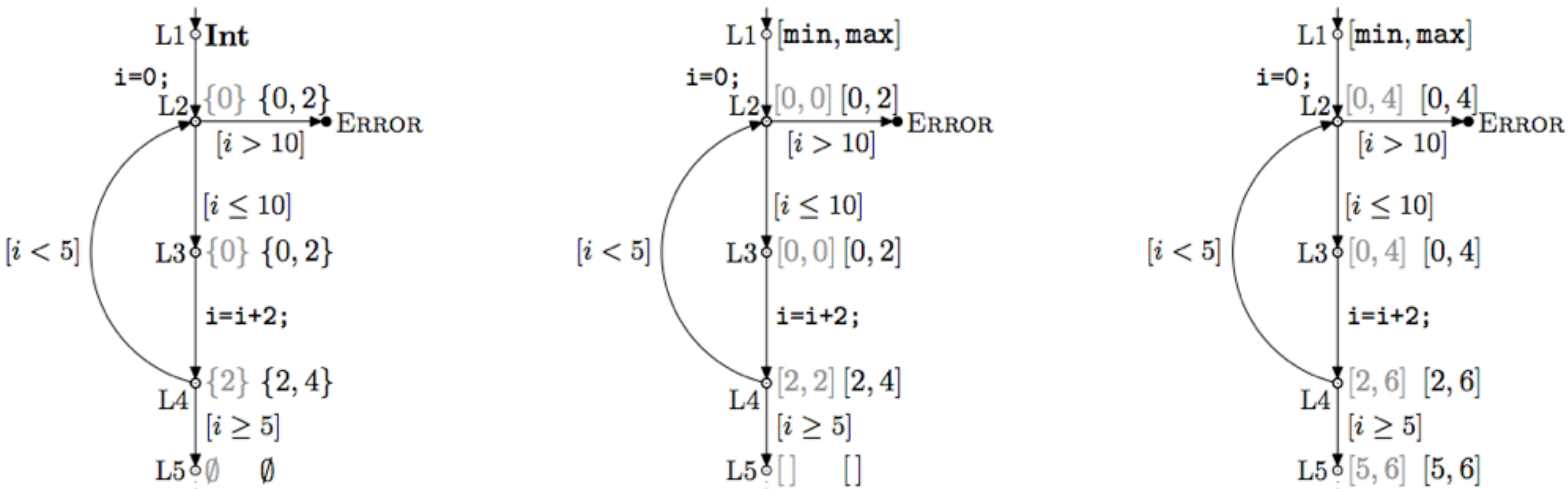
surely ✔

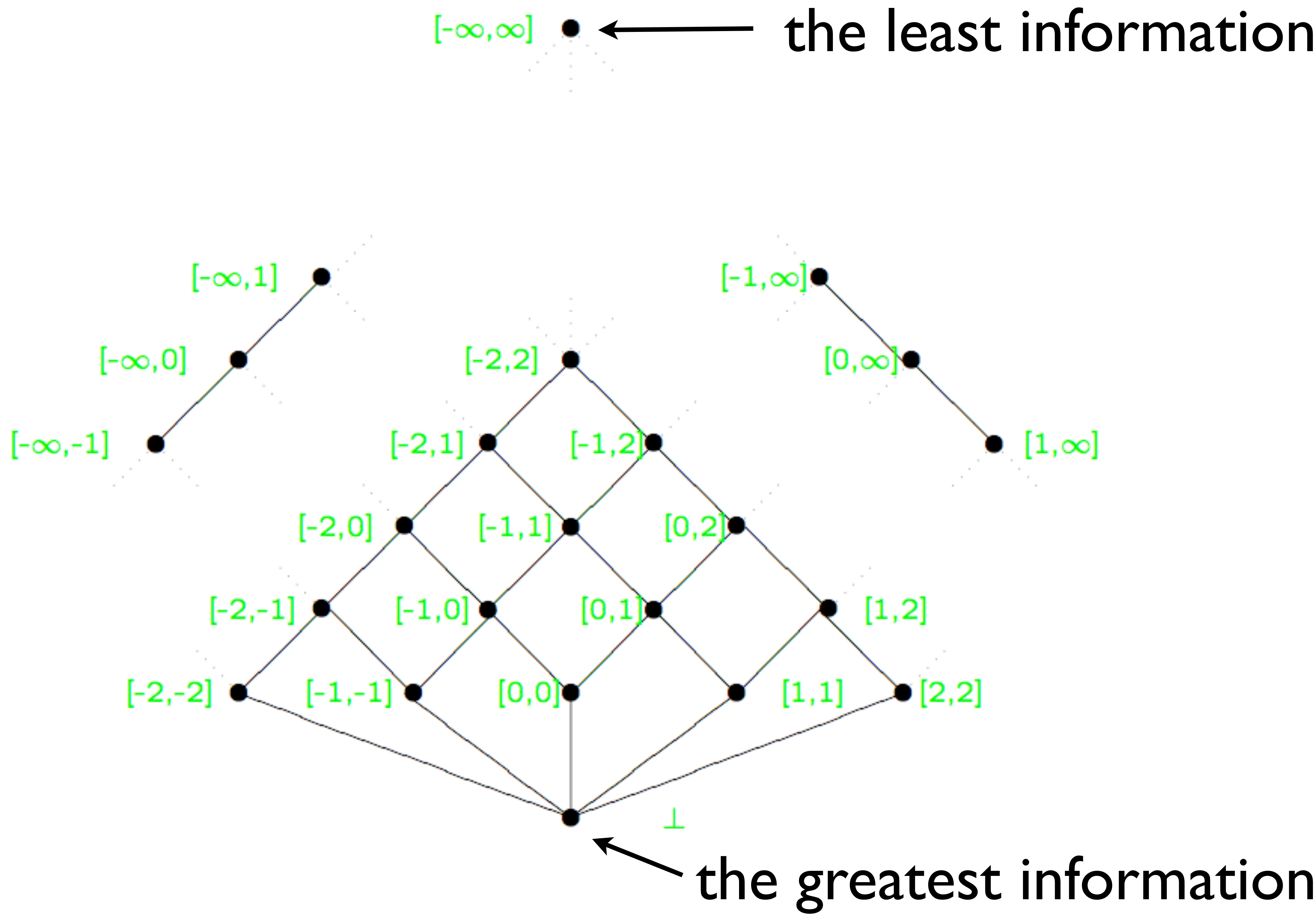
perhaps ✘

$$123 \cdot 457 + 76543 \quad \stackrel{?}{=} \quad 132654$$

$$
\begin{aligned}
123 \cdot 457 + 76543 &\stackrel{?}{=} 132654 \\
6 \cdot 7 + 7 &\stackrel{?}{=} 3 \pmod 9 \\
6 + 7 &\stackrel{?}{=} 3 \pmod 9
\end{aligned}
$$

```
int i = 0;
do {
  assert(i <= 10);
  i = i+2;
} while (i < 5);
```





[-∞,∞]
the least information
[-∞,1]
[-1,∞]
[-∞,0]
[-2,2]
[0,∞]
[-∞,-1]
[-2,1]
[-1,2]
[1,∞]
[-2,0]
[-1,1]
[0,2]
[-2,-1]
[-1,0]
[0,1]
[1,2]
[-2,-2]
[-1,-1]
[0,0]
[1,1]
[2,2]
⊥
the greatest information

Approximate analysis

- static analysis
- source code is analyzed (control-flow diagram)
- false alarms (false positives)
- typically oriented towards specific properties
- fully automatic
- scalable
- a diagnostic information if error

Approximate analysis - methods

- data-flow analysis
- control-flow analysis
- type analysis
- WCET analysis
- ...
- abstract interpretation

Approximate analysis - applications

- code optimization, program transformations
- program verification (static analysis)
- estimation of quality of code
- abstract interpretation - systematization

Code optimization

- constants propagation (at compile time)
- copies propagation
- available expressions analysis (elimination of comp.)
- live variables analysis (elimination of dead code)
- definition-use and use-definition analysis
- strictness analysis
- array bounds analysis
- ...

Program verification

- division by 0, exceptions
- pointers
  - NULL dereferences
  - static and dynamic data (stack and heap)
  - shape analysis
- aliasing analysis
- array bounds
- detection of invariants
- ...

Data-flow analyses

Data-flow analysis

$[y := x]^1$;

$[z := 1]^2$;

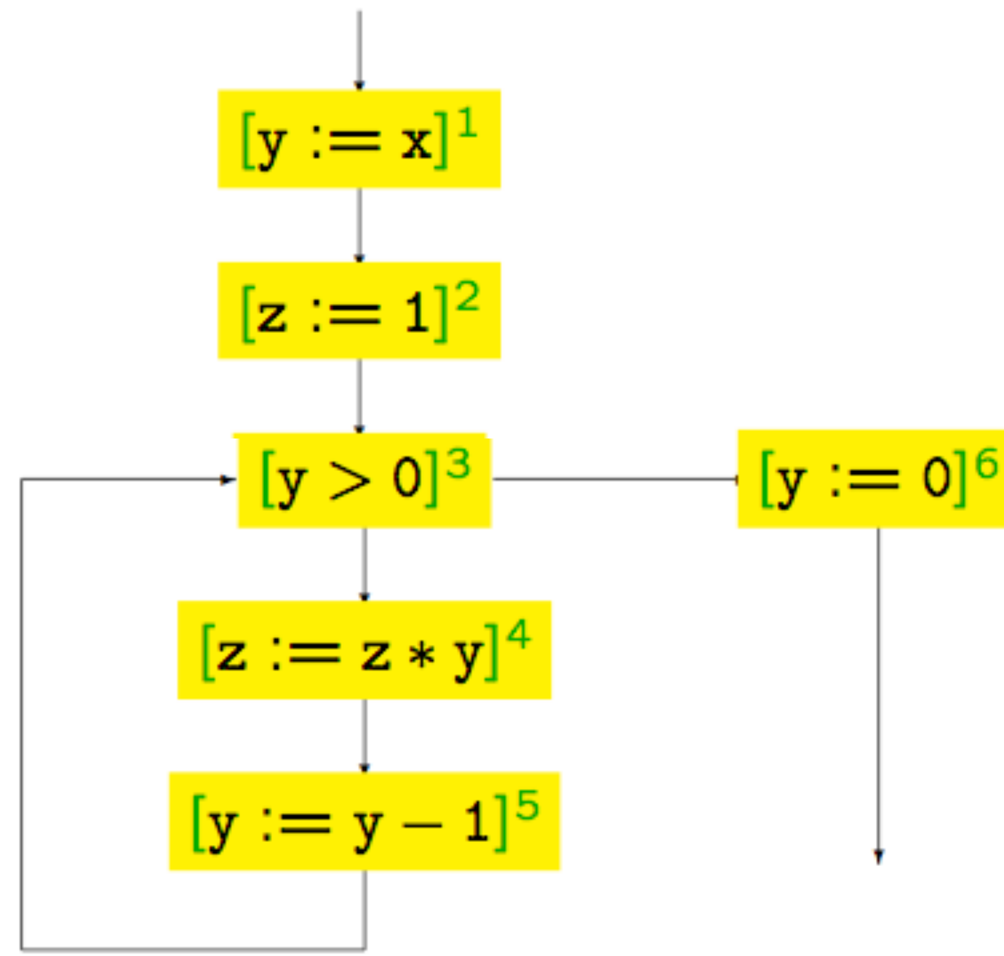
while $[y > 0]^3$ **do**

  $[z := z * y]^4$;

  $[y := y - 1]^5$

od;

$[y := 0]^6$



(while-programs)

finite set of control locations

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$\mathrm{State} = S \times \mathrm{Store}$$

$$\mathrm{Store} = \mathrm{Var} \to \mathrm{Val}$$

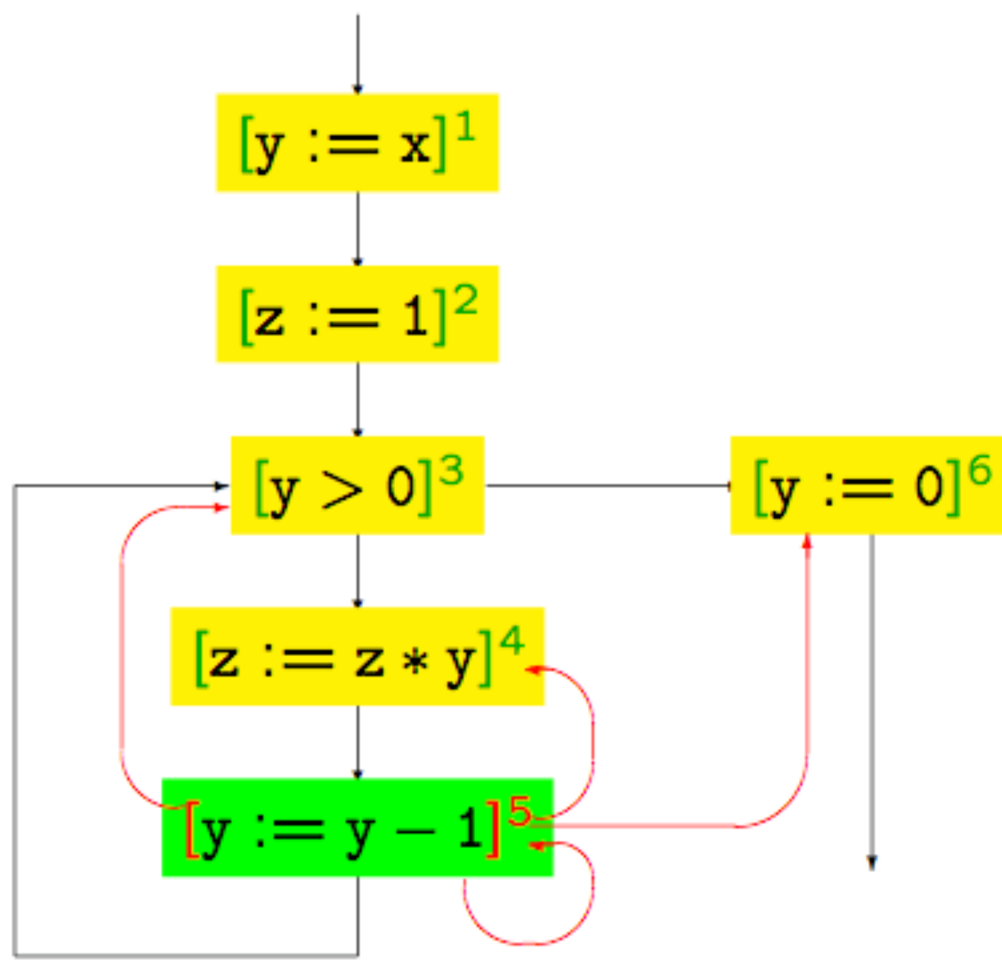
Data-flow analyses

- reaching definitions analysis
- available expressions analysis
- live variables analysis
- very busy expressions analysis
- constants propagation
- ...

