# Program analysis: from proving correctness to proving incorrectness

Roberto Bruni, Roberta Gori
(University of Pisa)

Lectures #04-05

BISS 2024
March 11-15, 2024

# Abstract Interpretation

# Abstract Interpretation

It is a technique to formally reason on approximations

It allows to derive effective methods to compute approximations

Generally used to compute overapproximations

Seldom used to compute underapproximations

# Example: out of bounds

```
function arrayOutOfBounds(int n, int x[10]) {

  a = 0

  if n >= 10 then

     n = n - 5

  else

     a = ++n


  a = max(0,a — n)

  return x[a] }
```

Let us assume $n \geq 0$

Is it a safe access? ($0 \leq a \leq 9$ ?)

# Using exact semantics

```
function arrayOutOfBounds(int n, int x[10]) {
(0,_)(1,_)(2,_)(3,_)(4,_)(5,_)(6,_)(7,_)(8,_)(9,_)(10,_)….
  a = 0
(0,0)(1,0)(2,0)(3,0)(4,0)(5,0)(6,0)(7,0)(8,0)(9,0)(10,0)….
  if n >= 10 then
(10,0)(11,0)(12,0)(13,0)(14,0)(15,0)(16,0)(17,0)(18,0)(19,0)….
     n = n - 5
(5,0)(6,0)(7,0)(8,0)(9,0)(10,0)(11,0)(12,0)(13,0)(14,0)….
  else

     a = ++n


  a = max(0,a – n)

  return x[a] }
```

We can't  track the infinite set of pairs!

**use intervals !**

# Example: interval abstraction

```
 function arrayOutOfBounds(int n, int x[10]) {
[0,∞]
    a = 0
[0,∞][0,0]
    if n >= 10 then
     [10,∞][0,0]
       n = n - 5
     [5,∞][0,0]
    else
     [0,9][0,0]
       a = ++n
     [1,10][1,10]
[1,∞][0,10]
    a = max(0,a – n)
[1,∞][0,9]
    return x[a] }
```

Merging branches looses precision

safe! $0 \leq a \leq 9$ !

# Abstract Interpretation: the idea

Goal: Compute the set S of possible values at each line of code

But… this is not feasible in general

We want to find an (over)approximation $S \subseteq S^{\#}$

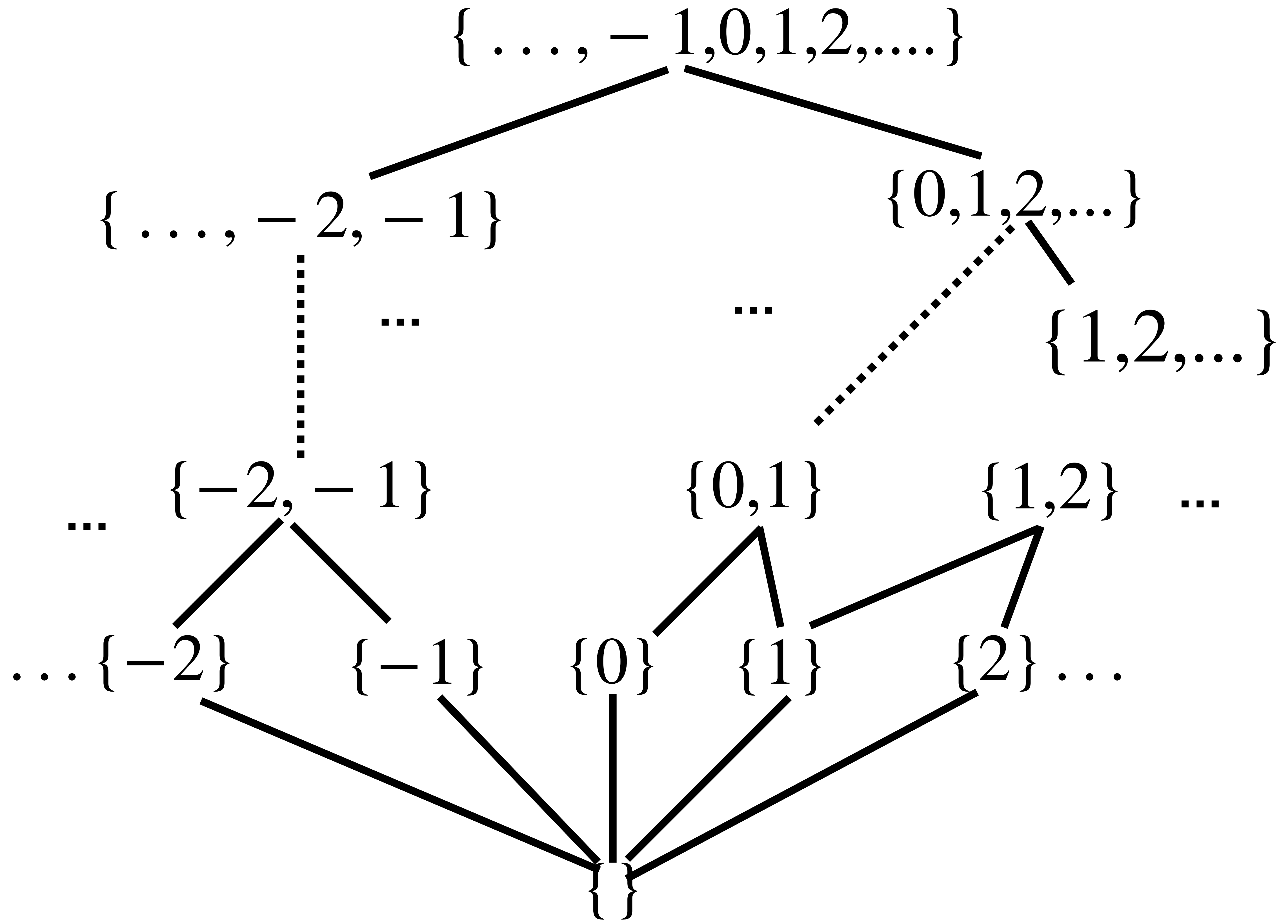
The theory of abstract interpretation allows to compute $S^{\#}$ as a set of abstract values obtained by applying abstract operations

# Abstraction and concretization

# Concrete domain

The set of values S that we would like to compute belongs to the concrete domain $C$

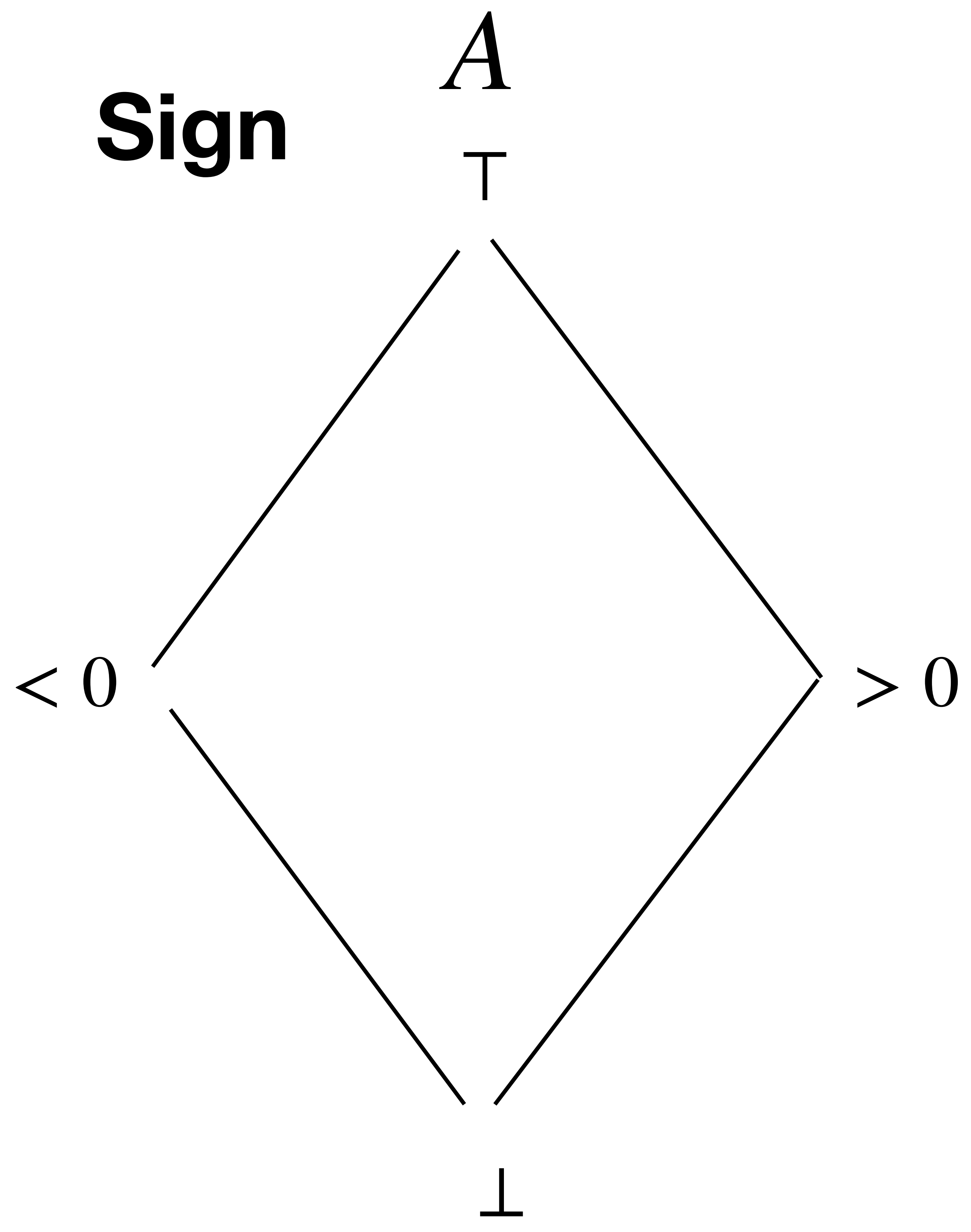
$$(\wp(\mathbb{Z}), \subseteq)$$



# Abstract Domain

$(A, \sqsubseteq)$ expresses some properties of the concrete values

For example




The order $\sqsubseteq$ on the abstract domain reflects the precision

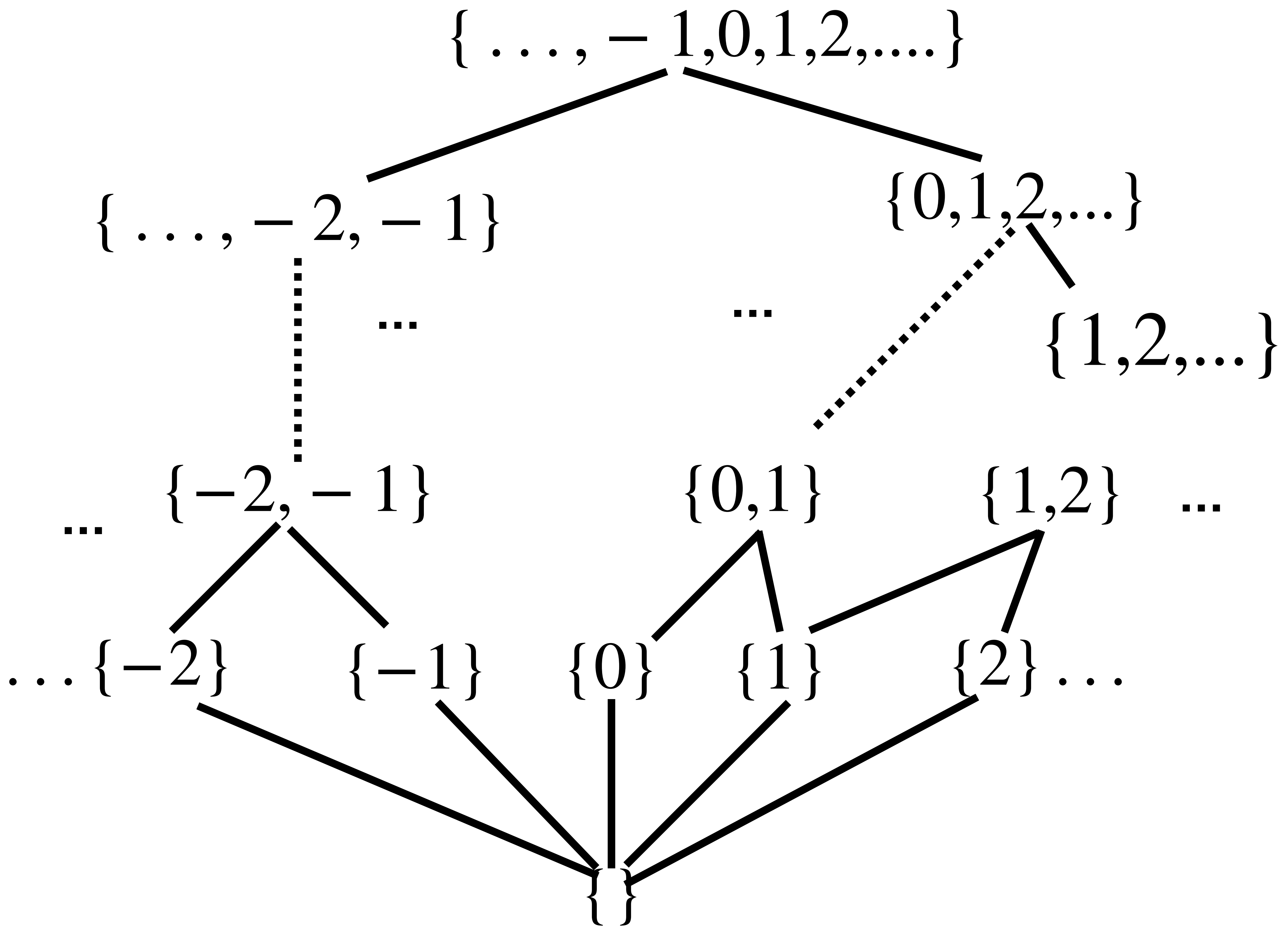
e.g $\bot \sqsubseteq <$

# Ingredients of Abstract Interpretation

- A concrete domain $C$
- An abstract domain $A$
- An abstraction function $\alpha$ that connects the concrete domain to the abstract one
- A concretisation function $\gamma$ that relates the abstract domain to the concrete one