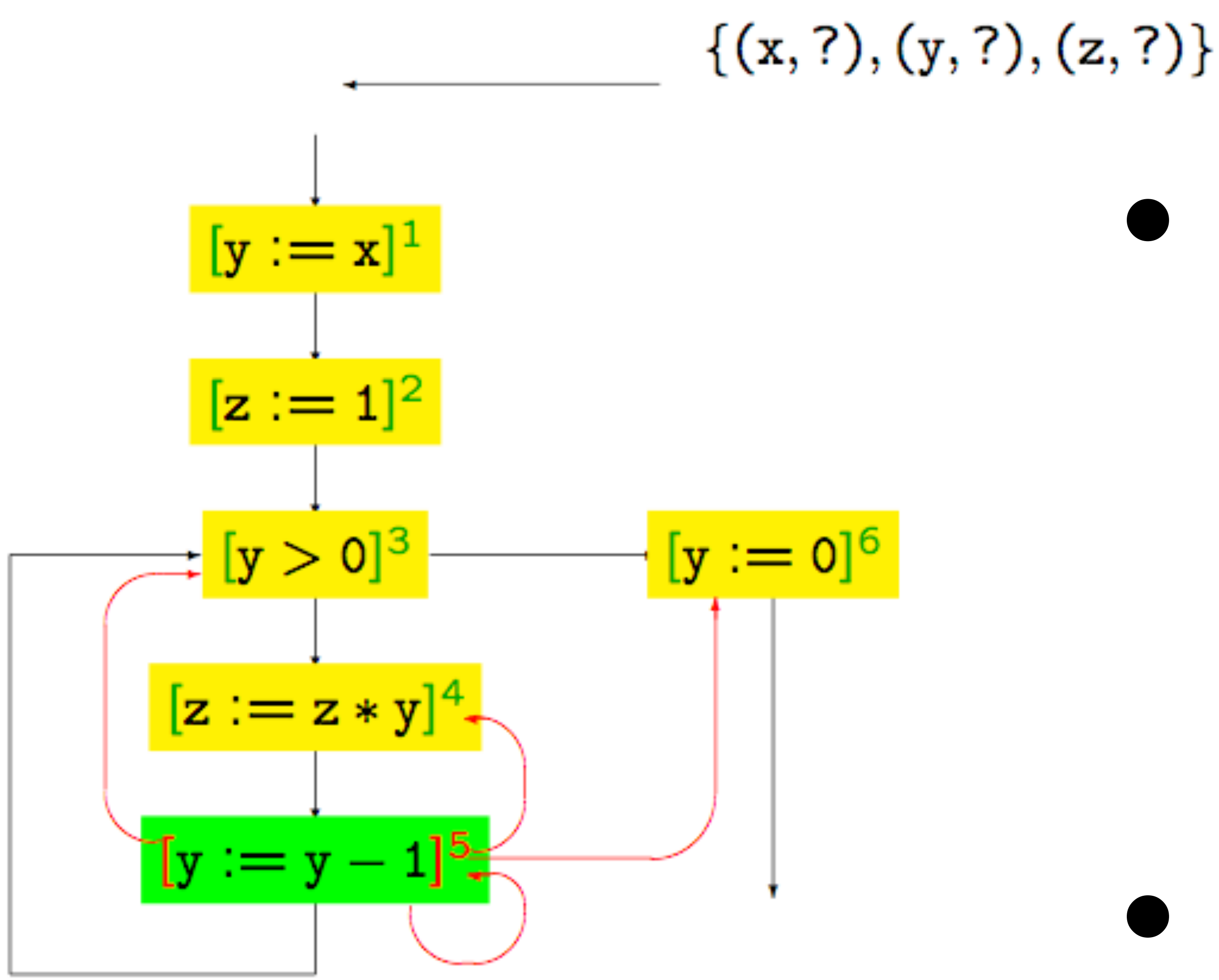
Reaching definitions analysis

In each control location compute the set of assignments that possibly have been executed (and not „overwritten”) prior to reaching this location.

Reaching definitions analysis



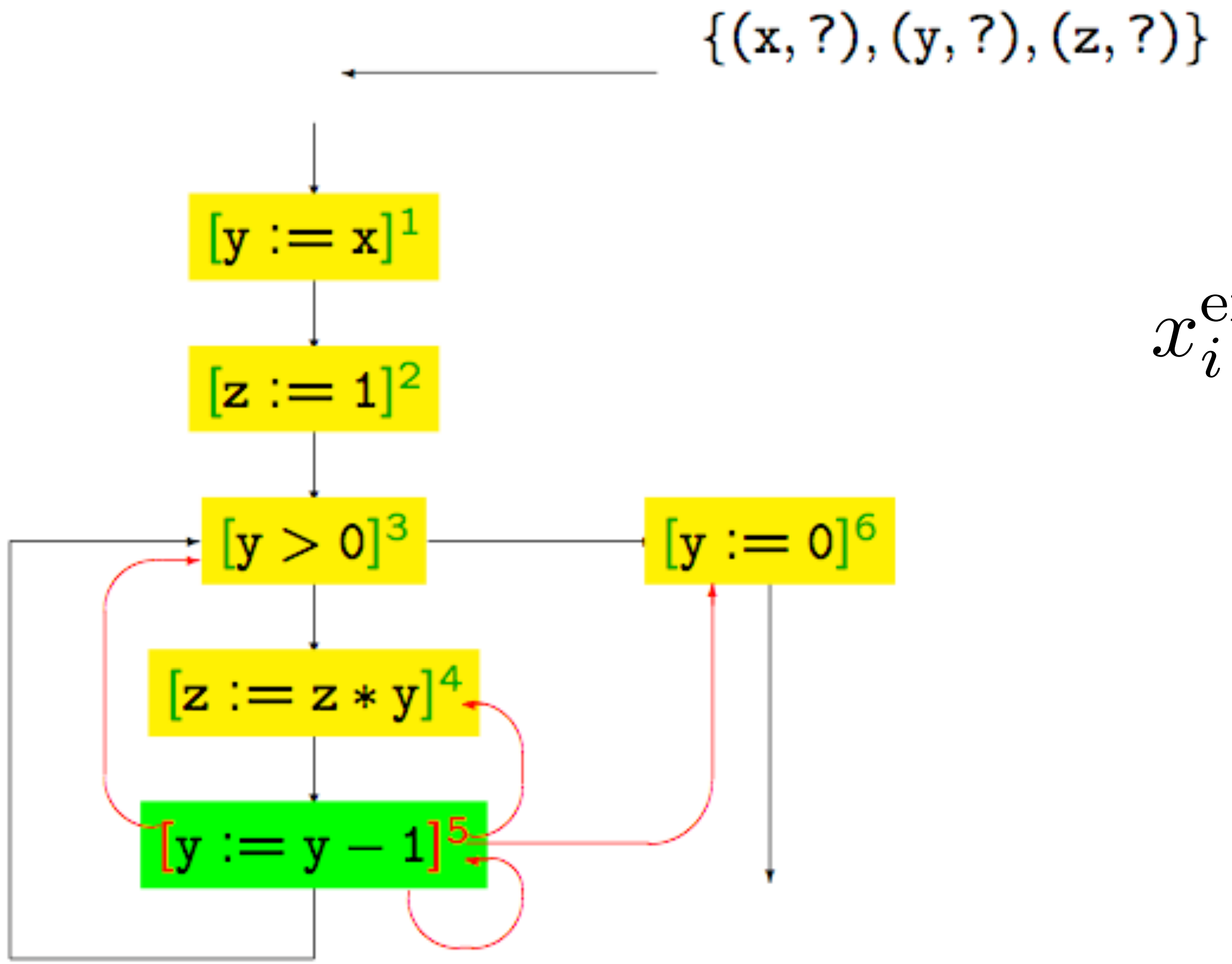
Reaching definitions analysis



kill(i)	gen(i)	U

- formalize the problem as a set of equations
  - variables represent information before and after each instruction
- the lest solution
- iterative algorithm

Reaching definitions analysis



$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \ \text{kill}(i) \ \cup \ \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcup_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \{(\mathrm{x}, ?), (\mathrm{y}, ?), (\mathrm{z}, ?)\}$$

we are interested in the least solution

Reaching definitions analysis

$$\mathrm{kill}(\mathrm{z} := \mathrm{e}) = \{(\mathrm{z}, ?)\} \cup \{(\mathrm{z}, j) \mid j \in S\}$$

$$\mathrm{kill}(\mathrm{b}) = \emptyset$$

$$\mathrm{kill}(\mathrm{skip}) = \emptyset$$

$$\mathrm{gen}([\mathrm{z} := \mathrm{e}]^i) = \{(\mathrm{z}, i)\}$$

$$\mathrm{gen}(\mathrm{b}) = \emptyset$$

$$\mathrm{gen}(\mathrm{skip}) = \emptyset$$

$$x_i^{\mathrm{exit}} = x_i^{\mathrm{entry}} \setminus \ \mathrm{kill}(i) \ \cup \ \mathrm{gen}(i)$$

$$x_i^{\mathrm{entry}} = \bigcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}}$$

$$x_1^{\mathrm{entry}} = \{(\mathtt{x}, ?), (\mathtt{y}, ?), (\mathtt{z}, ?)\}$$

we are interested in the least solution

Reaching definitions analysis

$\{(x,?),(y,?),(z,?)\}$

$[y := x]^1;$

$\{(x,?),(y,1),(z,?)\}$

$[z := 1]^2;$

$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$\texttt{while}\ [y > 0]^3\ \texttt{do}$

$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$\quad [z := z * y]^4;$

$\{(x,?),(y,1),(y,5),\boxed{(z,2)},(z,4)\}$

$\quad [y := y - 1]^5$

$\{(x,?),\boxed{(y,1)},(y,5),\boxed{(z,2)},(z,4)\}$

$\texttt{od};$

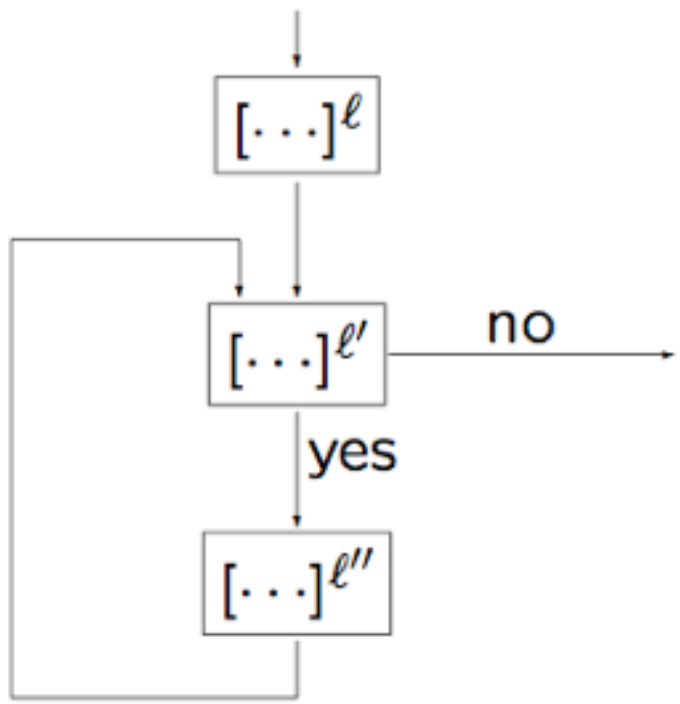
$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$[y := 0]^6$

$\{(x,?),(y,6),(z,2),(z,4)\}$

Reaching definitions analysis

$[\texttt{z:=x+y}]^{\ell}; \texttt{while}\ [\texttt{true}]^{\ell'}\ \texttt{do}\ [\texttt{skip}]^{\ell''}$



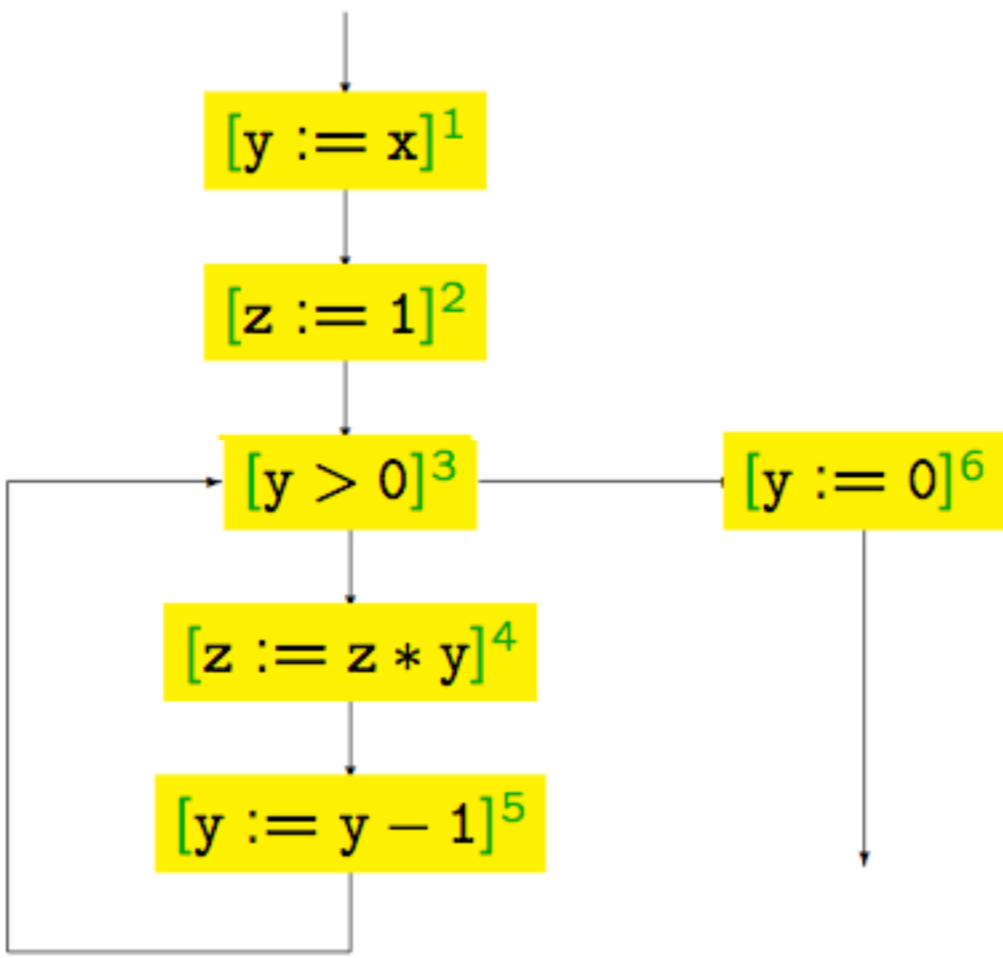
solutions: $x_{l'}^{\text{entry}} \supseteq \{(\texttt{x}, ?), (\texttt{y}, ?), (\texttt{z}, \texttt{l})\}$

we are interested in the least solution

Available expressions analysis

In each control location compute the set of expressions whose value is surely computed whenever this location is entered.

Available expressions analysis



$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \ \text{kill}(i) \ \cup \ \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{j \leadsto i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \emptyset$$

we are interested in the greatest solution

Available expressions analysis

`[x:=a+b]`^{1}; `[y:=a*b]`^{2}; `while [y> a+b ]`^{3} `do ([a:=a+1]`^{4}; `[x:=a+b]`^{5}`)`

i	x_i^{entry}	x_i^{exit}
1	$\emptyset$	{a+b}
2	{a+b}	{a+b, a*b}
3	{a+b}	{a+b}
4	{a+b}	$\emptyset$
5	$\emptyset$	{a+b}

$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \ \text{kill}(i) \ \cup \ \text{gen}(i)$$

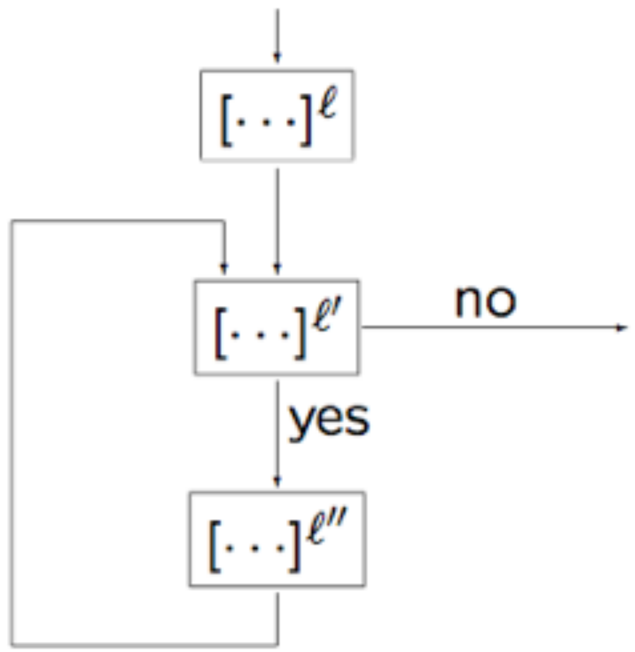
$$x_i^{\text{entry}} = \bigcap_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \emptyset$$

we are interested in the greatest solution

Available expressions analysis

[z:=x+y]$^{\ell}$; while [true]$^{\ell'}$ do [skip]$^{\ell''}$



$$x_i^{\text{exit}} = x_i^{\text{entry}} \setminus \ \text{kill}(i) \ \cup \ \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{j \rightsquigarrow i} x_j^{\text{exit}}$$

$$x_1^{\text{entry}} = \emptyset$$

two solutions:

$$x_{l'}^{\text{entry}} \ = \ \{x + y\}, \ \emptyset$$

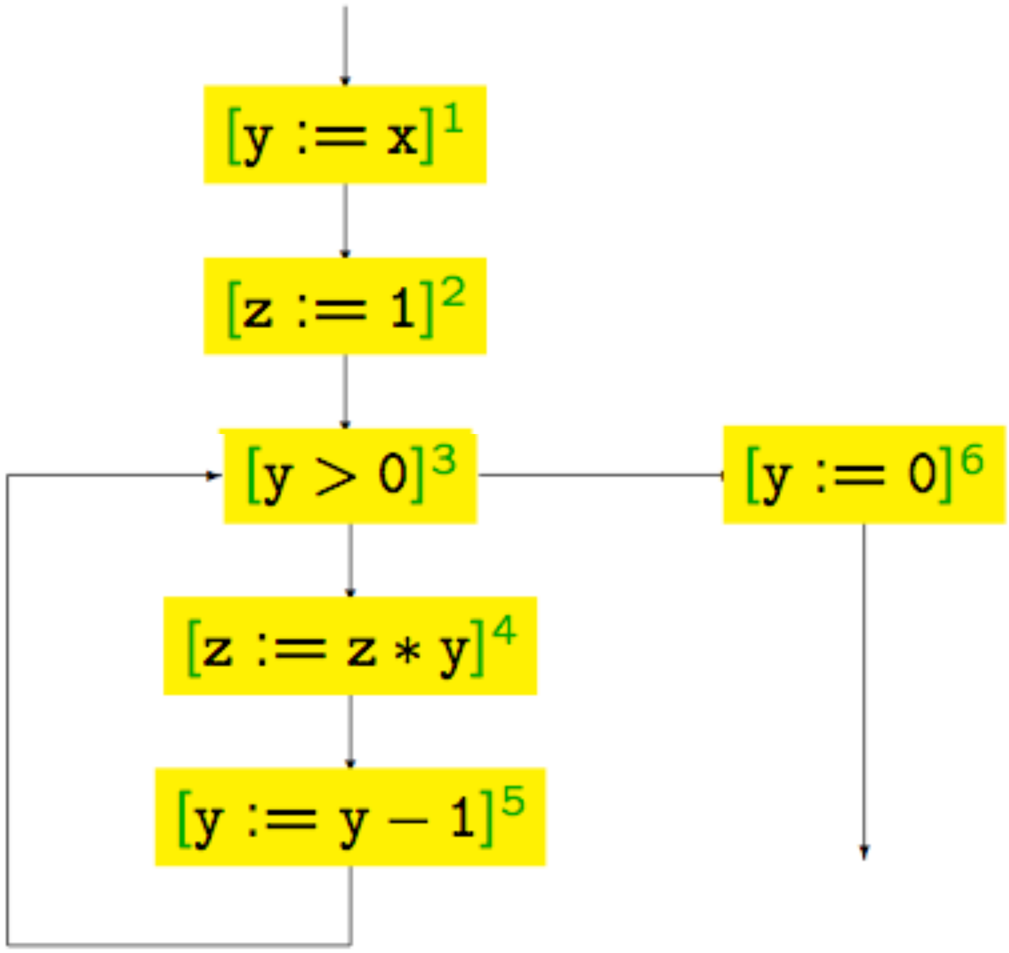
we are interested in the greatest solution

Live variables analysis

In each program location compute the set of variables that are live (possibly used in future before being redefined) when exiting this location.

backwards analysis

Live variables analysis



$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

$$x_i^{\text{exit}} = \bigcup_{i \rightsquigarrow j} x_j^{\text{entry}}$$

$$x_6^{\text{exit}} = \emptyset$$

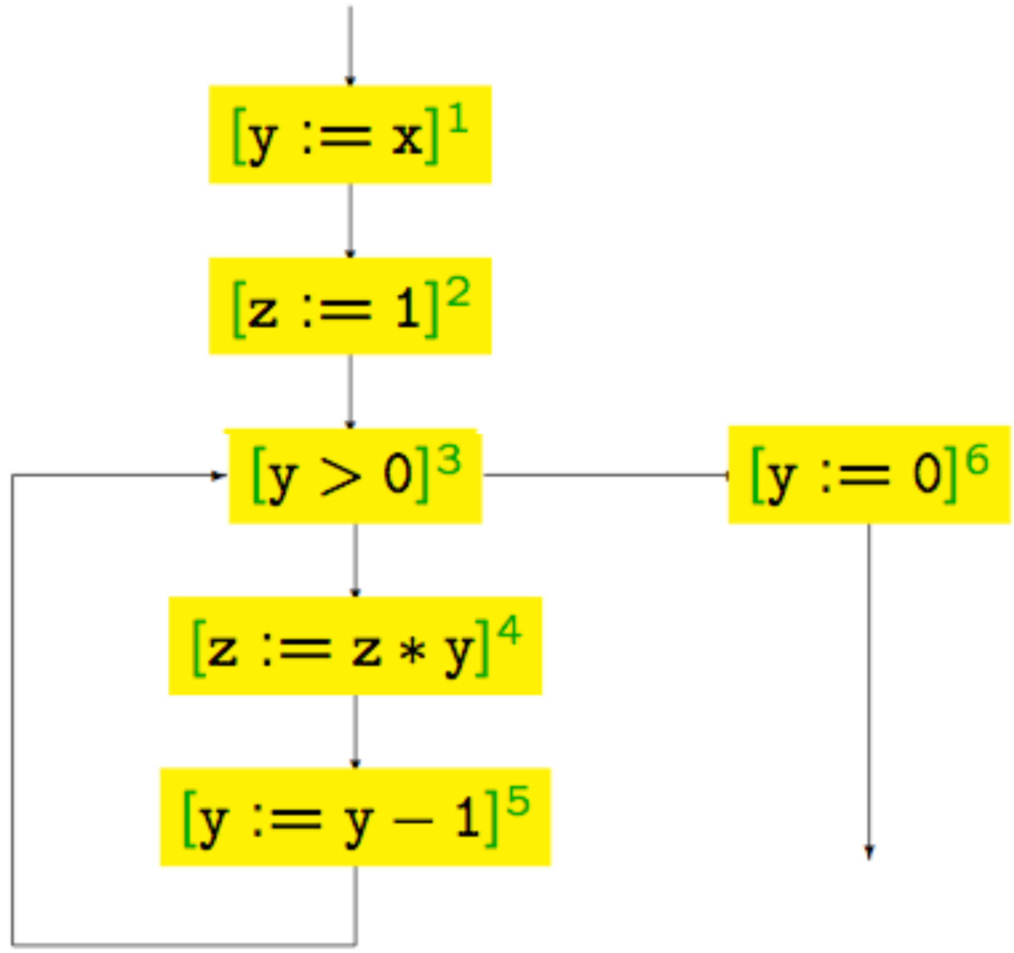
we are interested in the least solution

Very busy expressions analysis

In each control location compute the set of expressions that will surely be computed in future before redefinition of any of variables appearing in the expression.

backwards analysis

Very busy expressions analysis



$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \cup \text{gen}(i)$$

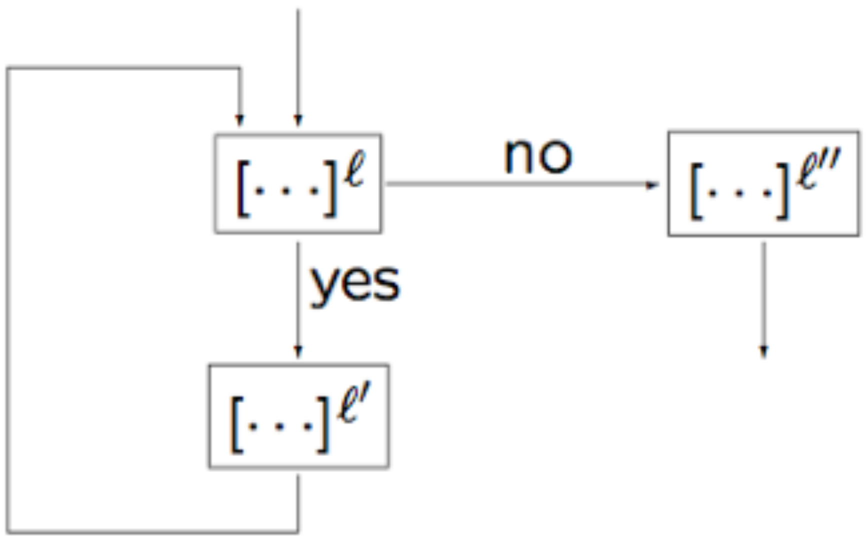
$$x_i^{\text{entry}} = \bigcap_{i \rightsquigarrow j} x_j^{\text{entry}}$$

$$x_6^{\text{exit}} = \emptyset$$

we are interested in the greatest solution

Very busy expressions analysis

(while [x>1]$^\ell$ do [skip]$^{\ell'}$); [x:=x+1]$^{\ell''}$



$$x_i^{\text{entry}} = x_i^{\text{exit}} \setminus \text{kill}(i) \; \cup \; \text{gen}(i)$$

$$x_i^{\text{entry}} = \bigcap_{i \rightsquigarrow j} x_j^{\text{entry}}$$

two solutions:

$$x_l^{\text{exit}} = \{x+1\}, \; \emptyset \qquad\qquad x_6^{\text{exit}} = \emptyset$$

we are interested in the greatest solution

Abstract interpretation

Generalization

L - abstract space

$(L, \sqsubseteq)$ - complete lattice

$\sqcup, \sqcap$ - bounds

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

Generalization

abstract interpretation

L - abstract space

$(L, \sqsubseteq)$ - complete lattice

$\sqcup, \sqcap$ - bounds

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

Data-flow analyses

- reaching definitions analysis
- available expressions analysis
- live variables analysis
- very busy expressions analysis
- constants propagation
- ...

Data-flow analyses

- reaching definitions analysis $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- available expressions analysis
- live variables analysis
- very busy expressions analysis
- constants propagation
- ...

Data-flow analyses

- reaching definitions analysis $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- available expressions analysis $\mathcal{P}(\mathrm{Expr})$
- live variables analysis
- very busy expressions analysis
- constants propagation
- ...

Data-flow analyses

- reaching definitions analysis $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- available expressions analysis $\mathcal{P}(\mathrm{Expr})$
- live variables analysis $\mathcal{P}(\mathrm{Var})$
- very busy expressions analysis
- constants propagation
- ...

Data-flow analyses

- reaching definitions analysis $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- available expressions analysis $\mathcal{P}(\mathrm{Expr})$
- live variables analysis $\mathcal{P}(\mathrm{Var})$
- very busy expressions analysis $\mathcal{P}(\mathrm{Expr})$
- constants propagation
- ...

Data-flow analyses

- reaching definitions analysis $\mathcal{P}(\mathrm{Var} \times (\mathcal{S} \cup \{?\}))$
- available expressions analysis $\mathcal{P}(\mathrm{Expr})$
- live variables analysis $\mathcal{P}(\mathrm{Var})$
- very busy expressions analysis $\mathcal{P}(\mathrm{Expr})$
- constants propagation $\mathrm{Var} \to \mathbb{Z}^{\top}$
- ...