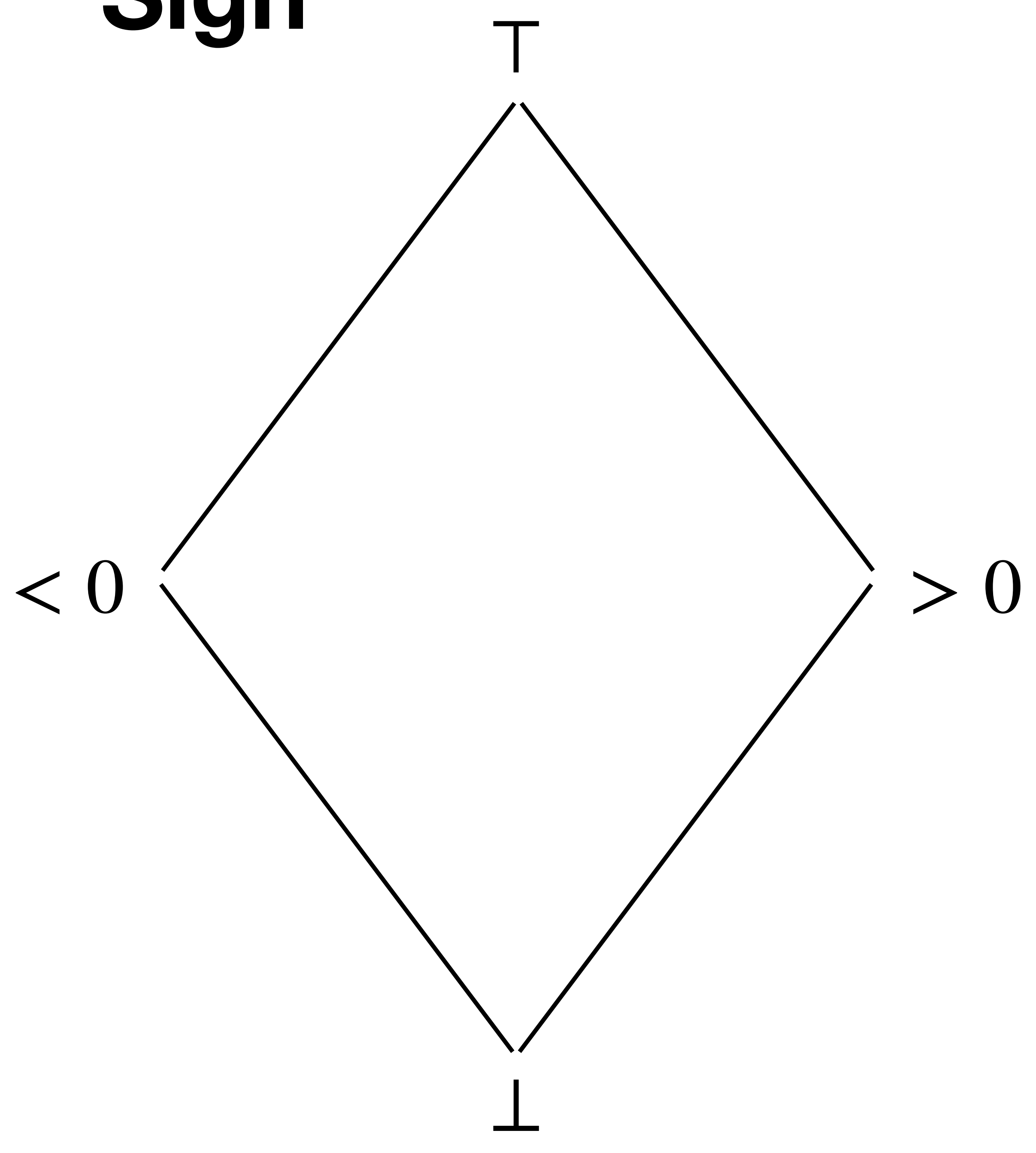
# Defining concretisation

$C$

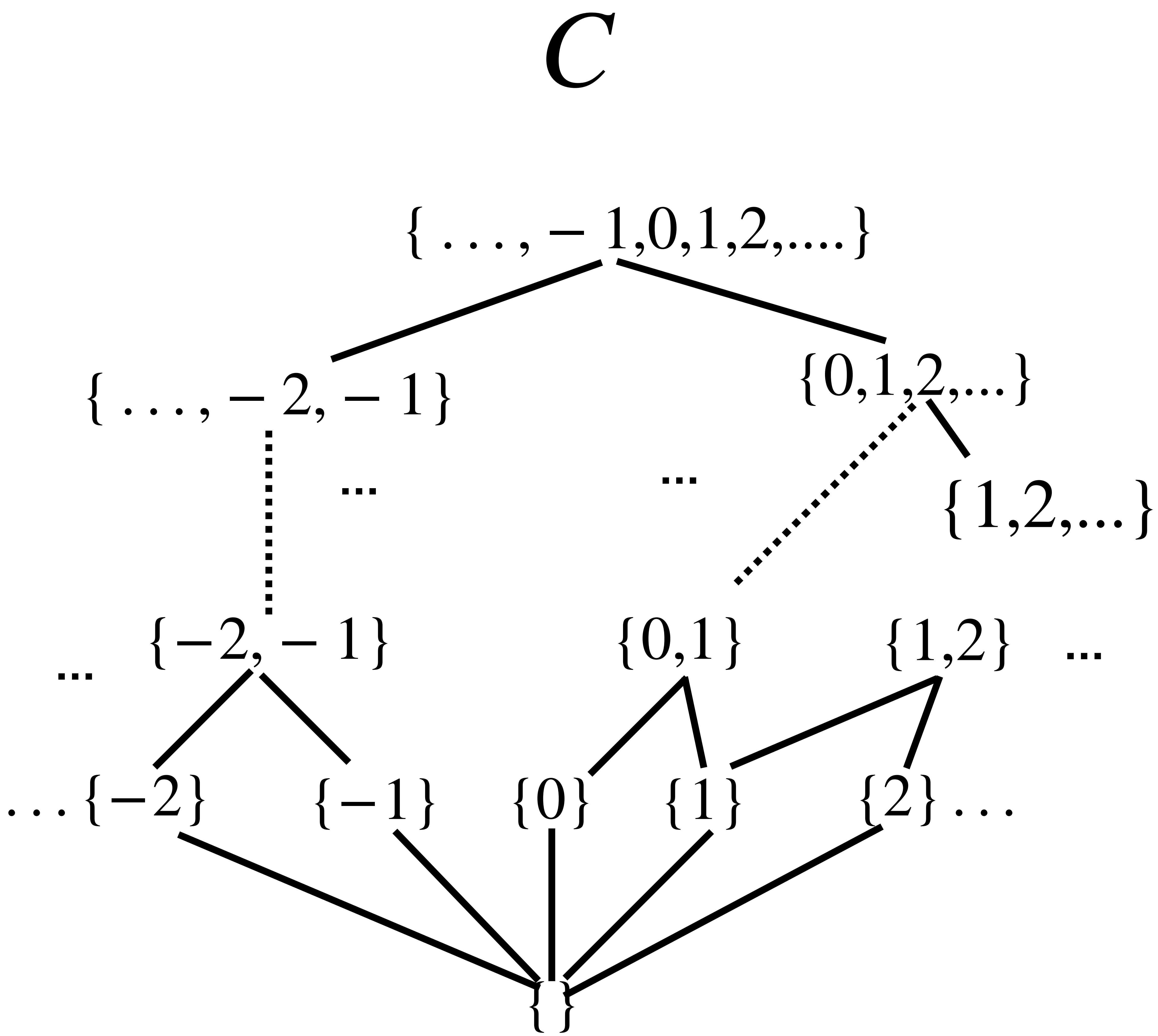


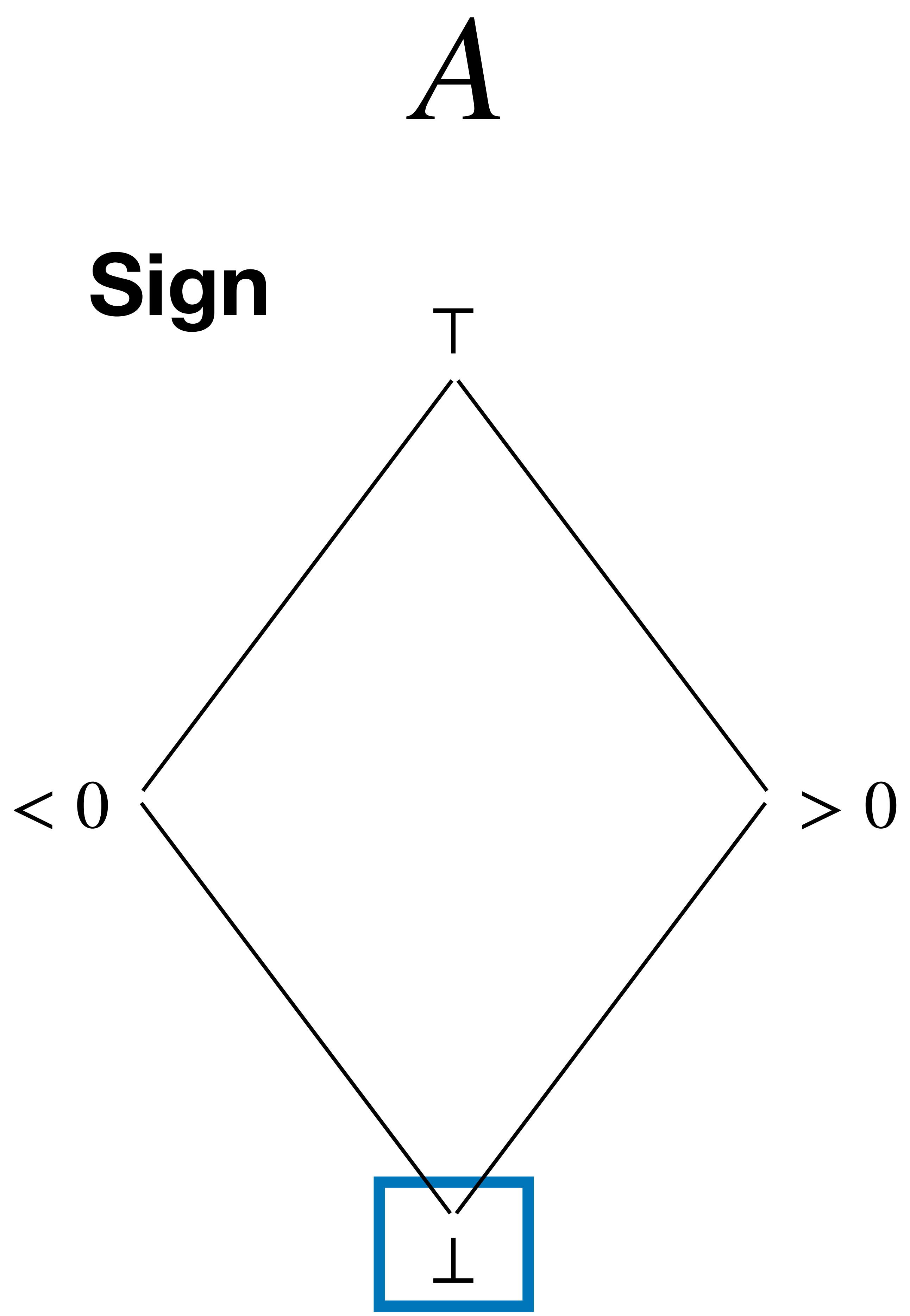

$A$

**Sign**



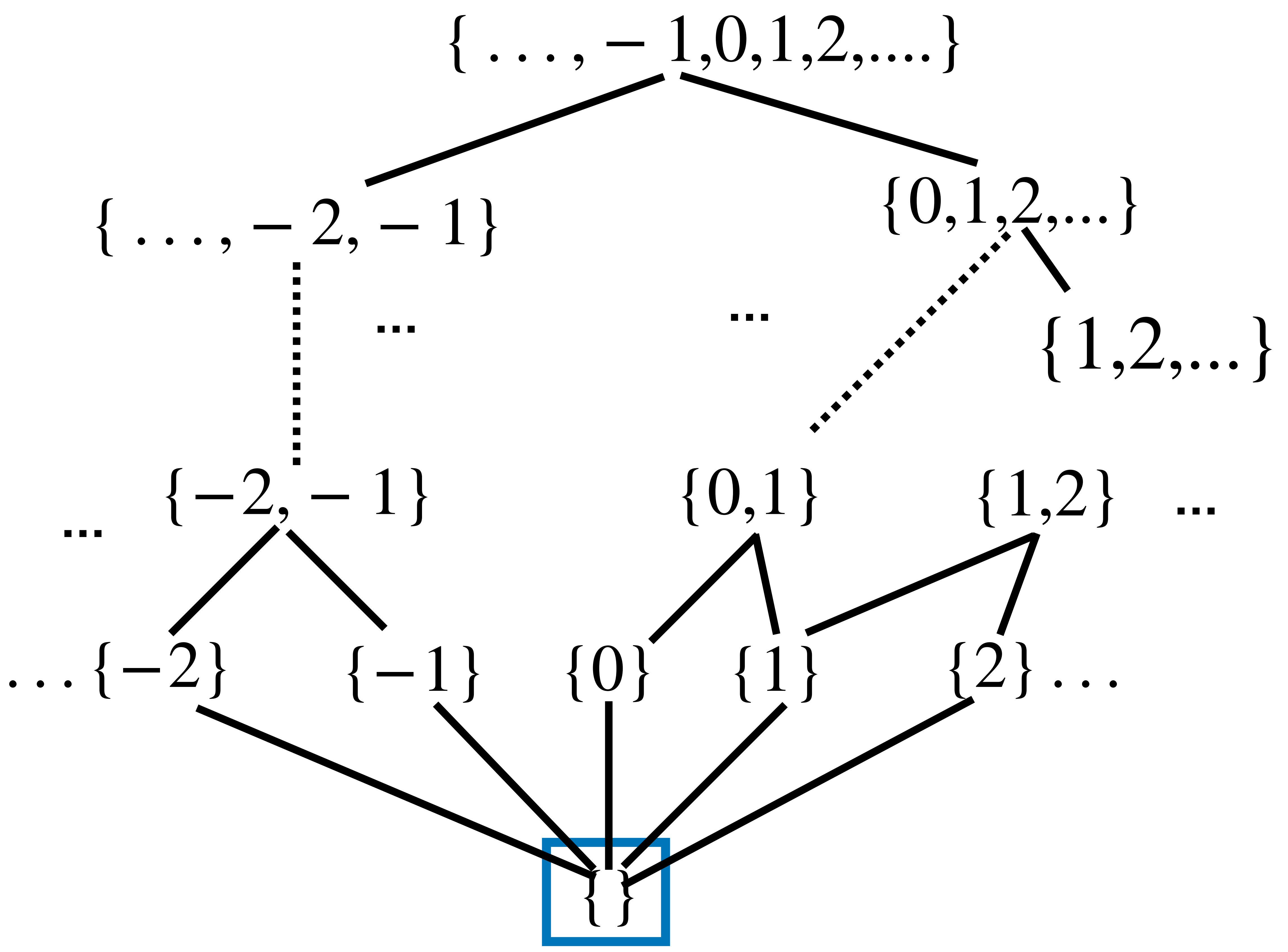
# Defining concretisation



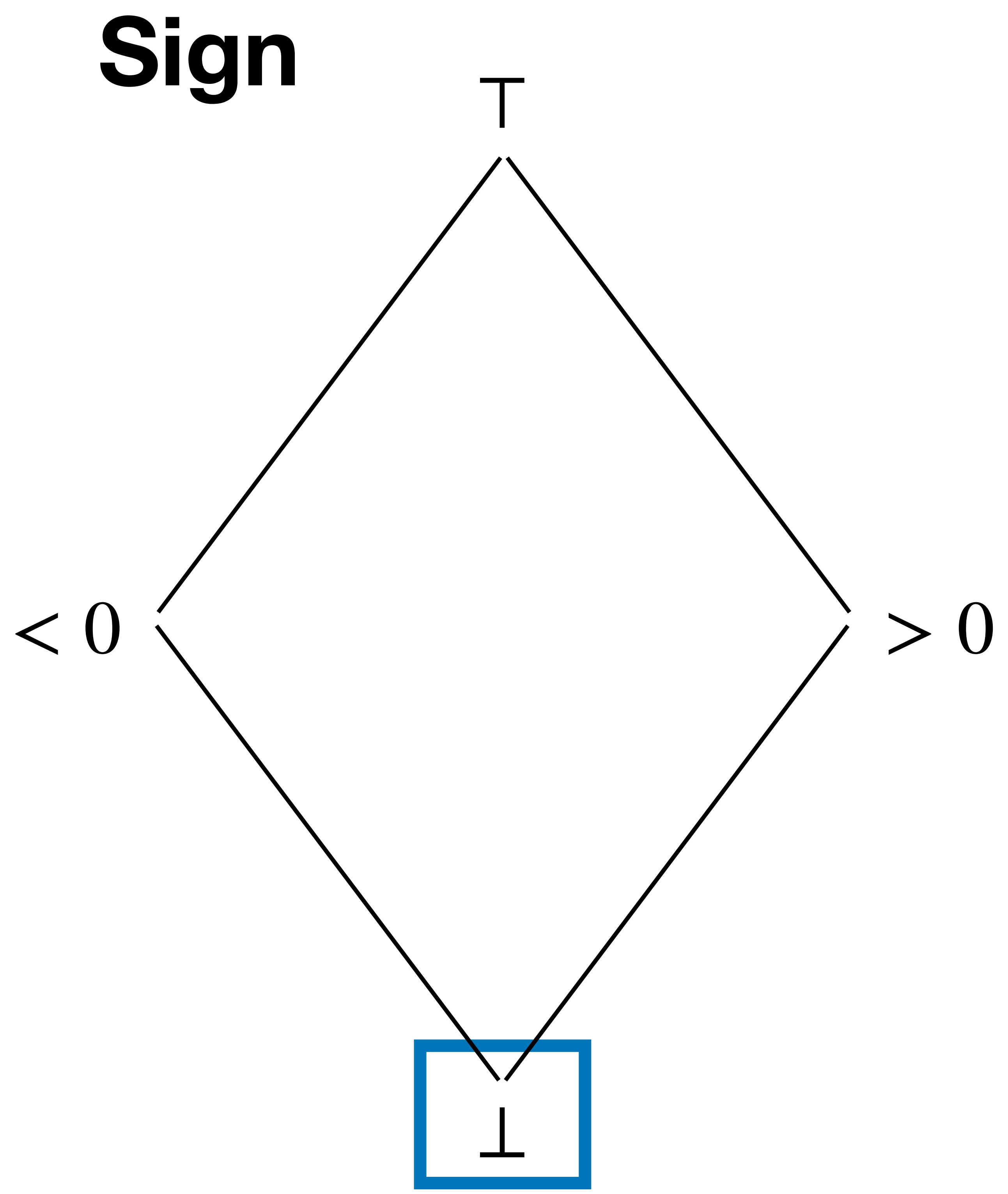



# Defining concretisation

$C$

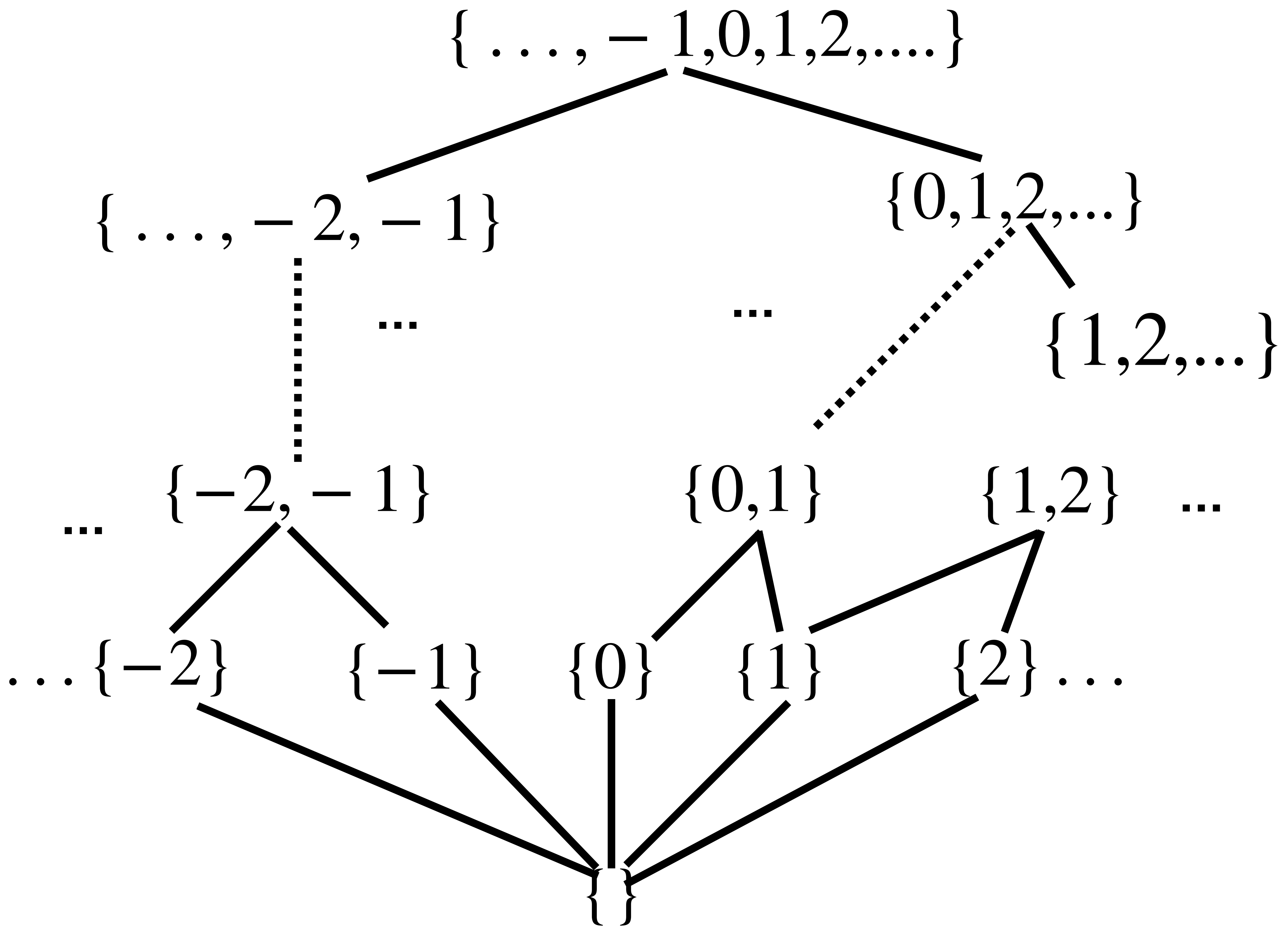



$A$



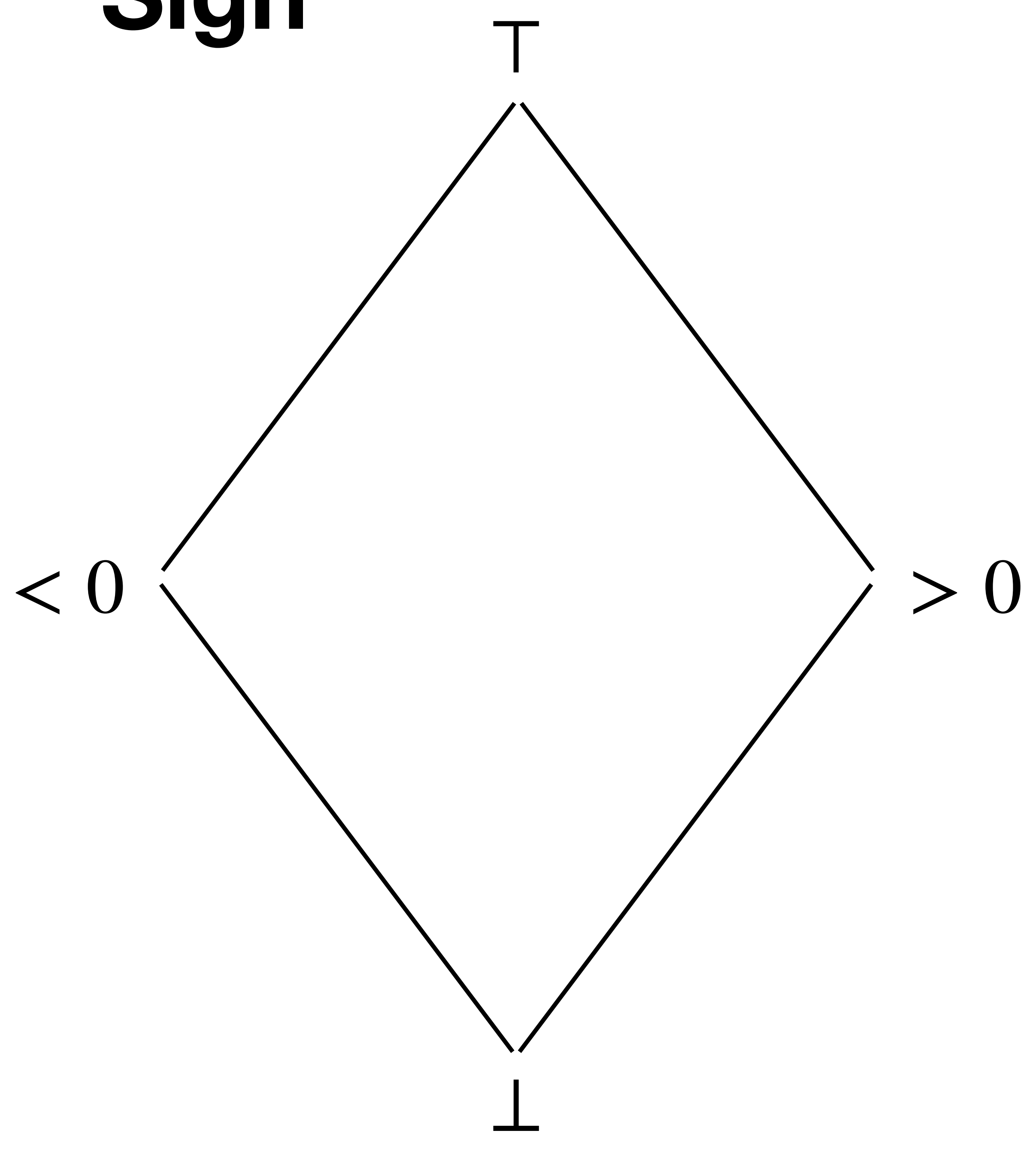
# Defining approximation

$C$



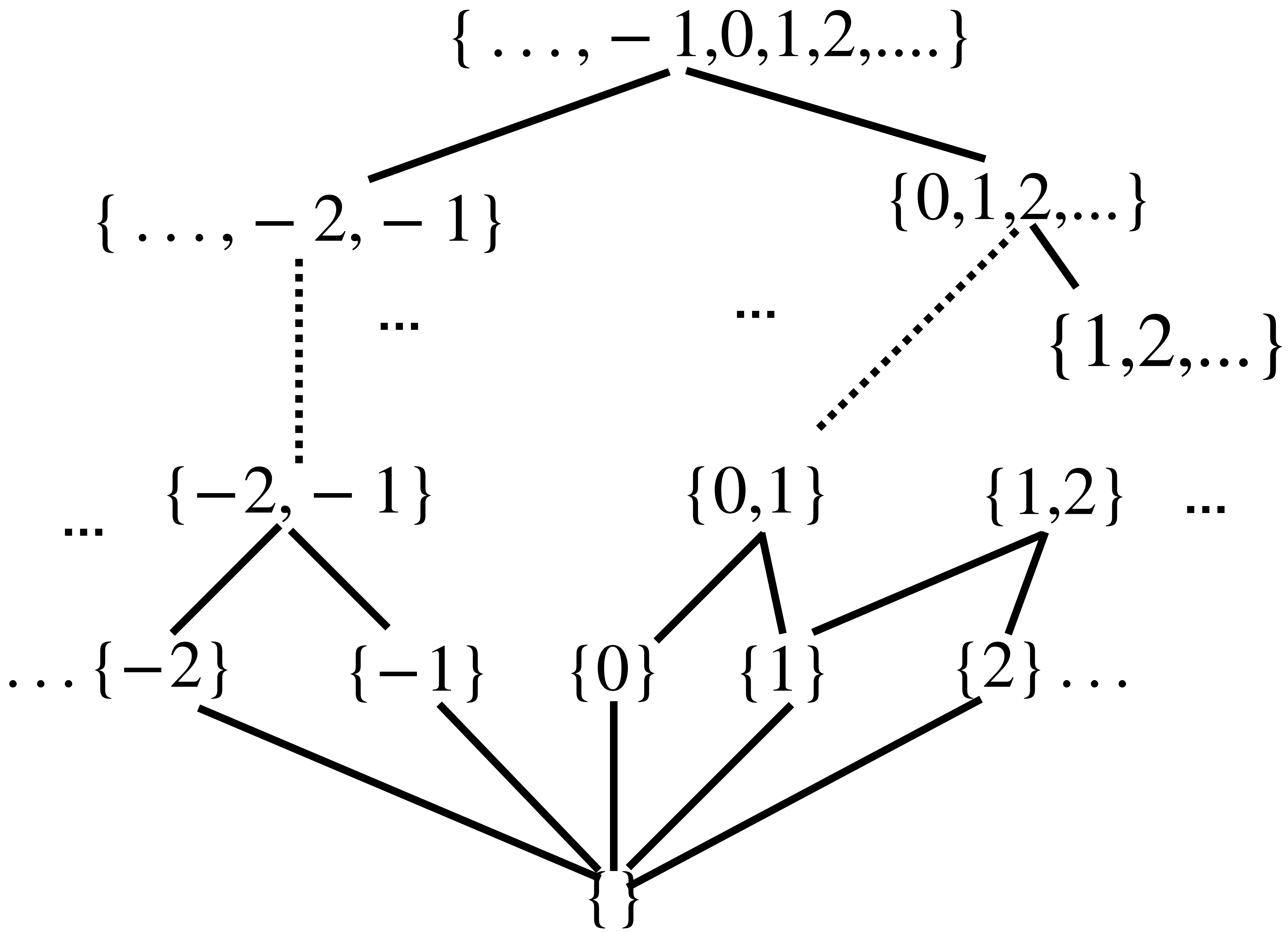

$A$

**Sign**

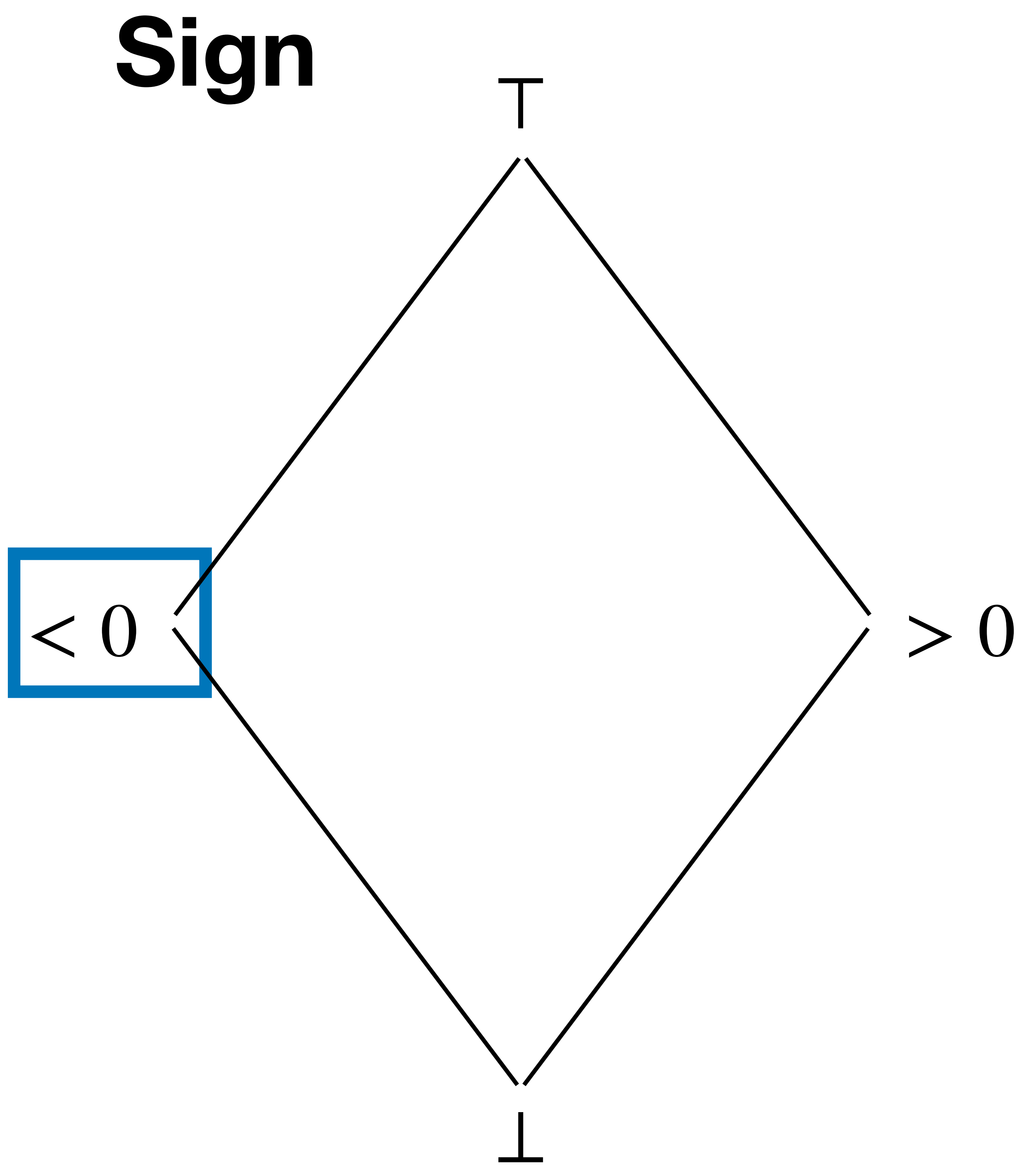


# Defining approximation

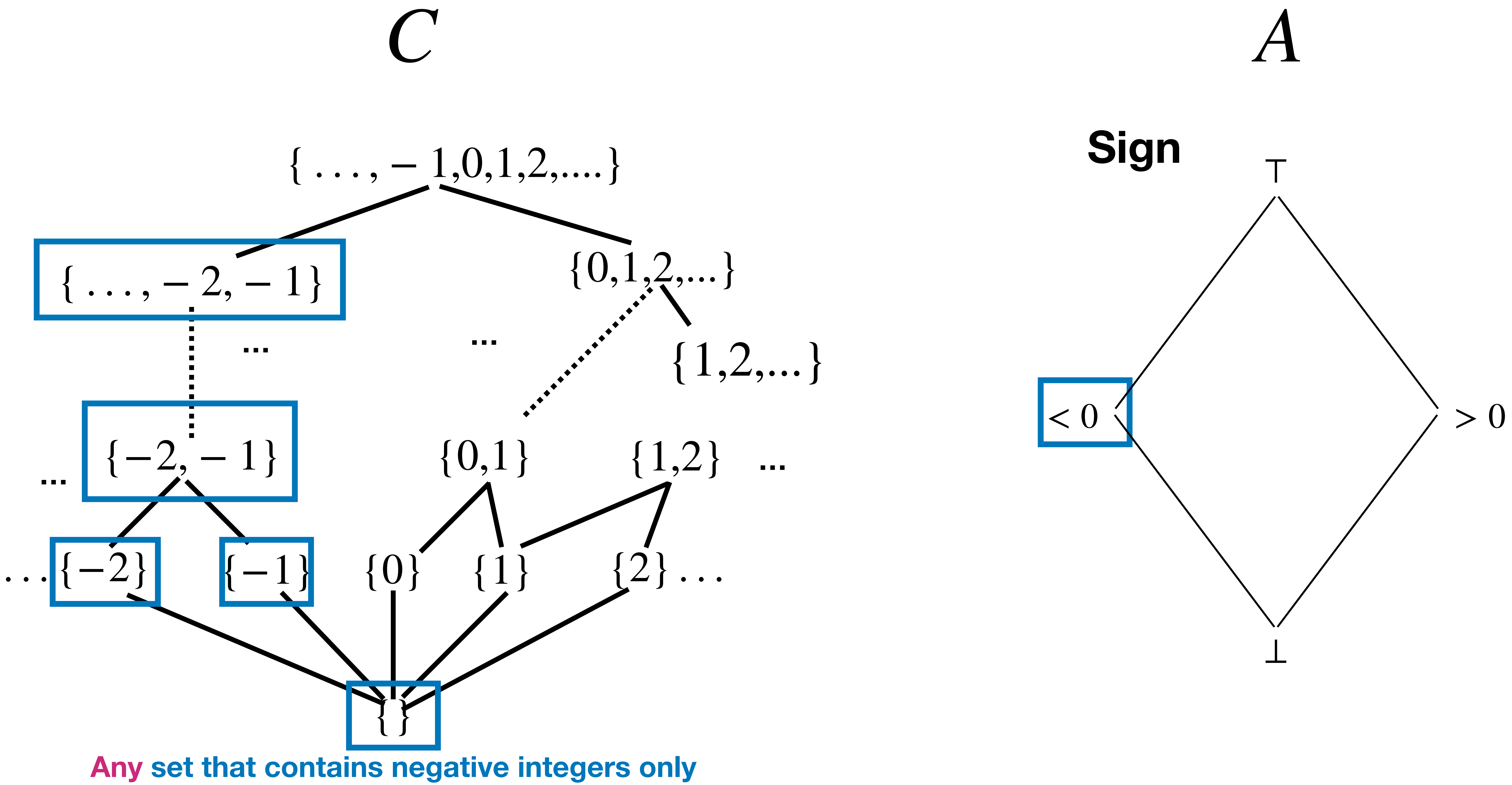
$C$



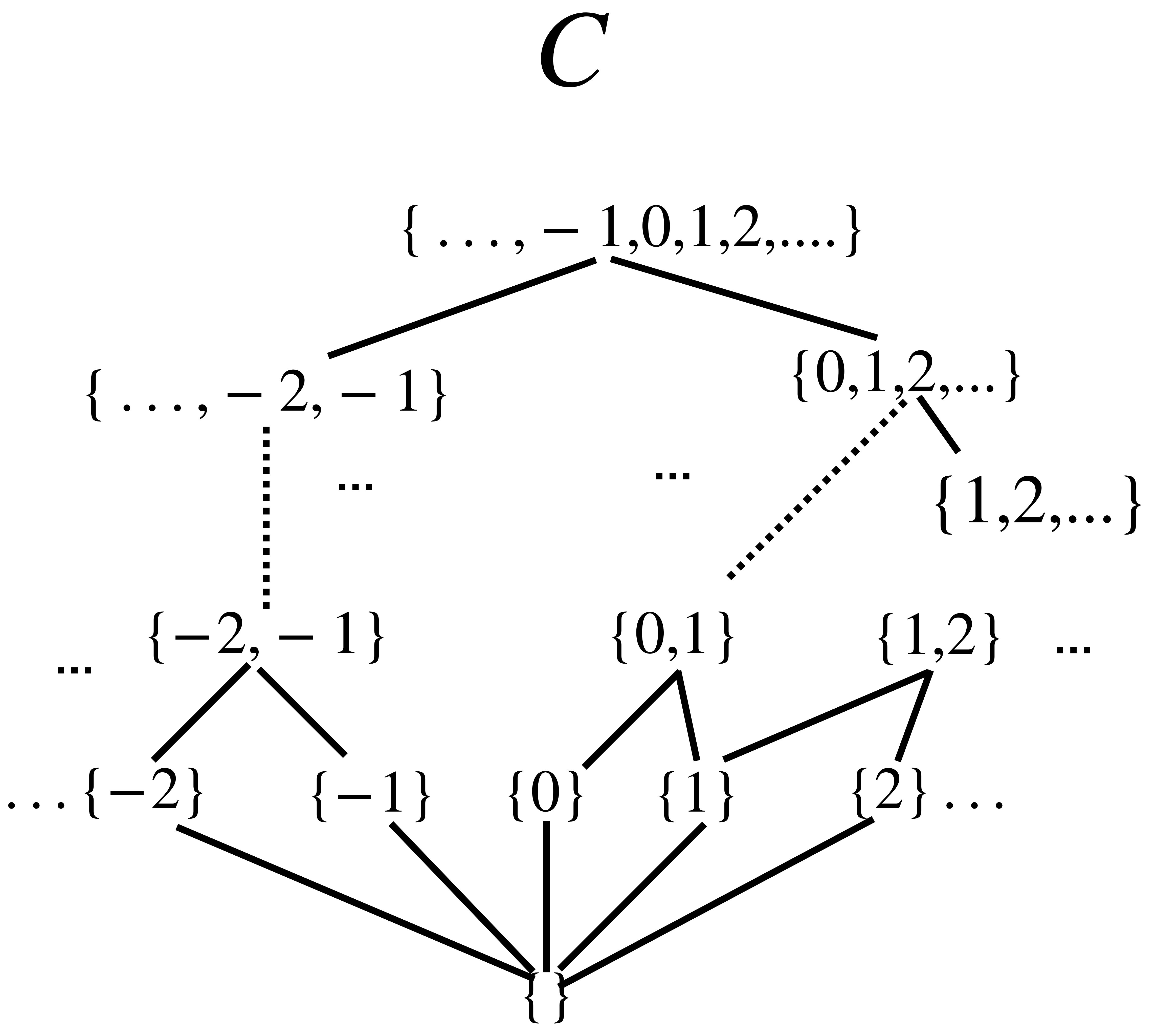

$A$

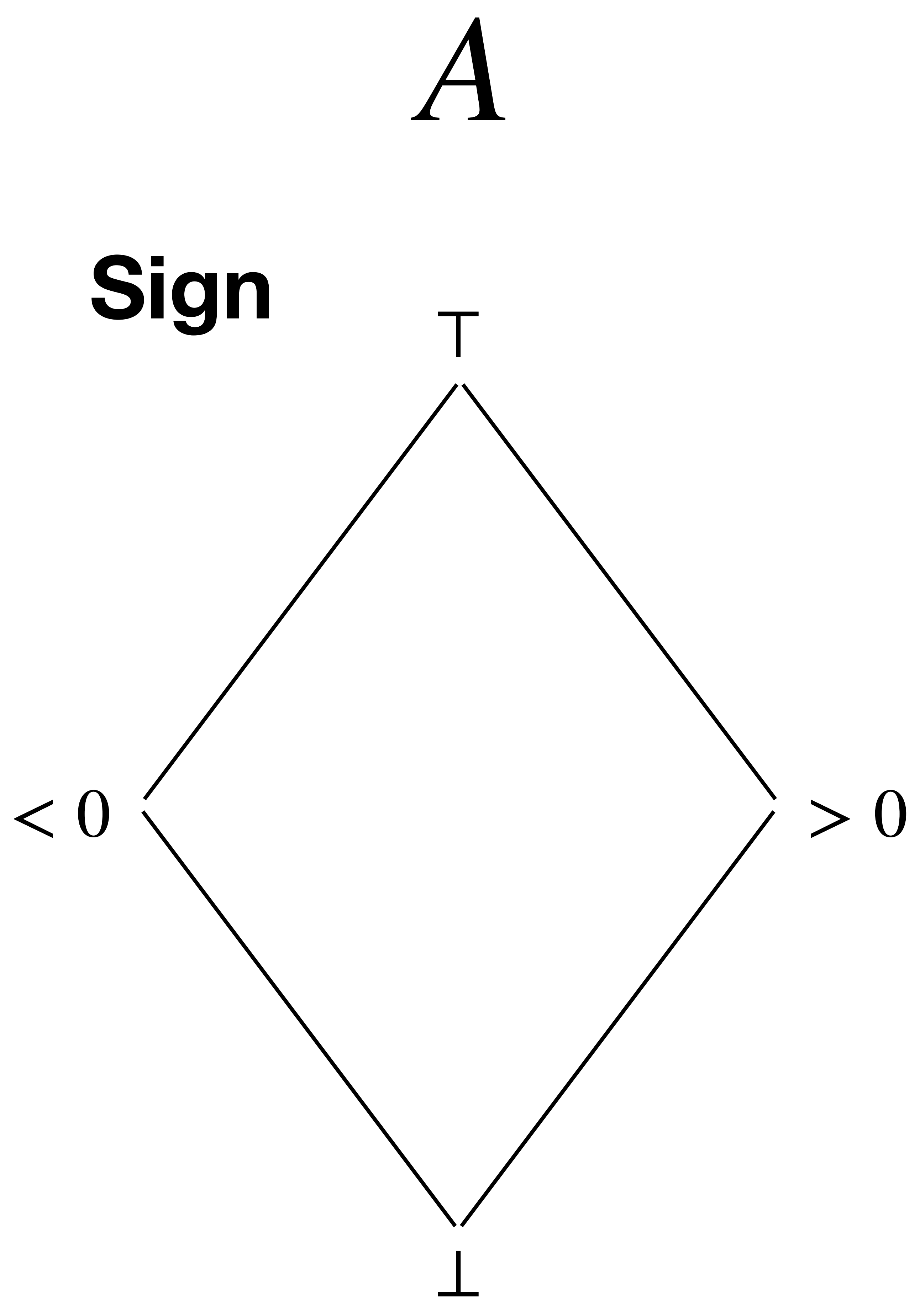


# Defining approximation

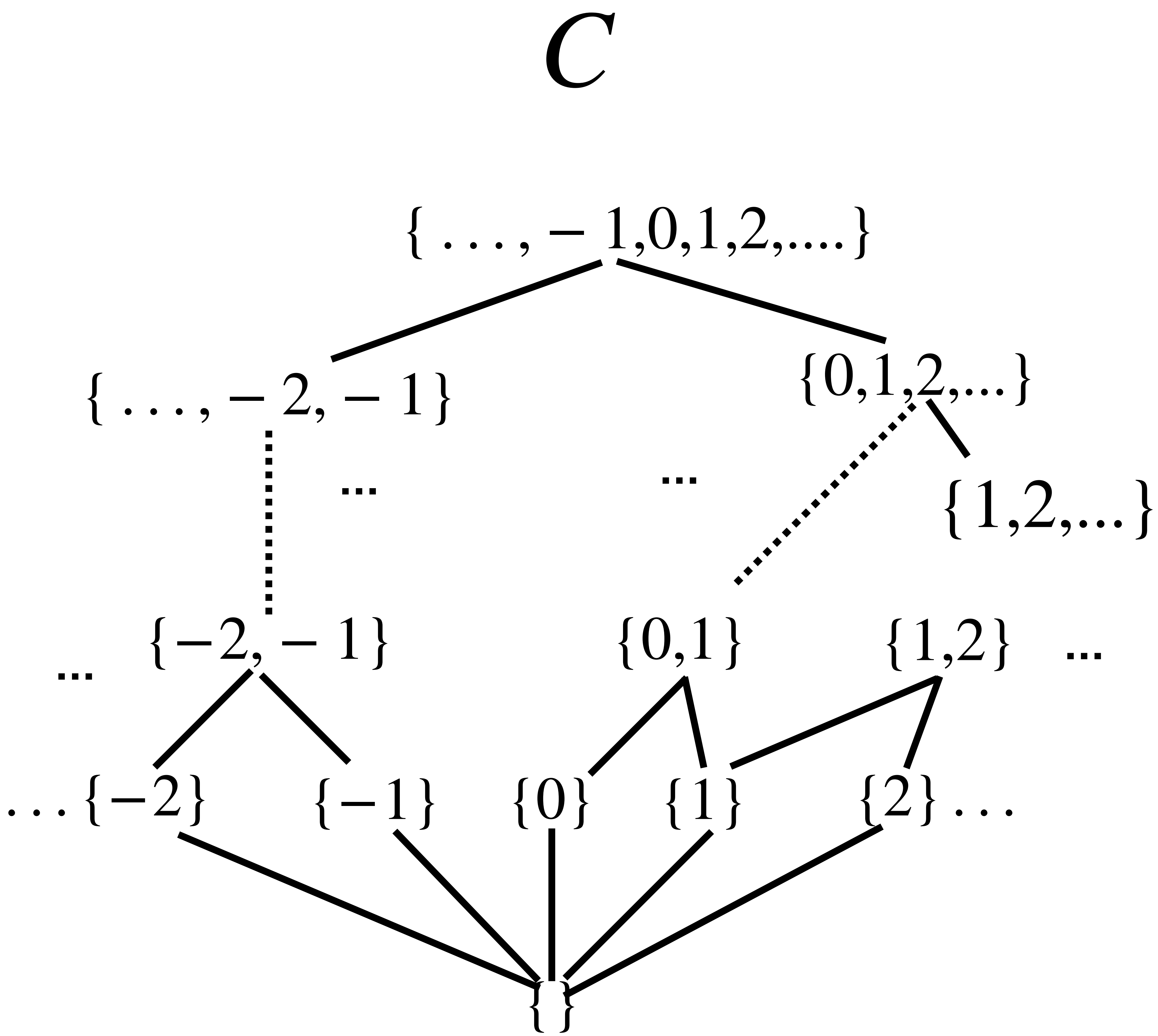


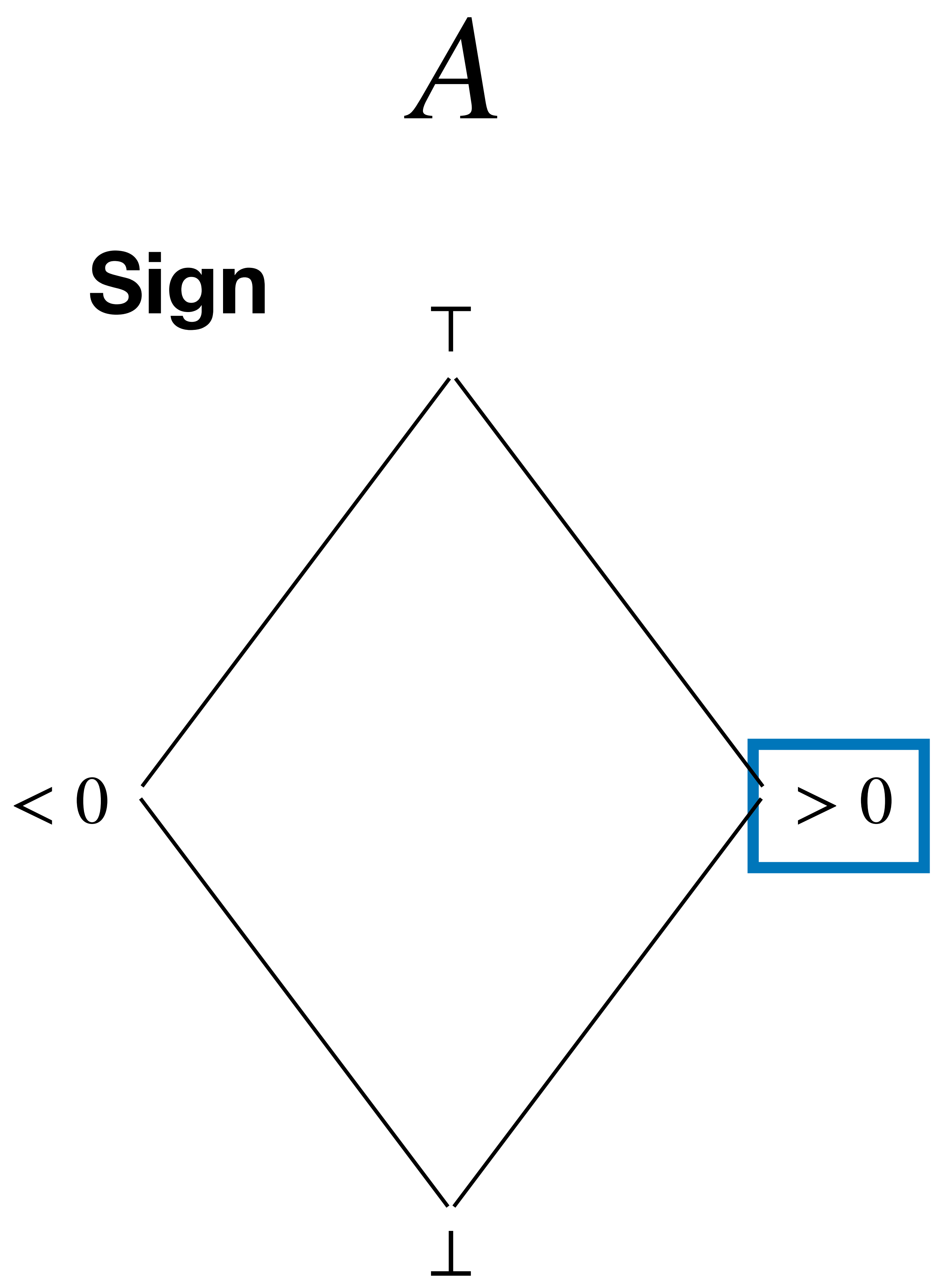
# Defining approximation



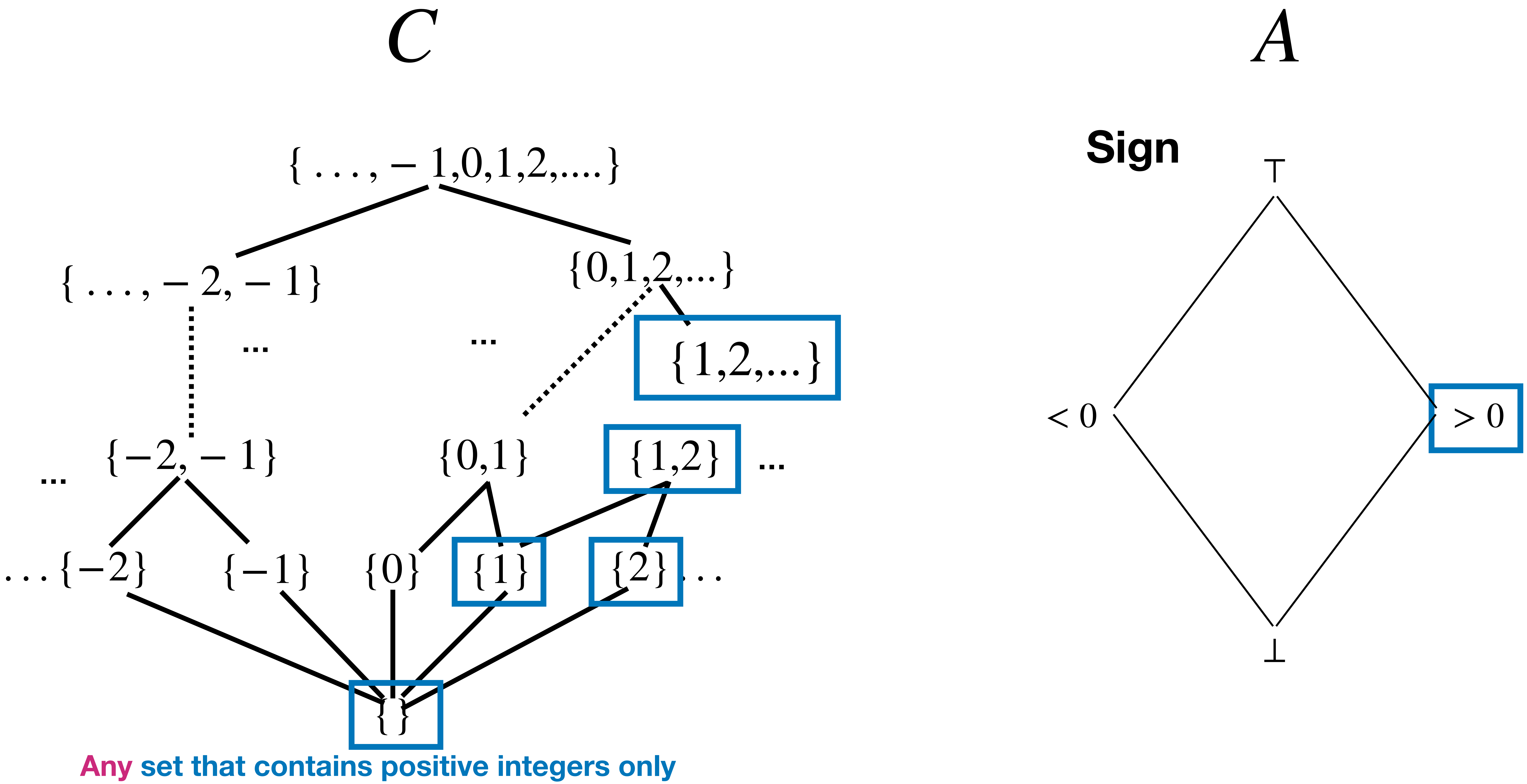



# Defining approximation



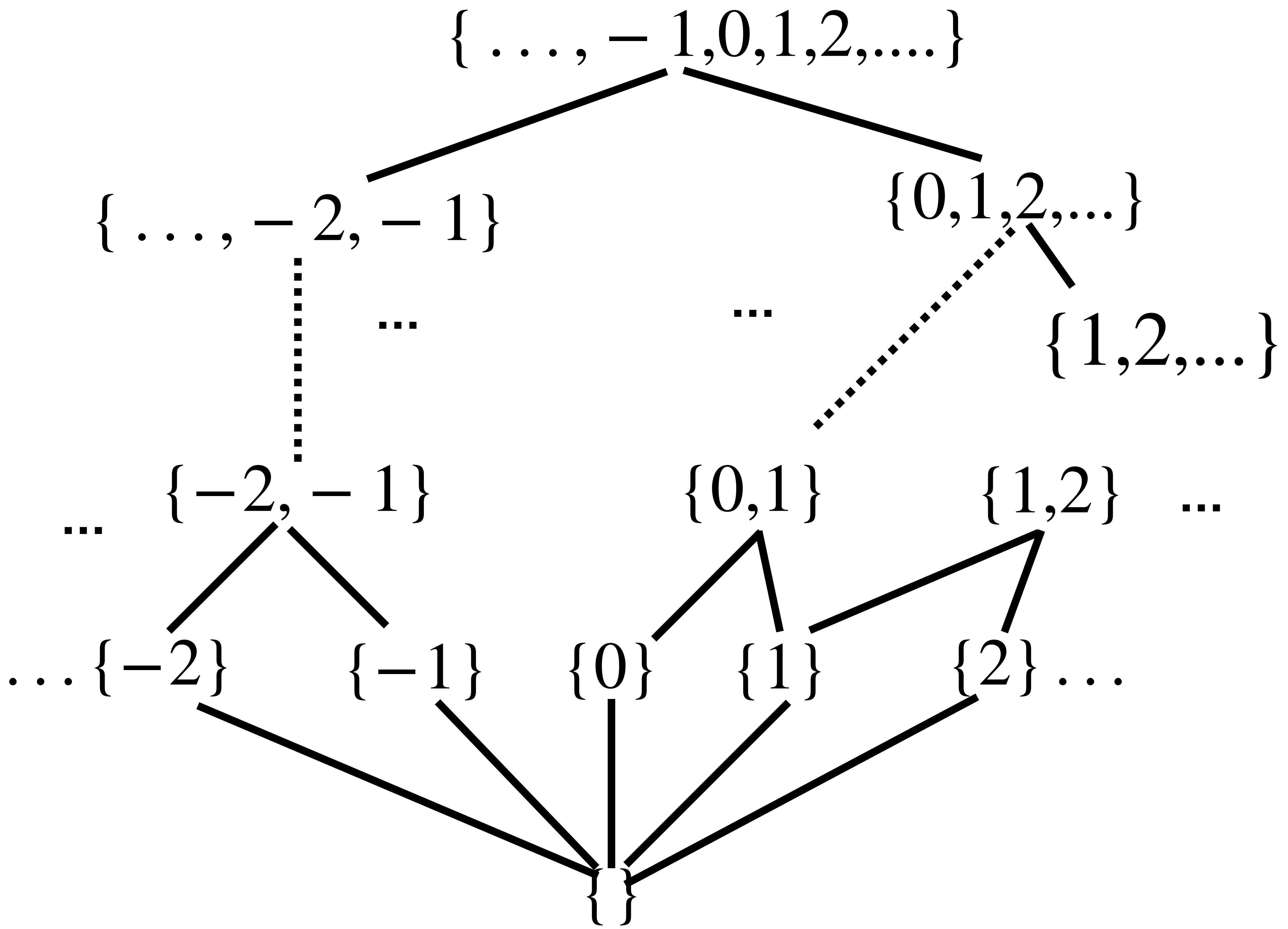



# Defining approximation

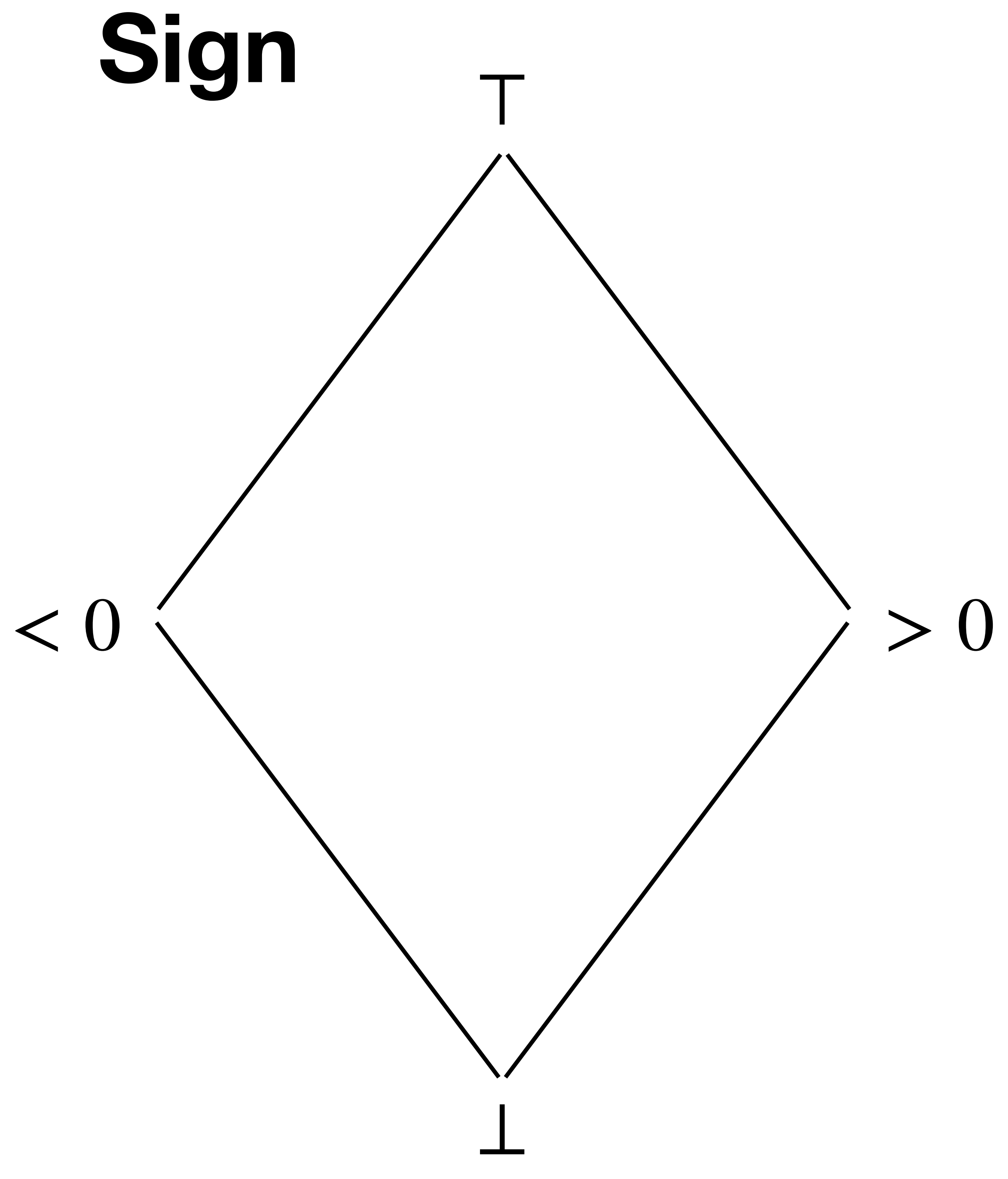


# Defining approximation

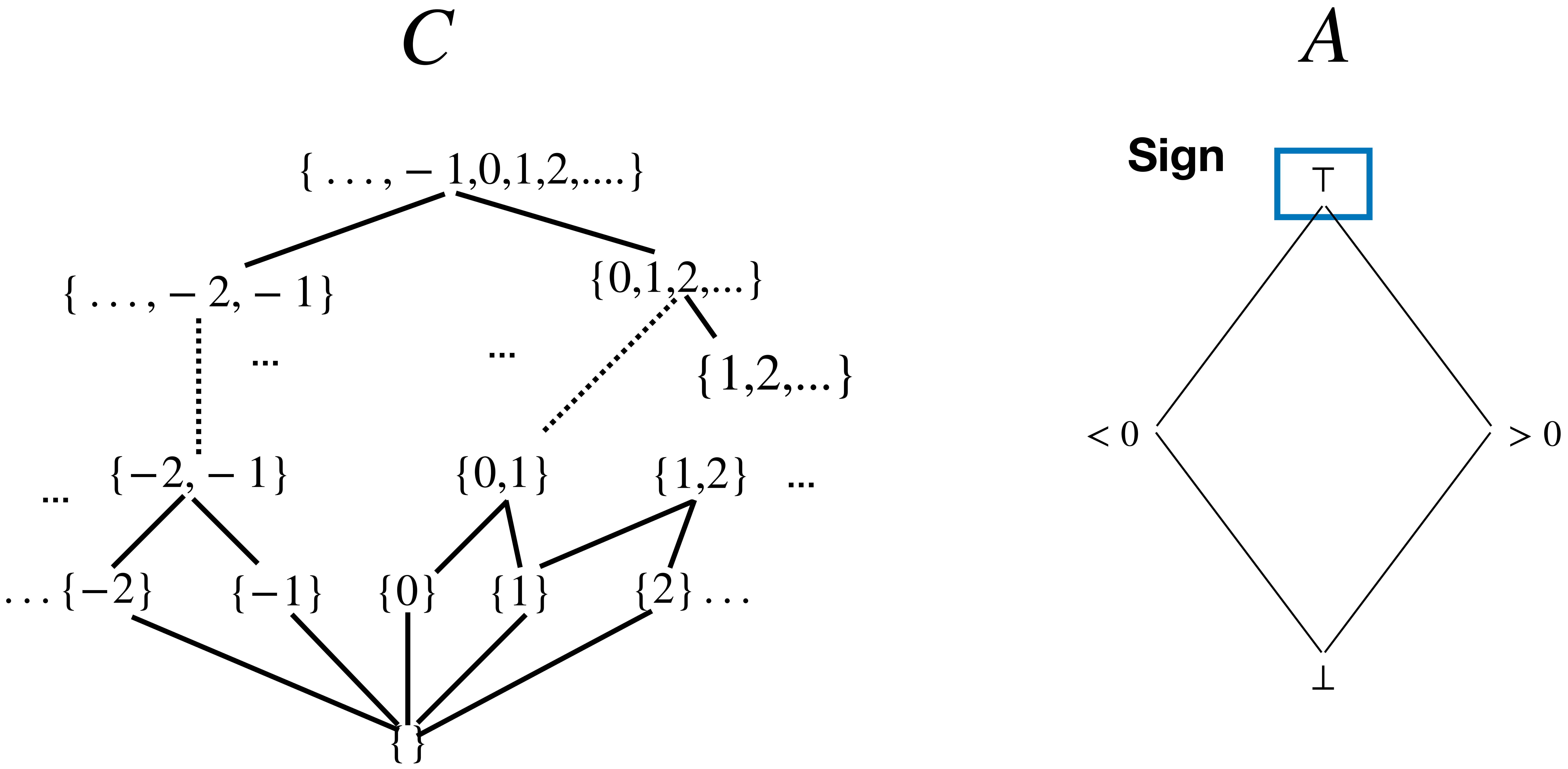
$C$



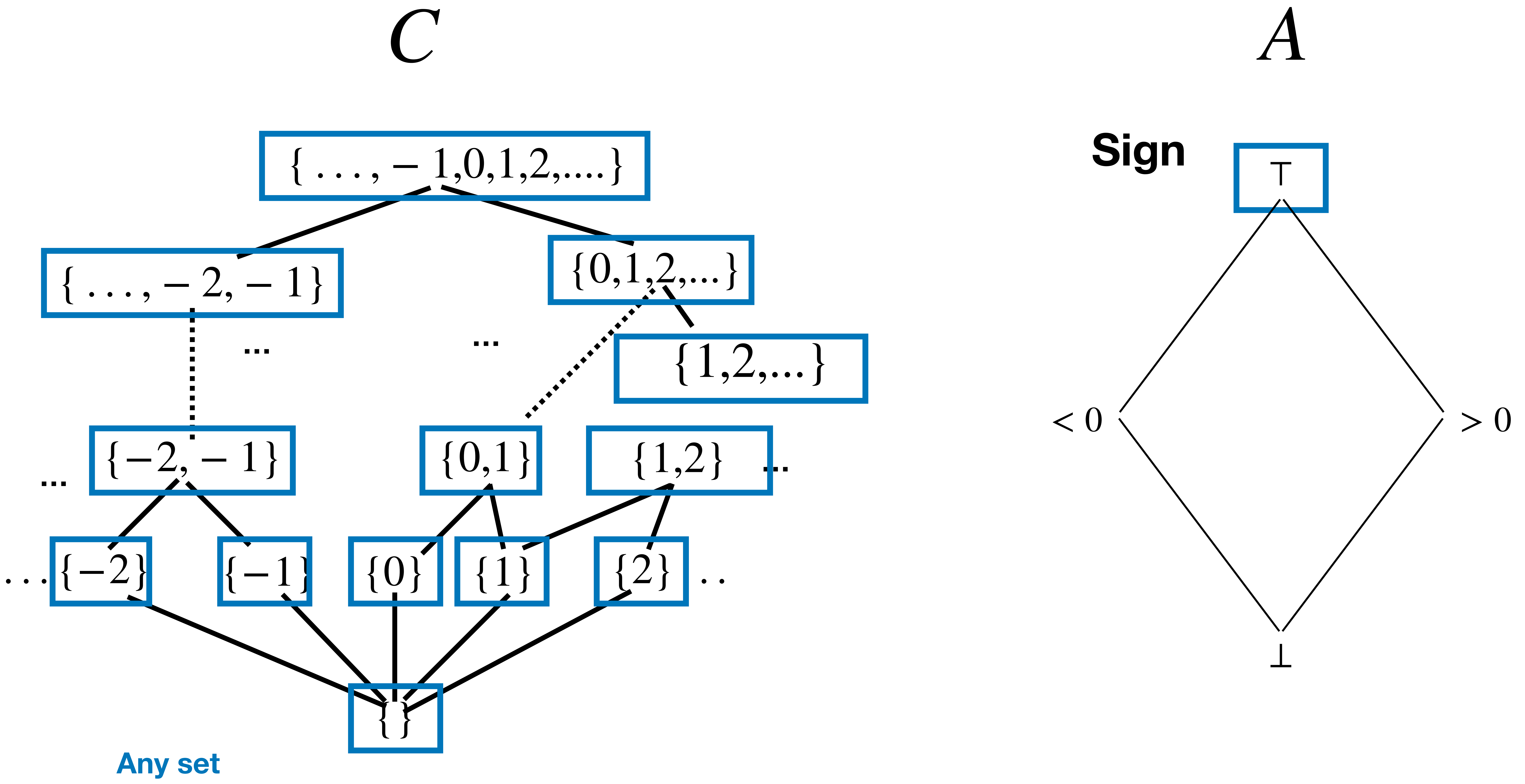

$A$



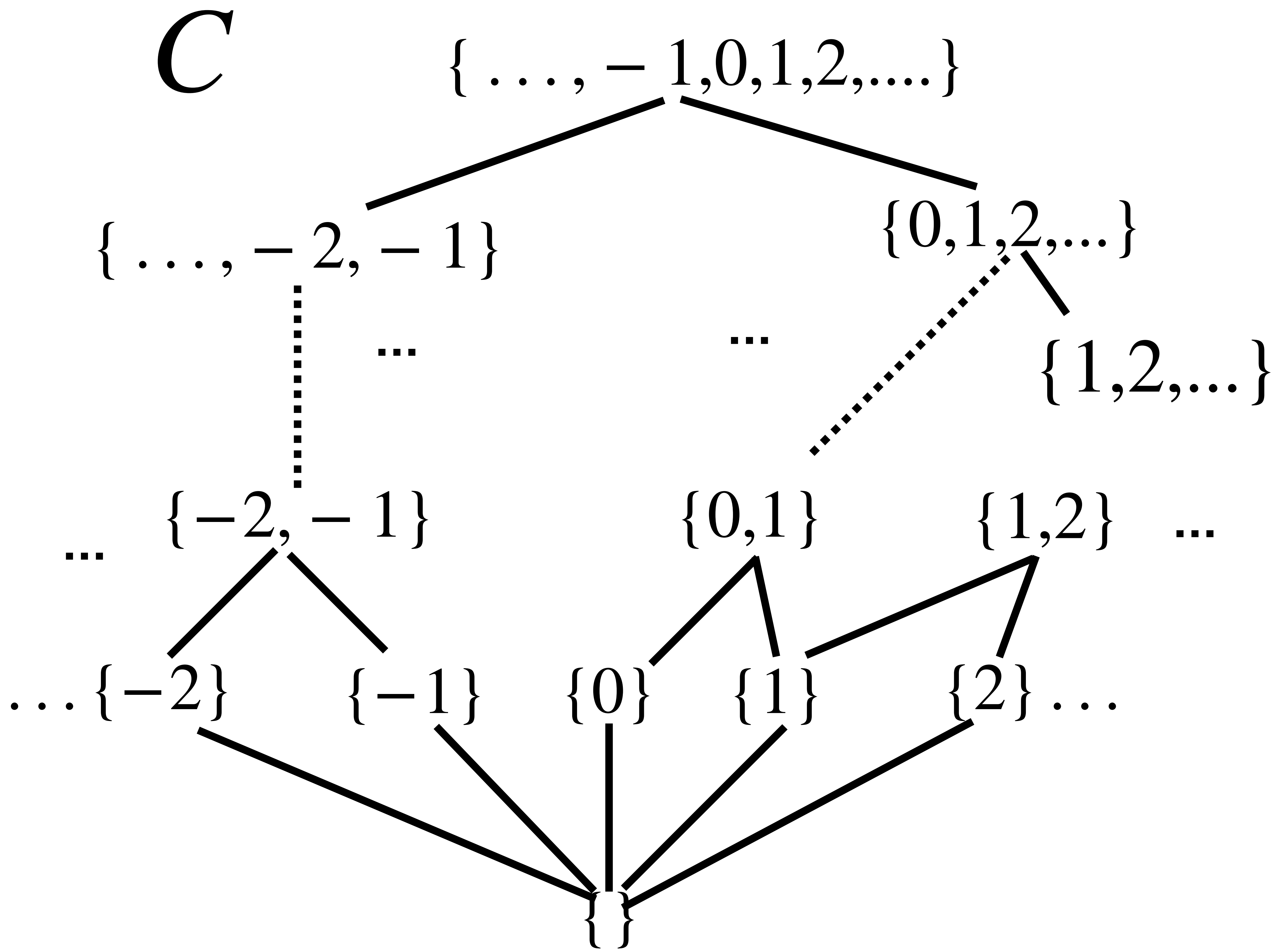
# Defining approximation

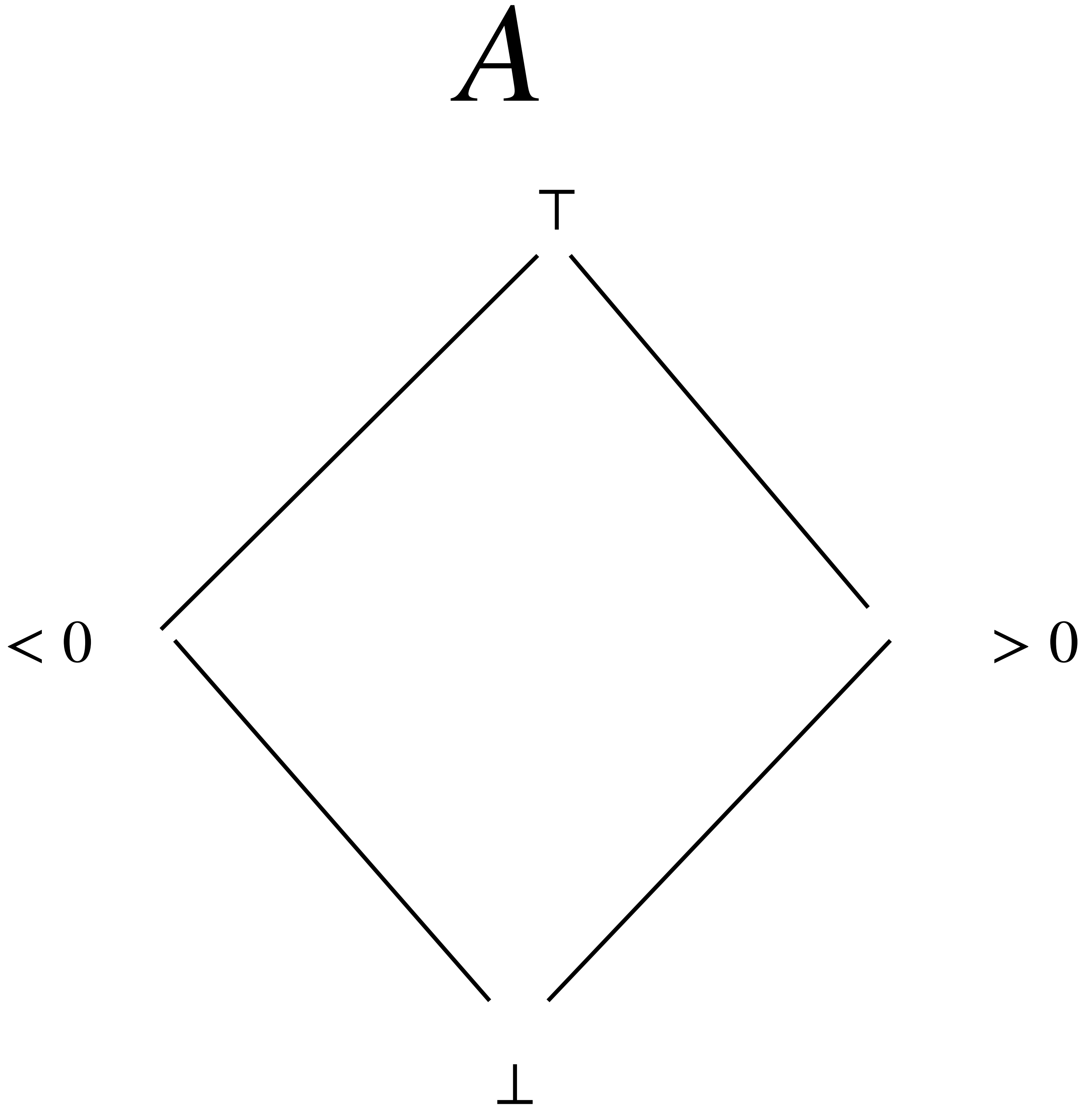


# Defining approximation



# Concretization function



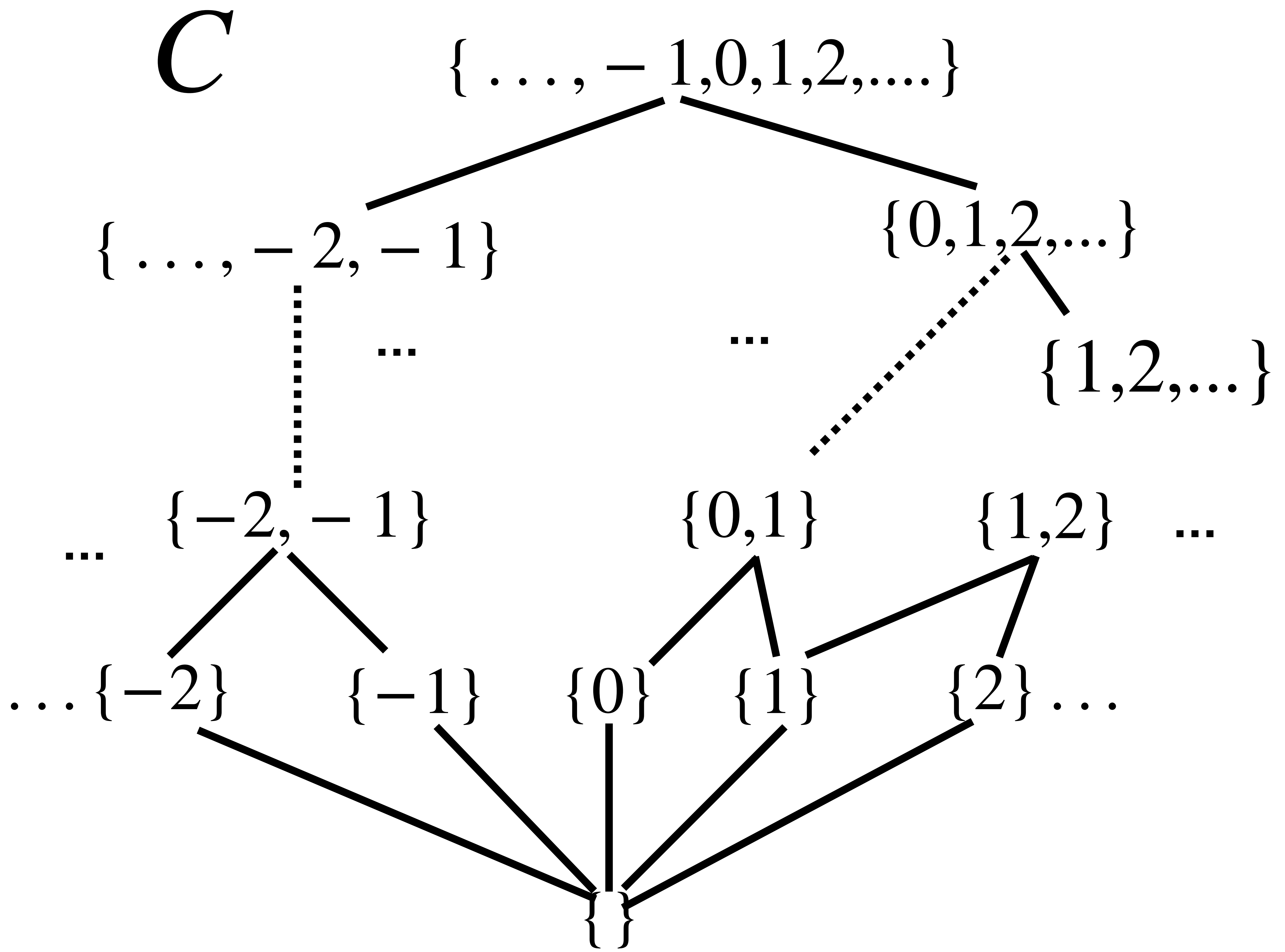



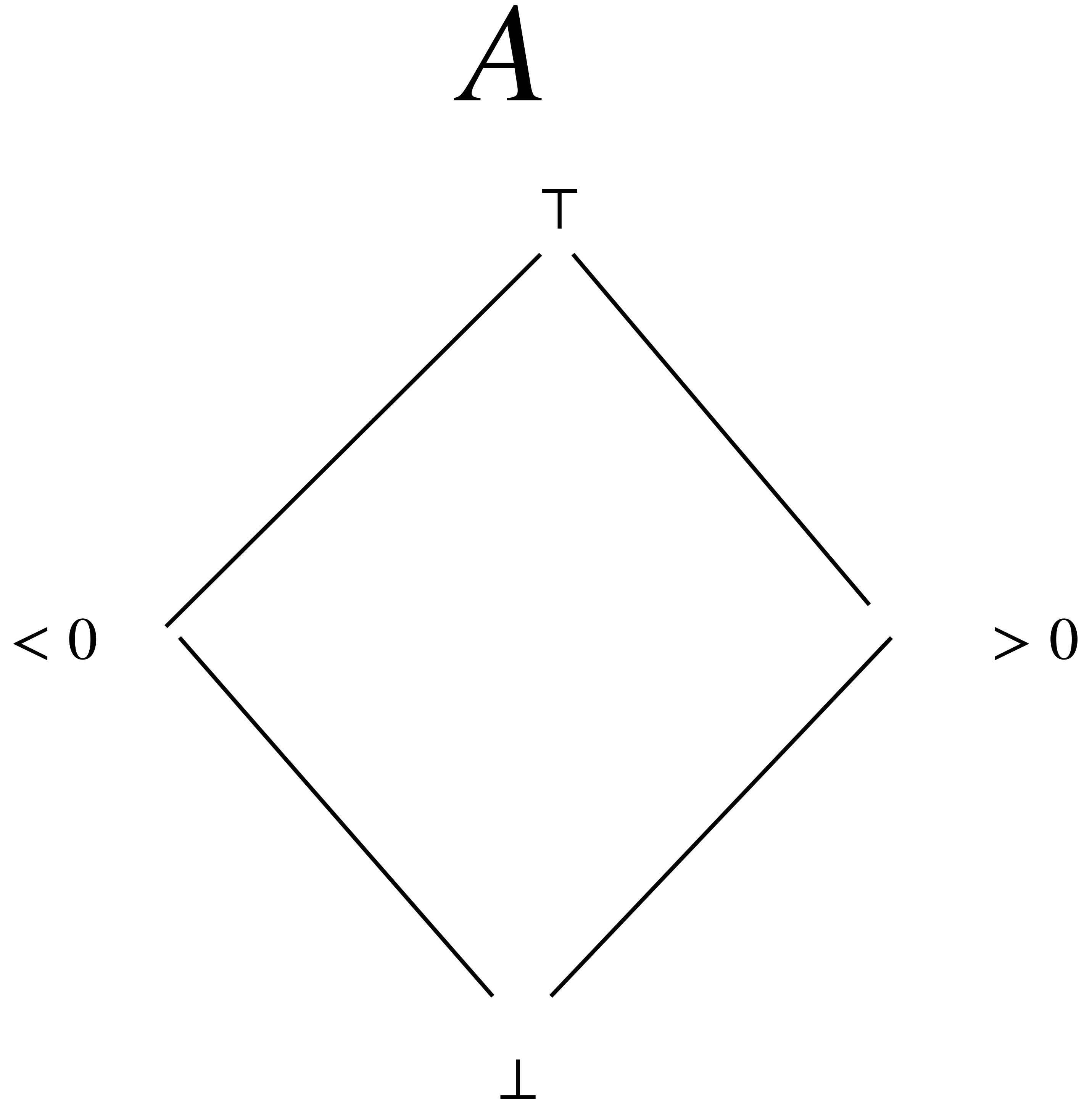
# Concretization function

**Definition**

Concretization function $\gamma : A \to C$ is a monotone function

that maps abstract $a$ into the greatest concrete $c$ that it approximates



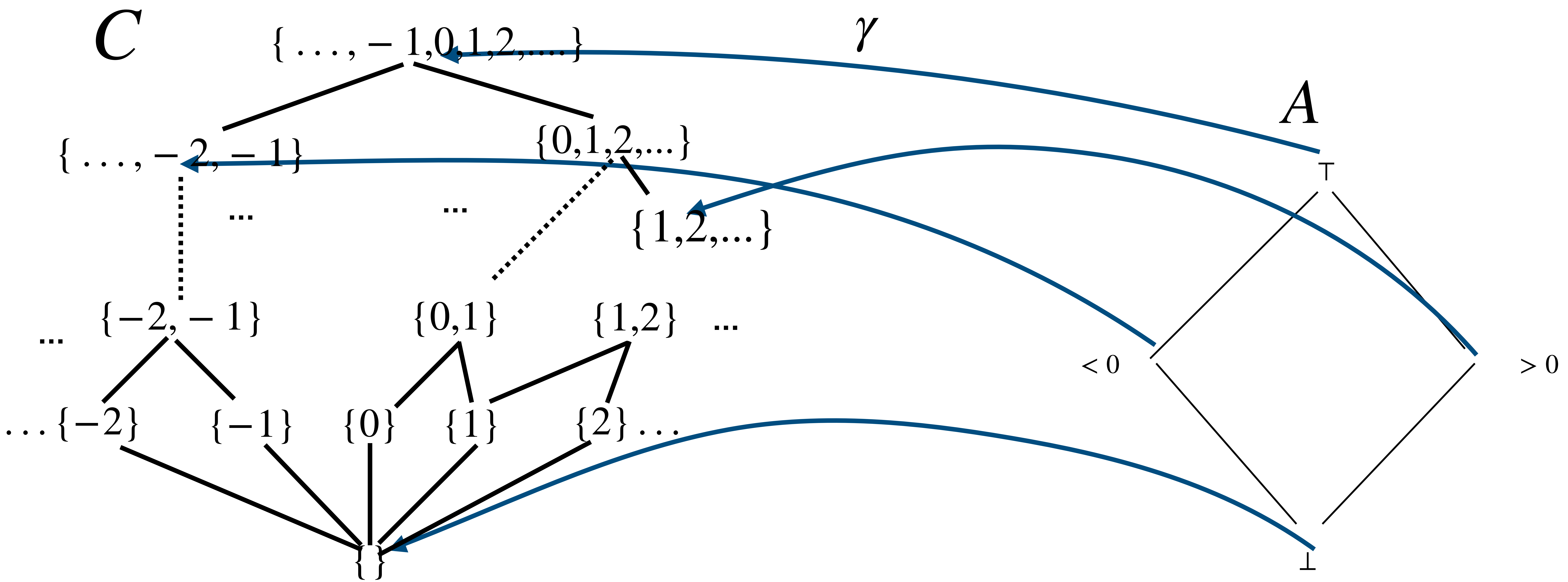



# Concretization function

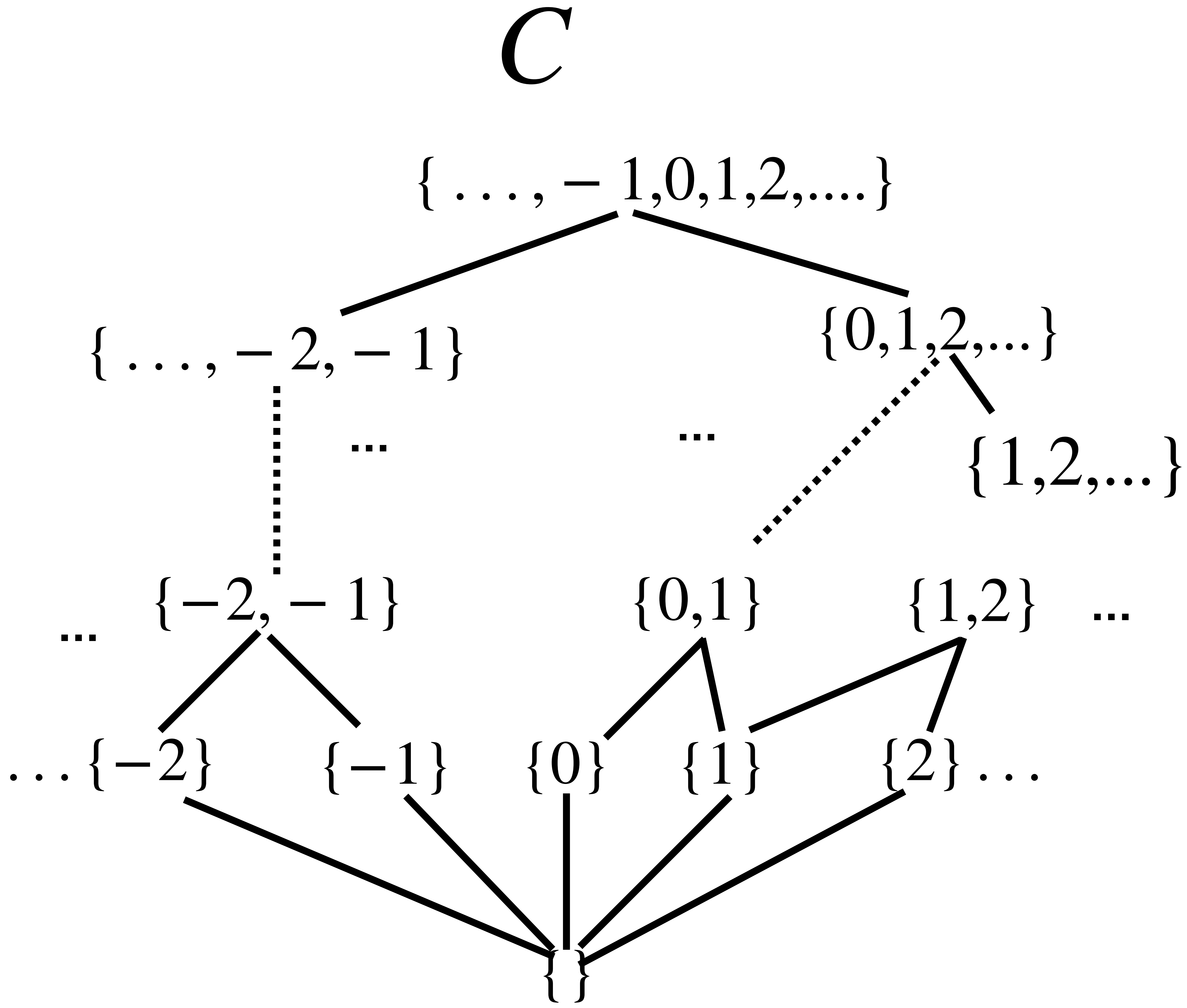
**Definition**

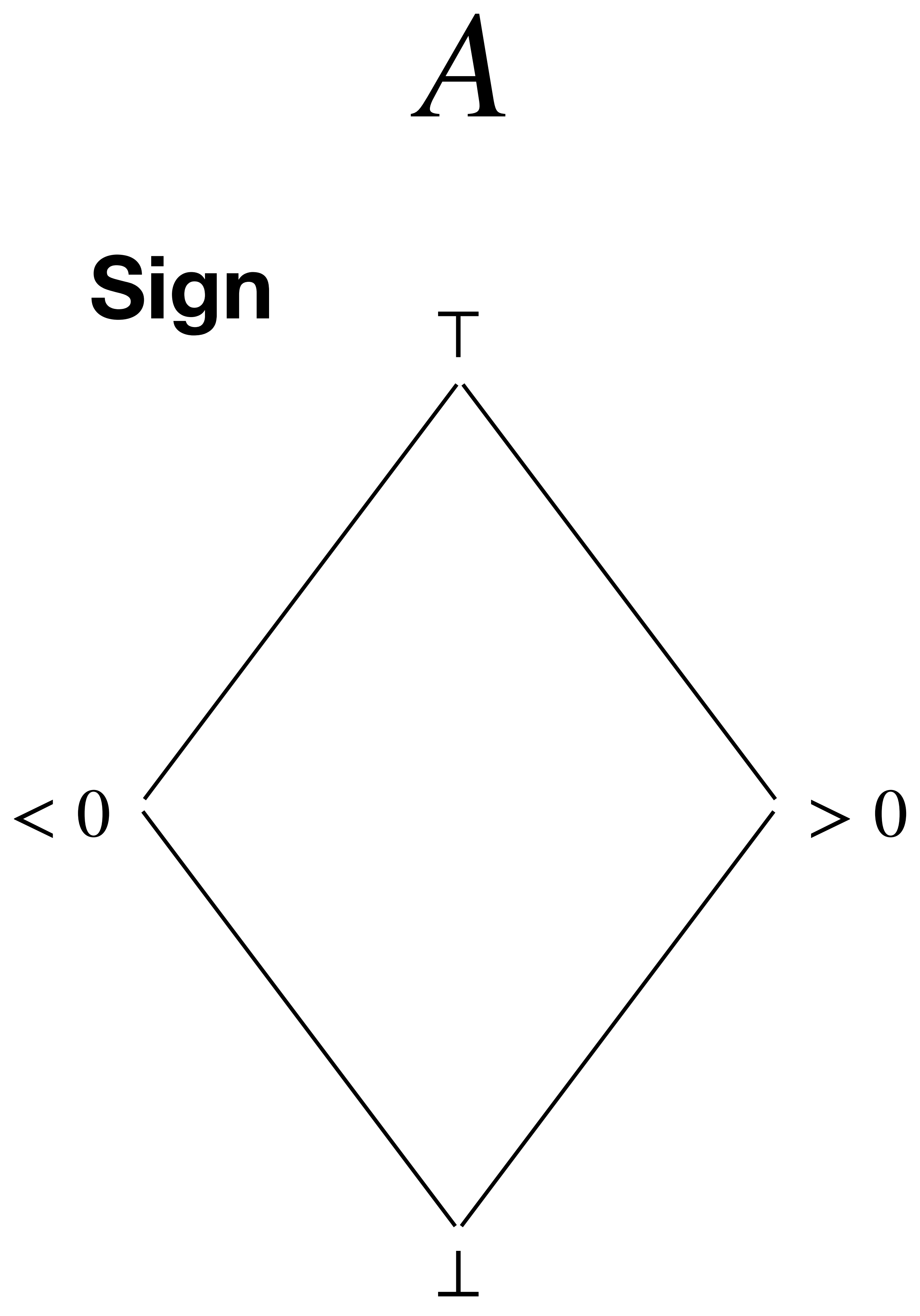
Concretization function $\gamma : A \to C$ is a monotone function