Distributivity

$$f(s)(l_1 \sqcup l_2) \quad = \quad f(s)(l_1) \;\sqcup\; f(s)(l_2)$$

holds whenever:

$$L = \mathcal{P}(\mathcal{D}) \quad \mathcal{D} - finite$$

$$f(s)(l) = l \setminus l_1 \cup l_2$$

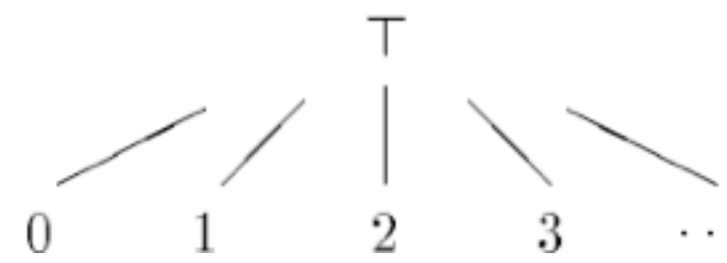
may not hold

Constants propagation

In each control location compute the set of variables that have a constant value independent from the history.

Constants propagation

$$L = \mathrm{Var} \to \mathbb{Z}^{\top}$$



$$[\mathtt{x}:=6]^1; [\mathtt{y}:=3]^2; \mathtt{while}\ [\mathtt{x} > \mathtt{y}]^3\ \mathtt{do}\ ([\mathtt{x}:=\mathtt{x}-1]^4; [\mathtt{z}:=\mathtt{y} * \mathtt{y}]^6)$$

$$[\mathtt{x}:=6]^1; [\mathtt{y}:=3]^2; \mathtt{while}\ [\mathtt{x} > 3]^3\ \mathtt{do}\ ([\mathtt{x}:=\mathtt{x}-1]^4; [\mathtt{z}:=9]^6)$$

$$l_1(\mathrm{y}) = 5 \qquad l_2(\mathrm{y}) = -5$$

$$f(6)(l_1 \sqcup l_2)(\mathtt{z}) = \top$$

$$f(6)(l_1)(\mathtt{z}) = f(6)(l_2)(\mathtt{z}) = 25$$

Algorithm

$$x_i^{\text{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\text{exit}} \quad \sqcup \quad f_{\text{init}}(x)$$

$$x_i^{\text{exit}} = f(x)(x_i^{\text{entry}})$$

$$\Rightarrow \qquad \vec{x} = \vec{f}(\vec{x})$$

complete lattice $L^{S \times \{\text{entry},\text{exit}\}}$ with the coordinate-wise order

the lest fix-point of the monotonic function $\vec{f}$

iterative algorithm

we assume that L has only finite chains

LFP

L - abstract space

$(L, \sqsubseteq)$ - complete lattice

$\sqcup, \sqcap$ - bounds

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$x_i^{\mathrm{entry}} = \bigsqcup_{j \rightsquigarrow i} x_j^{\mathrm{exit}} \quad \sqcup \quad f_{\mathrm{init}}(x)$$

$$x_i^{\mathrm{exit}} = f(x)(x_i^{\mathrm{entry}})$$

MOP

L - abstract space

$(L, \sqsubseteq)$ - complete lattice

$\sqcup, \sqcap$ - bounds

$f : S \to \mathrm{Mon}(L \to L)$

$f_{\mathrm{init}} : \mathrm{init}(S) \to L$

$S = \{1, \ldots, n\}, \quad \rightsquigarrow \subseteq S \times S$

$\mathrm{init}(S) \subseteq S$

$$y_i^{\mathrm{entry}} = \bigsqcup \{ f(p) \mid p \in \mathrm{paths}^{\mathrm{entry}}(i) \}$$

$$y_i^{\mathrm{exit}} = \bigsqcup \{ f(p) \mid p \in \mathrm{paths}^{\mathrm{exit}}(i) \}$$

$$\text{MOP} \sqsubseteq \text{LFP}$$

$$\vec{y} \sqsubseteq \vec{x}$$

$$\text{MOP} \sqsubseteq \text{LFP}$$

not always computable

$$\vec{y} \sqsubseteq \vec{x}$$

MOP = LFP

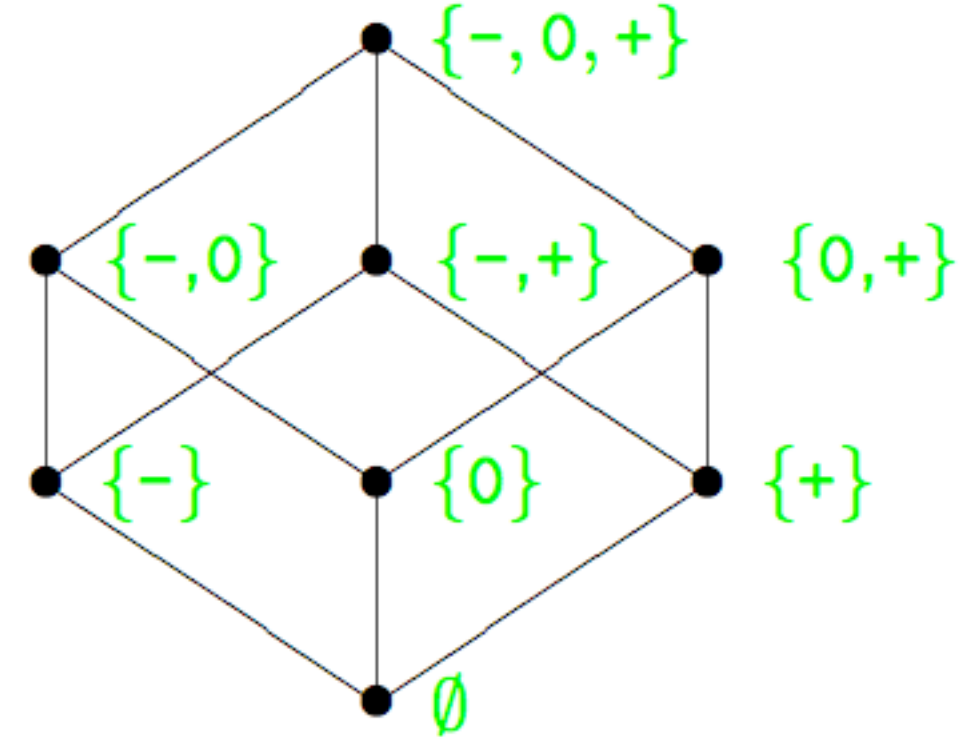
$$\vec{y} = \vec{x}$$

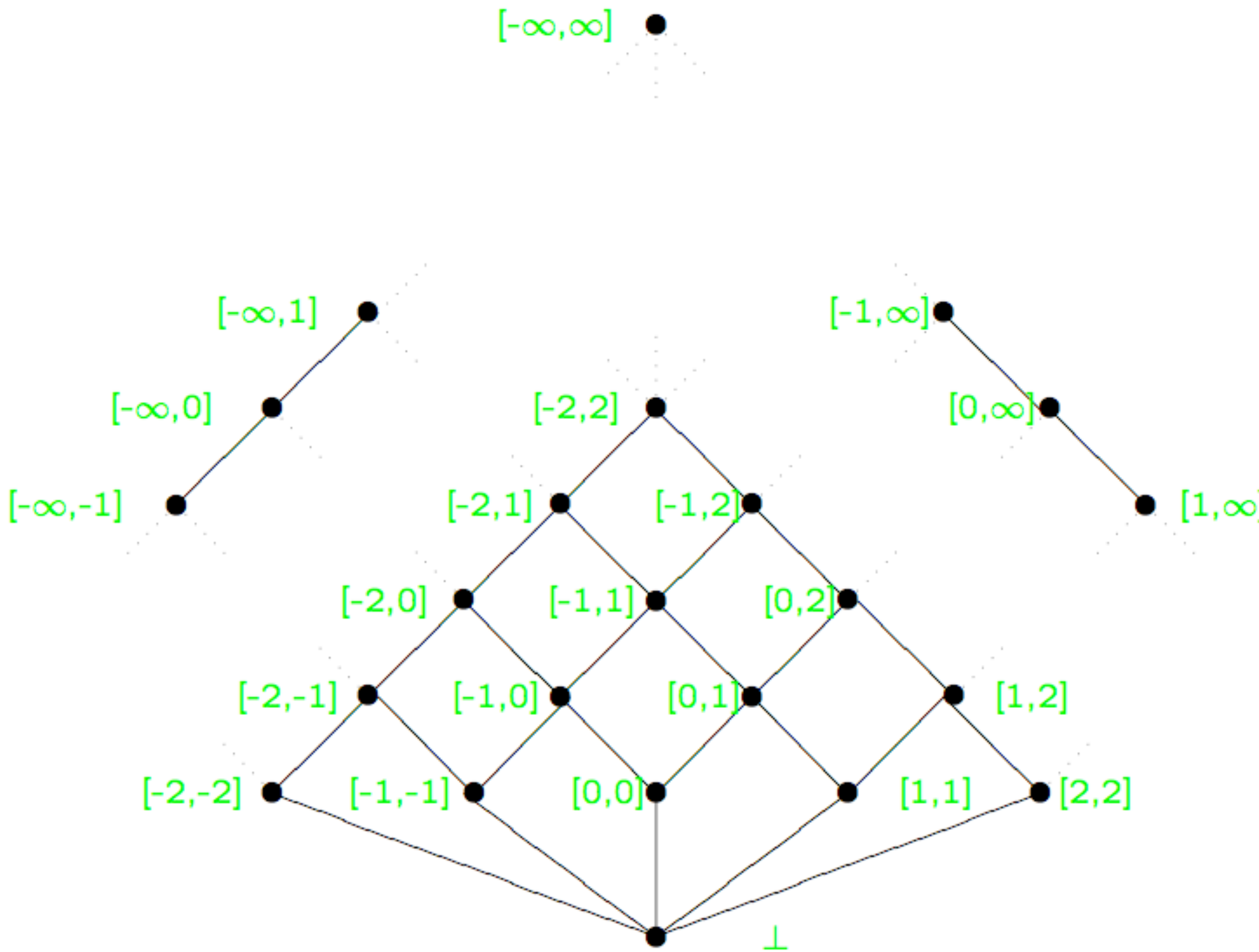
when distributivity holds

Abstract domains

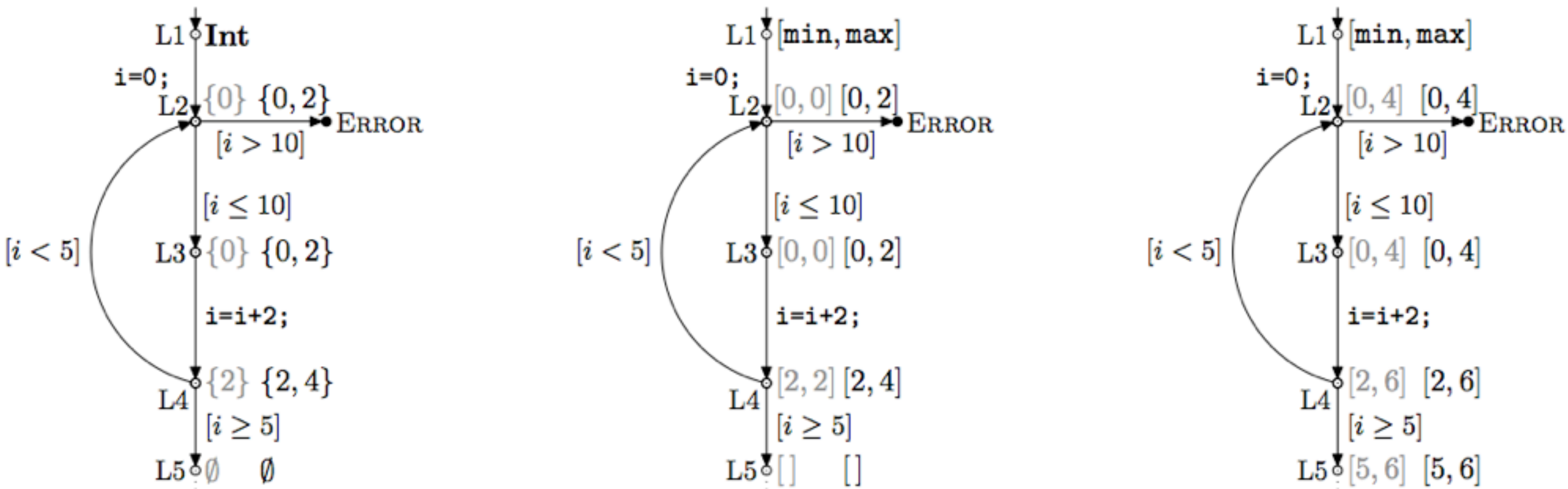
Non-relational domains

- signs $\mathcal{P}(-, 0, +)$
- intervals $[n, m]$
- parity
- congruence modulo k




[-∞,∞]
[-∞,1]
[-1,∞]
[-∞,0]
[-2,2]
[0,∞]
[-∞,-1]
[-2,1]
[-1,2]
[1,∞]
[-2,0]
[-1,1]
[0,2]
[-2,-1]
[-1,0]
[0,1]
[1,2]
[-2,-2]
[-1,-1]
[0,0]
[1,1]
[2,2]
⊥

```
int i = 0;
do {
   assert(i <= 10);
   i = i+2;
} while (i < 5);
```



Expressive power

precision

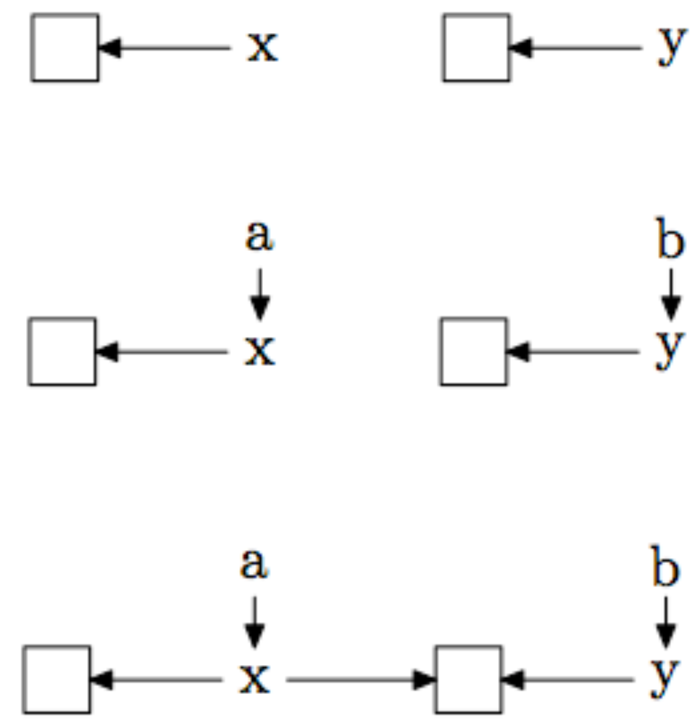
- signs
- intervals
- DBM (difference bounds matrices) $x - y \leq c$
- octagon $\pm x \pm y \leq c$
- octahedra $\pm x_1 \ldots \pm x_n \leq c$
- polyhedra $a_1 x_1 + \ \ldots \ + a_n x_n \leq c$