that maps abstract $a$ into the greatest concrete $c$ that it approximates

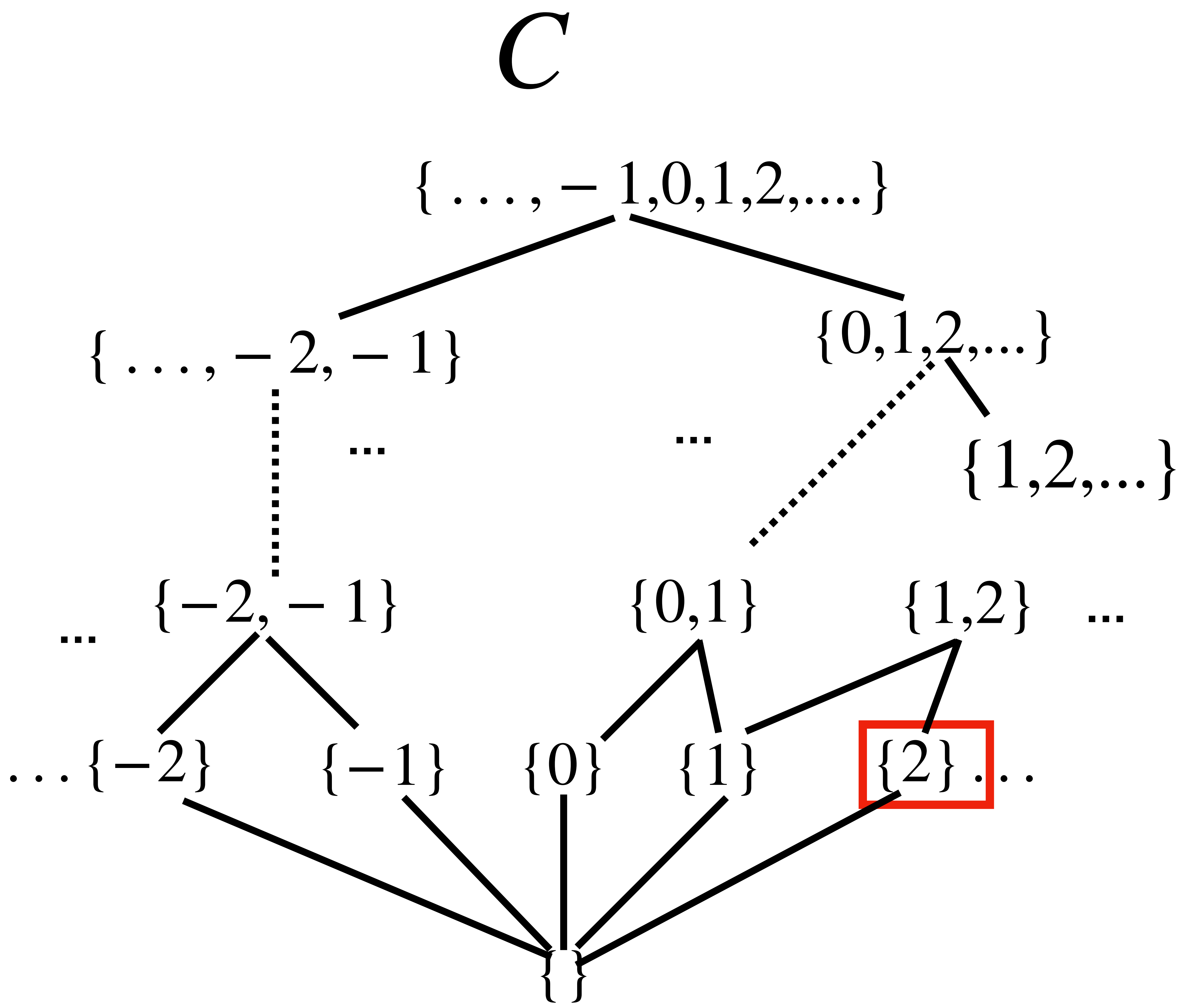


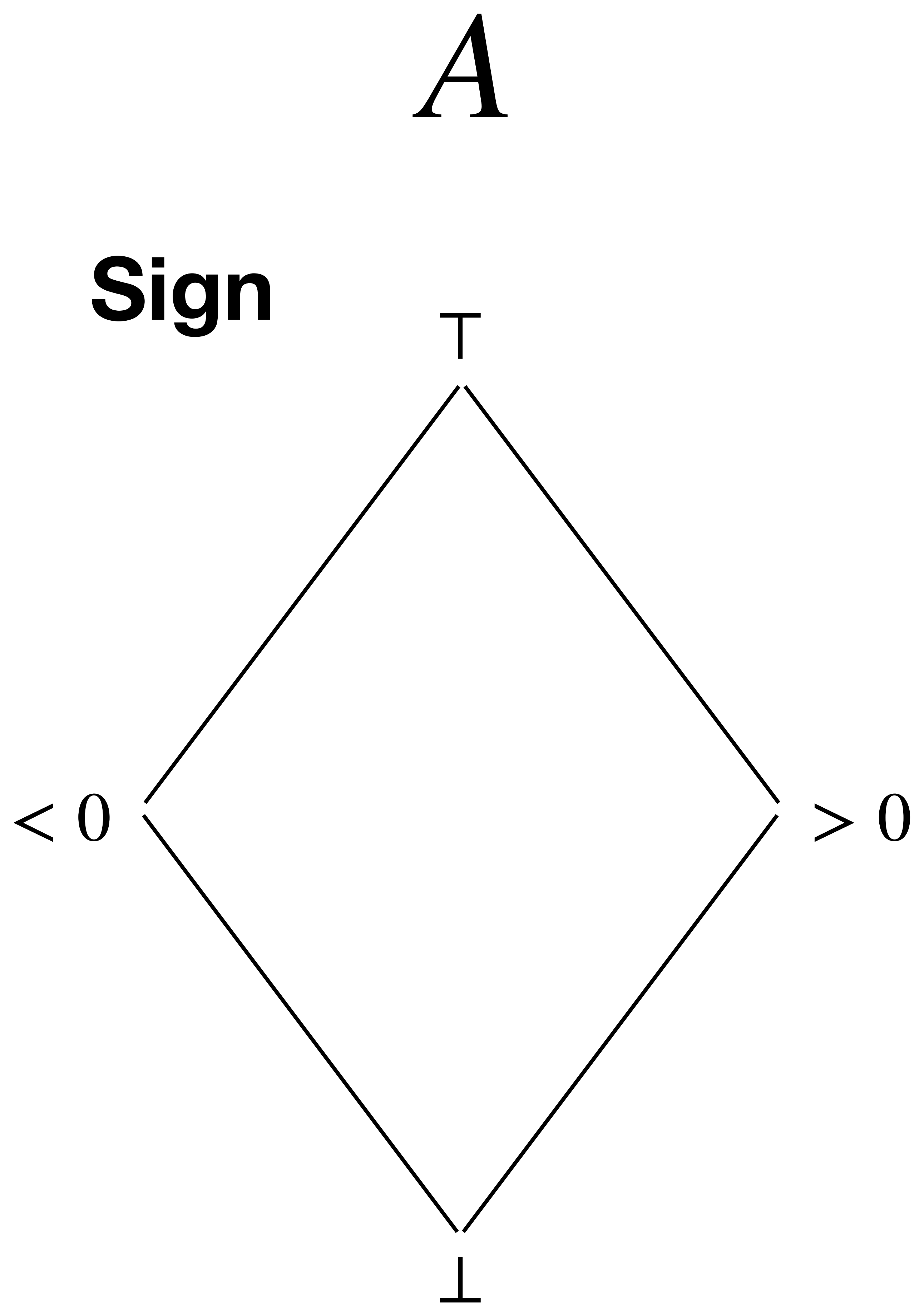
# Defining approximation



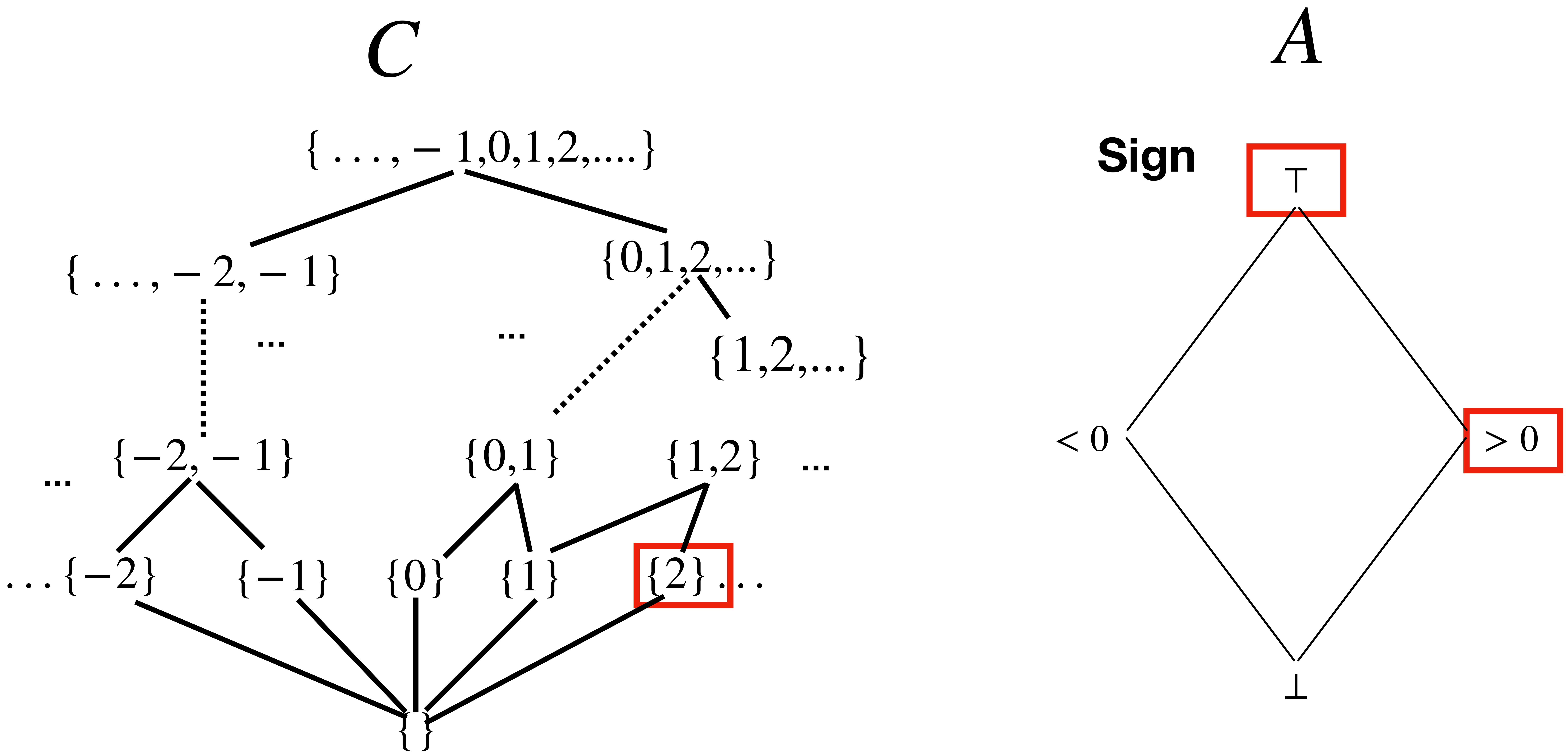



# Defining approximation

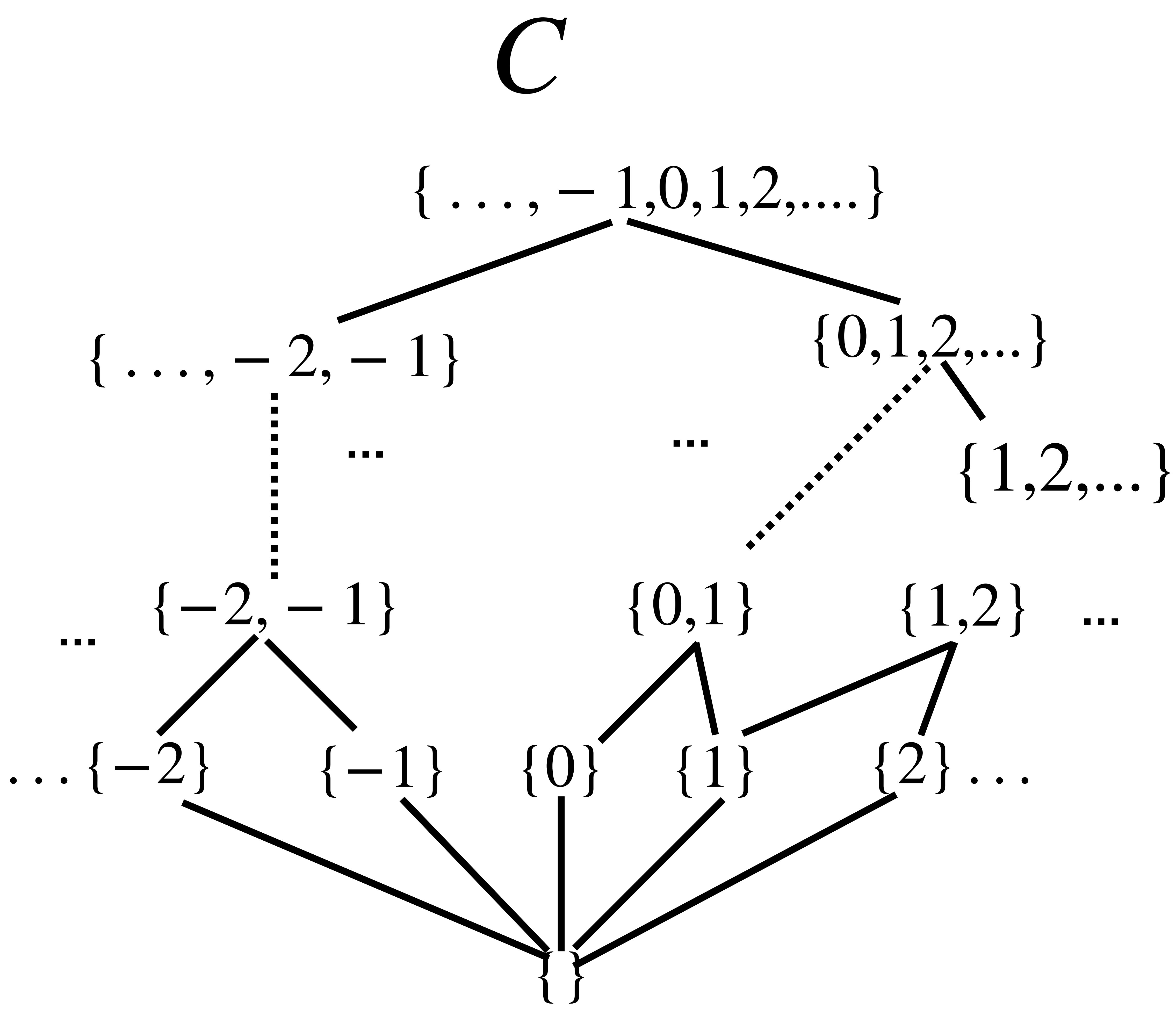


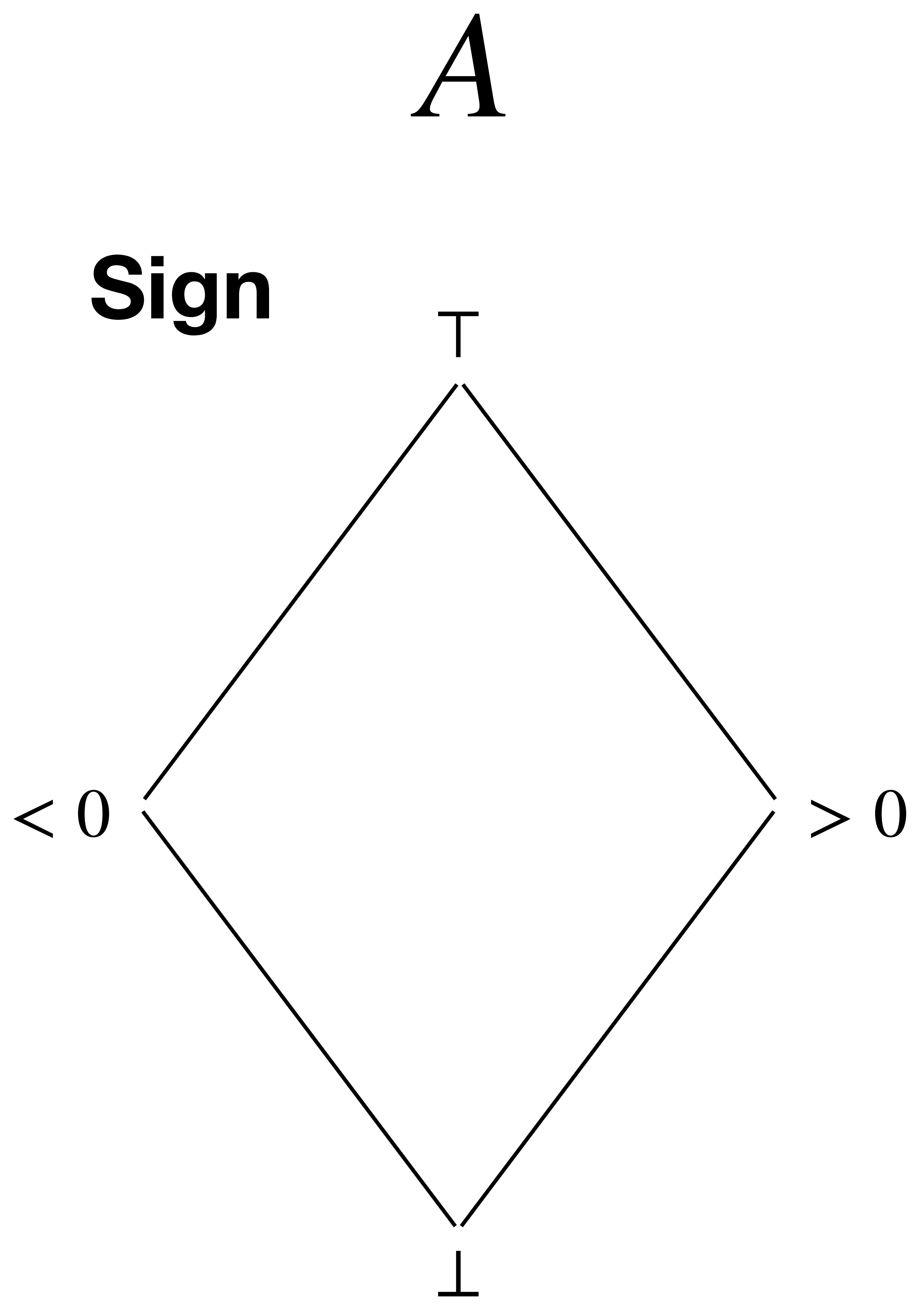



# Defining approximation

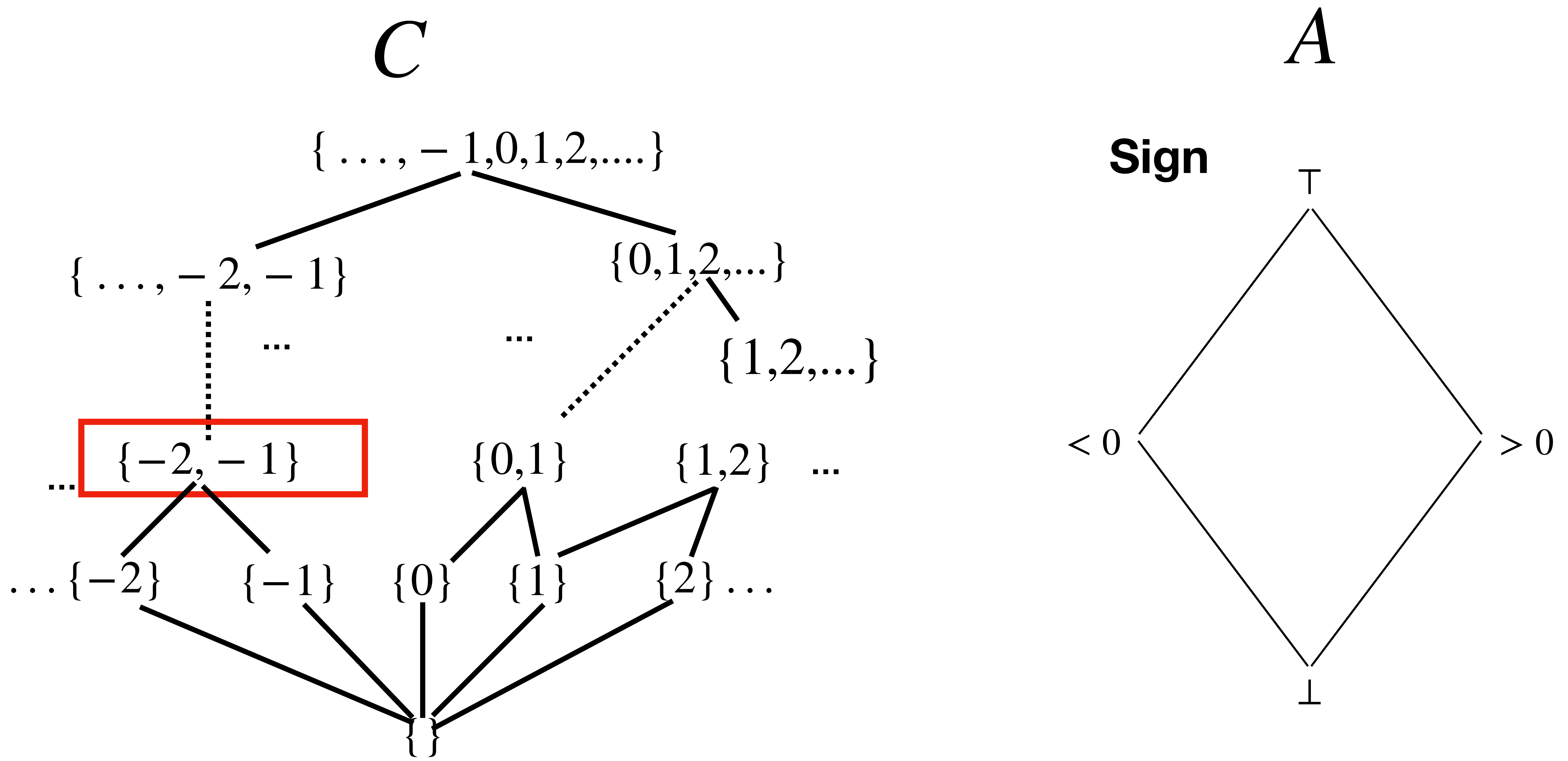


# Defining approximation

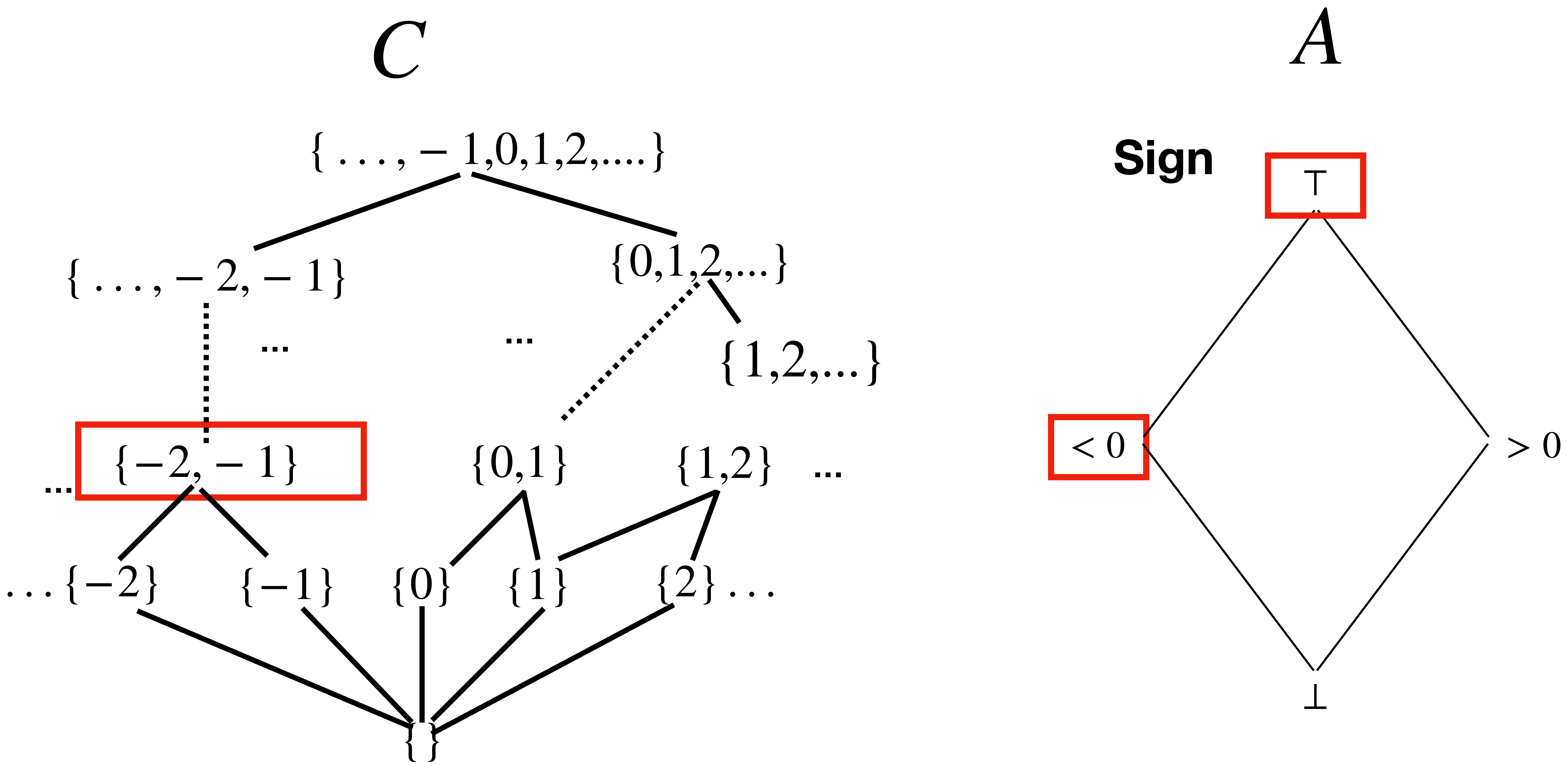


# Defining approximation

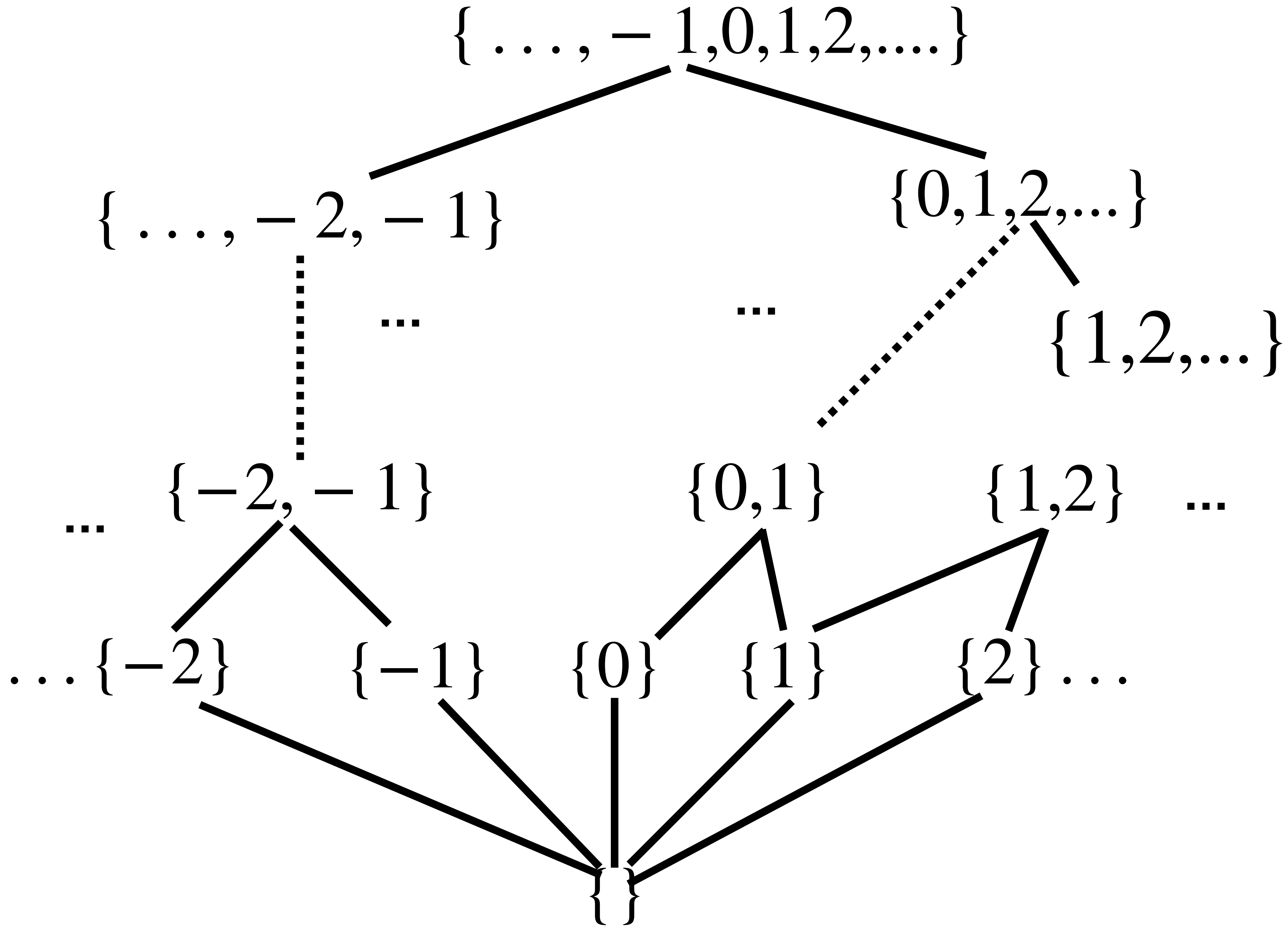


# Defining approximation



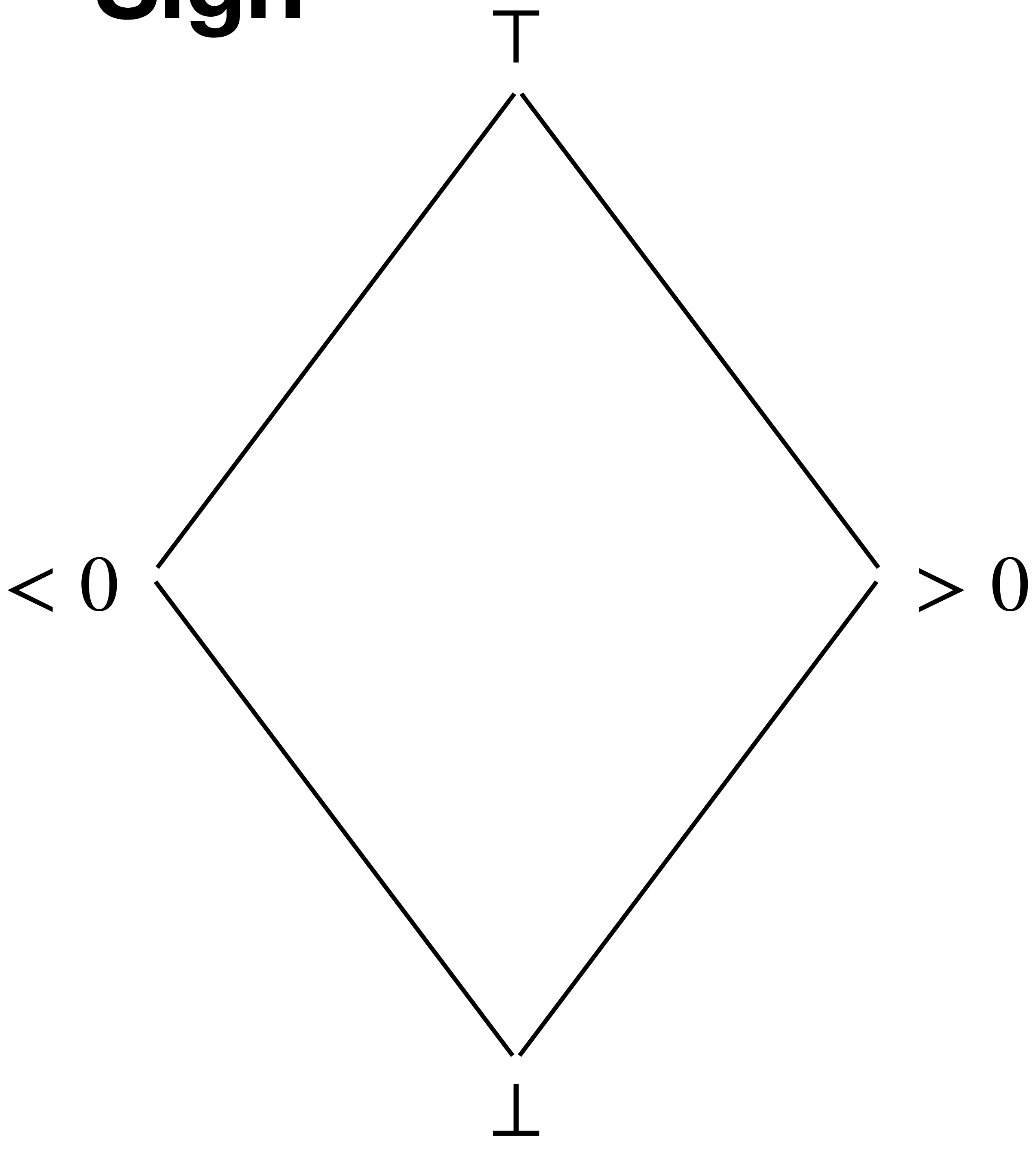
# Defining approximation

$C$

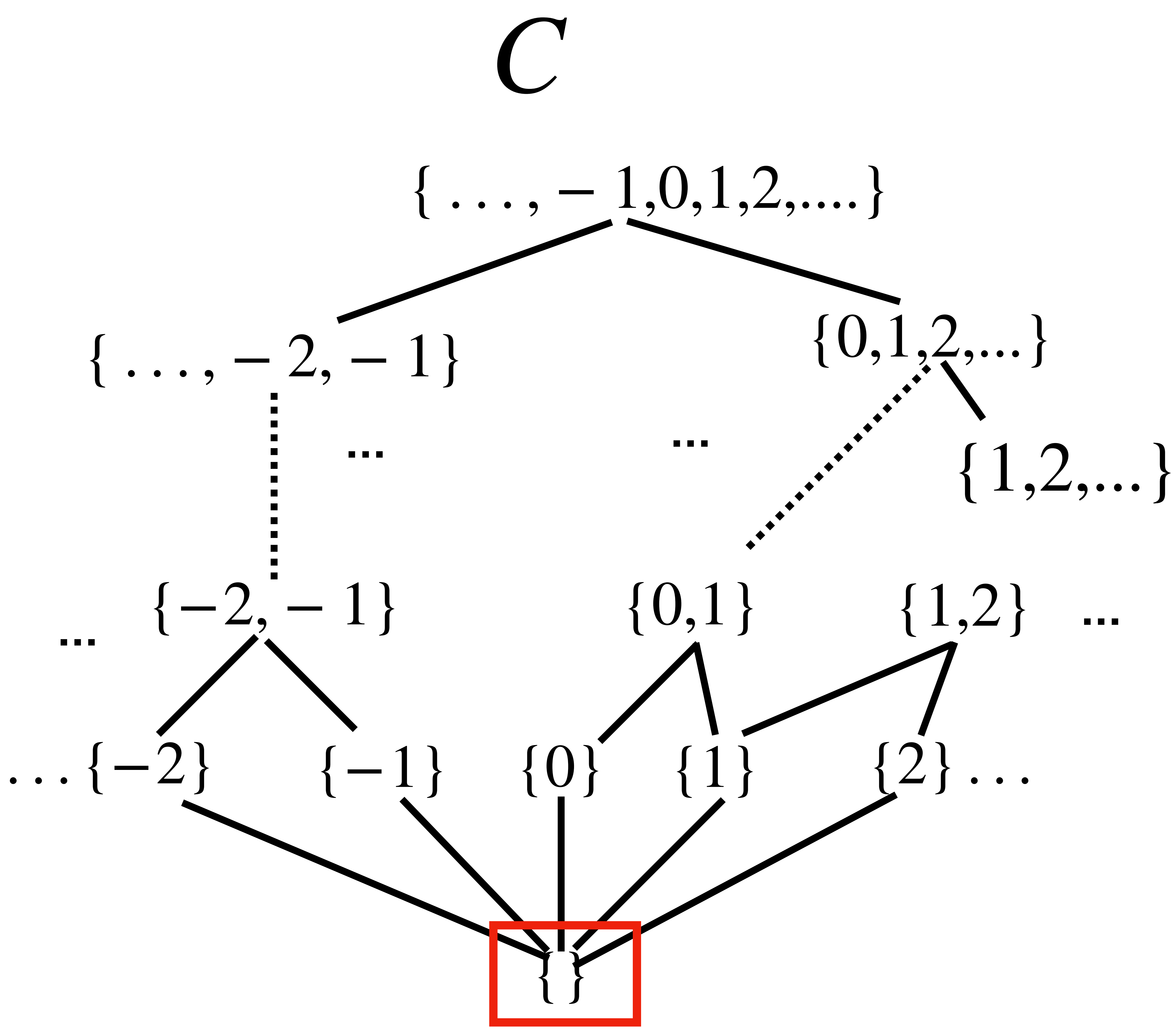


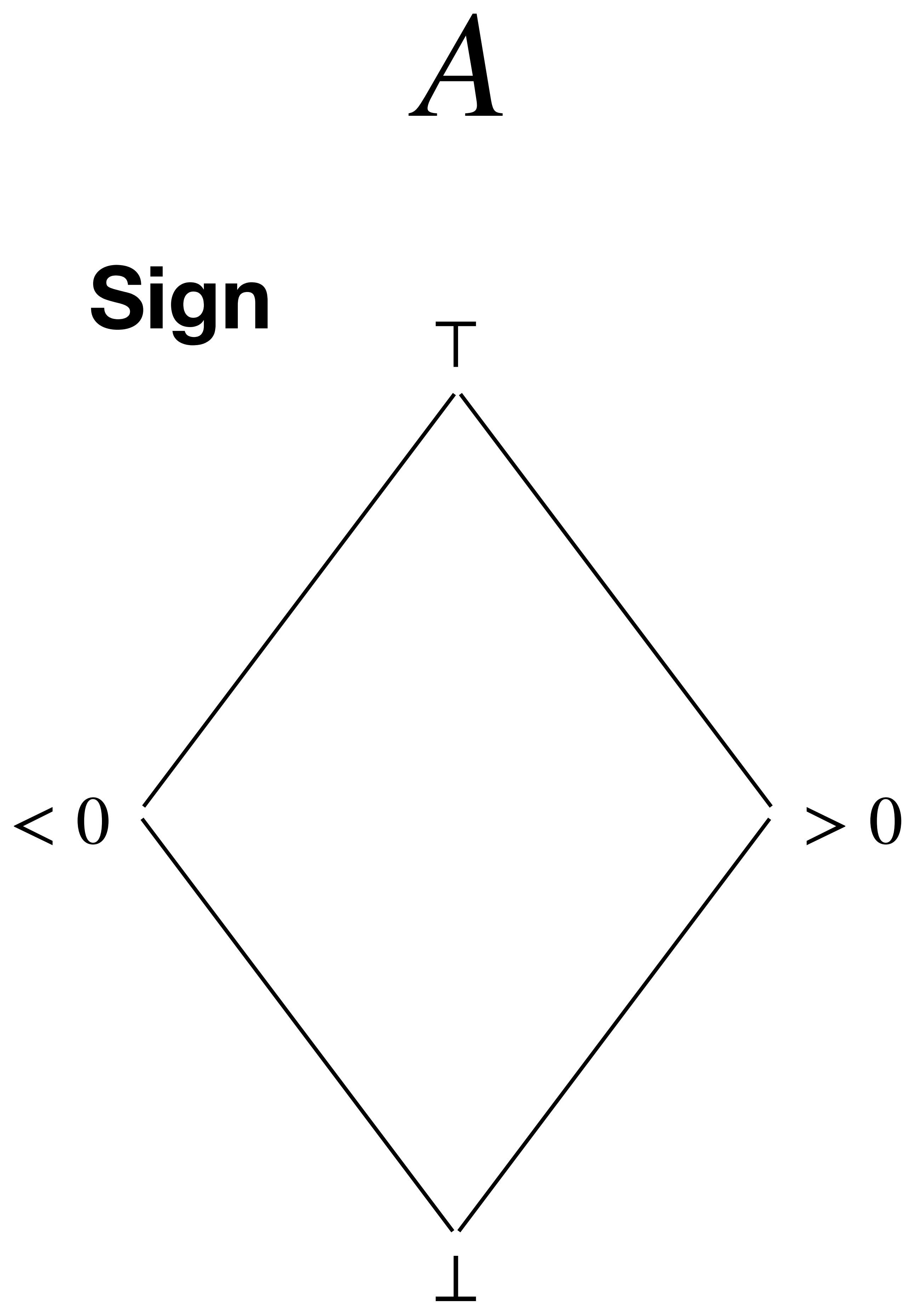

$A$

**Sign**

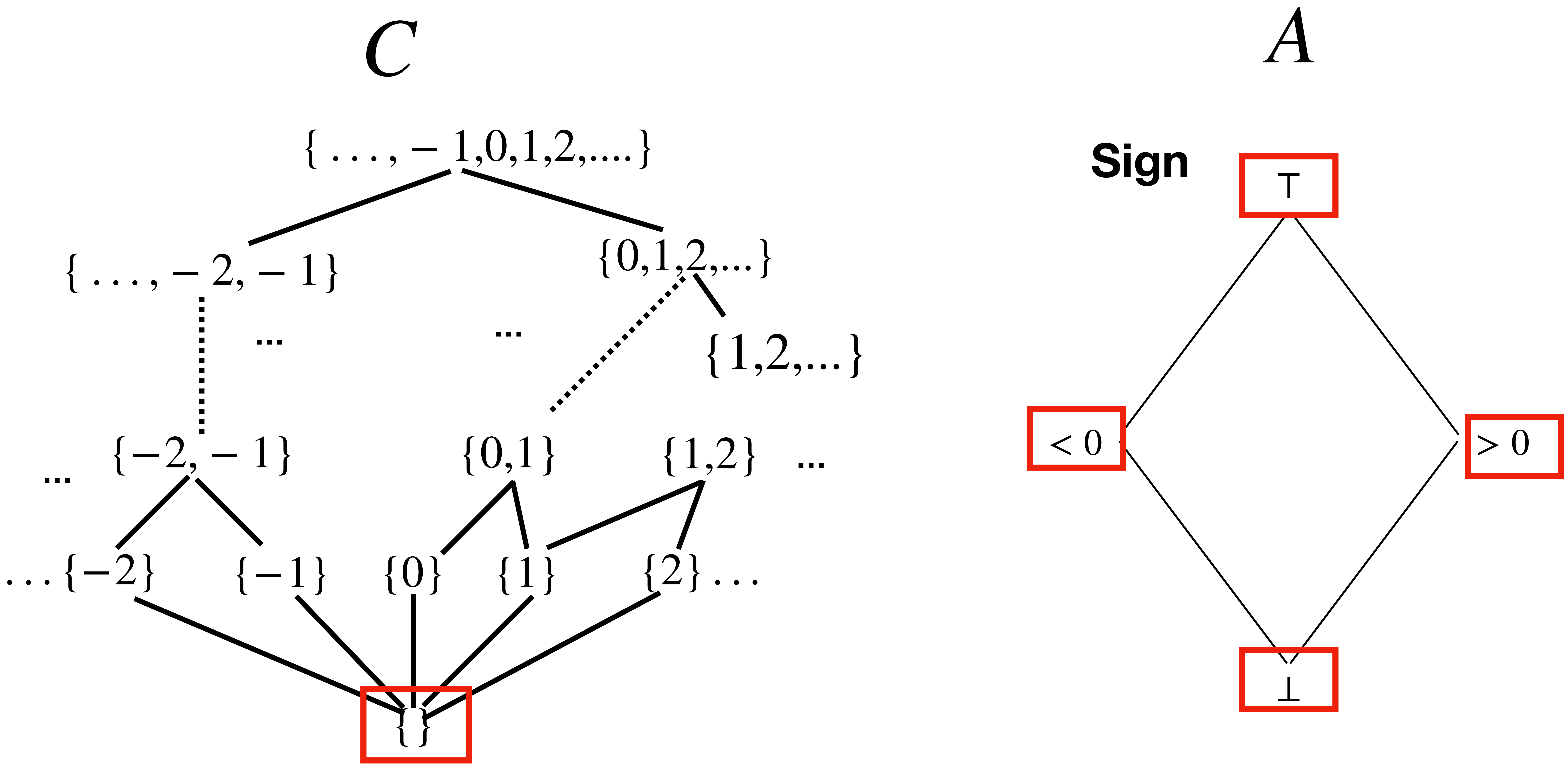


# Defining approximation



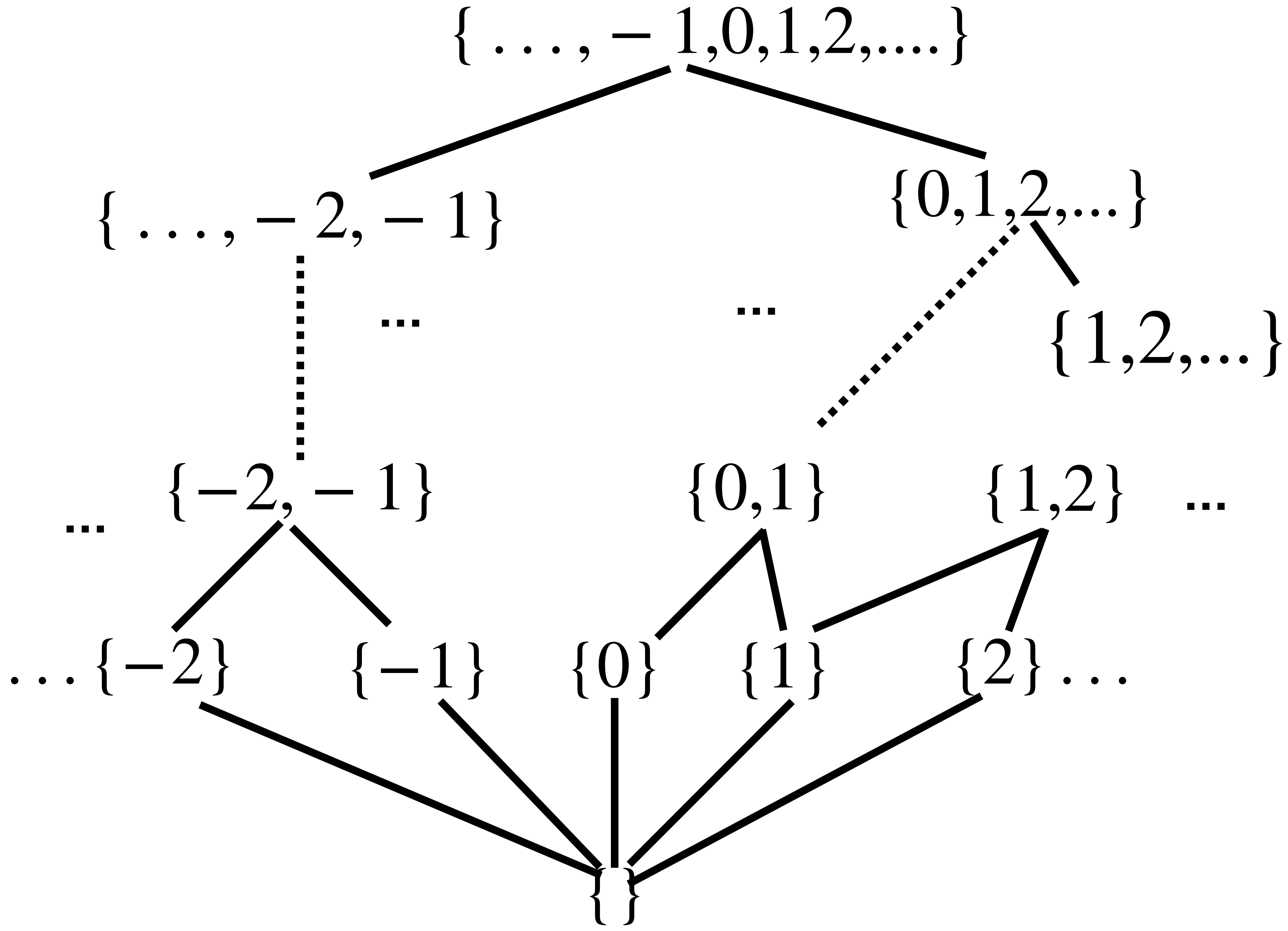



# Defining approximation



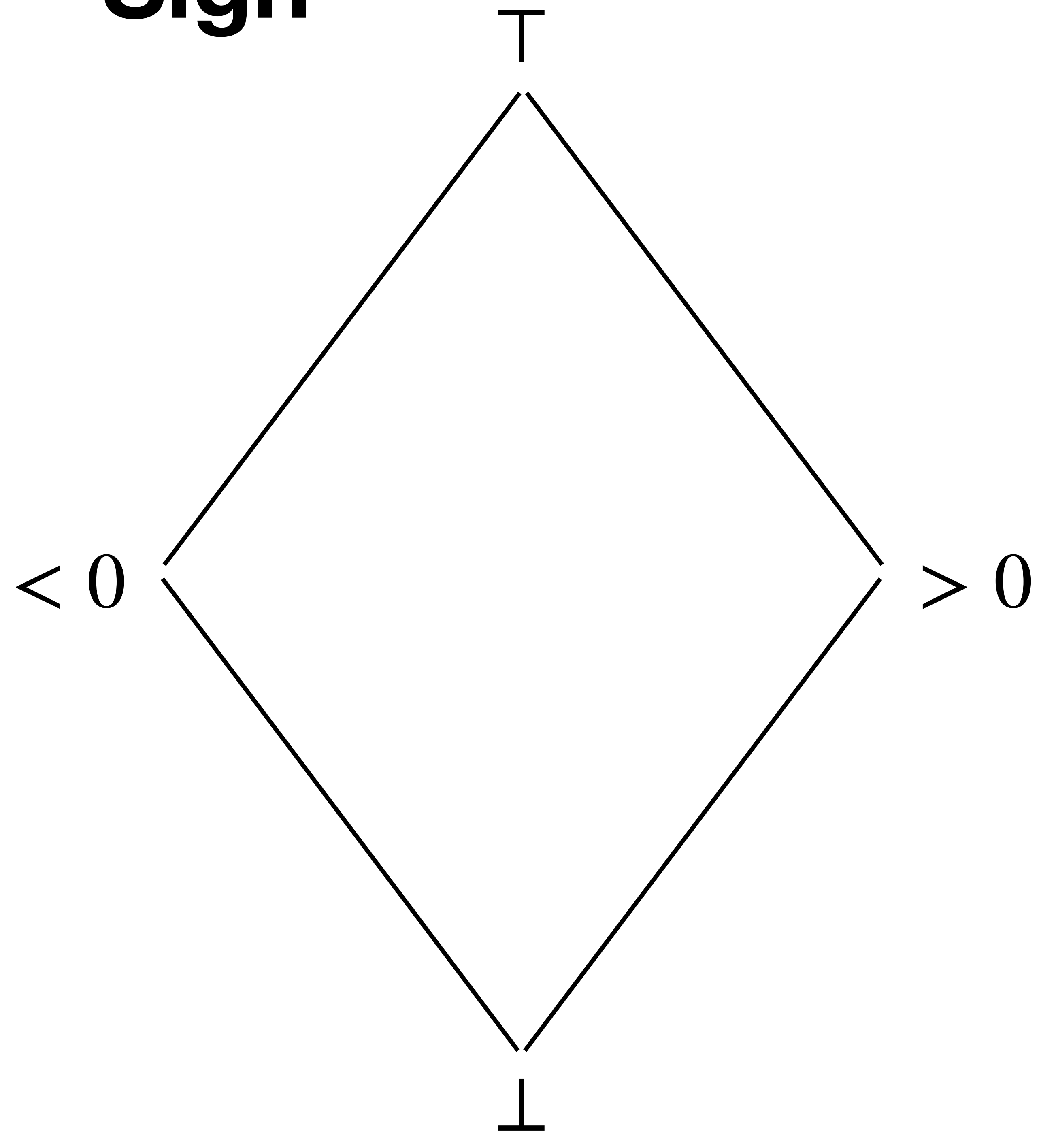
# Defining approximation

$C$

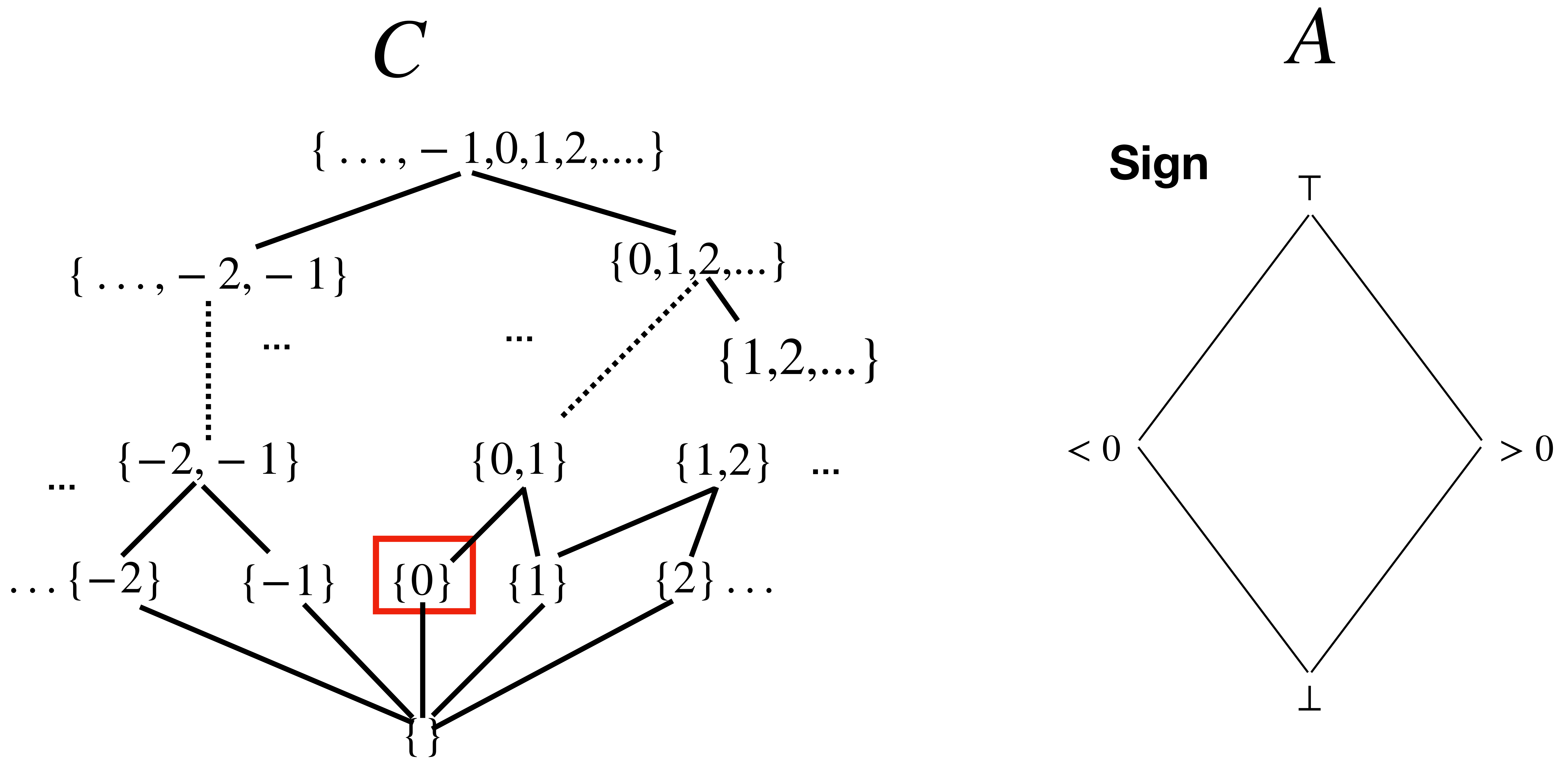



$A$

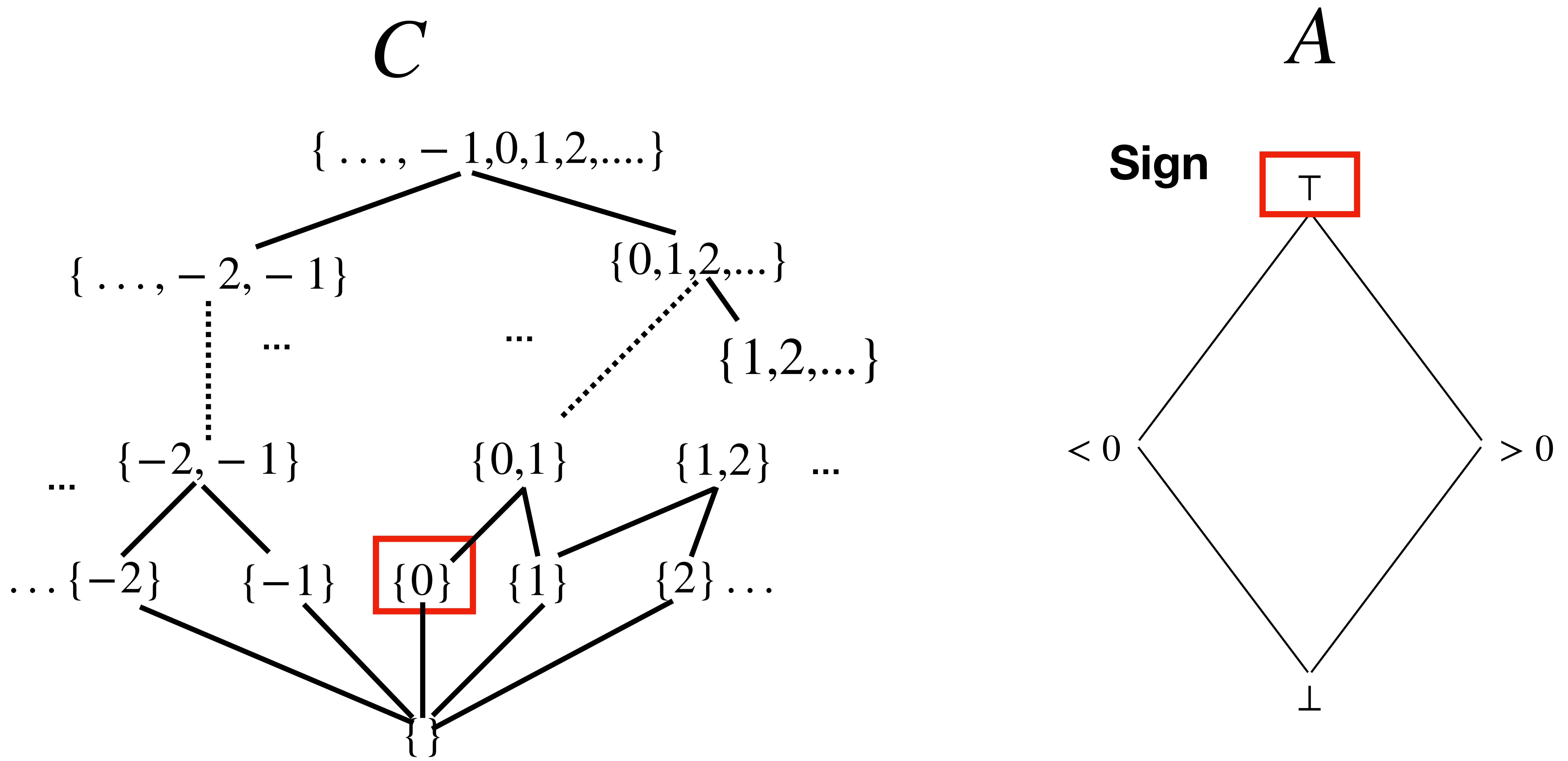
**Sign**



# Defining approximation



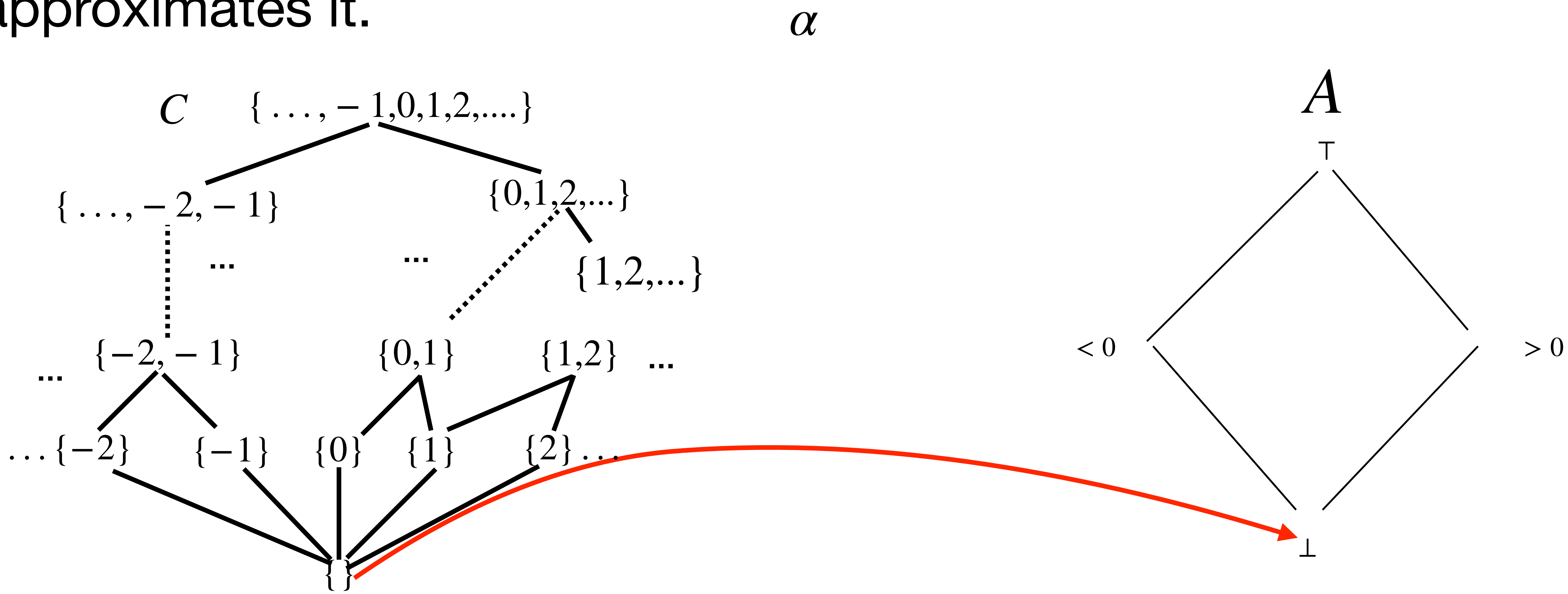
# Defining approximation



# Abstraction function

**Definition**

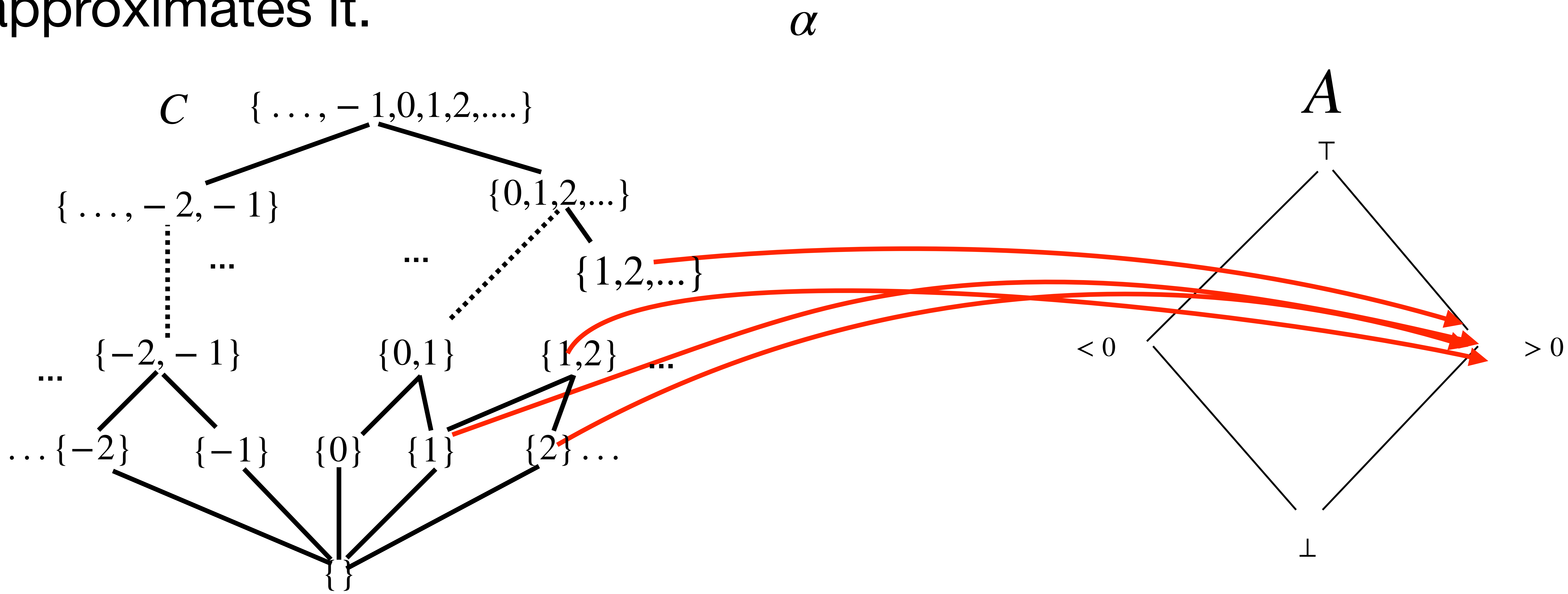
Abstraction function $\alpha : C \to A$ is a monotone function that maps concrete $c$ into the most precise abstract element that approximates it.