Pointer analyses domains

- points-to graphs

Example: alias analysis

```
int **a, **b, *x, *y;
x = (int*) malloc(sizeof(int));
y = (int*) malloc(sizeof(int));
a = &x;
b = &y;
*a = y;
```

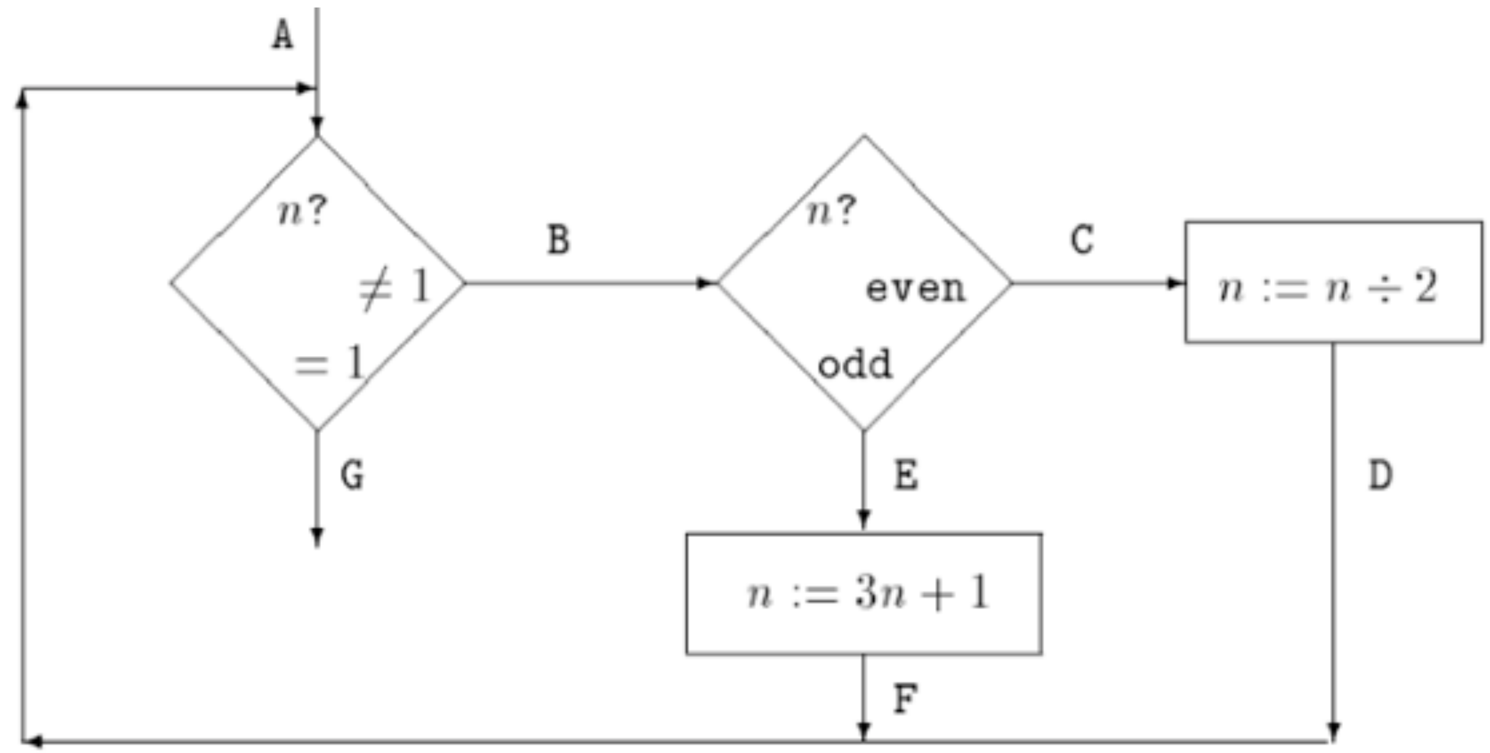


a and b do not point to the same location

x and y may point to the same location

Abstract semantics

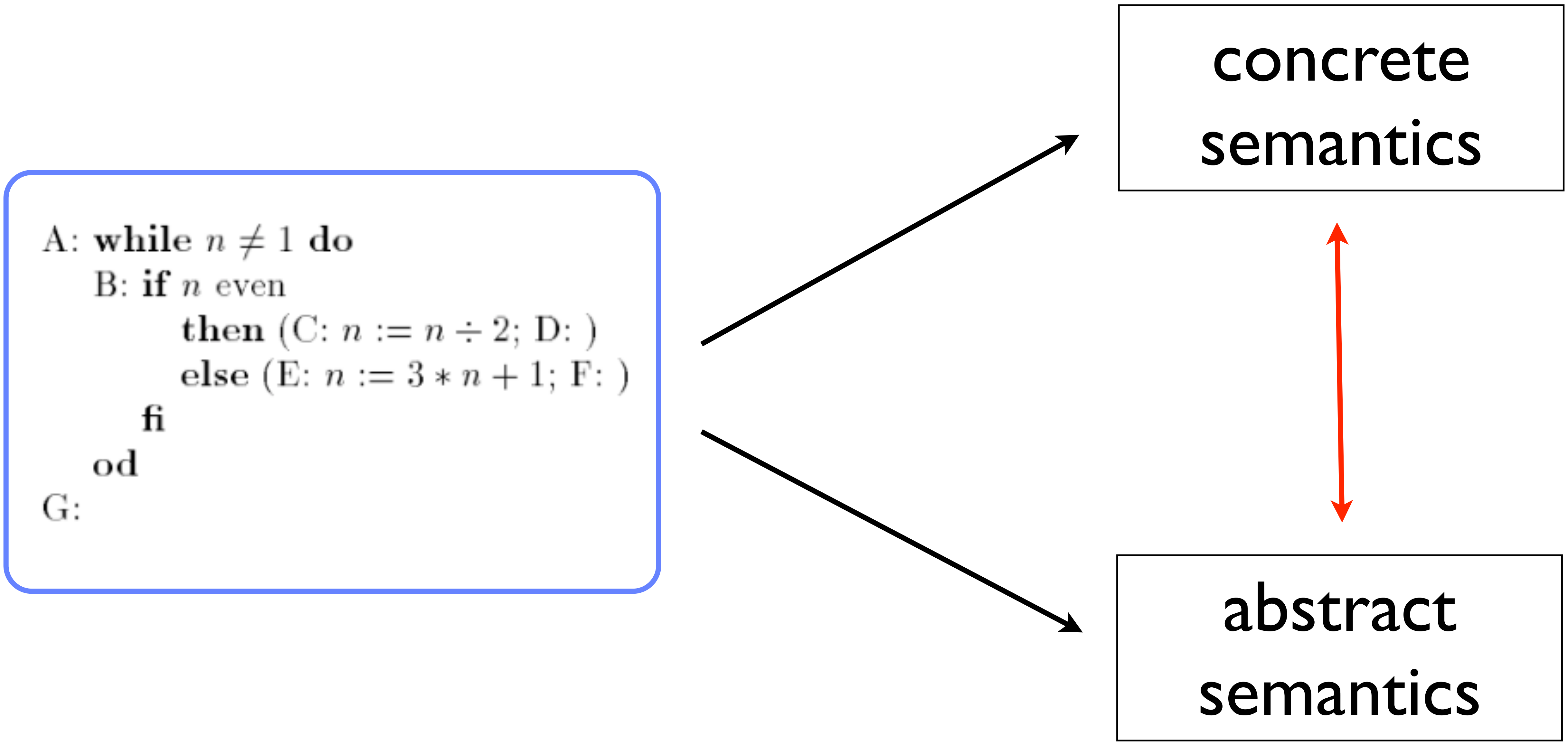
A: **while** $n \neq 1$ **do**
B: **if** n even
then (C: $n := n \div 2$; D:)
else (E: $n := 3 * n + 1$; F:)
fi
od
G:



$$S = \{\text{A, B, C, D, E, F, G}\}$$

$$\text{State} \;=\; S \times \text{Store}$$

$$\text{Store} \;=\; \text{Var} \to \text{Val}$$


A: while $n \neq 1$ do
B: if n even
then (C: $n := n \div 2$; D:)
else (E: $n := 3 * n + 1$; F:)
fi
od
G:
concrete semantics
abstract semantics

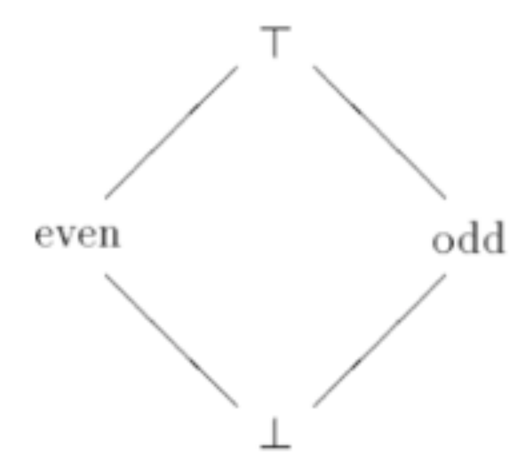
Domains

concrete semantics

$$V = \text{Store} = \text{Var} \to \mathbb{Z}$$

abstract semantics

$$L = \text{Var} \to \{\bot, \text{even}, \text{odd}, \top\}$$



Abstract semantics

$n := n \div 2;$

$$\bot \mapsto \bot$$

$$\text{odd}, \text{even}, \top \mapsto \top$$

$n := 3 * n + 1;$

$$\bot \mapsto \bot$$

$$\text{odd} \mapsto \text{even}$$

$$\text{even} \mapsto \text{odd}$$

$$\top \mapsto \top$$

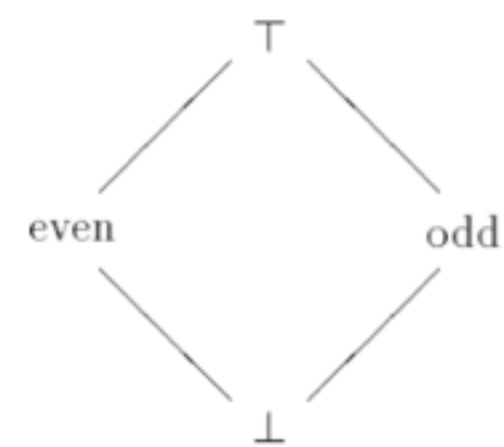
concrete semantics

abstract semantics

$$V = \text{Store} = \text{Var} \to \mathbb{Z}$$

do these two semantics agree?

$$L = \text{Var} \to \{\bot, \text{even}, \text{odd}, \top\}$$



Representation function

concrete semantics

$$V = \text{Store} = \text{Var} \to \mathbb{Z}$$

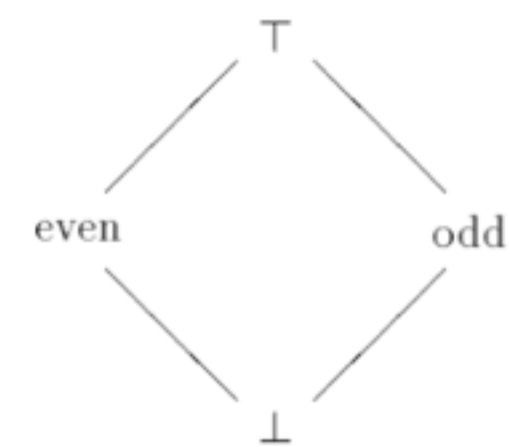
$\beta : V \to L$

monotonic

$$\beta(v) = \begin{cases} \text{even} & \text{if v even} \\ \text{odd} & \text{if v odd} \end{cases}$$

abstract semantics

$$L = \text{Var} \to \{\bot, \text{even}, \text{odd}, \top\}$$



Representation function

concrete semantics

the best approximation

$$V = \text{Store} = \text{Var} \to \mathbb{Z}$$

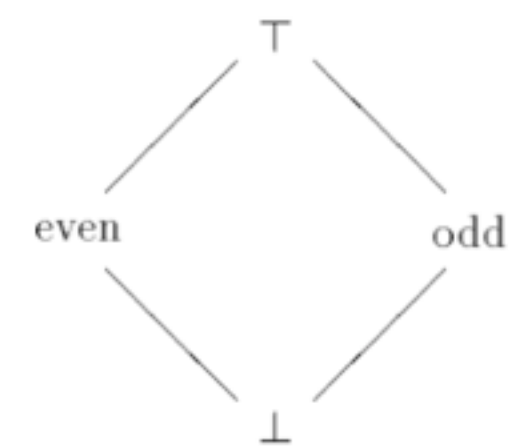
$\beta : V \to L$

monotonic

$$\beta(v) = \begin{cases} \text{even} & \text{if v even} \\ \text{odd} & \text{if v odd} \end{cases}$$

abstract semantics

$$L = \text{Var} \to \{\bot, \text{even}, \text{odd}, \top\}$$



$$\begin{tikzcd}
7 \arrow[r, "n := 3 * n + 1"] \arrow[d, "\beta"] & 22 \arrow[d, "\beta"] \\
\text{odd} \arrow[r, "n := 3 * n + 1"] & \text{even}
\end{tikzcd}$$

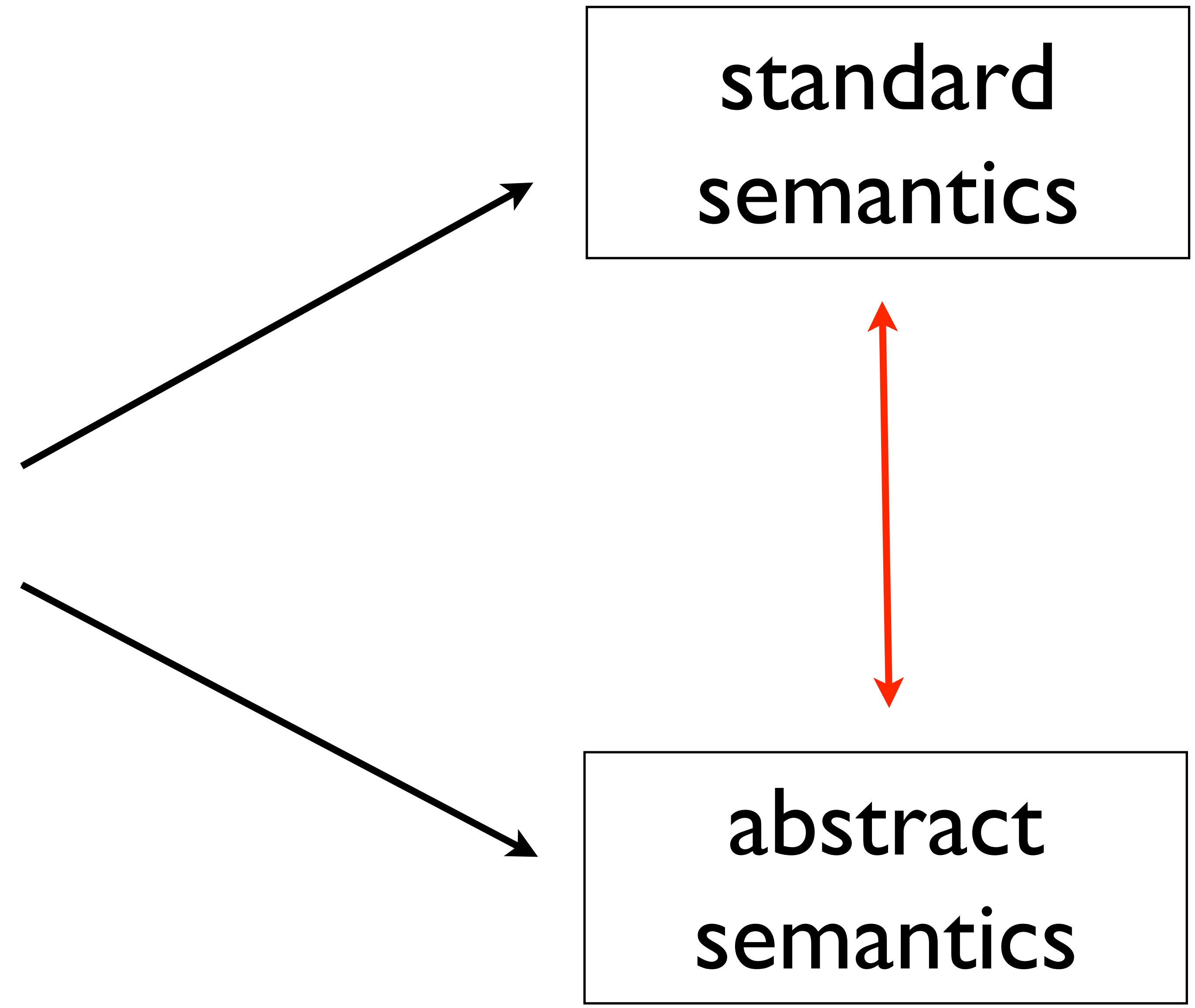
$$\begin{tikzcd} 7 \arrow[r, "n := 3 * n + 1"] \arrow[d, "\beta"] & 22 \arrow[d, "\beta"] \\ \text{odd} \arrow[r, "n := 3 * n + 1"] & \text{even} \end{tikzcd}$$

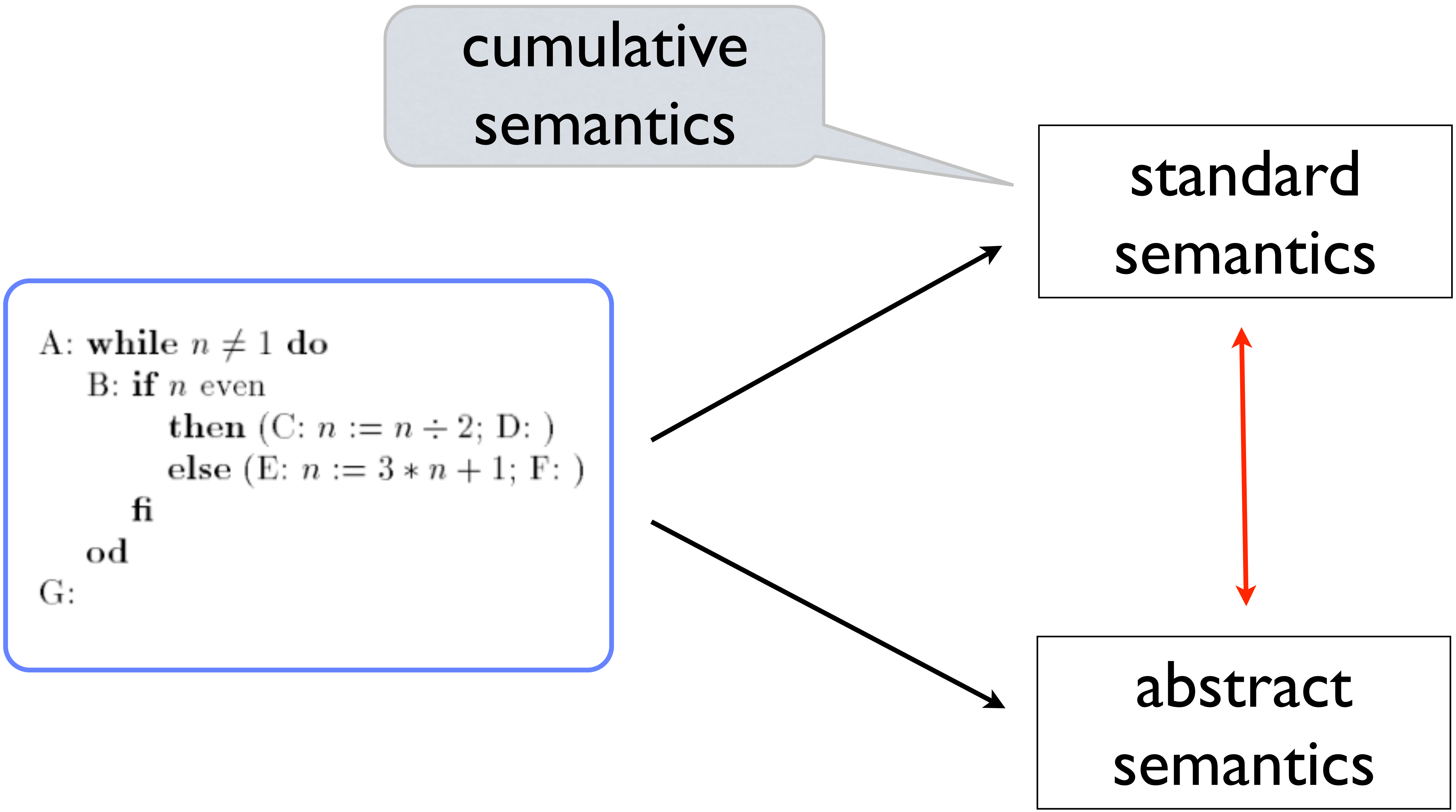
β is not always a homomorphism!

β is not always a homomorphism!

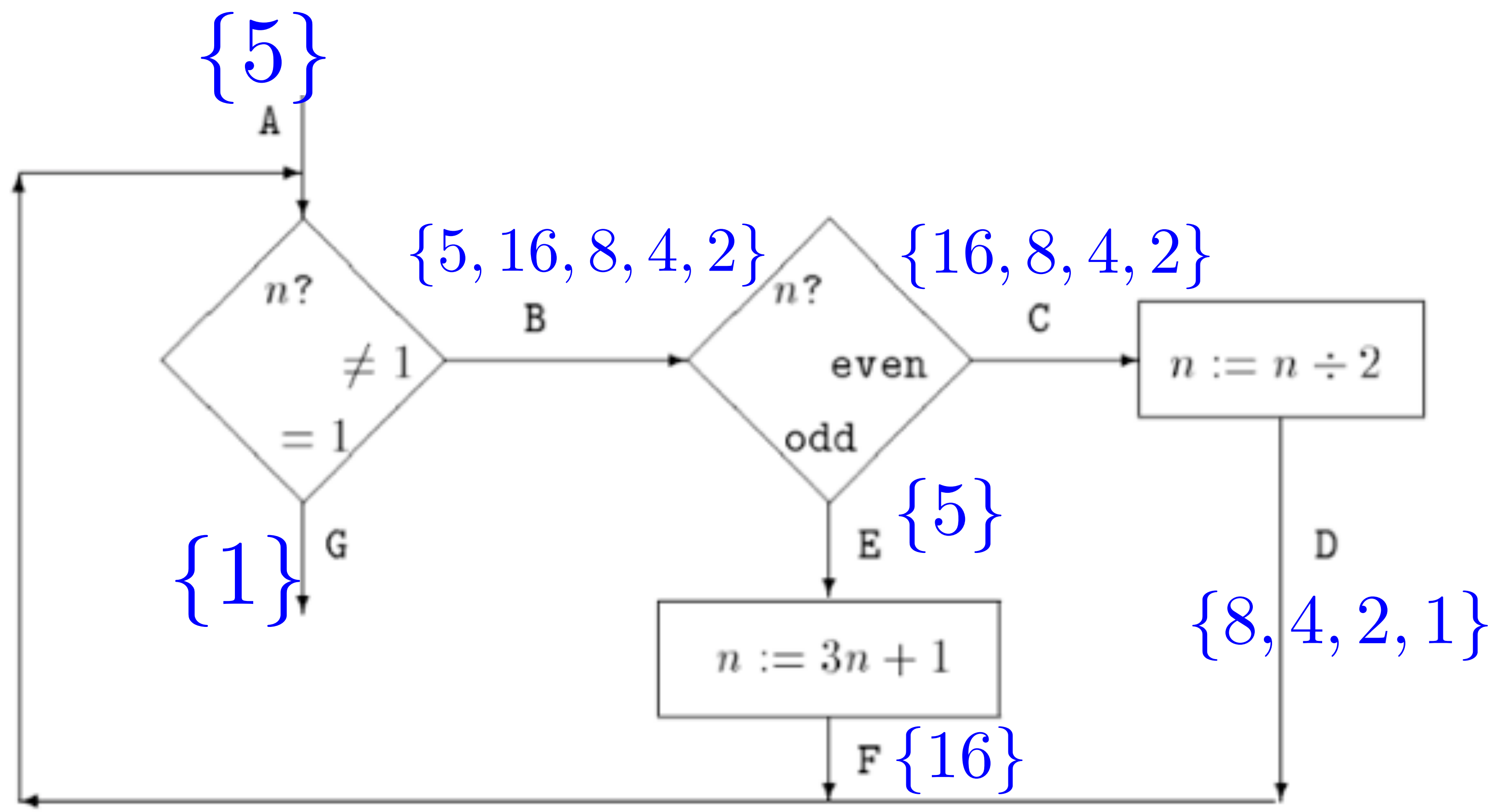
$$\begin{tikzcd}
{[n \mapsto 1, m \mapsto -2]} \arrow[r, "m := n + m"] \arrow[dd, "\beta"] & {[n \mapsto 1, m \mapsto -1]} \arrow[d, "\beta"] \\
 & {[n \mapsto +, m \mapsto -]} \arrow[d, phantom, sloped, "\sqsubseteq"] \\
{[n \mapsto +, m \mapsto -]} \arrow[r, "m := n + m"] & {[n \mapsto +, m \mapsto \top]}
\end{tikzcd}$$

```
A: while n ≠ 1 do
     B: if n even
          then (C: n := n ÷ 2; D: )
          else (E: n := 3 * n + 1; F: )
        fi
   od
G:
```




cumulative semantics
standard semantics
abstract semantics
A: while $n \neq 1$ do
B: if n even
then (C: $n := n \div 2$; D:)
else (E: $n := 3 * n + 1$; F:)
fi
od
G:

A: **while** $n \neq 1$ **do**
  B: **if** n even
    then (C: $n := n \div 2$; D:)
    else (E: $n := 3 * n + 1$; F:)
  fi
od
G:



Abstraction function

concrete semantics

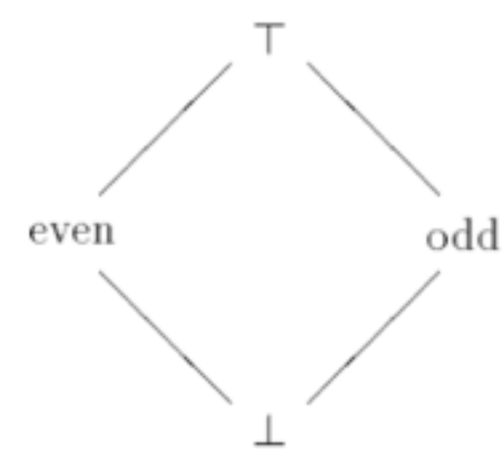
$$\mathcal{P}(V)$$

abstraction

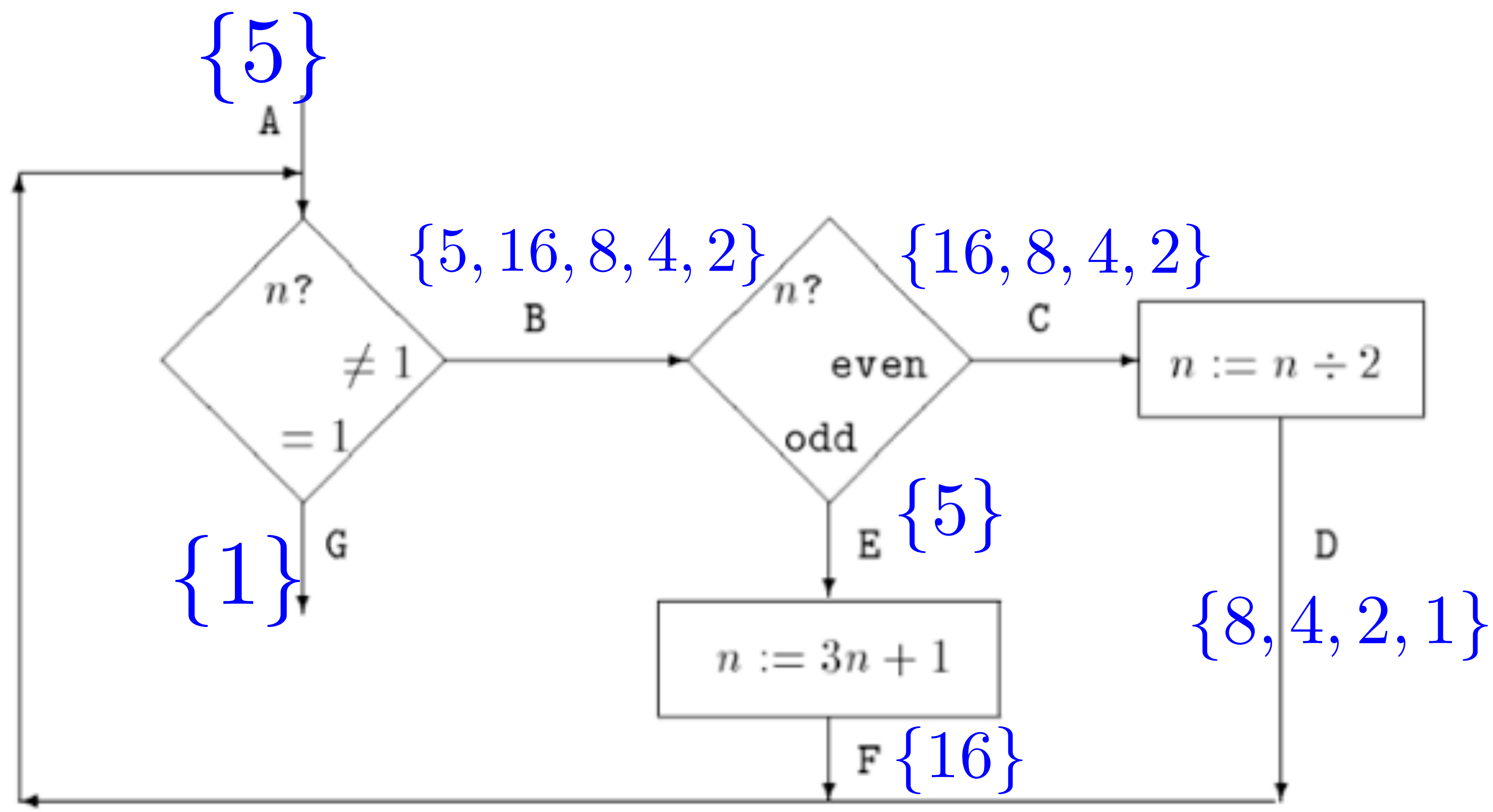
$$\alpha : \mathcal{P}(V) \to L \qquad \alpha(X) = \sqcup\{\beta(v) \mid v \in X\}$$

abstract semantics

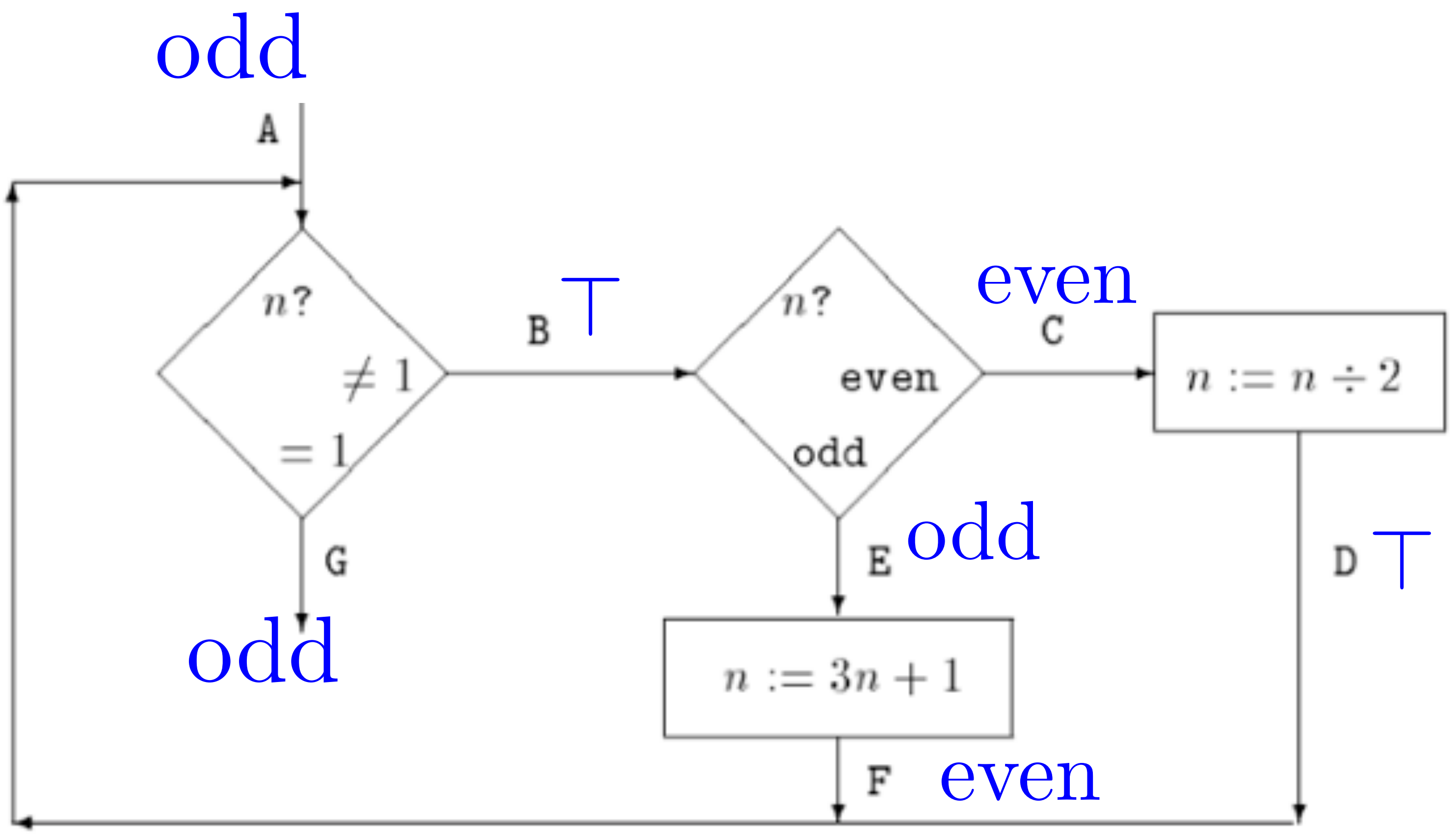
$$L \;=\; \mathrm{Var} \to \{\bot, \mathrm{even}, \mathrm{odd}, \top\}$$

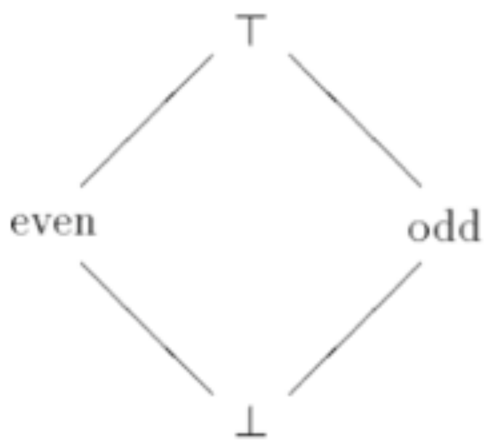


A: **while** $n \neq 1$ **do**
  B: **if** n even
    then (C: $n := n \div 2$; D:)
    else (E: $n := 3 * n + 1$; F:)
   fi
  od
G:



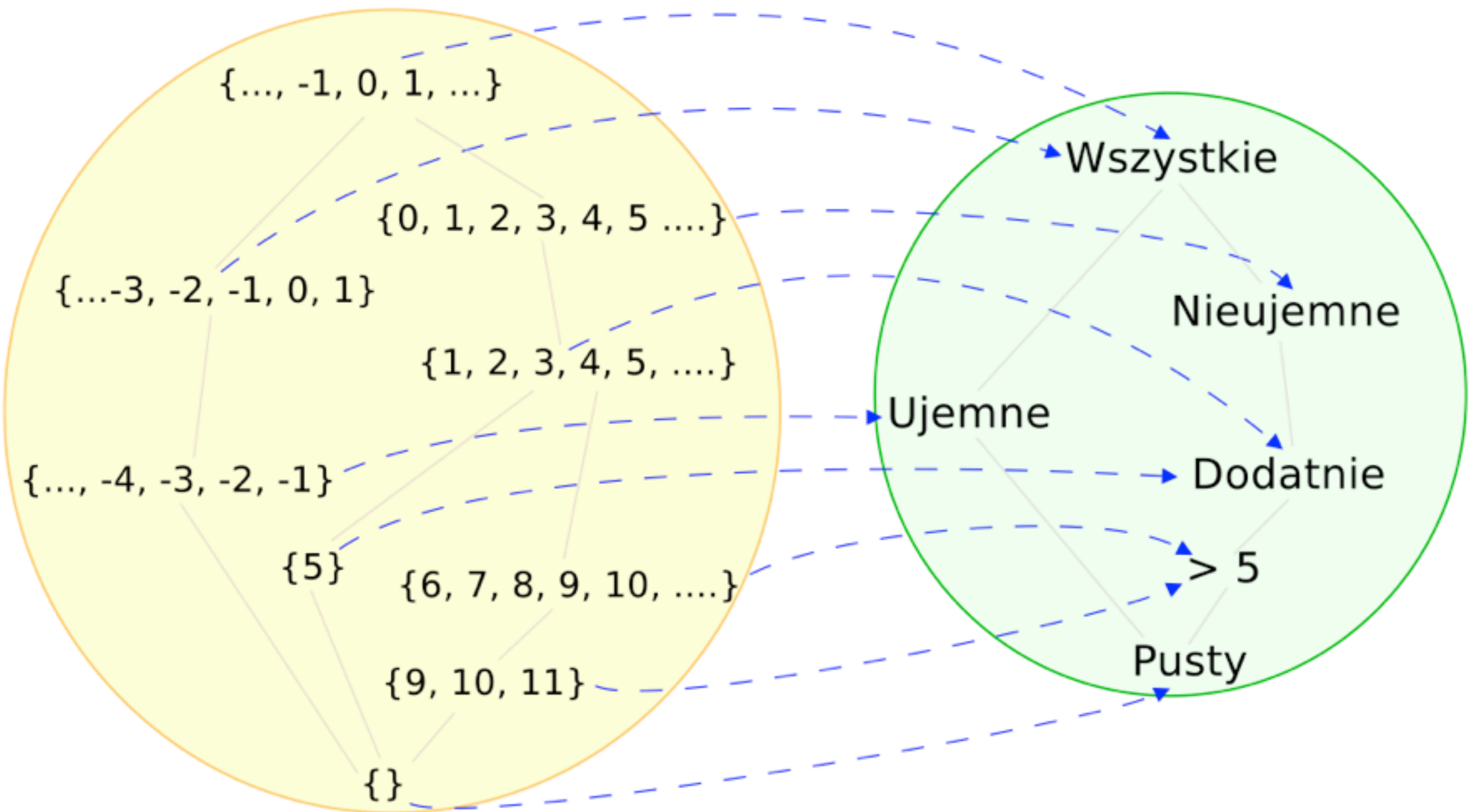
A: **while** $n \neq 1$ **do**
    B: **if** n even
        then (C: $n := n \div 2$; D:)
        else (E: $n := 3 * n + 1$; F:)
    fi
od
G:





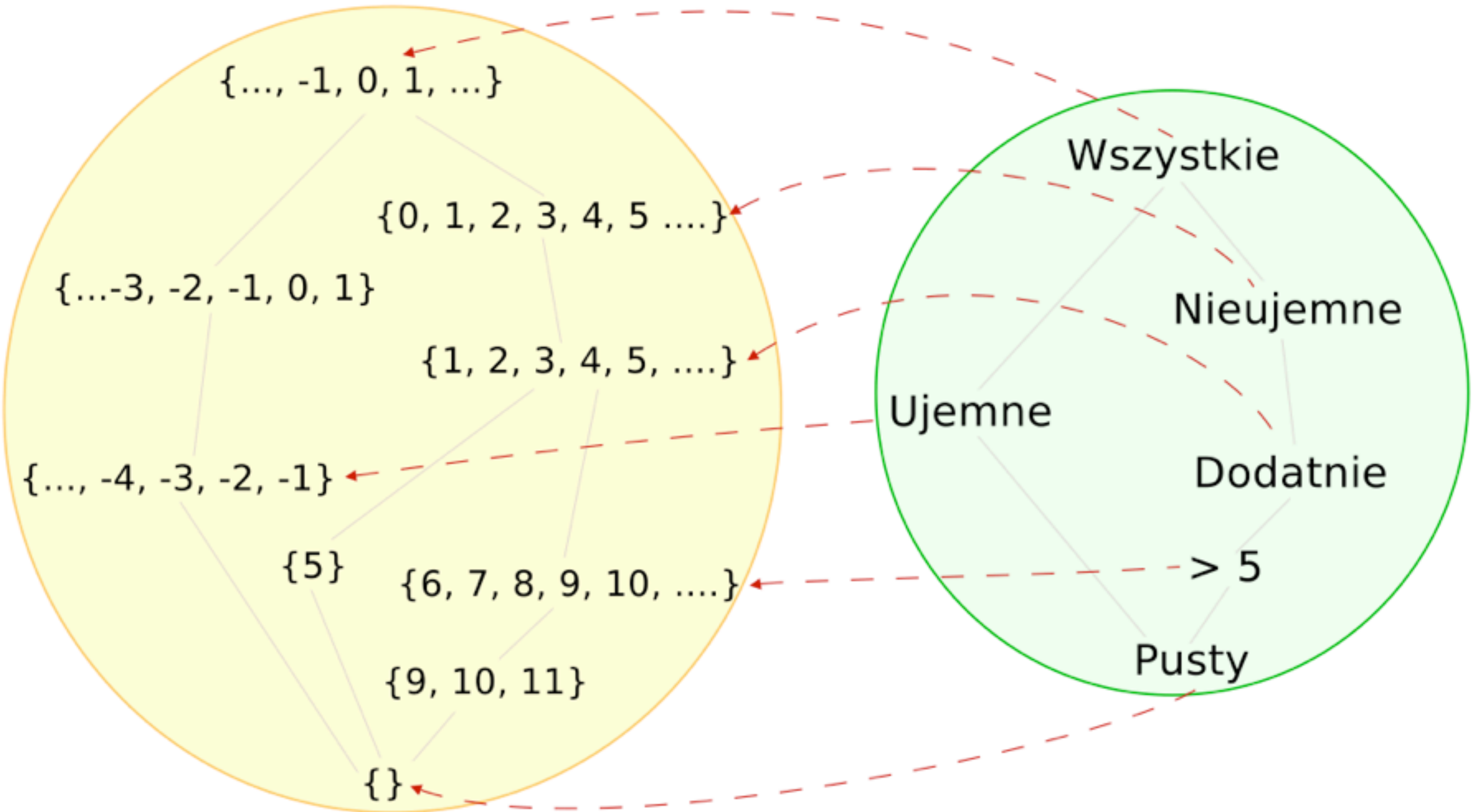
Abstraction function (example)