# Abstraction function

**Definition**

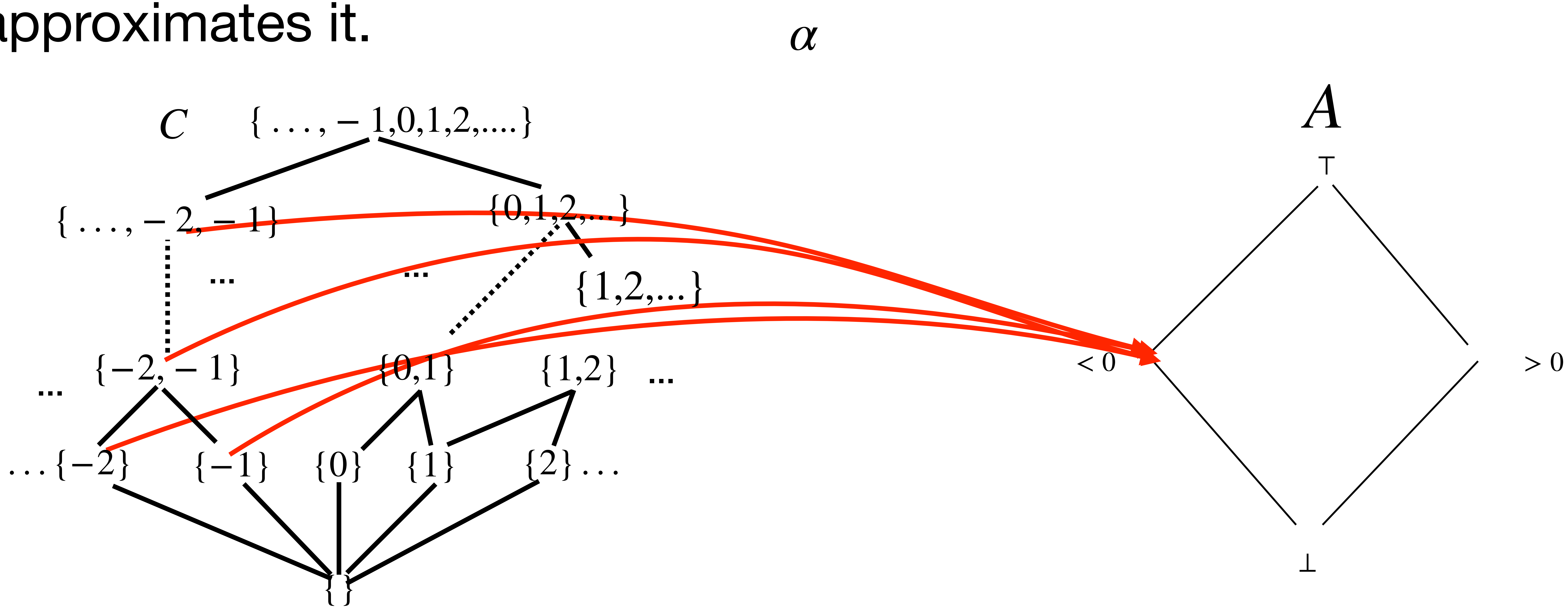
Abstraction function $\alpha : C \to A$ is a monotone function that maps concrete $c$ into the most precise abstract element that approximates it.



# Abstraction function

**Definition**

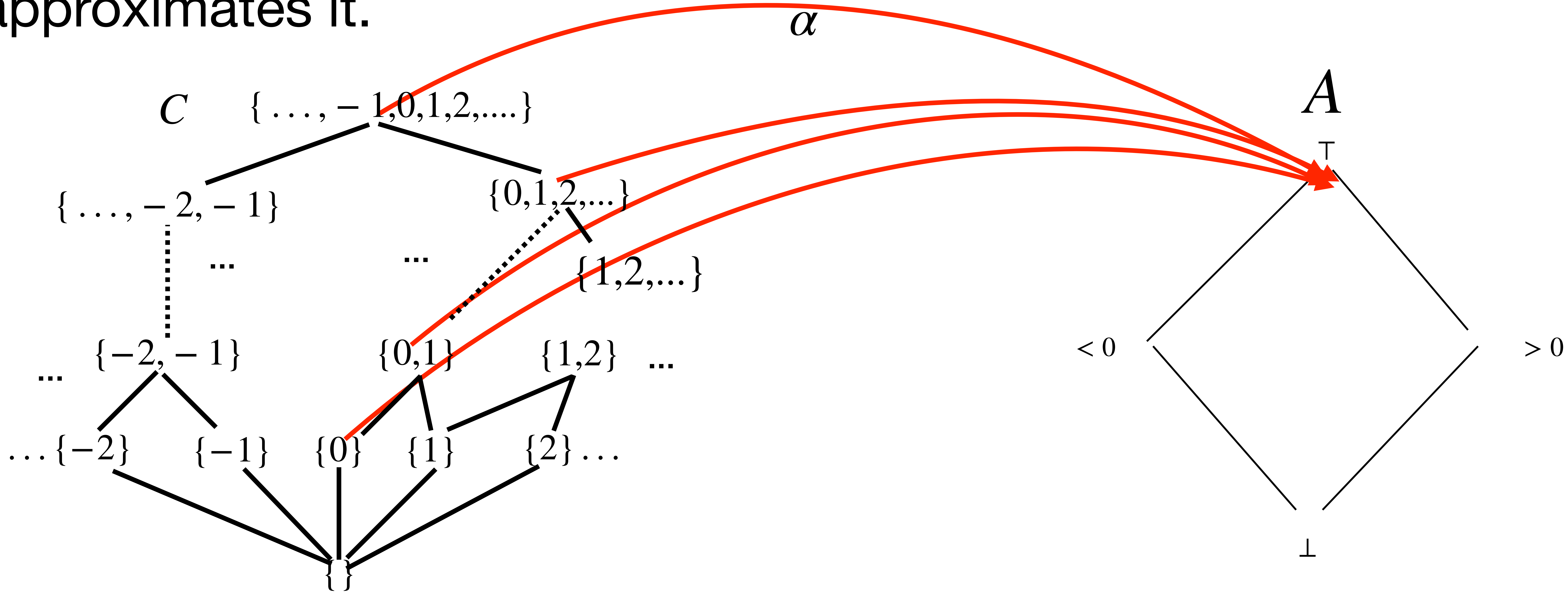
Abstraction function $\alpha : C \rightarrow A$ is a monotone function that maps concrete $c$ into the most precise abstract element that approximates it.



# Abstraction function

**Definition**

Abstraction function $\alpha : C \to A$ is a monotone function that maps concrete $c$ into the most precise abstract element that approximates it.

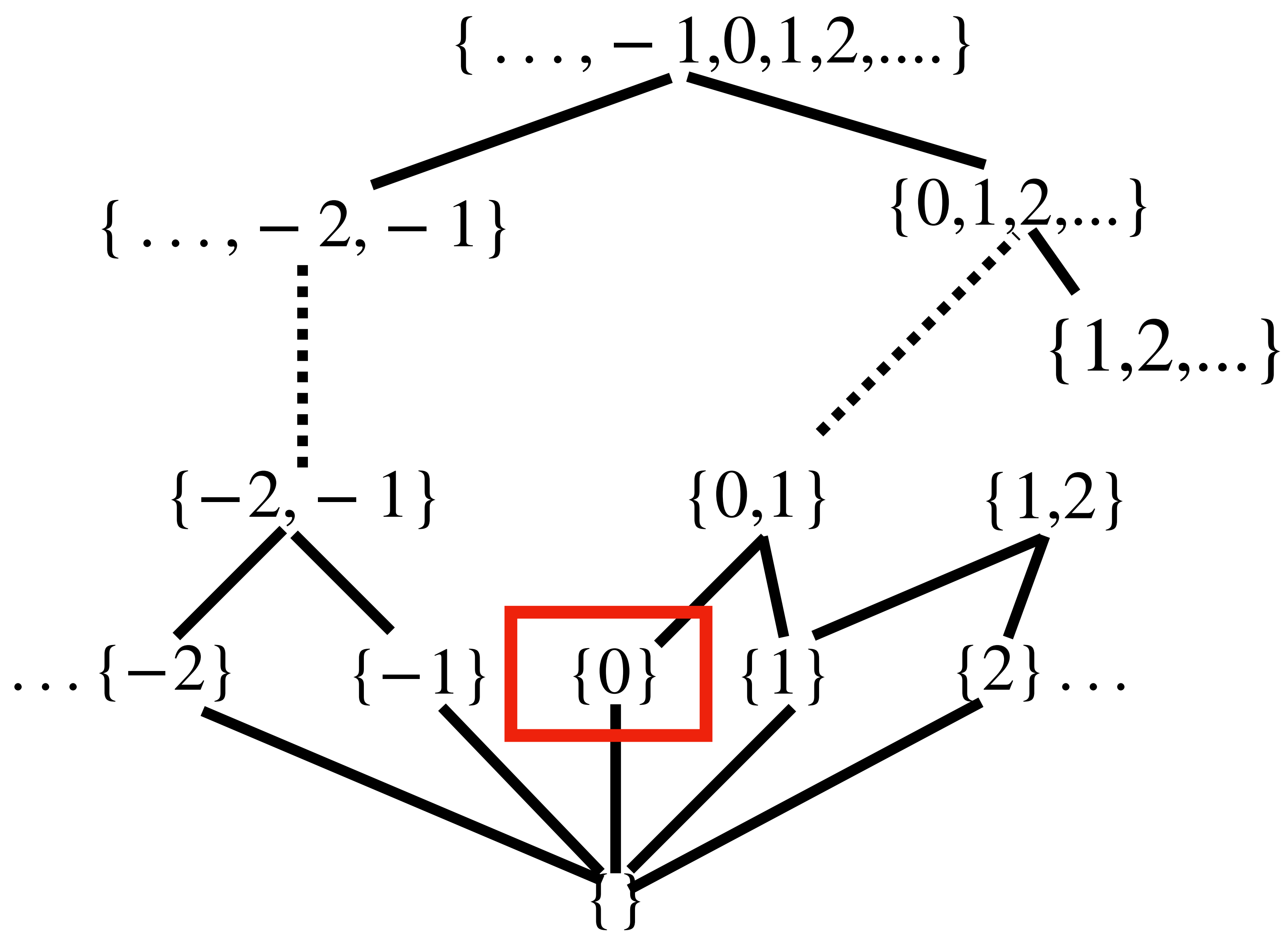


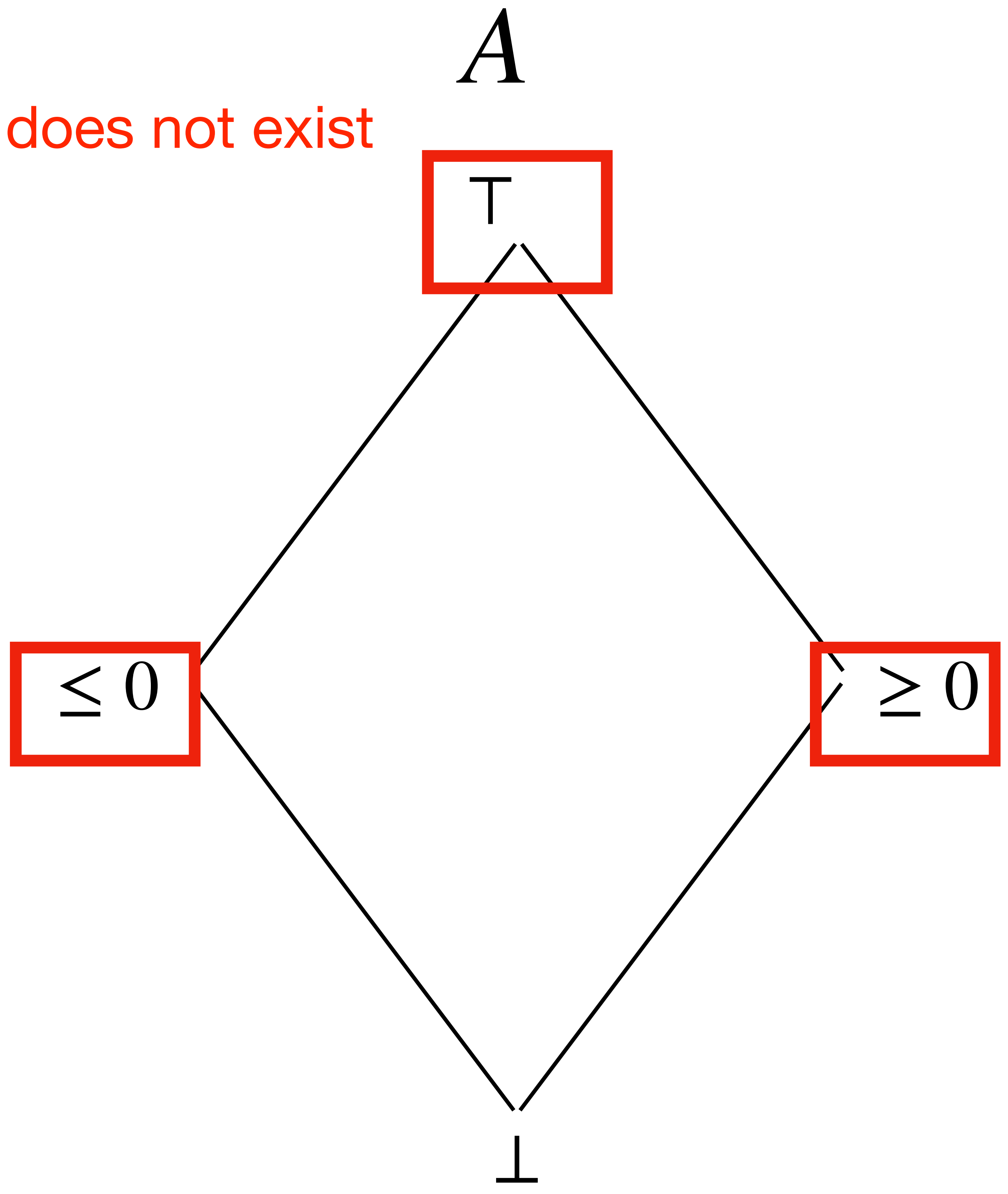
# Abstraction function

**Remark**

To design an abstraction function $\alpha : C \to A$ the abstract domain must be closed under meet.

In this abstract domain the most precise element that approximates $\{0\}$ does not exist

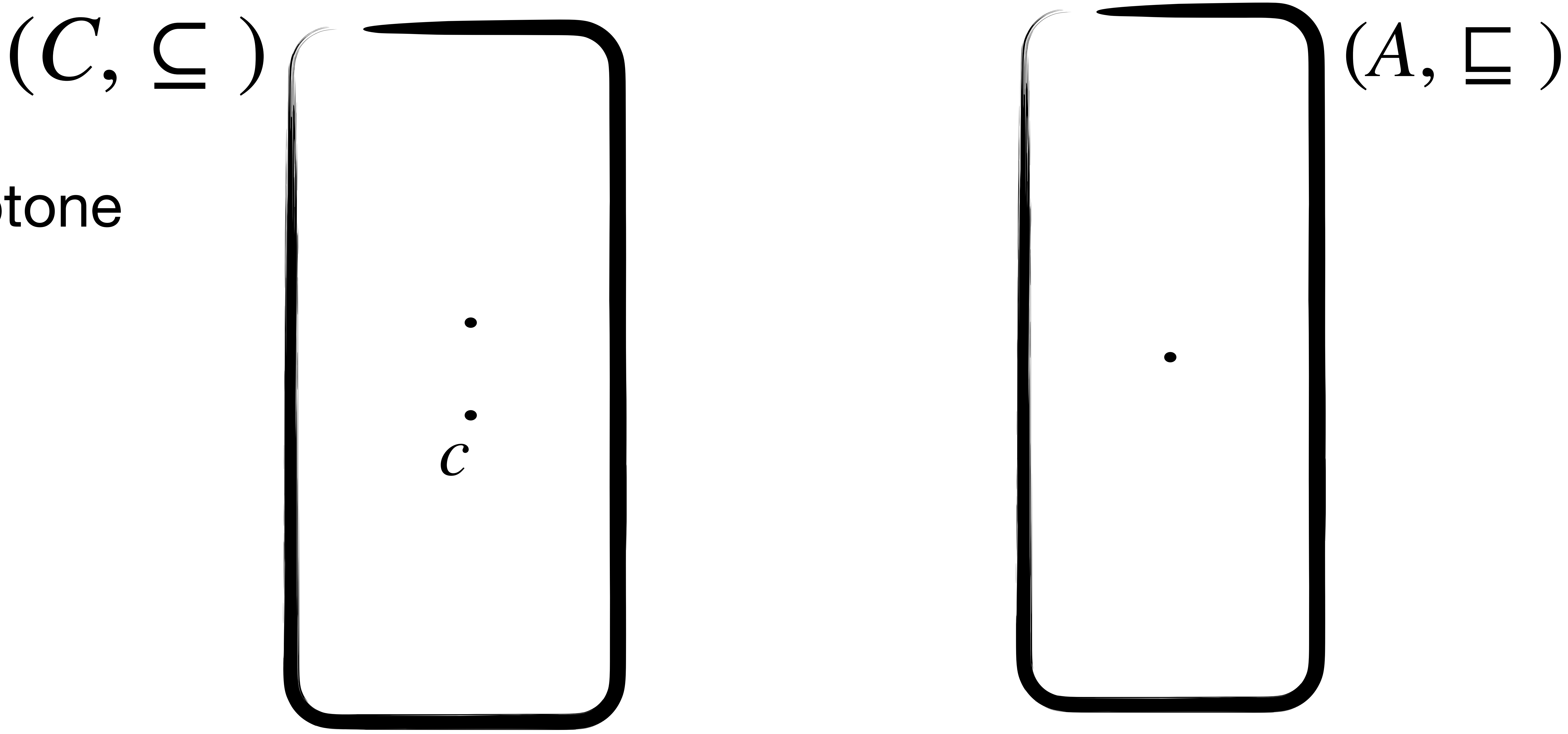


# Abstract Interpretation
# (AI)

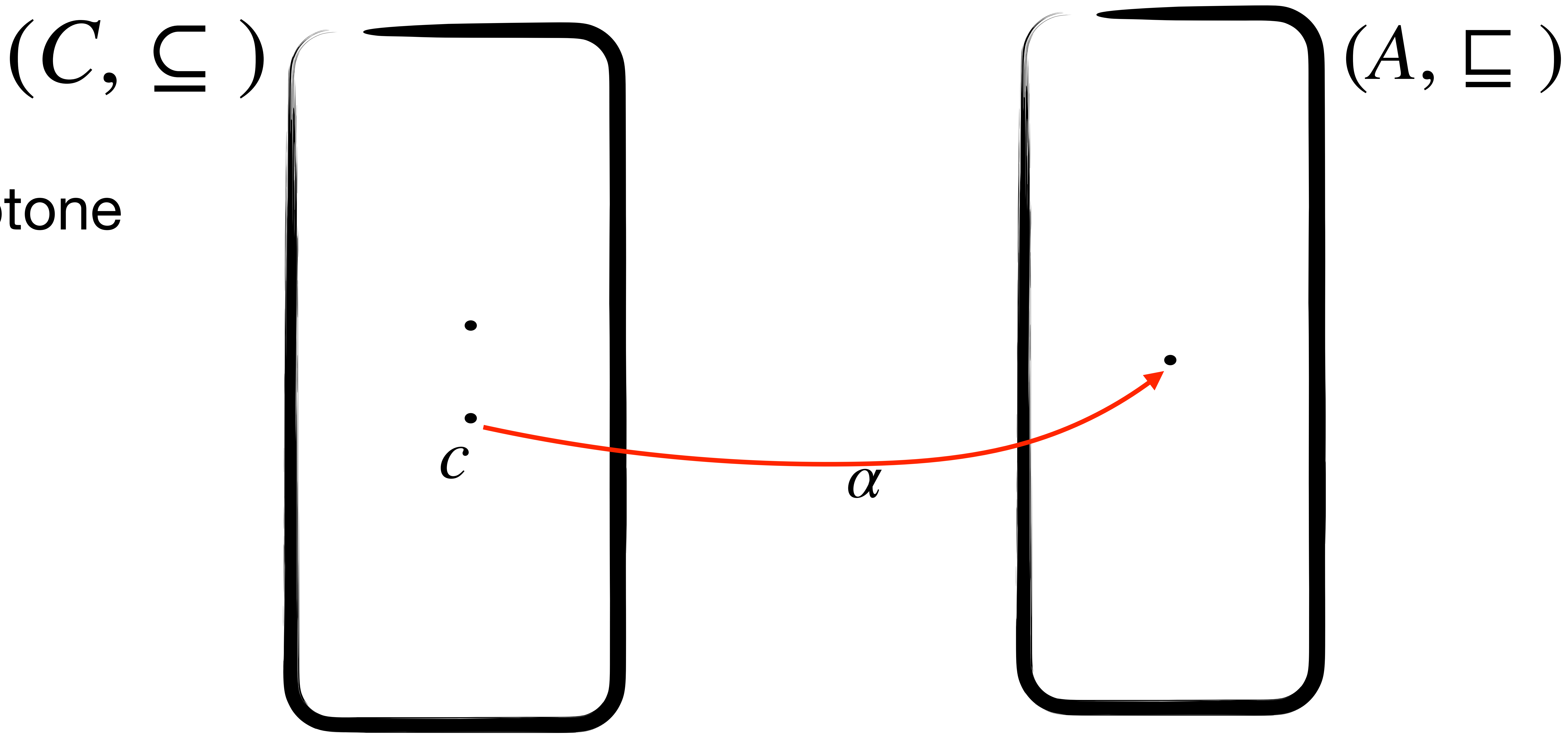
# Properties of Galois insertions

- $\alpha$ and $\gamma$ are monotone
- $c \subseteq \gamma(\alpha(c))$
- $\alpha(\gamma(a)) = a$



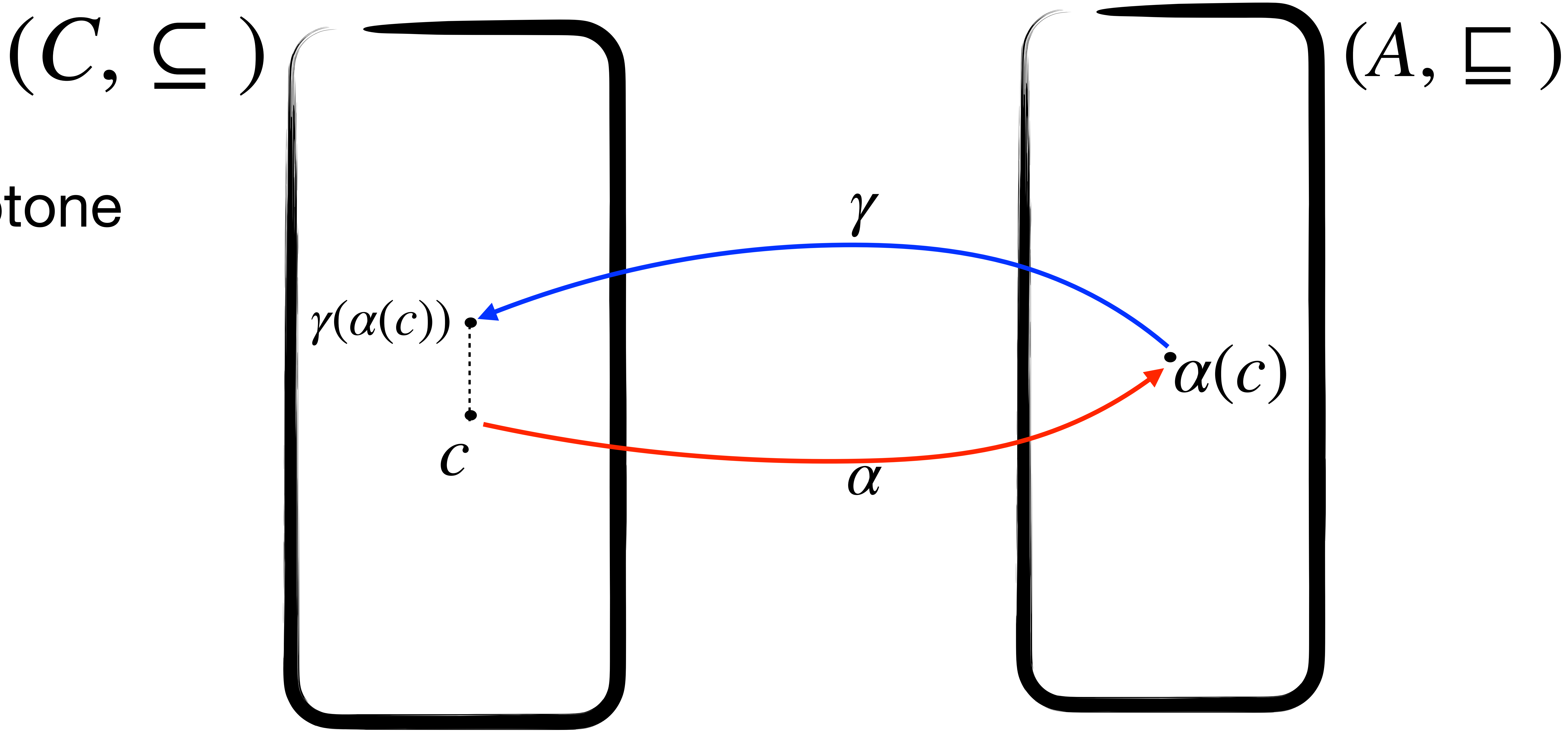
# Properties of Galois insertions

- $\alpha$ and $\gamma$ are monotone
- $c \subseteq \gamma(\alpha(c))$
- $\alpha(\gamma(a)) = a$



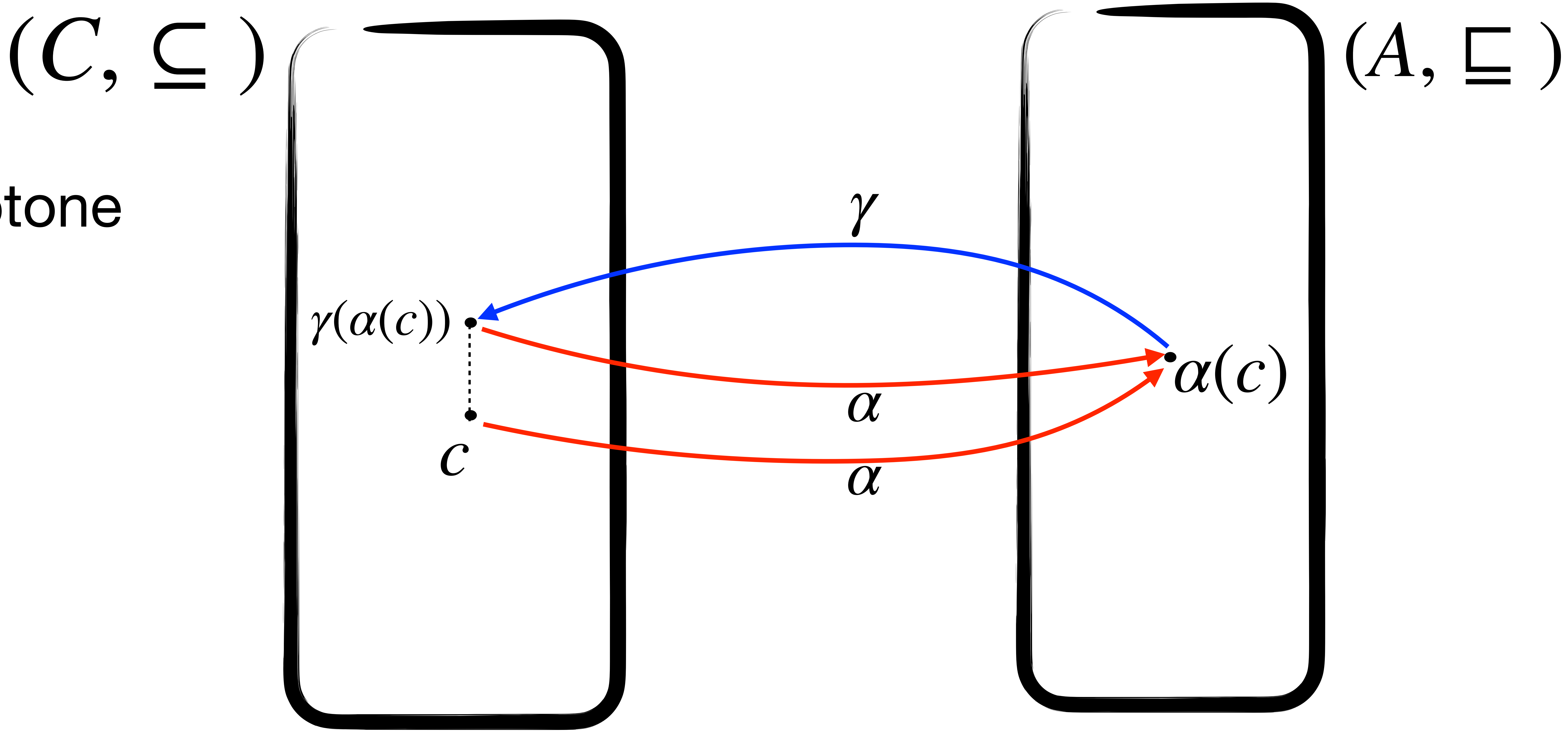
# Properties of Galois insertions

- $\alpha$ and $\gamma$ are monotone
- $c \subseteq \gamma(\alpha(c))$
- $\alpha(\gamma(a)) = a$

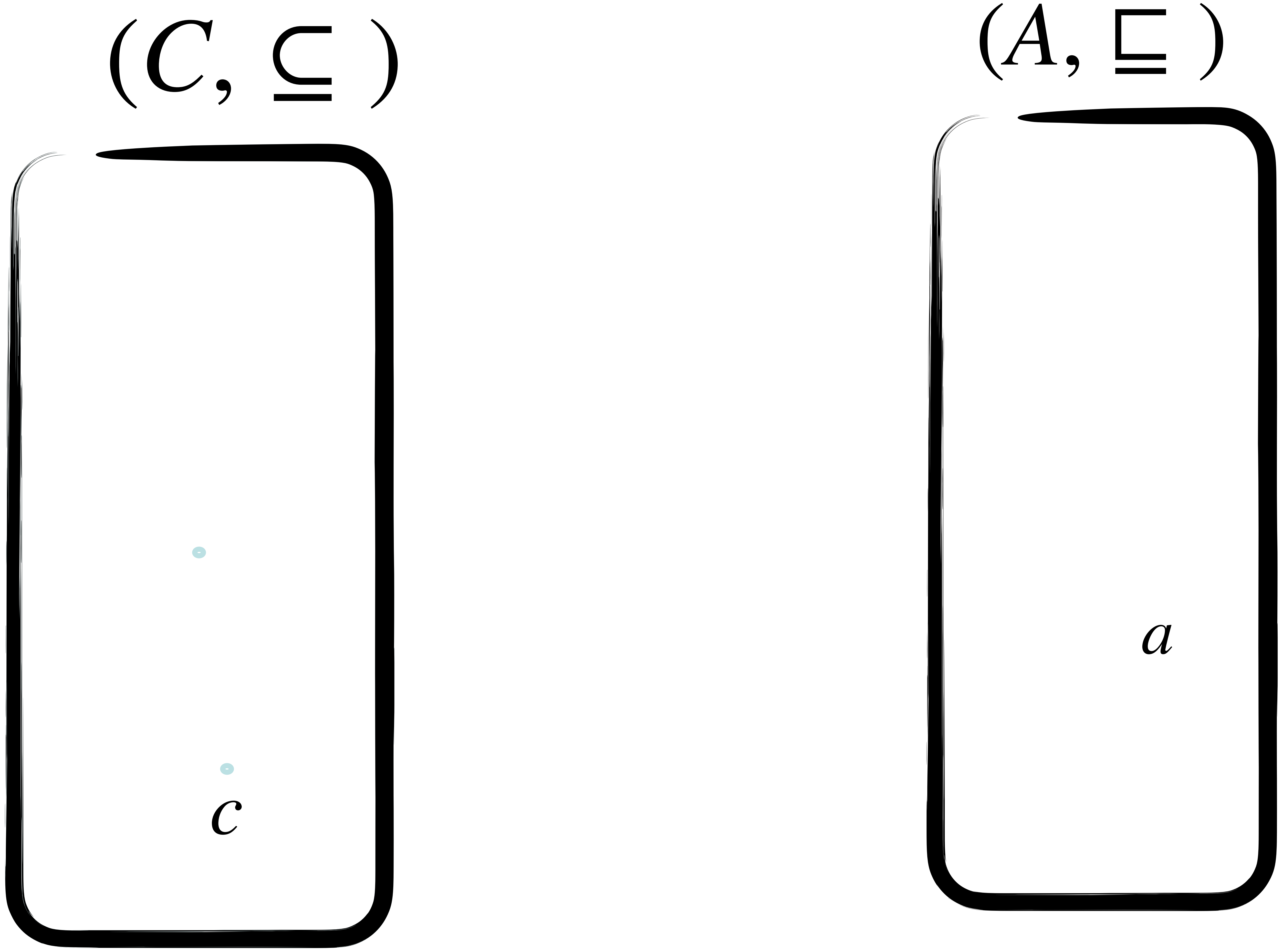


# Properties of Galois insertions

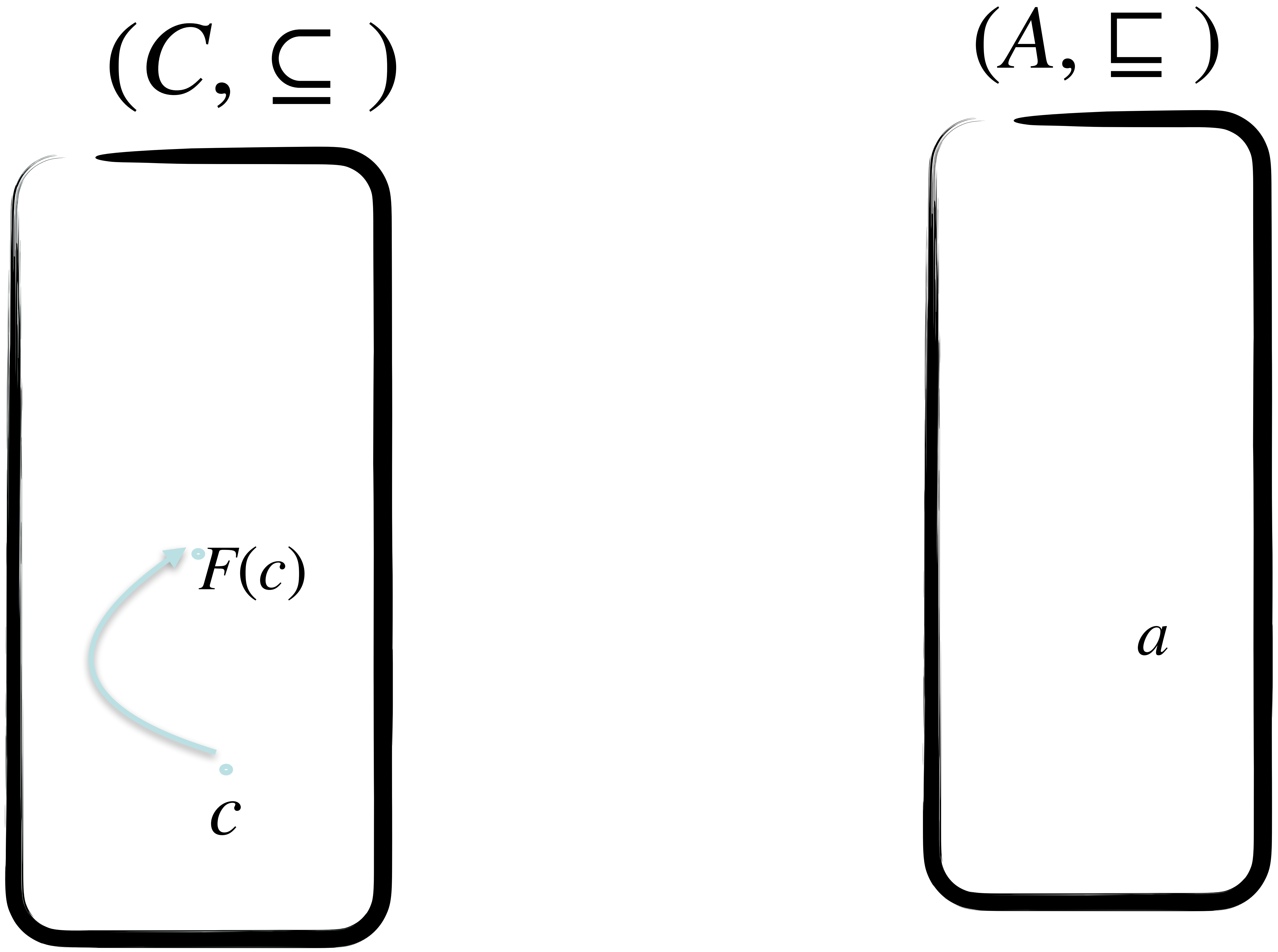
- $\alpha$ and $\gamma$ are monotone
- $c \subseteq \gamma(\alpha(c))$
- $\alpha(\gamma(a)) = a$



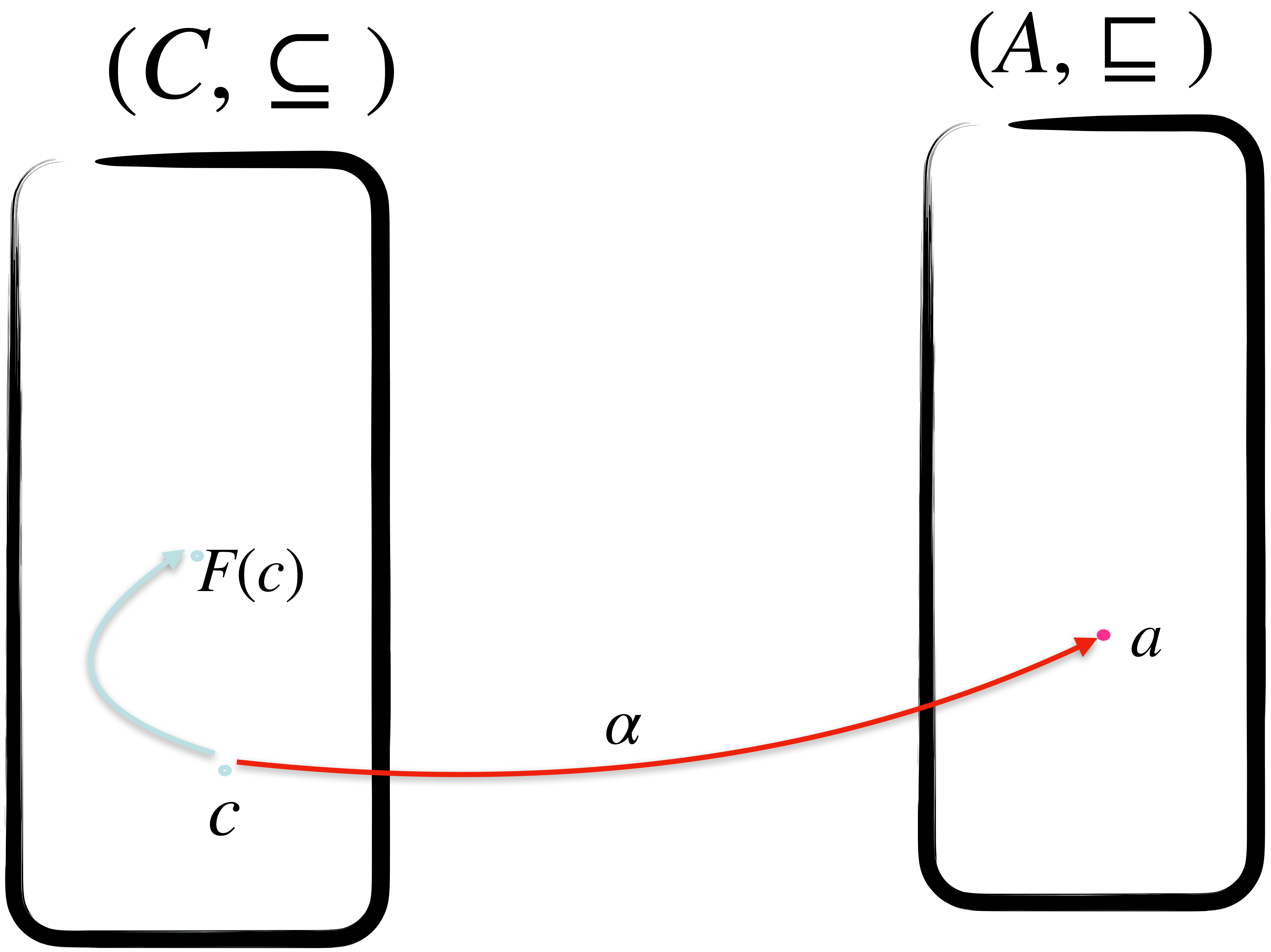
# Correct approximations



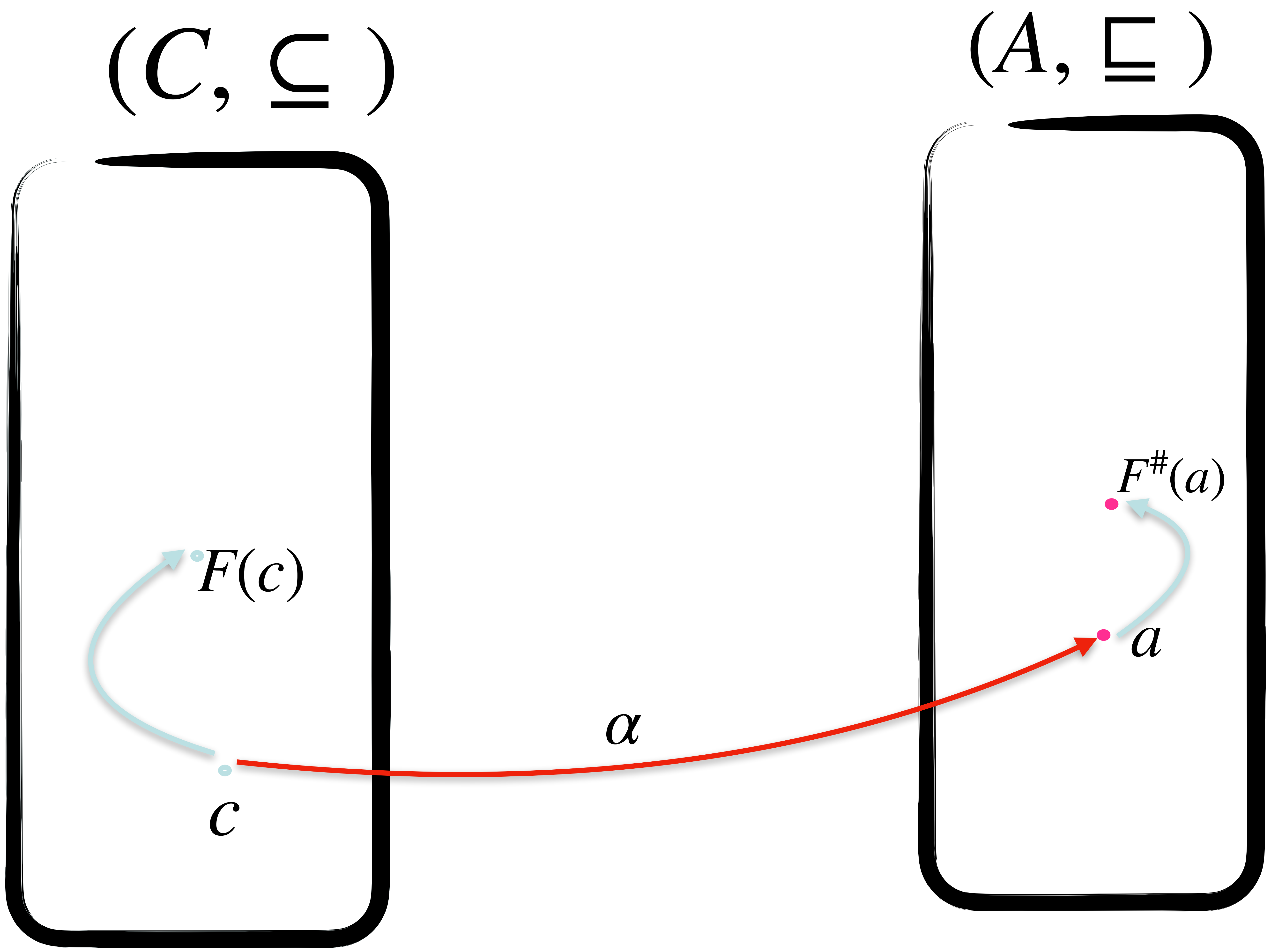
# Correct approximations



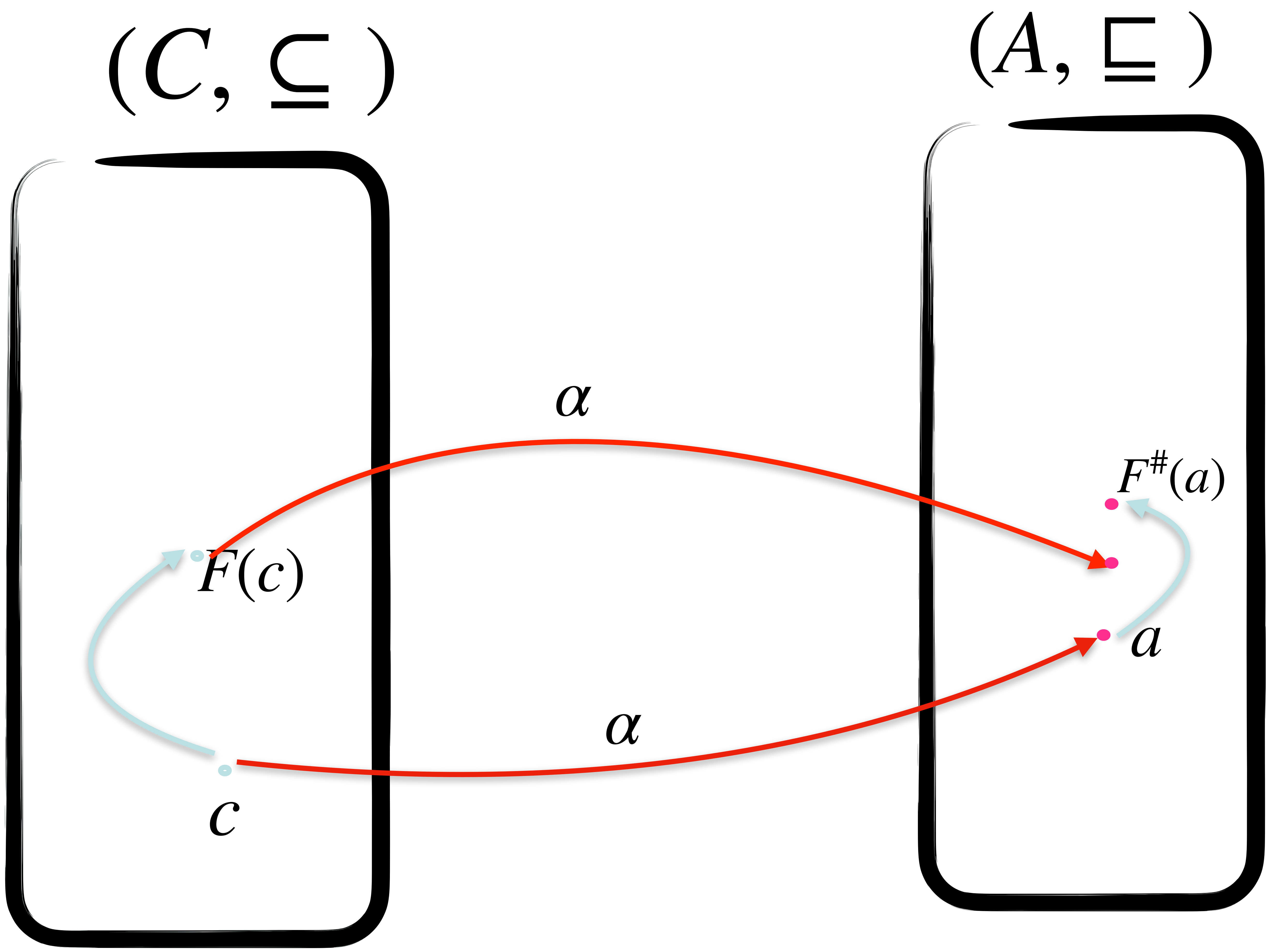
# Correct approximations



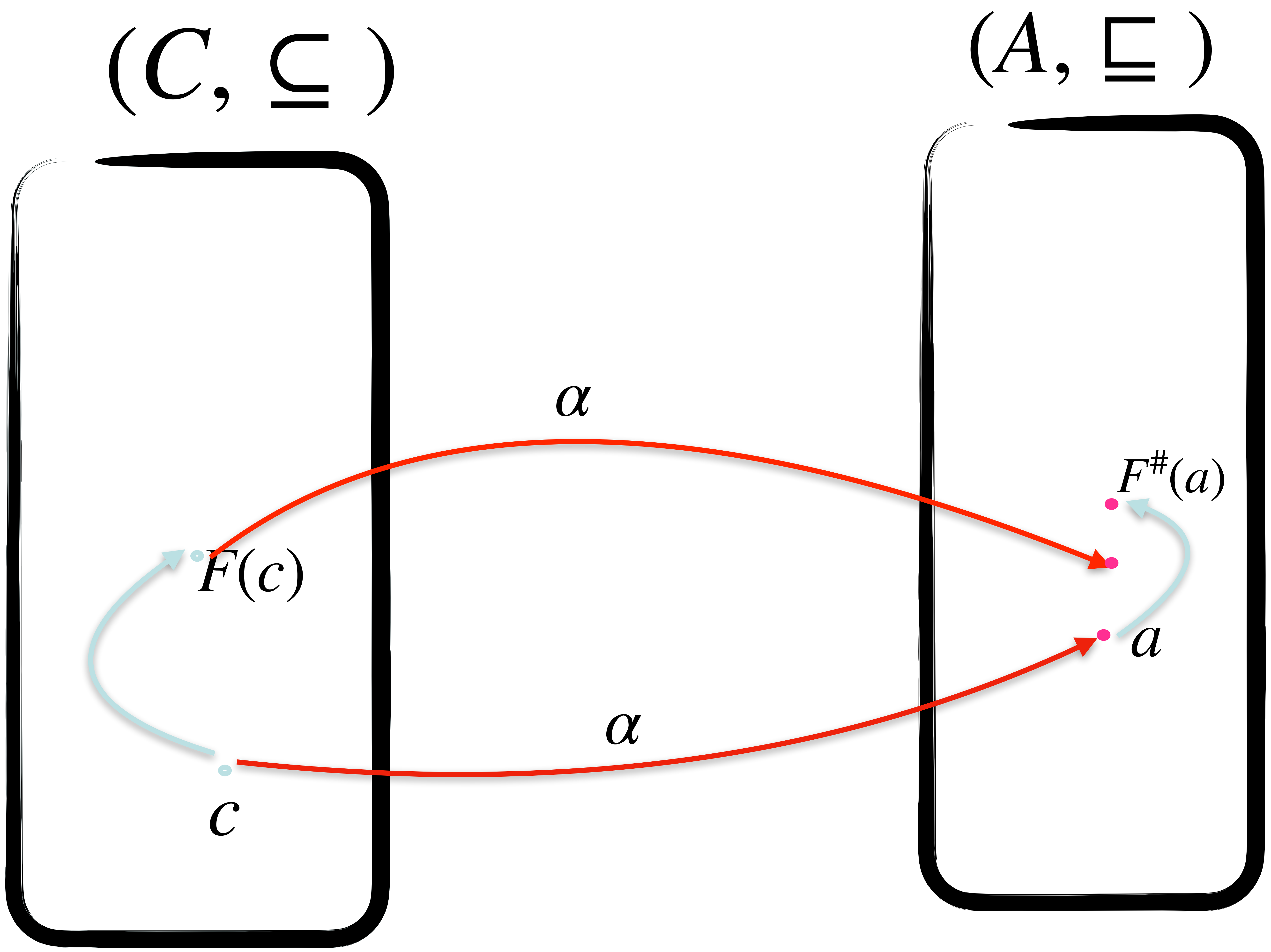
# Correct approximations



# Correct approximations

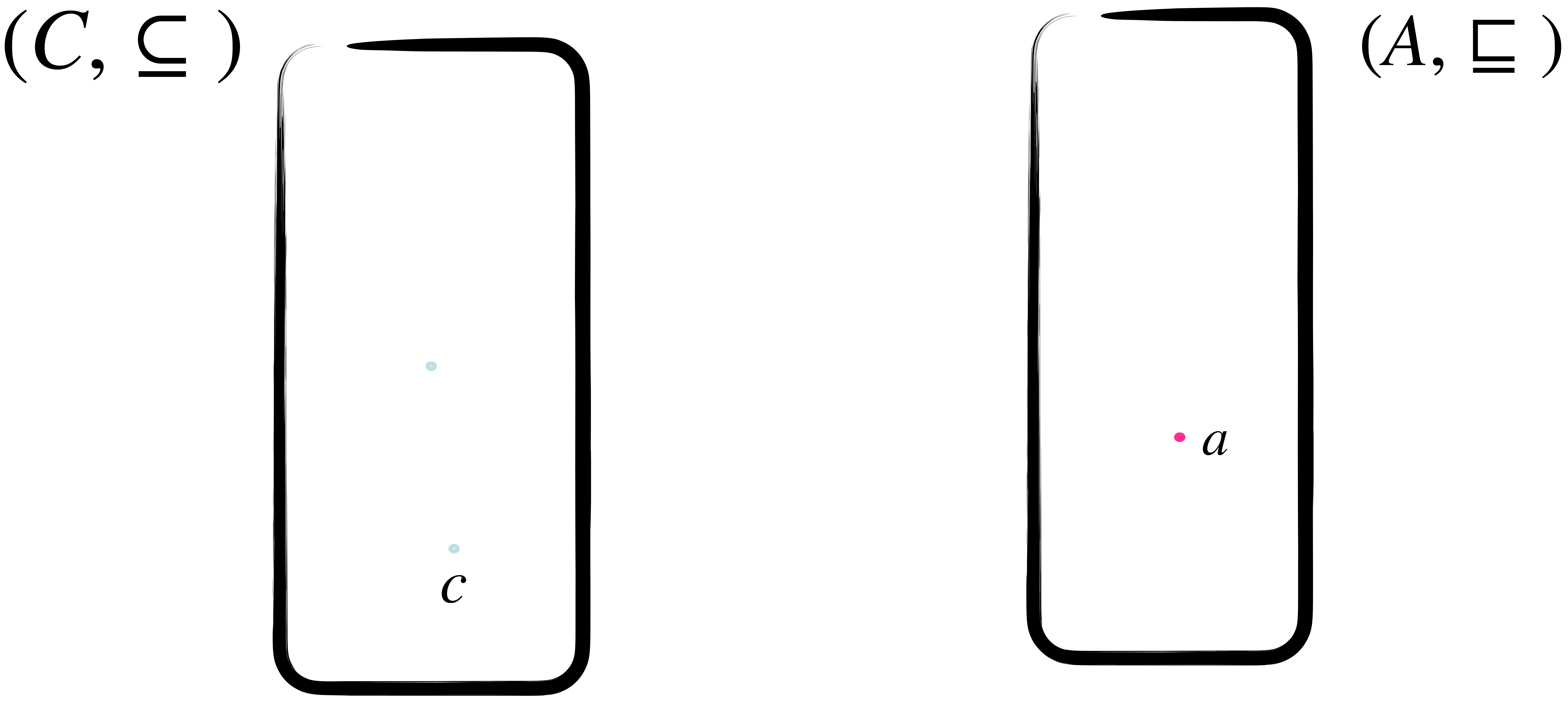


# Correct approximations

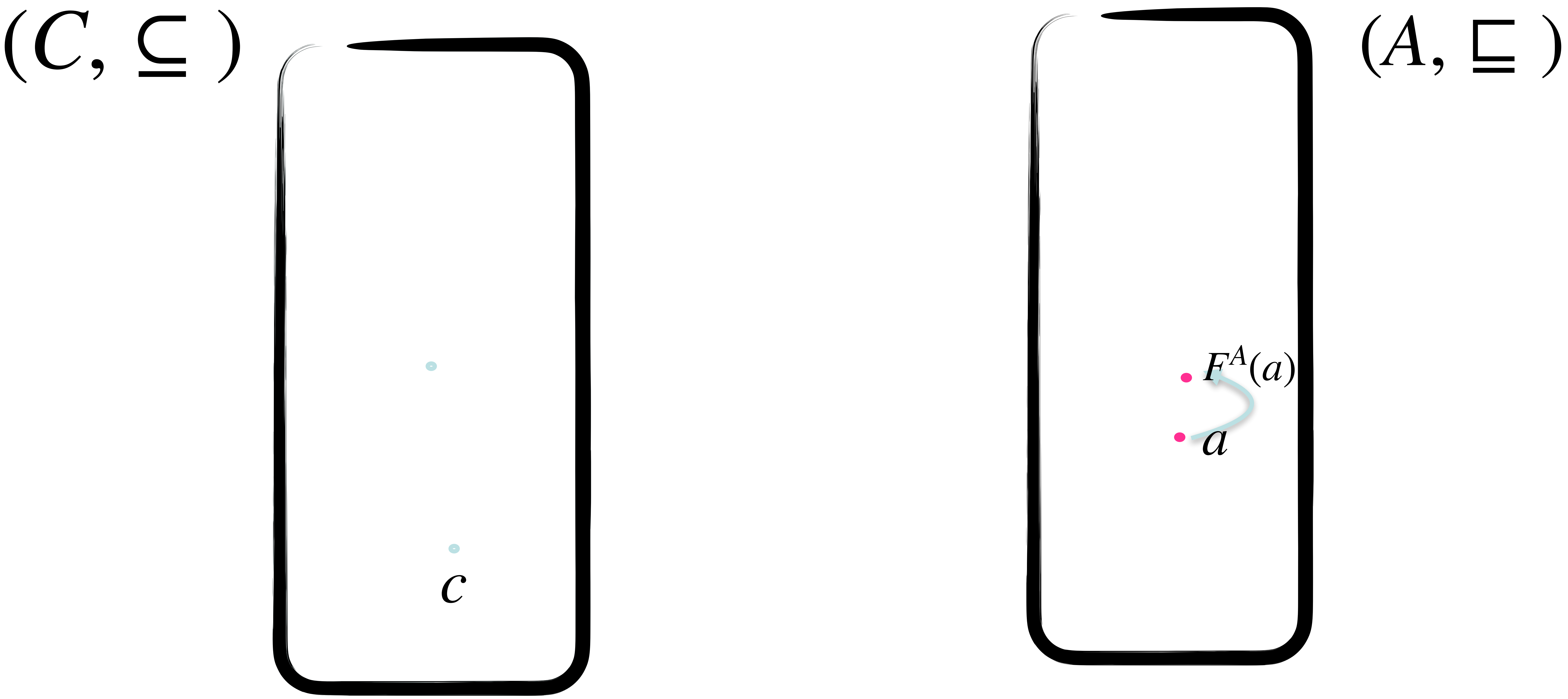



$F^{\#}\alpha \sqsupseteq \alpha F$

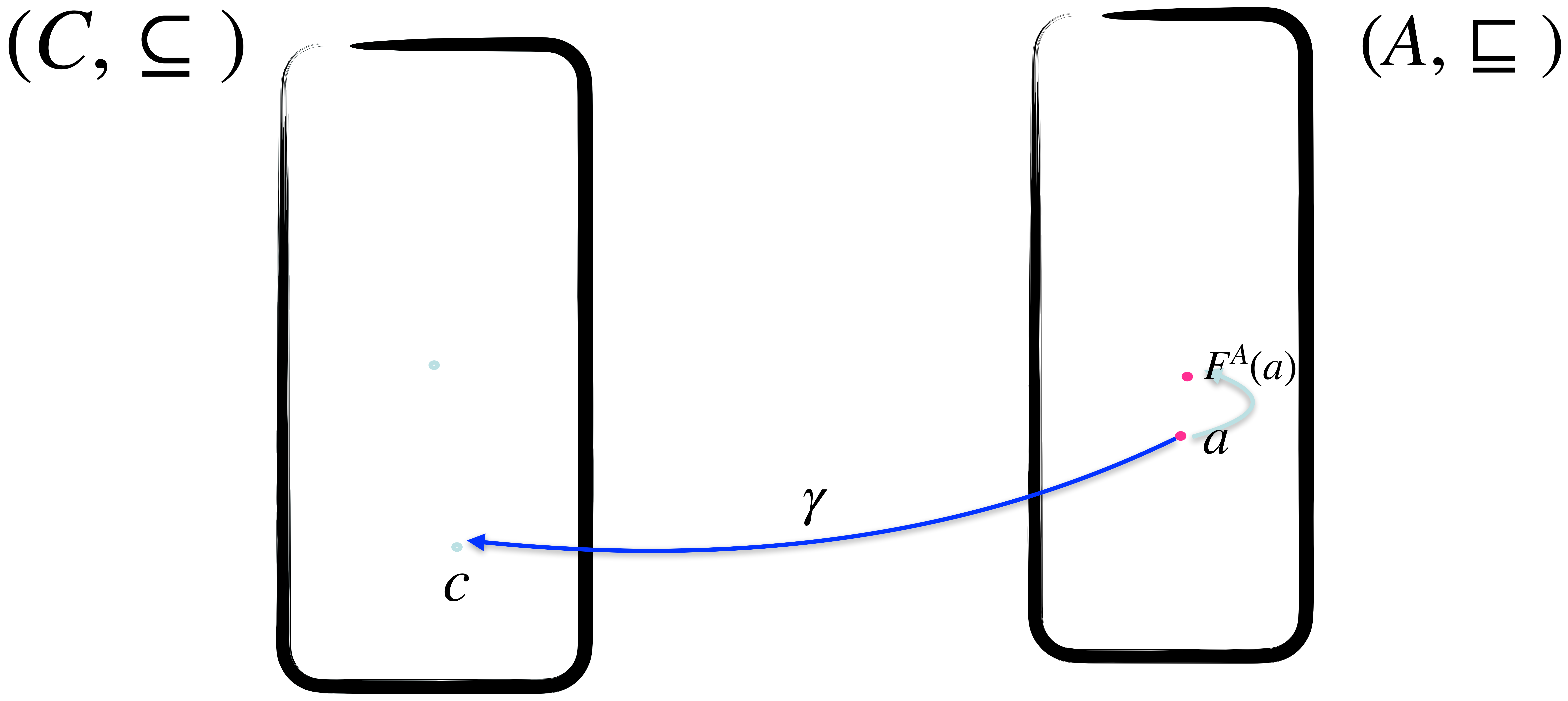
# Best correct approximation (bca)



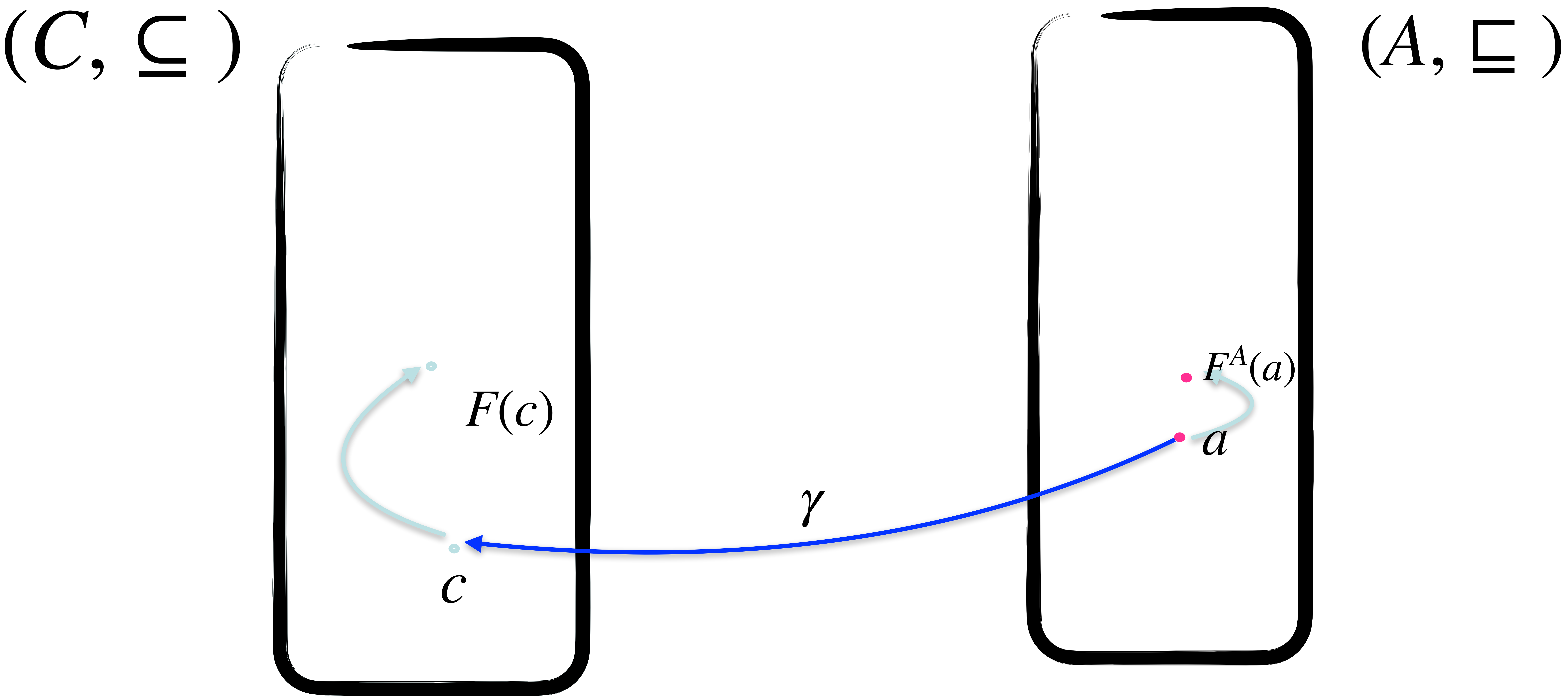
# Best correct approximation (bca)