α maps a set of concrete values to the most exact abstract value

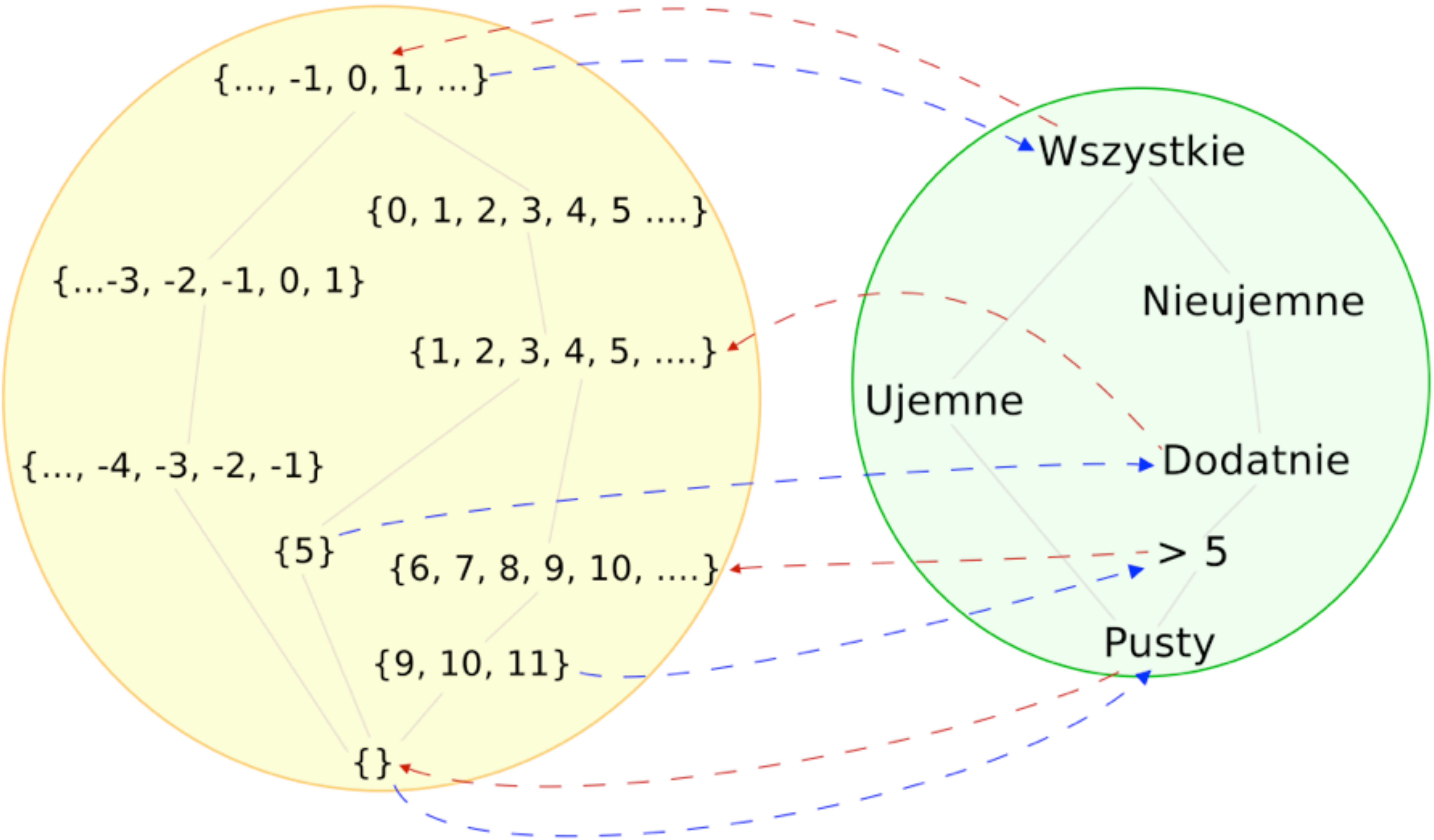


Concretization function (example)

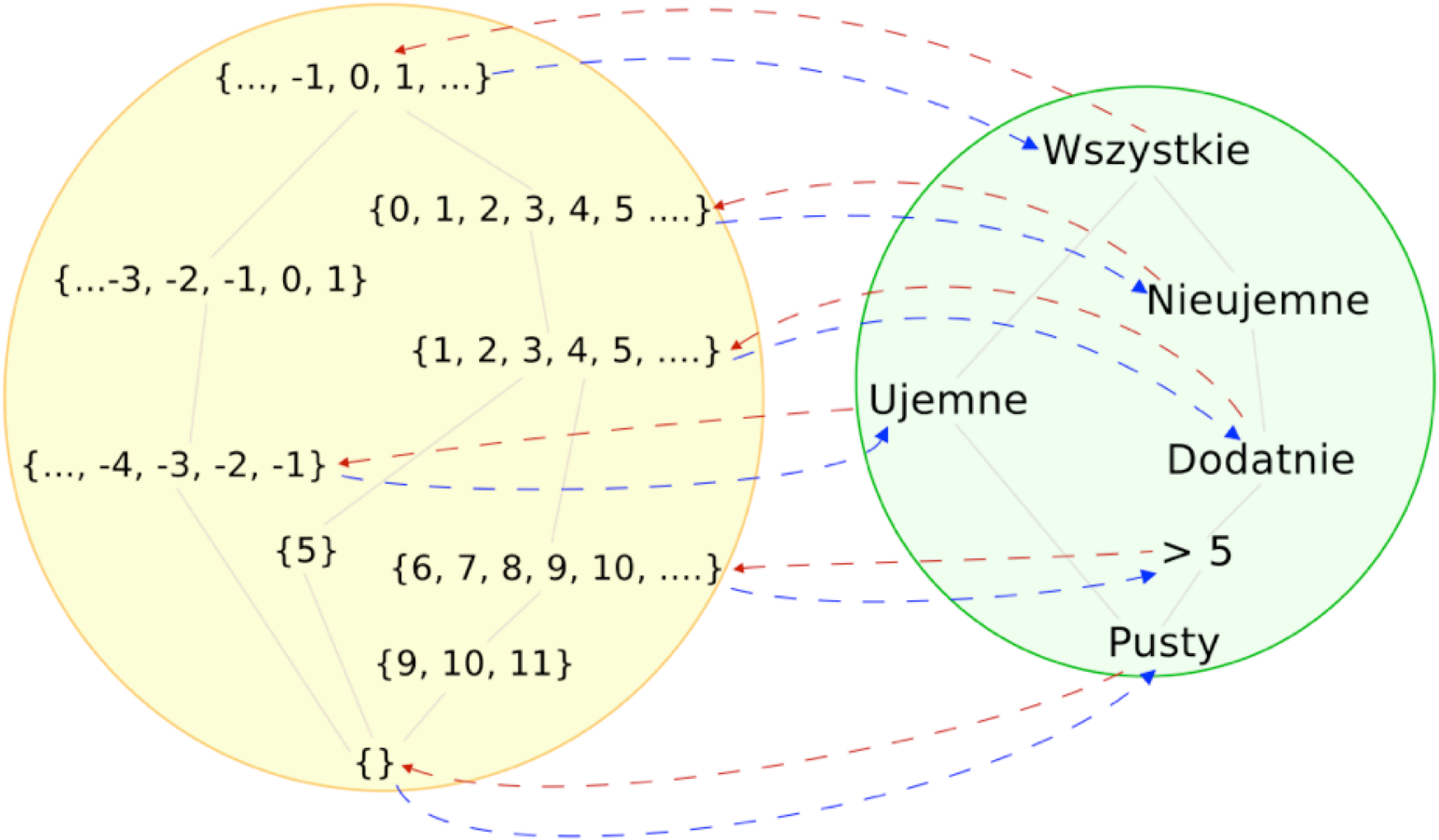
γ maps an abstract value to the set of represented concrete values



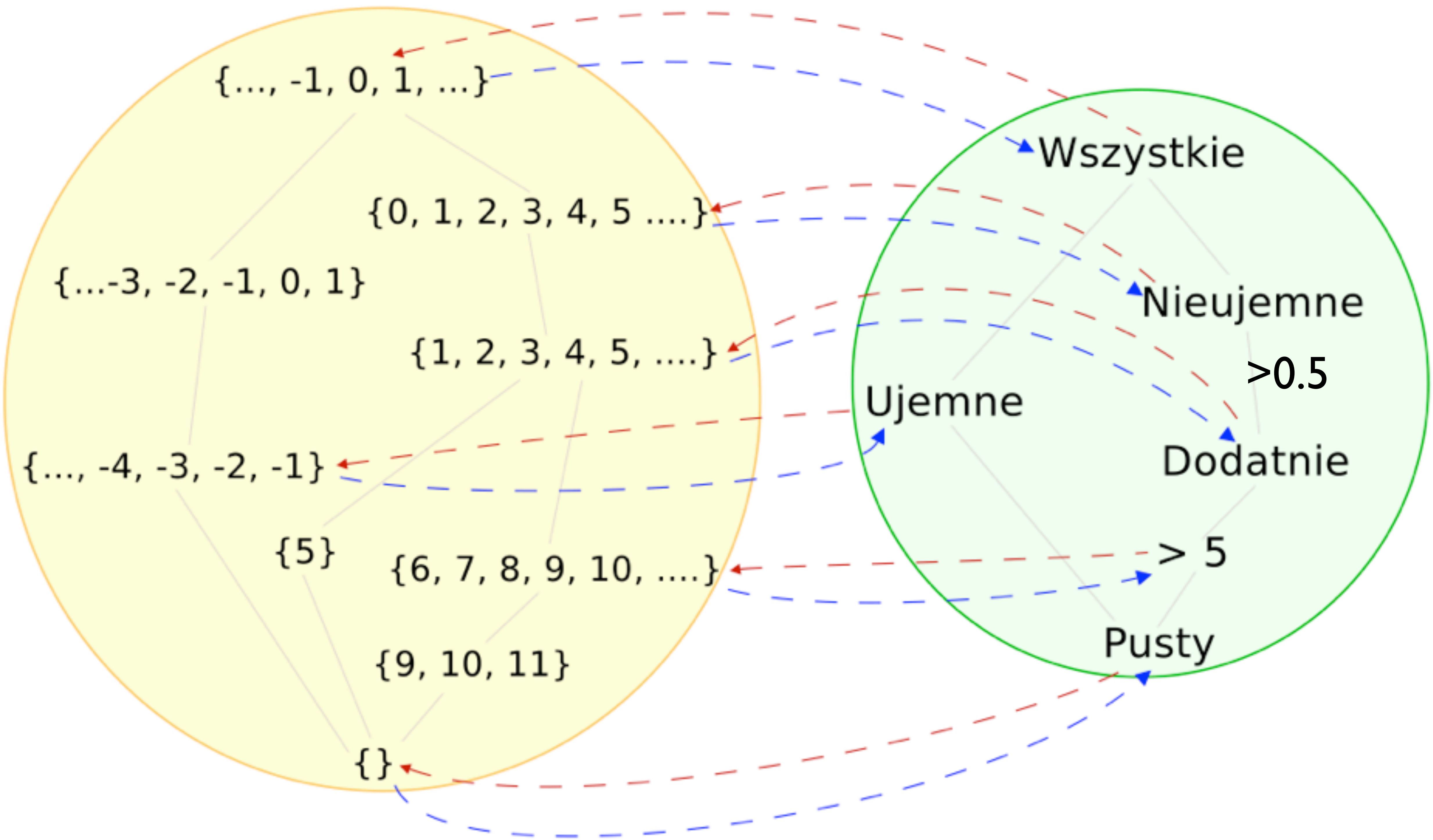
$$x \subseteq \gamma \cdot \alpha(x)$$



$$\alpha \cdot \gamma(a) \ \leq \ a$$



$$\alpha \cdot \gamma(a) \leq a$$



Galois connection

Concrete and abstract domain

set of represented values

$$\mathcal{P}(V) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} L$$

the most exact abstraction

Example

$$\mathcal{P}(\mathbb{Z}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \text{Intervals}$$

$\alpha(X) =$ the smallest interval containing *X*

$$\gamma(I) = I$$

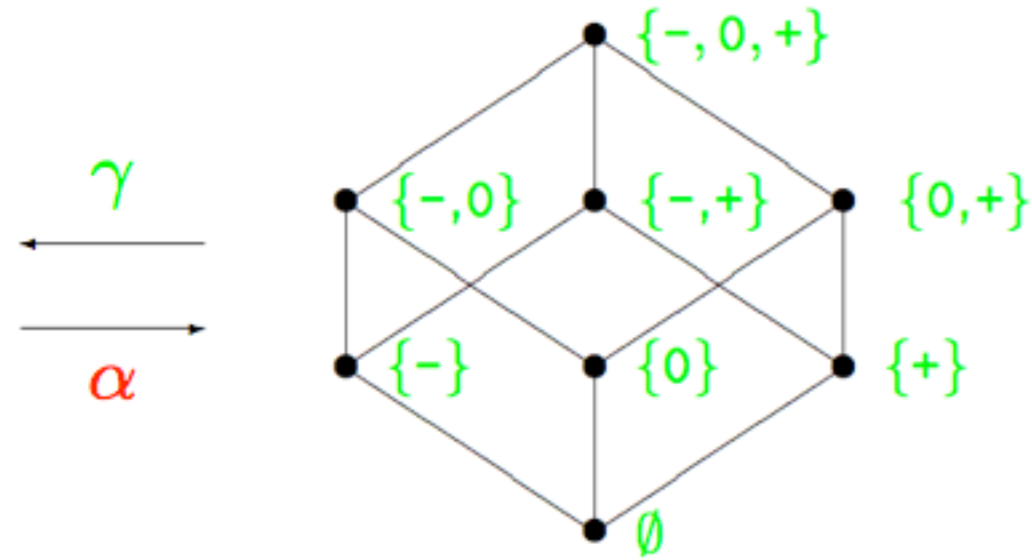
Two abstract domains

$$\begin{tikzcd} L \arrow[r, shift right, "\alpha"'] & M \arrow[l, shift right, "\gamma"'] \end{tikzcd}$$

M is more abstract (less exact) than *L*

Example

Intervals



Example

Intervals