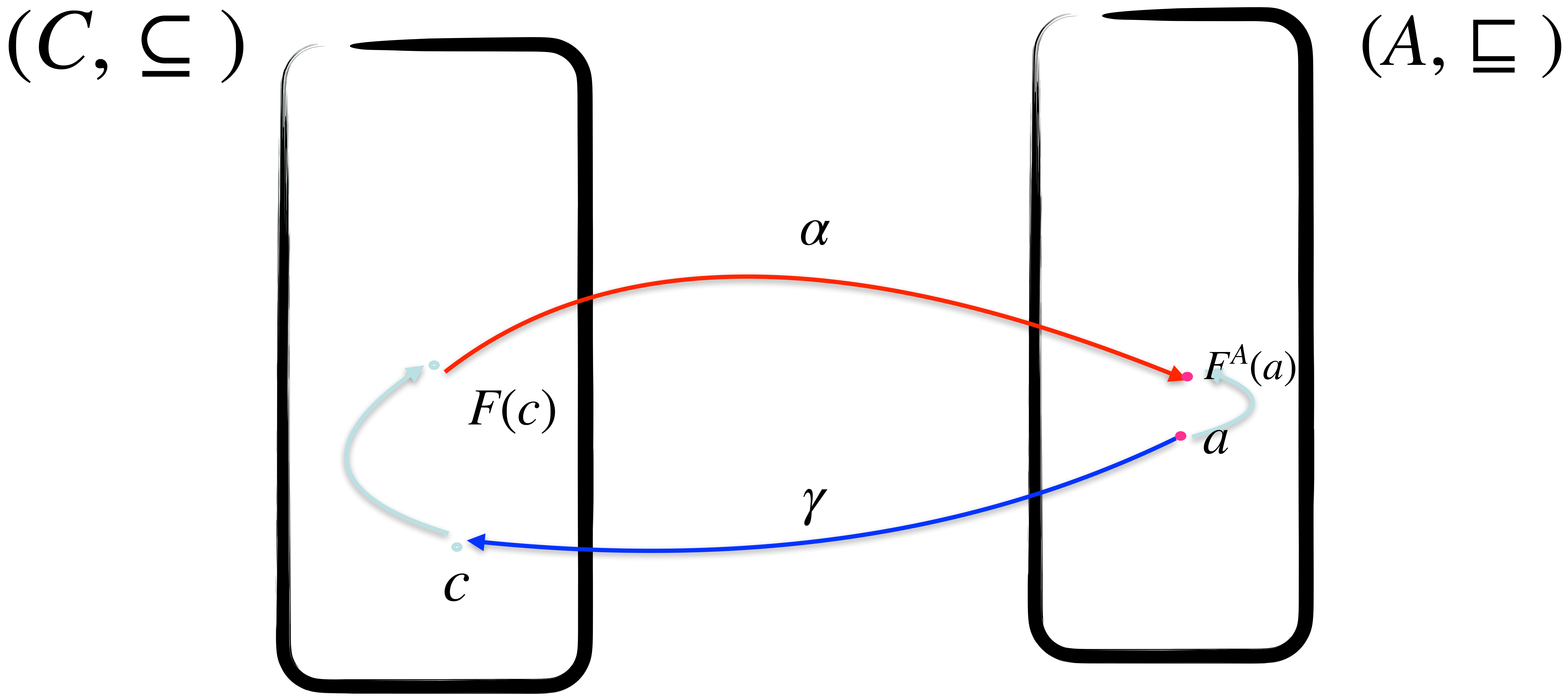
# Best correct approximation (bca)



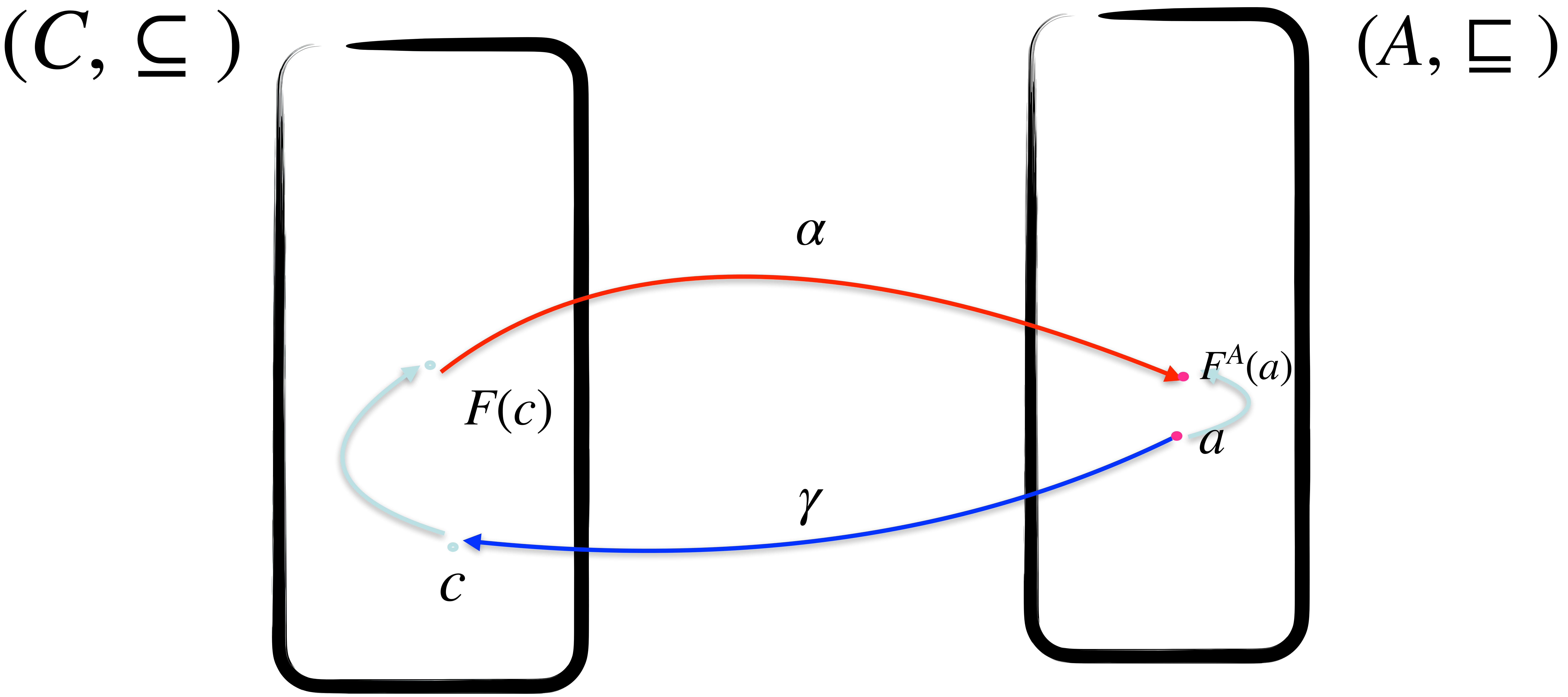
# Best correct approximation (bca)



# Best correct approximation (bca)



# Best correct approximation (bca)




$F^A \triangleq \alpha F \gamma$

# Abstract operations: +

$$(\wp(\mathbb{Z}), \subseteq)$$

$$+ : \wp(Z) \to \wp(Z)$$

$$\{3,5\} + \{-2,4\} = \{1,7,3,9\}$$

$\{1,2,...\}$ $\{\ldots, -2, -1\}$ $\top$

$> 0$ $< 0$ $> 0$

$$\{3\} + \{-2\} = \{1\}$$

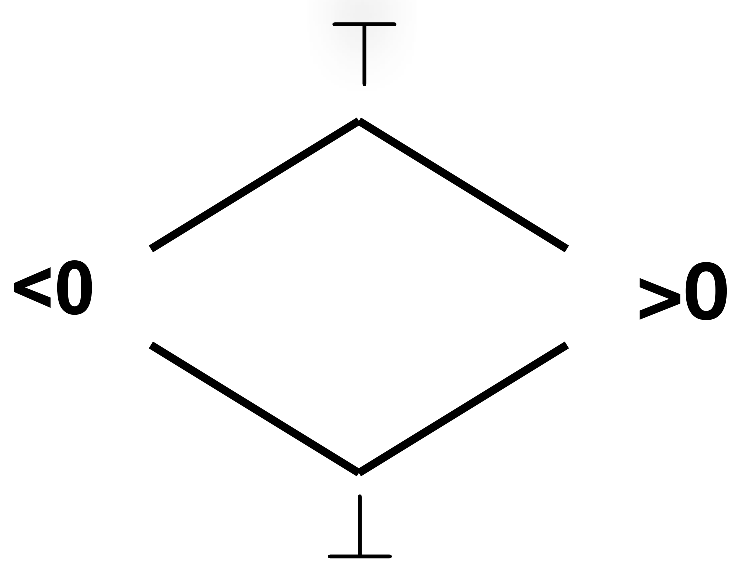
We lost precision

$> 0$ $> 0$ $> 0$

$\{1,2,...\}$ $\{1,2,...\}$ $\{2,3,...\}$

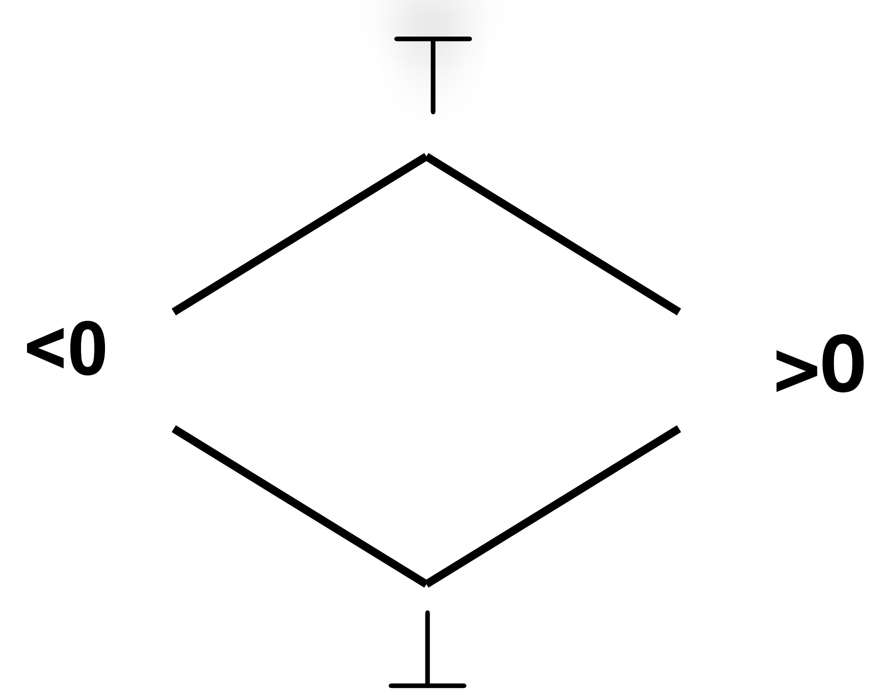
$$\{3\} + \{2\} = \{5\}$$

Precise result!




| $+^\#$ | ⊥ | <0 | >0 | ⊤ |
|---|---|---|---|---|
| ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |
| <0 | ⊥ | <0 | ⊤ | ⊤ |
| >0 | ⊥ | ⊤ | >0 | ⊤ |
| ⊤ | ⊥ | ⊤ | ⊤ | ⊤ |

# Abstract operations: $\times$




$$(\wp(\mathbb{Z}), \subseteq)$$

$$\times : \wp(Z) \rightarrow \wp(Z)$$

$$\{3,5\} \times \{-2,4\} = \{-6, 12, -10, 20\}$$

$> 0$ $\quad$ $< 0$ $\quad$ $< 0$ ✓

$\{1,2,\ldots\}$ $\quad$ $\{\ldots, -2, -1\}$ $\quad$ $\{\ldots, -2, -1\}$

$$\{3\} \times \{-2\} = \{-6\}$$

Precise result!

$> 0$ $\quad$ $> 0$ $\quad$ $> 0$ ✓

$\{1,2,\ldots\}$ $\quad$ $\{1,2,\ldots\}$ $\quad$ $\{1,2,\ldots\}$

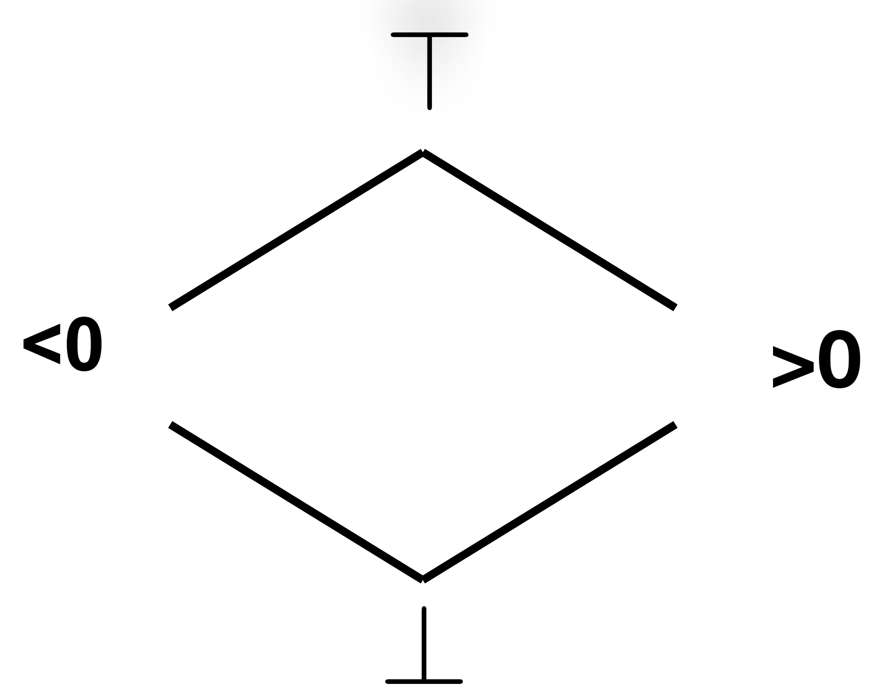
$$\{3\} \times \{2\} = \{6\}$$

Precise result!

| $\times^{\#}$ | ⊥ | <0 | >0 | T |
|---|---|---|---|---|
| ⊥ | ⊥ | ⊥ | ⊥ | ⊥ |
| <0 | ⊥ | >0 | <0 | T |
| >0 | ⊥ | <0 | >0 | T |
| T | ⊥ | T | T | T |

# Correctness

The abstract operations $+^{\#}$ and $\times^{\#}$ are correct on the domain Sign:



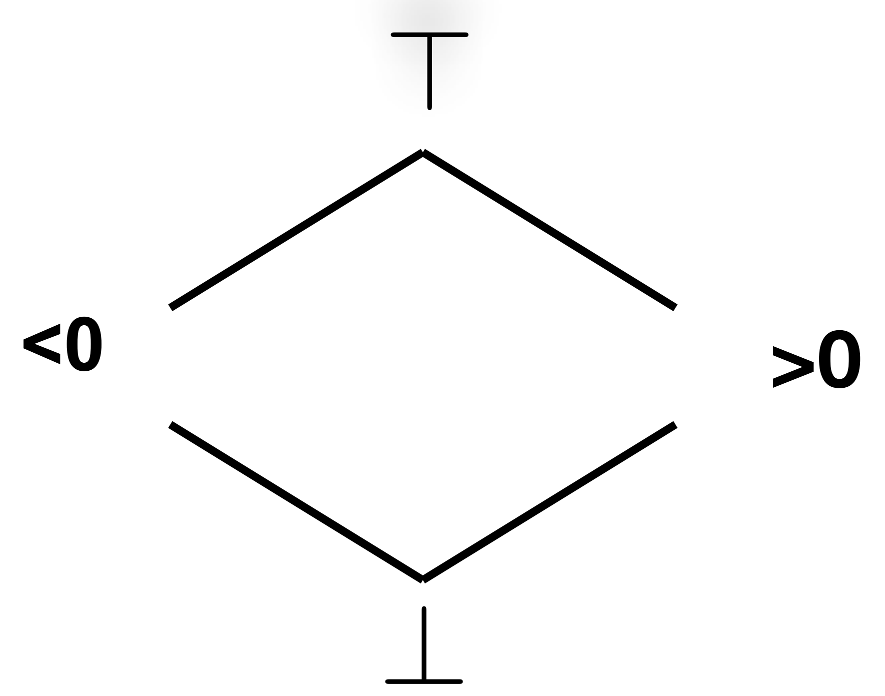

$$\forall n, m \in C \,.\, \alpha(n) +^{\#} \alpha(m) \sqsupseteq \alpha(n+m)$$

Remember $F^{\#}$ is correct on an abstract domain A
whenever it returns an approximation of the result of the concrete computation:

$$F^{\#}\alpha \sqsupseteq \alpha F$$

# Completeness

The abstract operation $\times^{\#}$ has a very nice property on the domain Sign:




$$\forall n, m \in C \,.\, \alpha(n) \times^{\#} \alpha(m) \overset{\not\sqsupset}{=} \alpha(n \times m)$$

$F^{\#}$ is complete on an abstract domain A whenever it also holds:

$$F^{\#}\alpha = \alpha F$$

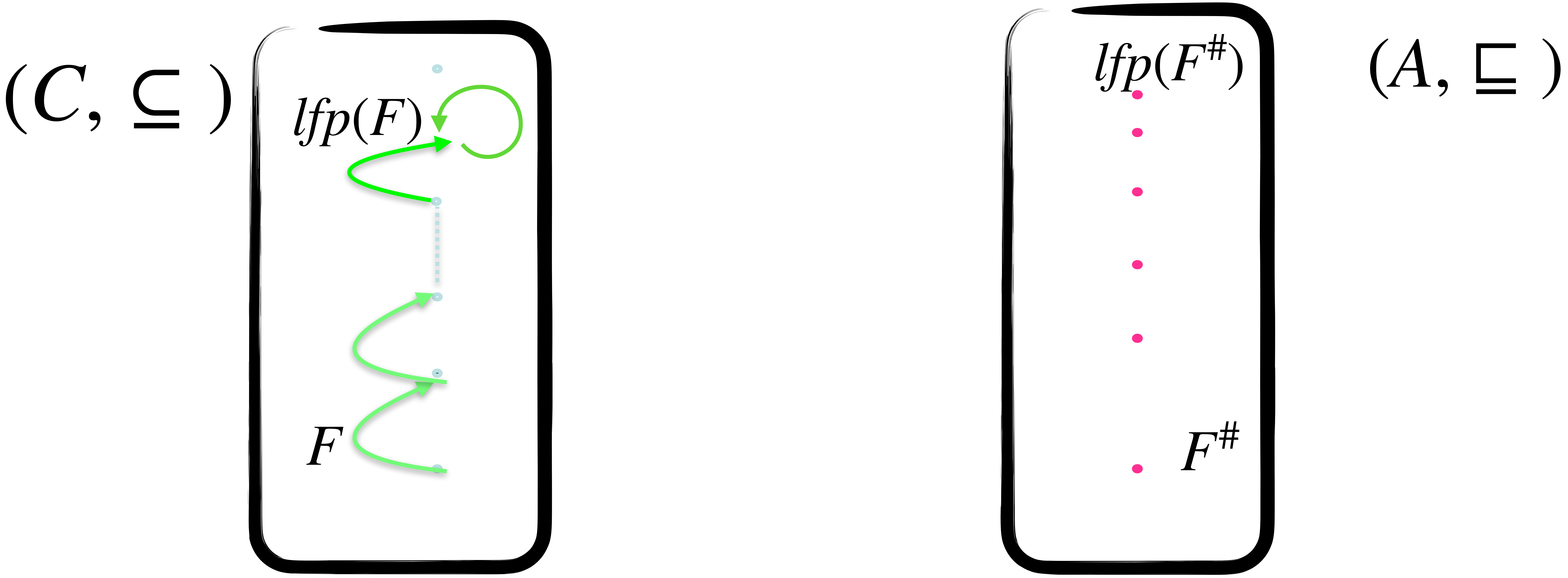
# Completeness and bcas

$F^{\#}$ is complete $\implies$ $F^{\#} = F^{A}$

$$\alpha F = F^{\#}\alpha \implies F^{A} = \alpha F \gamma = F^{\#}\alpha\gamma = F^{\#}$$

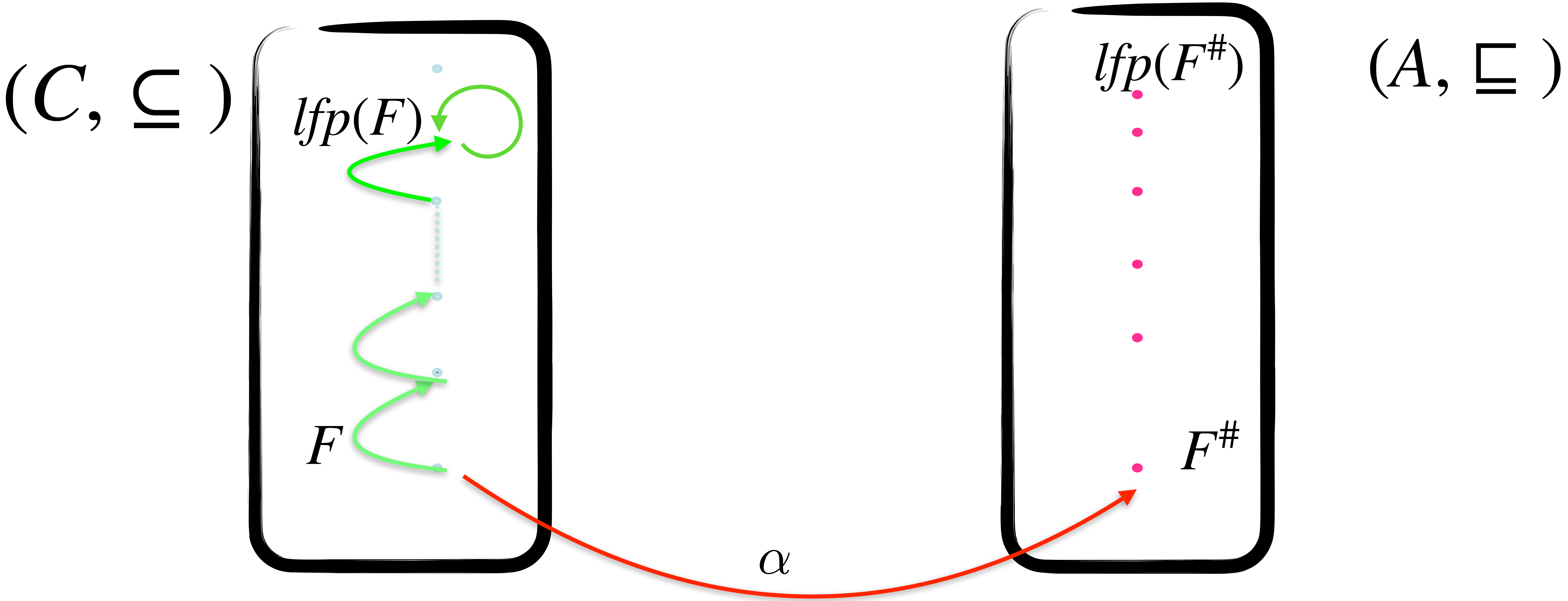
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



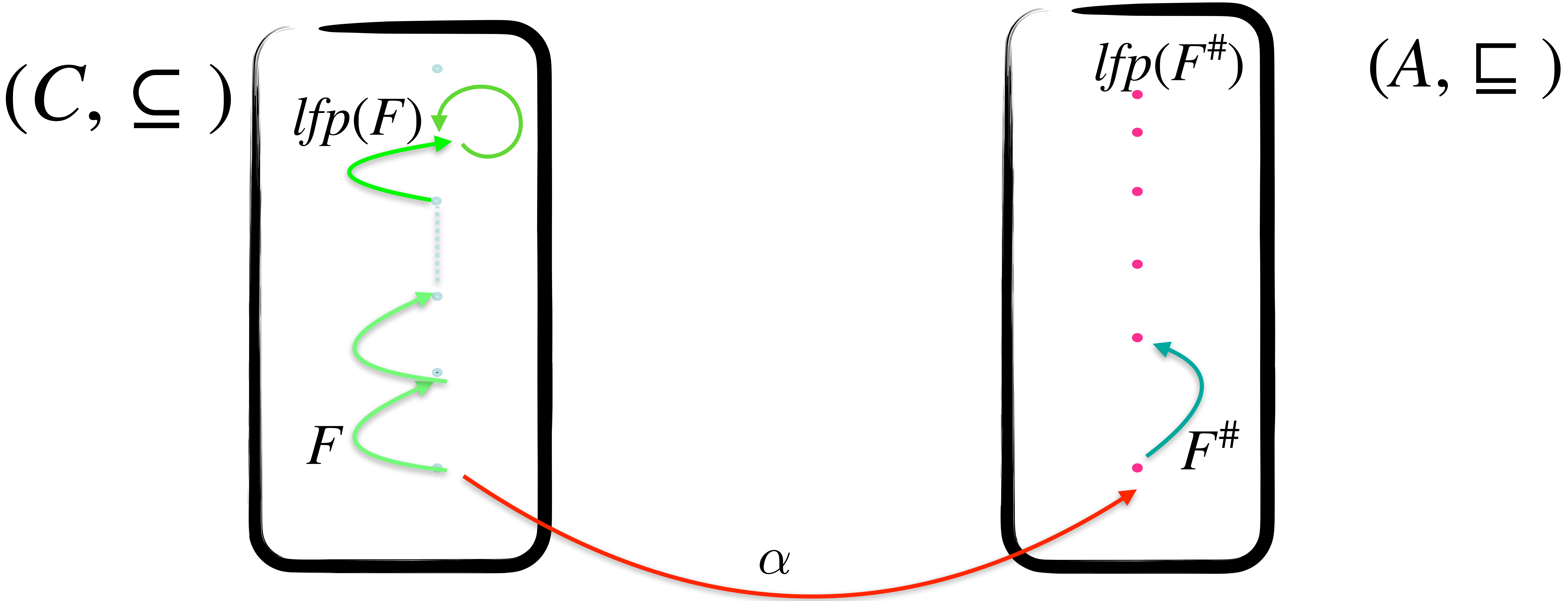
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



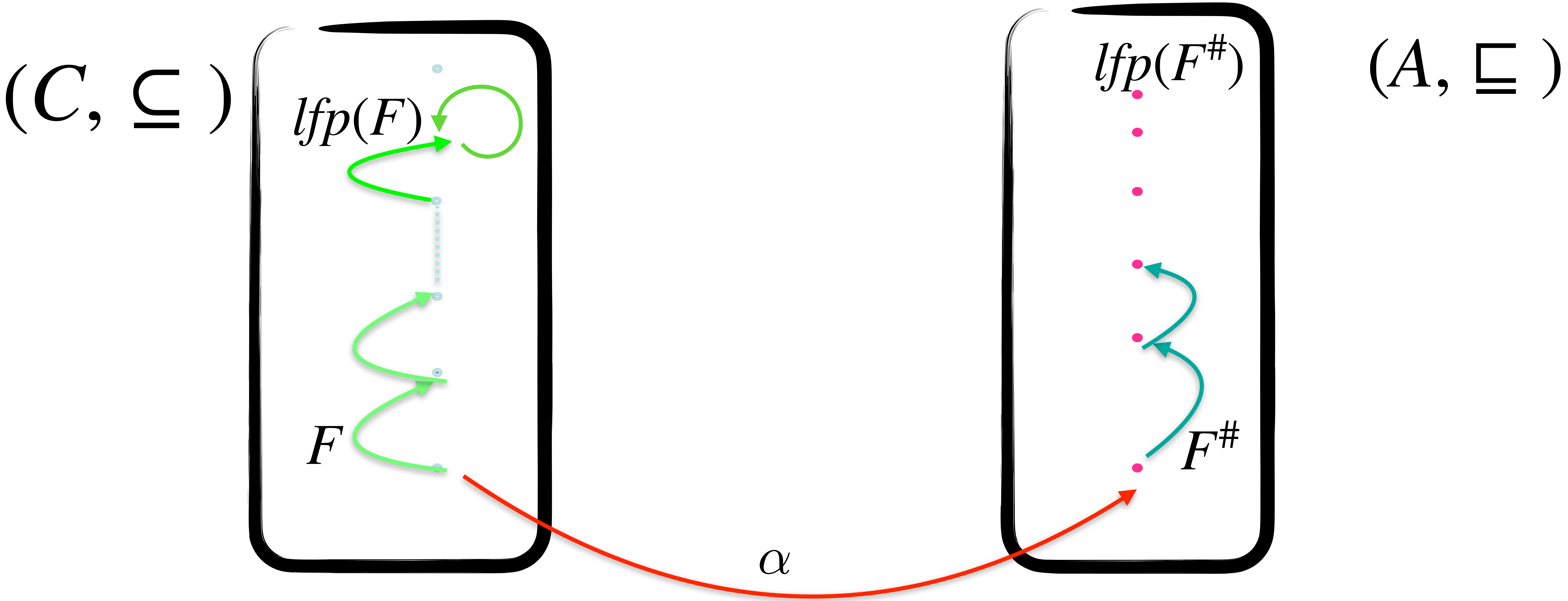
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



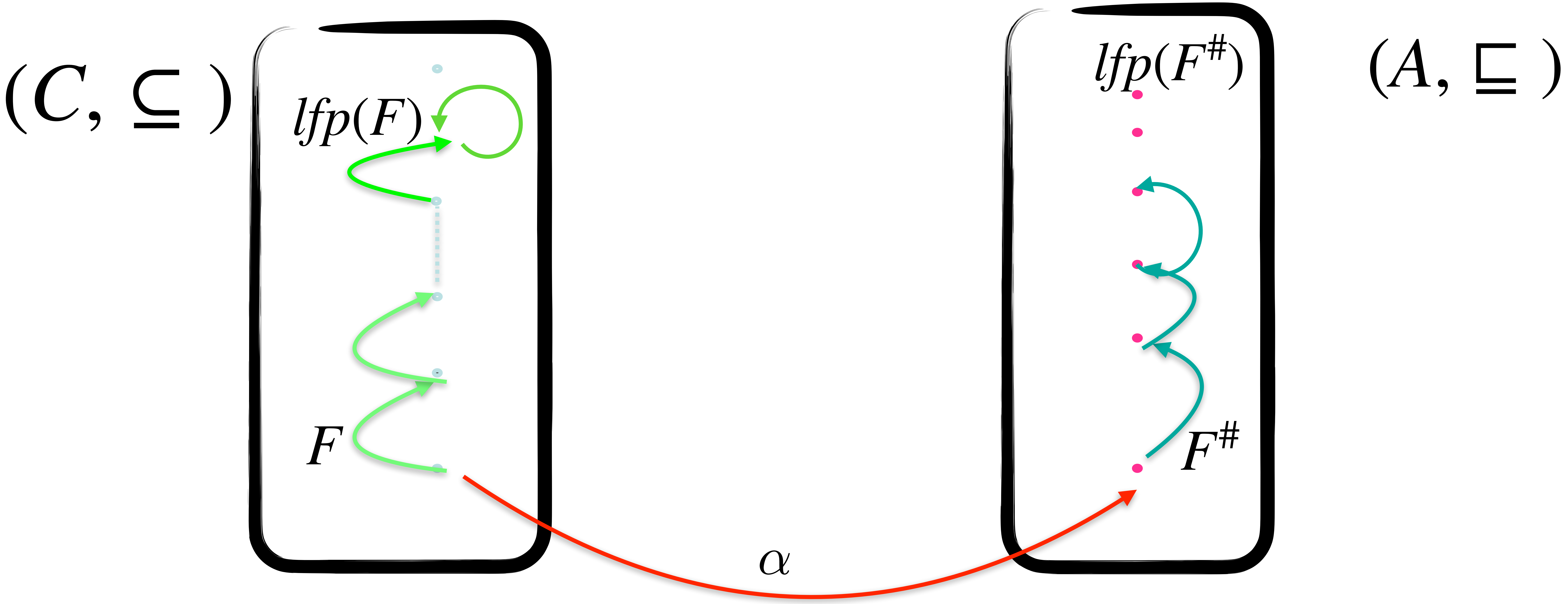
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



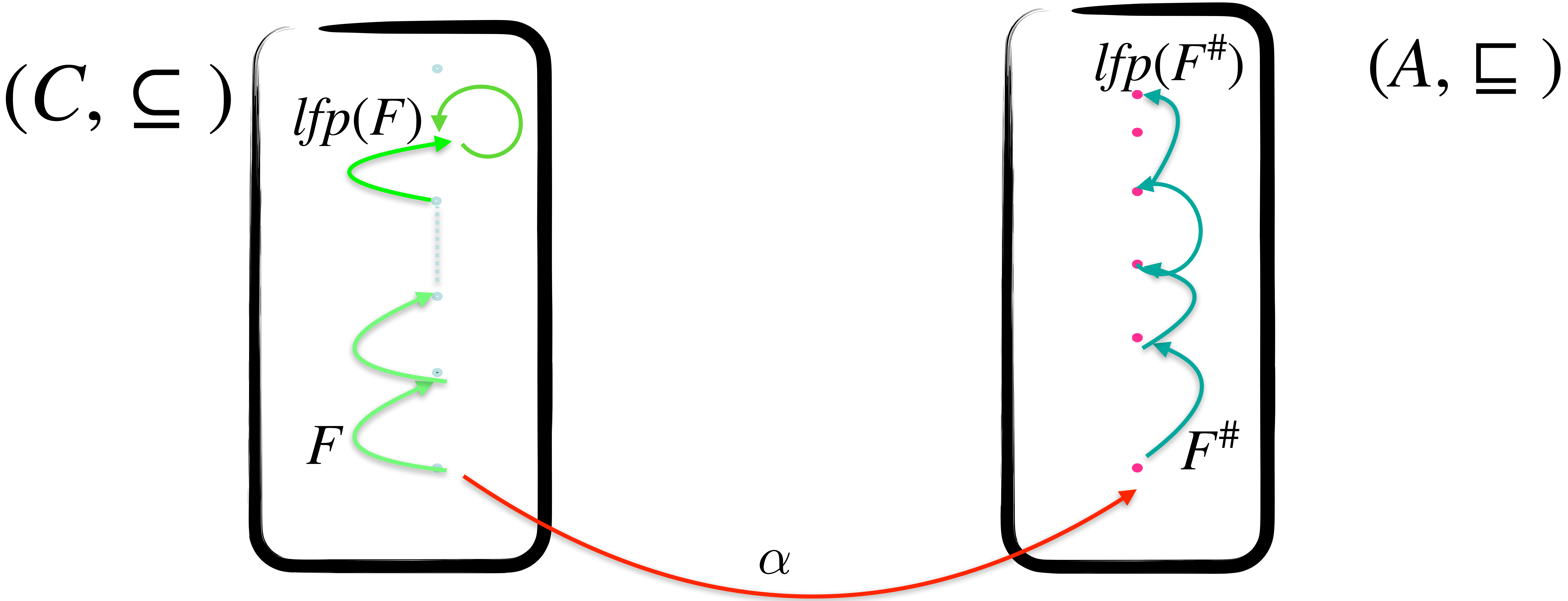
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



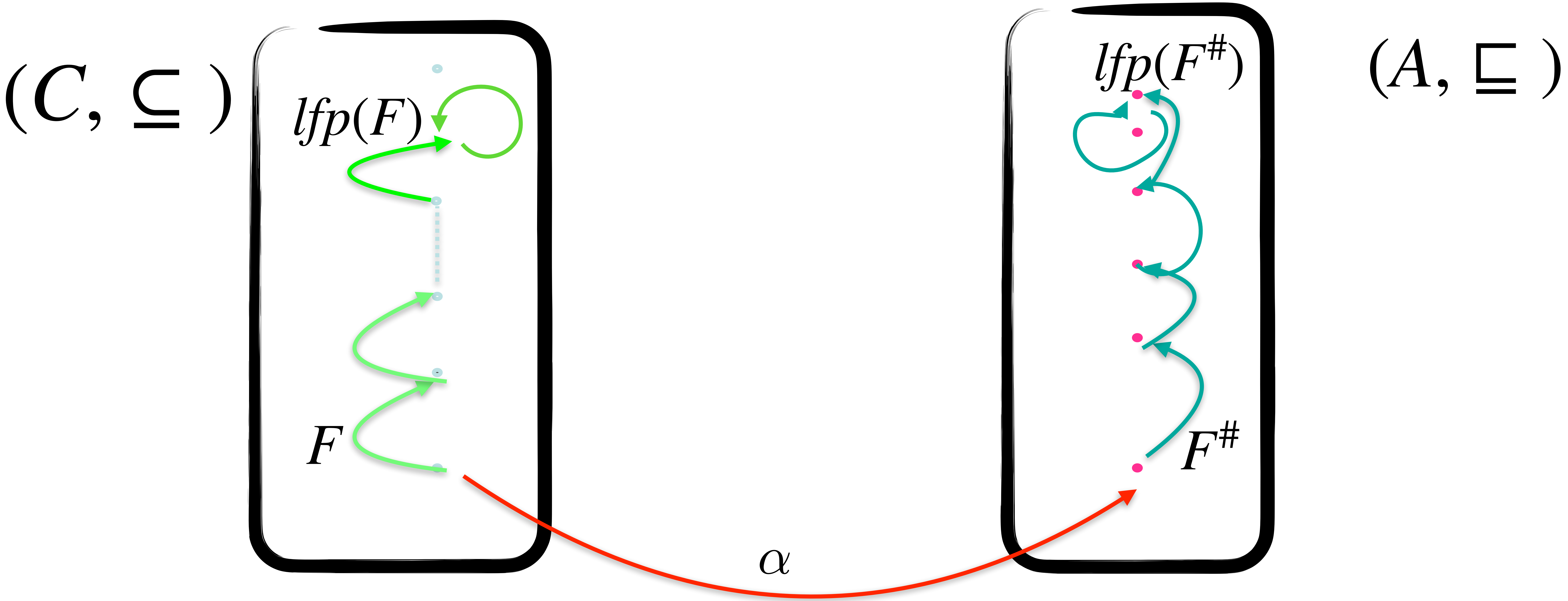
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



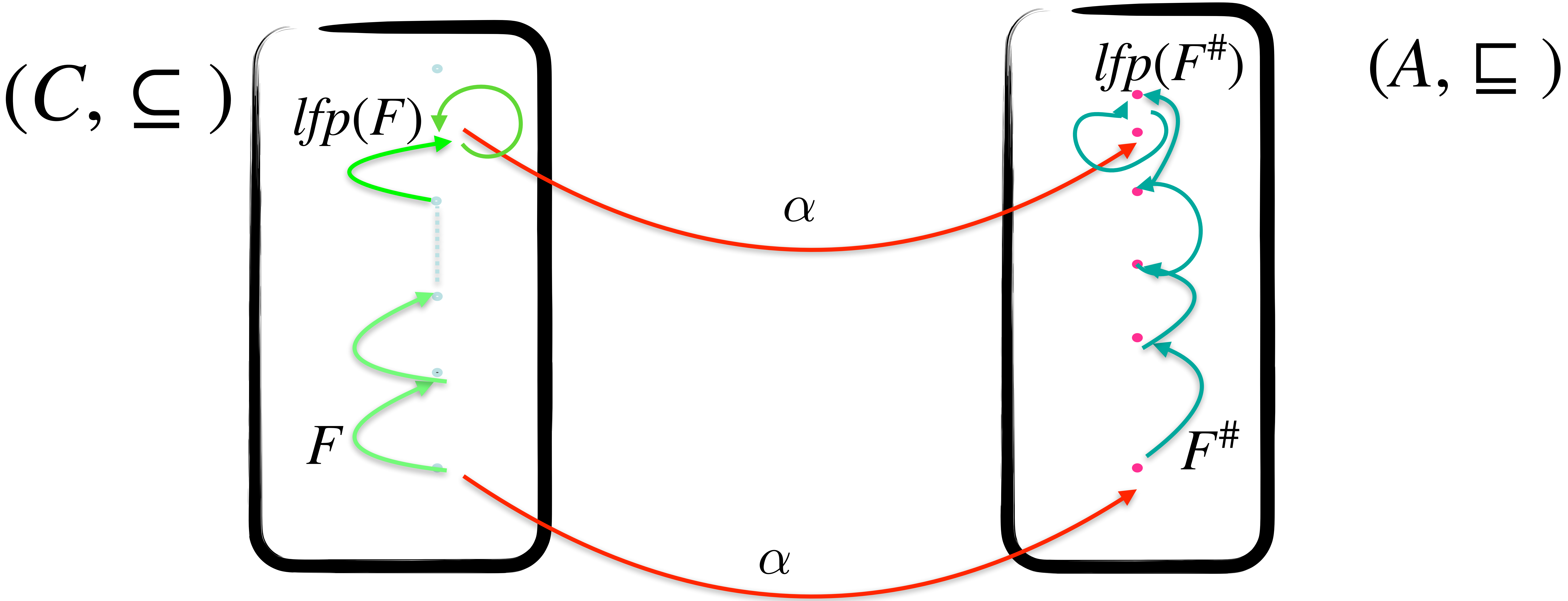
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



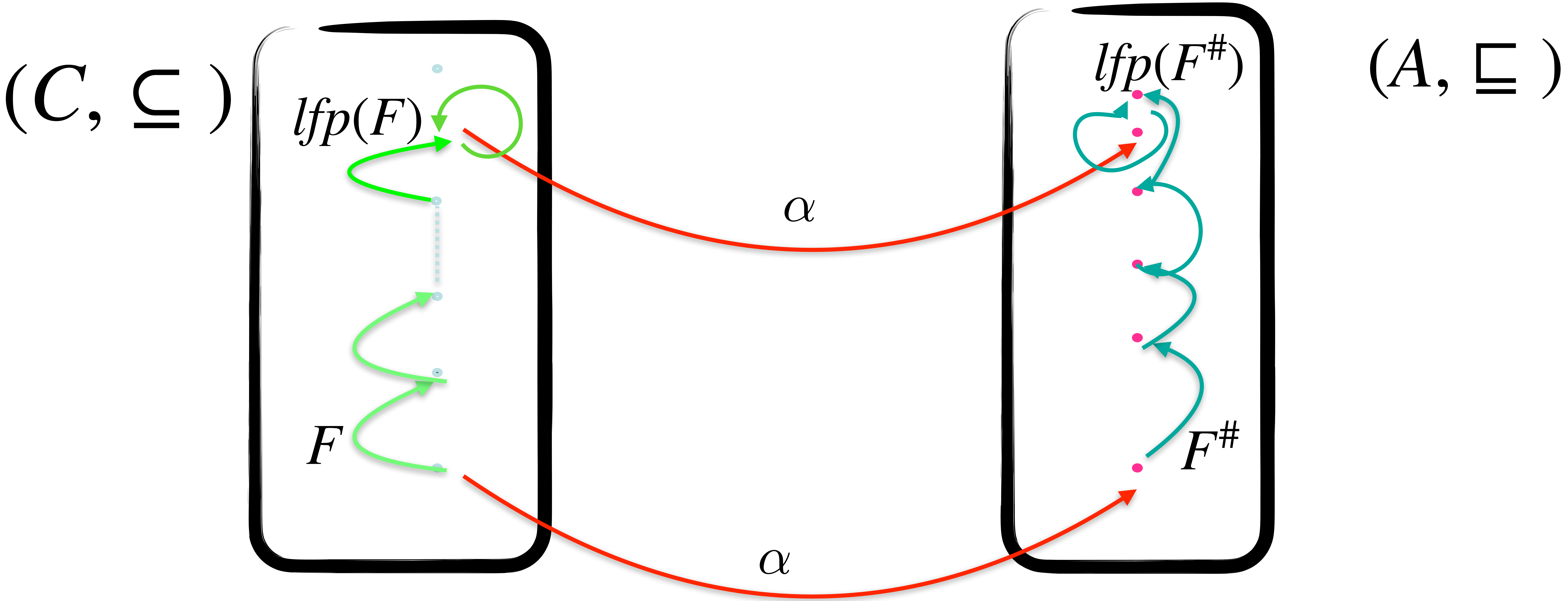
# Fixpoint computation approximation

If F monotone and $F^{\#}$ correct



# Fixpoint computation approximation

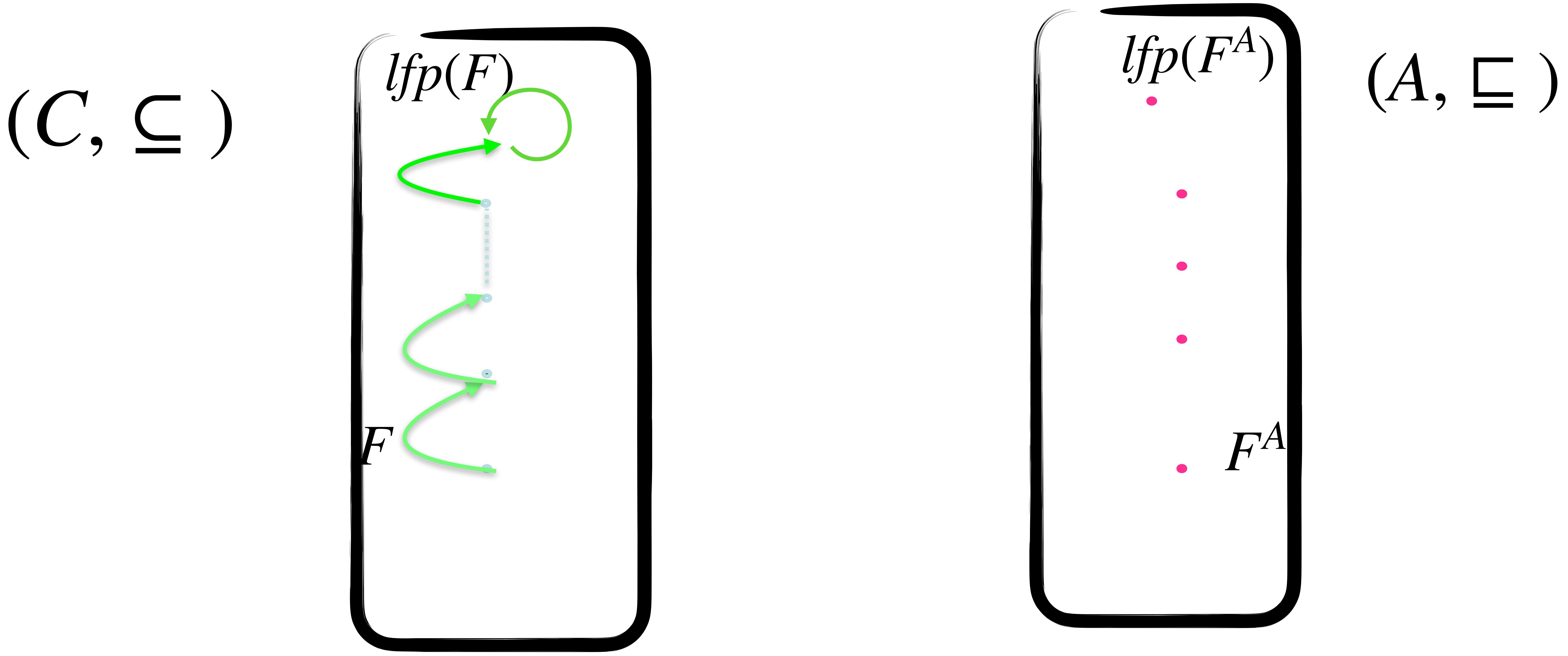
If F monotone and $F^{\#}$ correct




$lfp(F^{\#})$ is a correct over approximation of $lfp(F)$

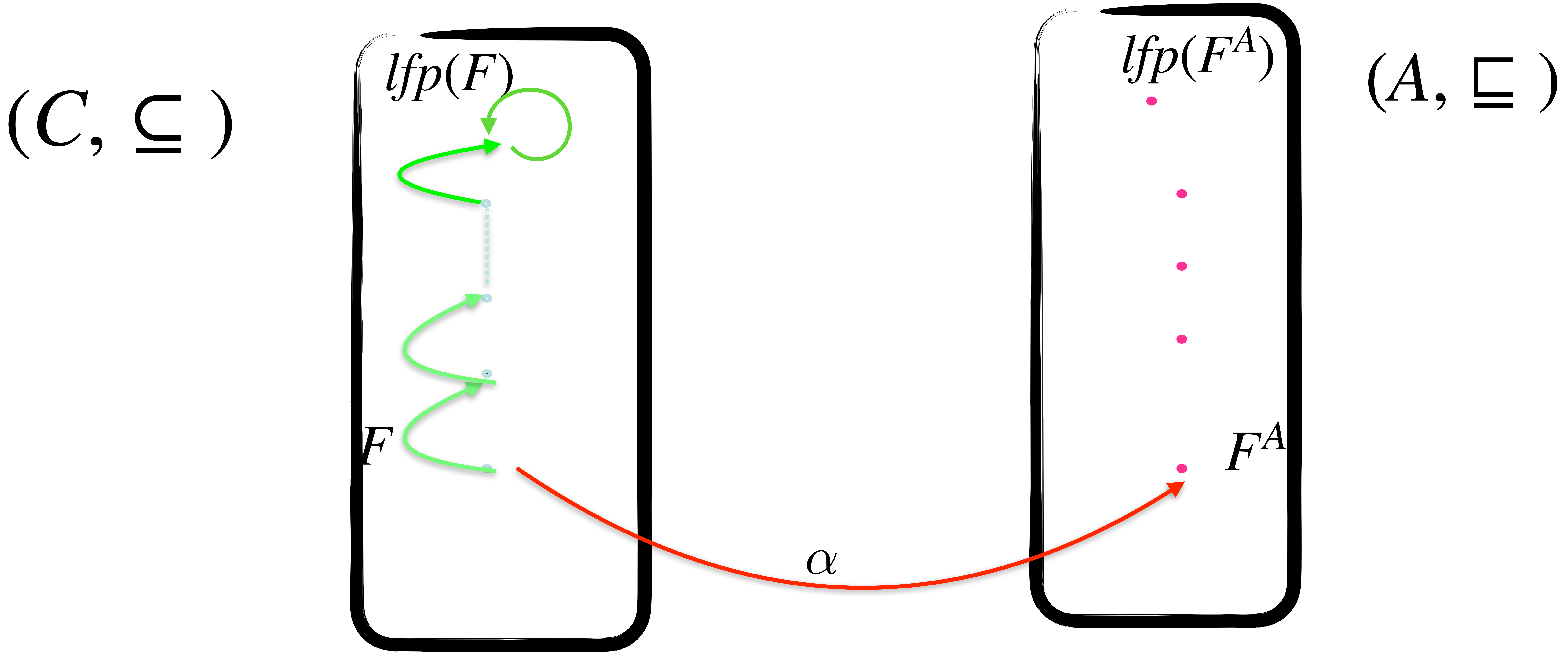
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



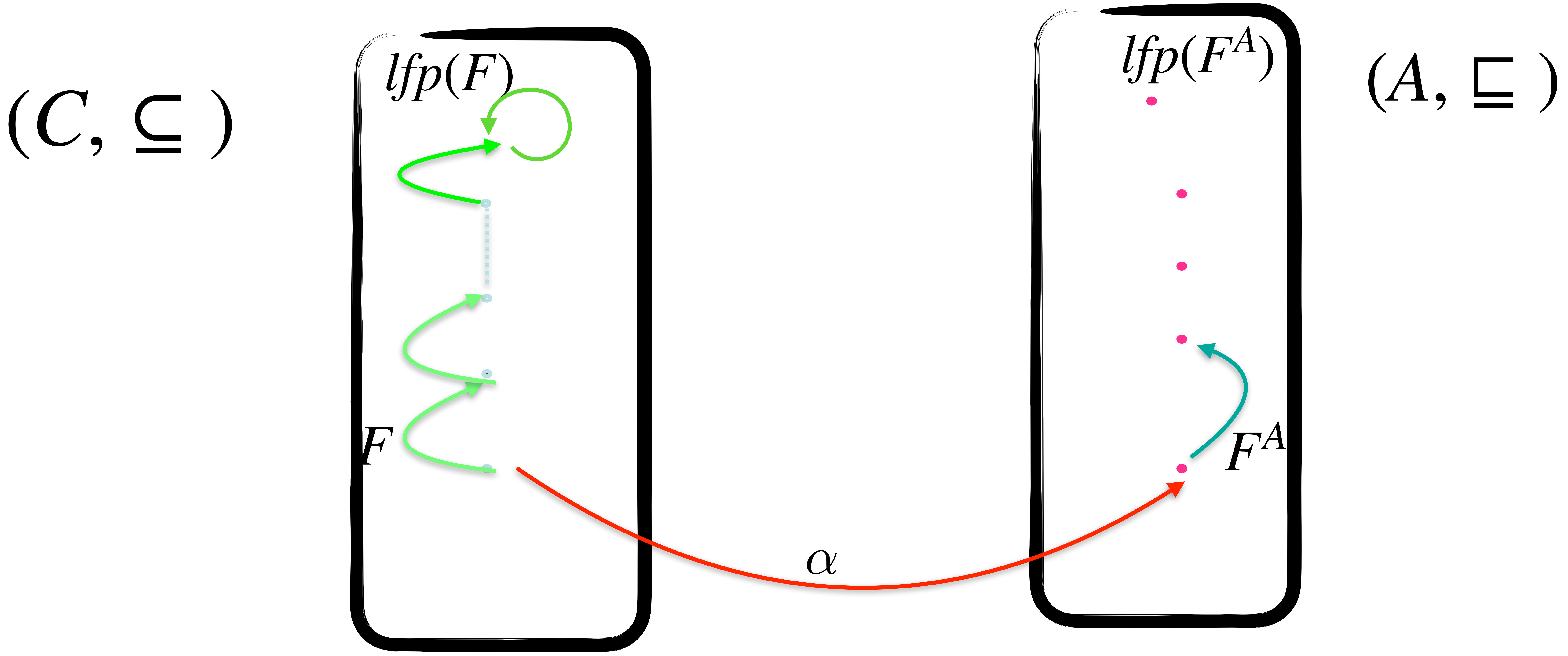
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



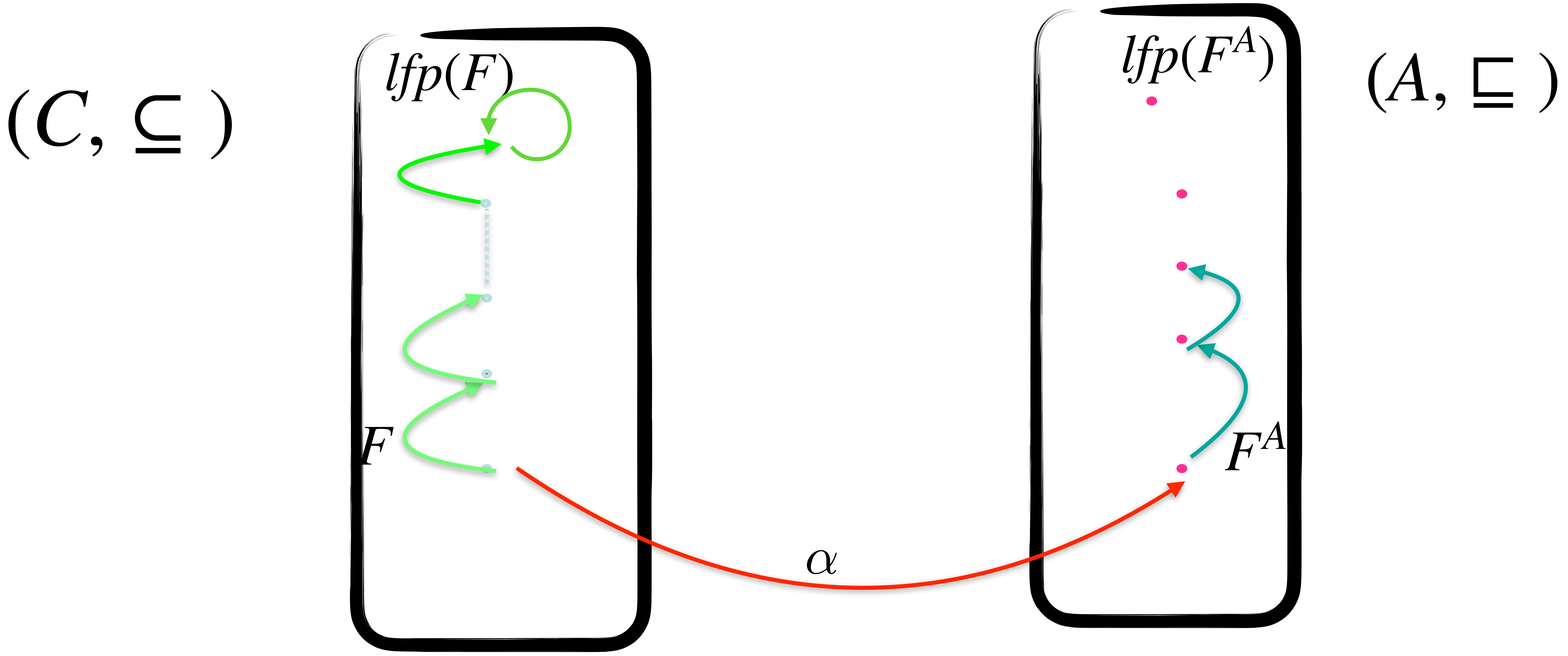
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



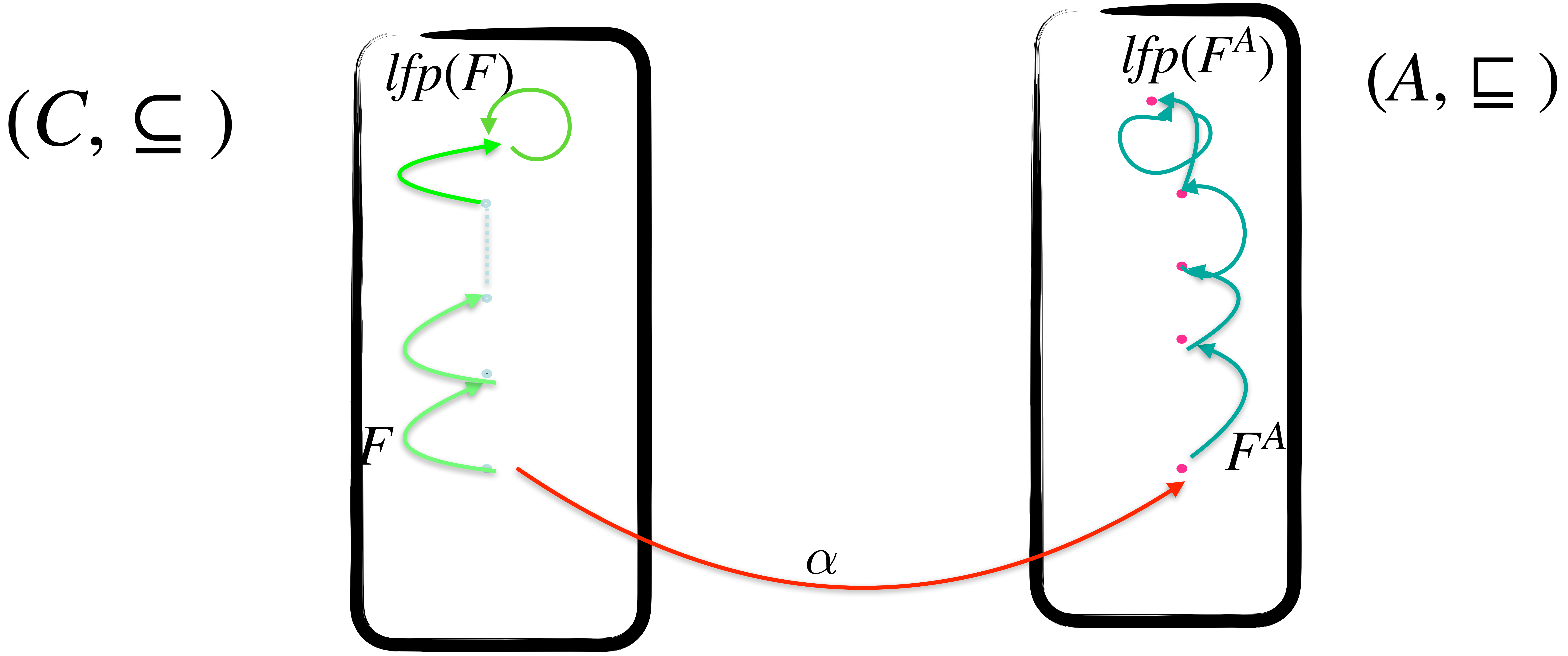
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



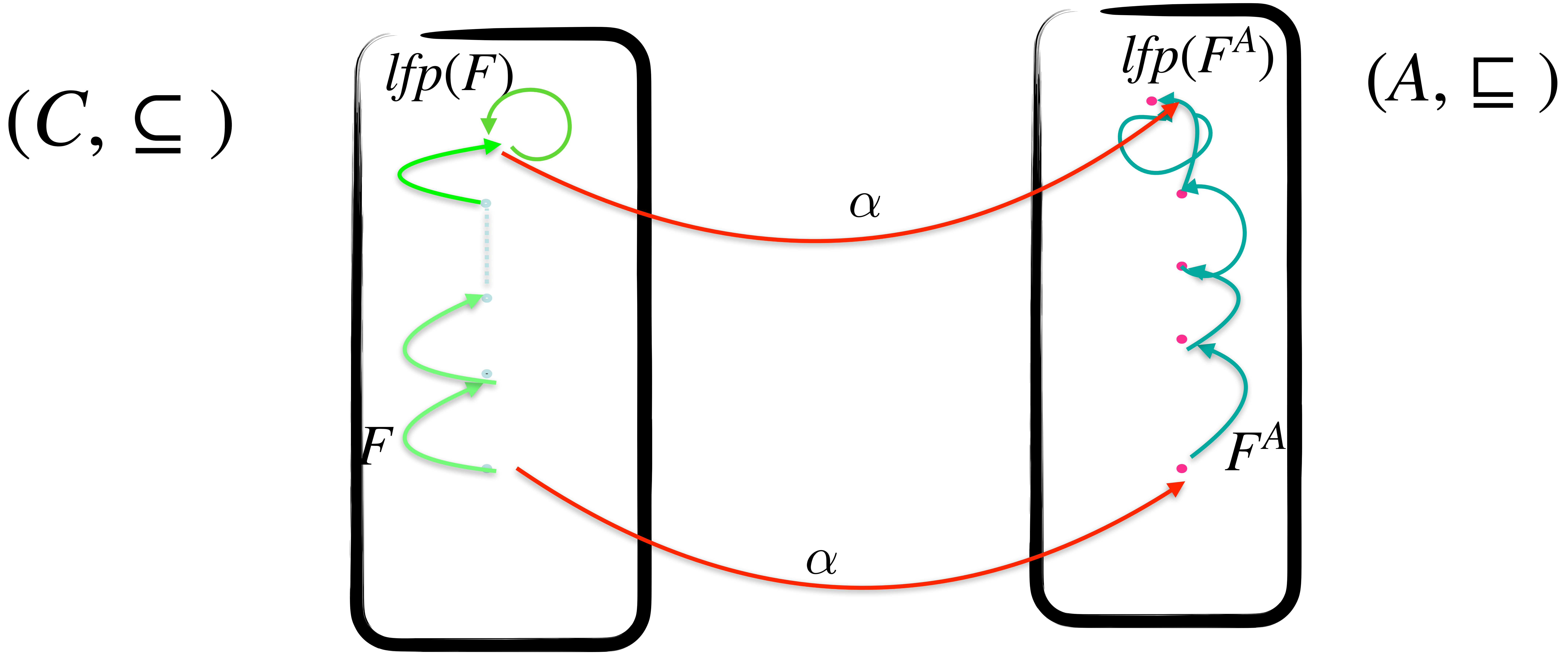
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



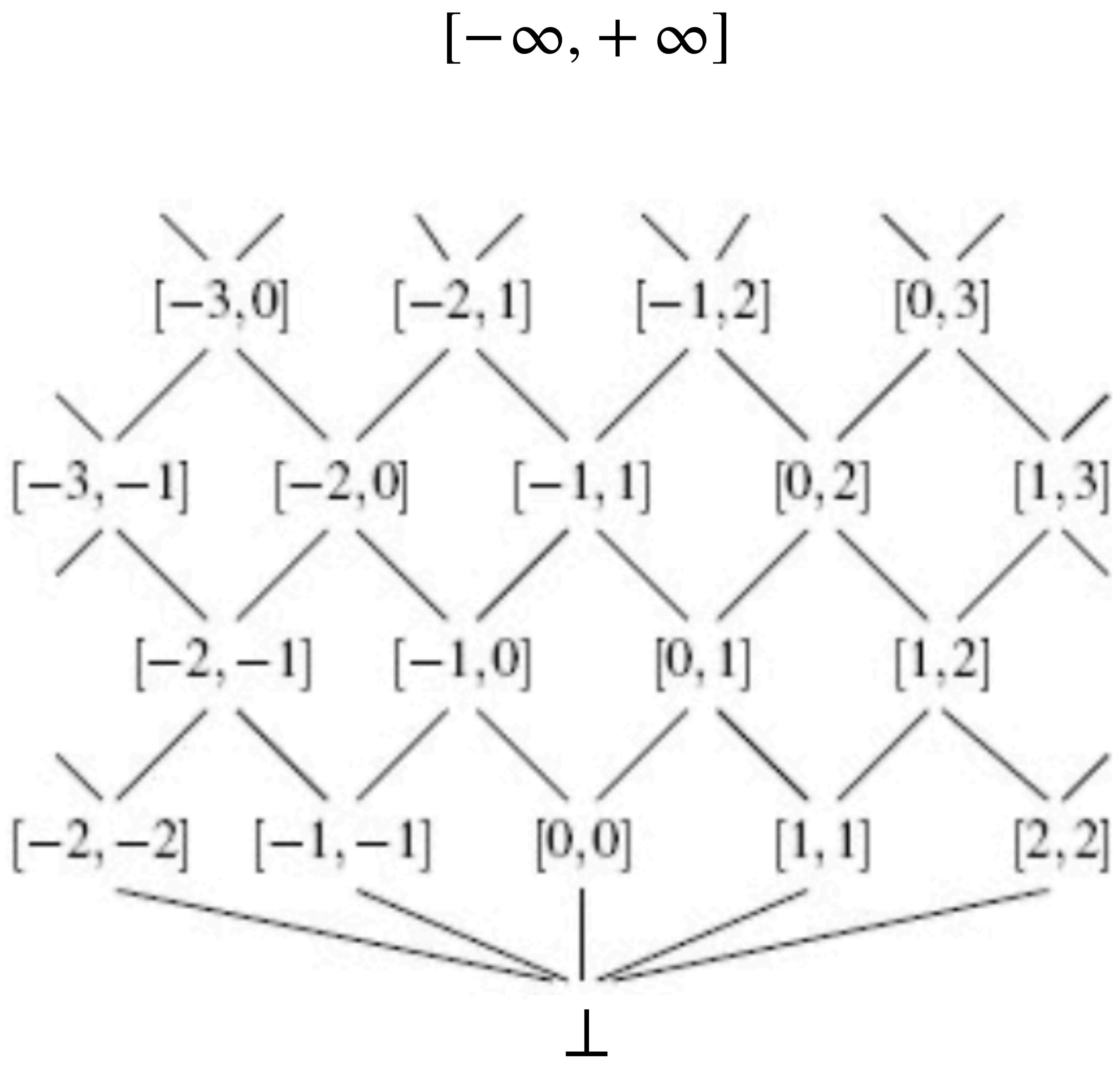
# Fixpoint computation approximation

If F monotone and $F^A$ is complete



# Abstract domains

# Intervals




Elements of A:

- ⊥ the empty set of values
- $[n_0, n_1],\ n_0 \in (\mathbb{Z} \cup \{-\infty\}),\ \ n_1 \in (\mathbb{Z} \cup \{+\infty\}), n_0 \leq n_1$

$\sqsubseteq$ is the interval inclusion

$$\begin{aligned}
\gamma(\bot) &= \{\} \\
\gamma([n_0, n_1]) &= \{\ n \in \mathbb{Z}\ \mid n_0 \leq n \leq n_1\} \\
\gamma([-\infty, n_1]) &= \{\ n \in \mathbb{Z} \mid n \leq n_1\} \\
\gamma([n_0, +\infty]) &= \{\ n \in \mathbb{Z} \mid n_0 \leq n\} \\
\gamma([-\infty, +\infty]) &= \mathbb{Z}
\end{aligned}$$

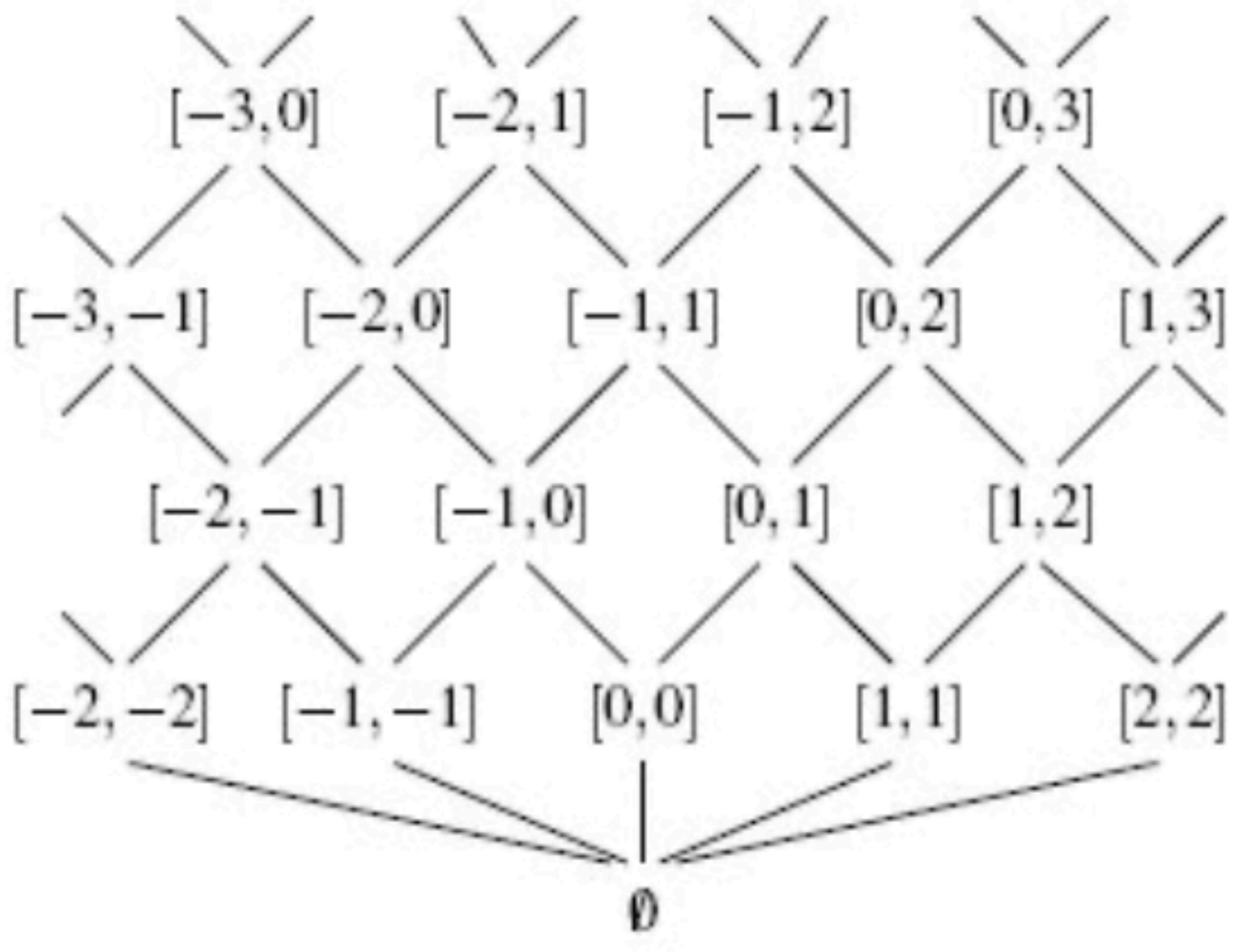
$$\begin{aligned}
&\alpha(c) = \bot \text{ if } c = \emptyset, \\
&\alpha(c) = [min(c), max(c)] \text{ if } c \neq \emptyset, min(c) \text{ and } max(c) \text{ exists} \\
&\alpha(c) = [min(c), +\infty] \text{ if } c \neq \emptyset, min(c) \text{ exists} \\
&\alpha(c) = [-\infty, max(c)] \text{ if } c \neq \emptyset, max(c) \text{ exists} \\
&\alpha(c) = [-\infty, +\infty] \text{ otherwise}
\end{aligned}$$

# $+^A$ and $\times^A$ are complete on Int

$$[n,m] +^A [p,r] = [n+p, m+r]$$

$$[n,m] \times^A [p,r] = [n \times p, m \times r]$$

if all positives, otherwise pay attention




$[1,6]$ $[-3,1]$ $[-2,7]$

Precise result!

$$\{1,4,6\} + \{-3,1\} = \{-2,1,2,3,5,7\}$$

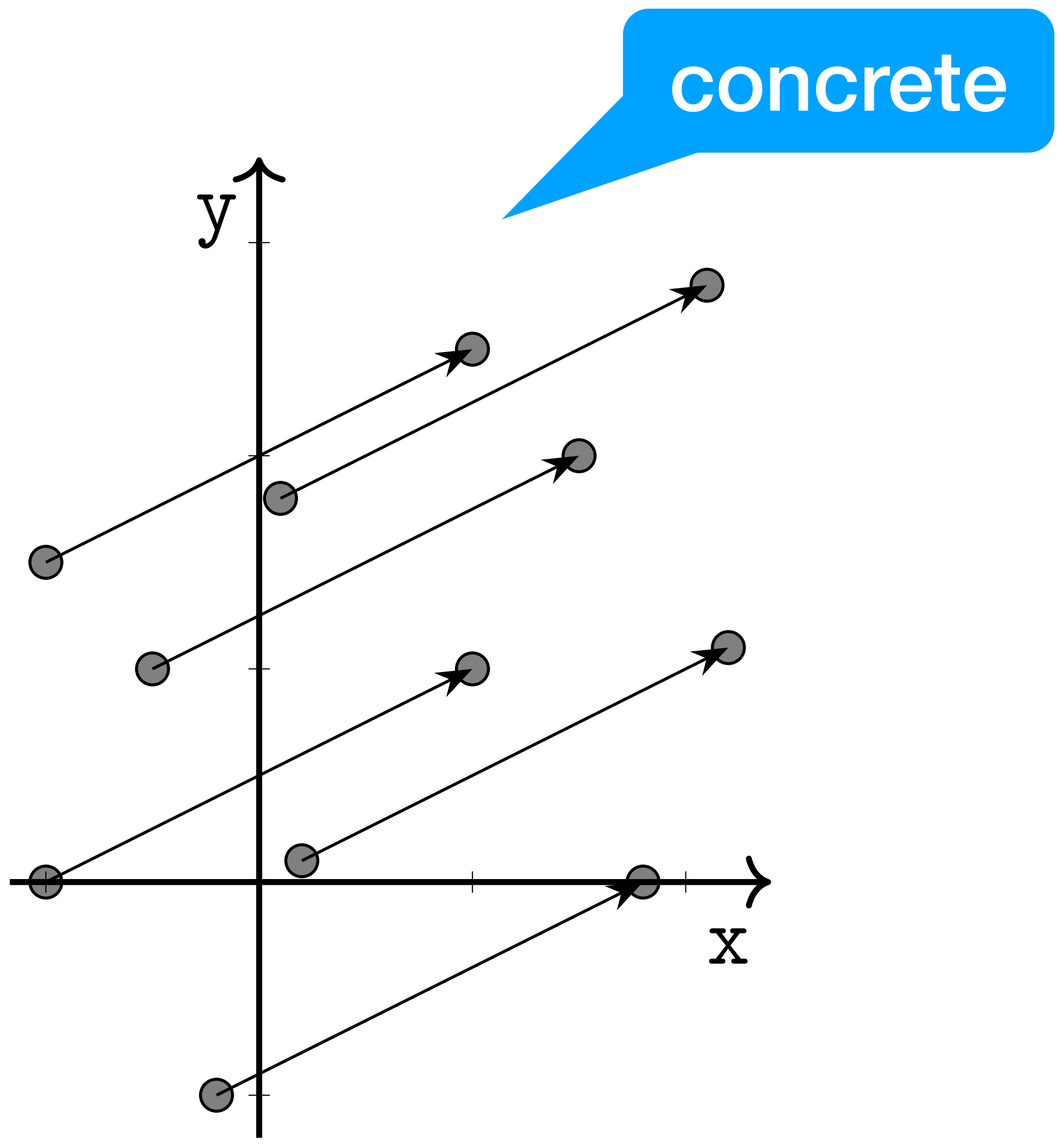
# Tests are not complete on Int

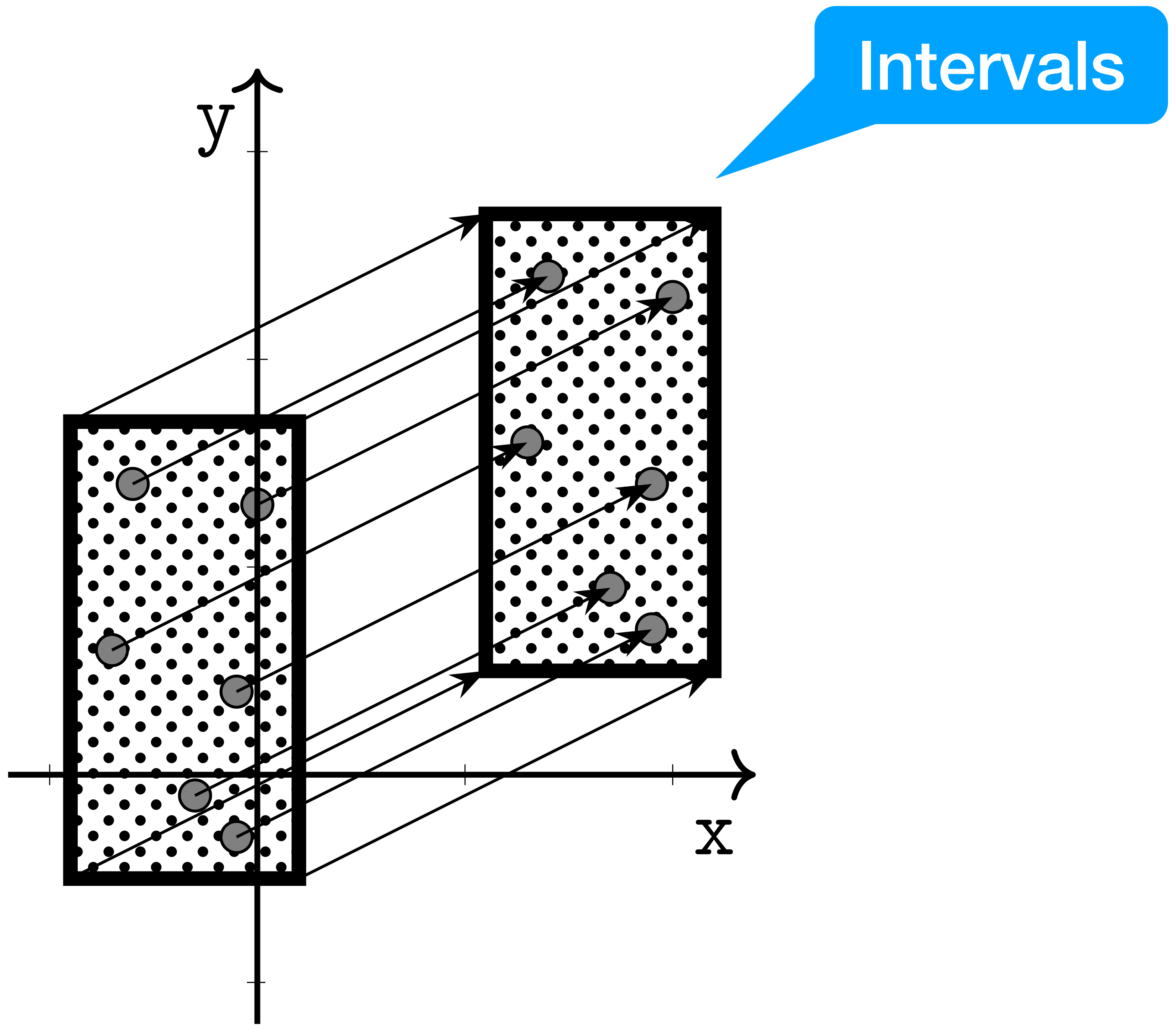
$A(P) = [-7,0]$ → $(x < 0)$ → $[-7, -1] \sqsupseteq [-7, -7] = A([\![\mathsf{c}]\!]P)$

Concrete $\qquad (x < 0)$

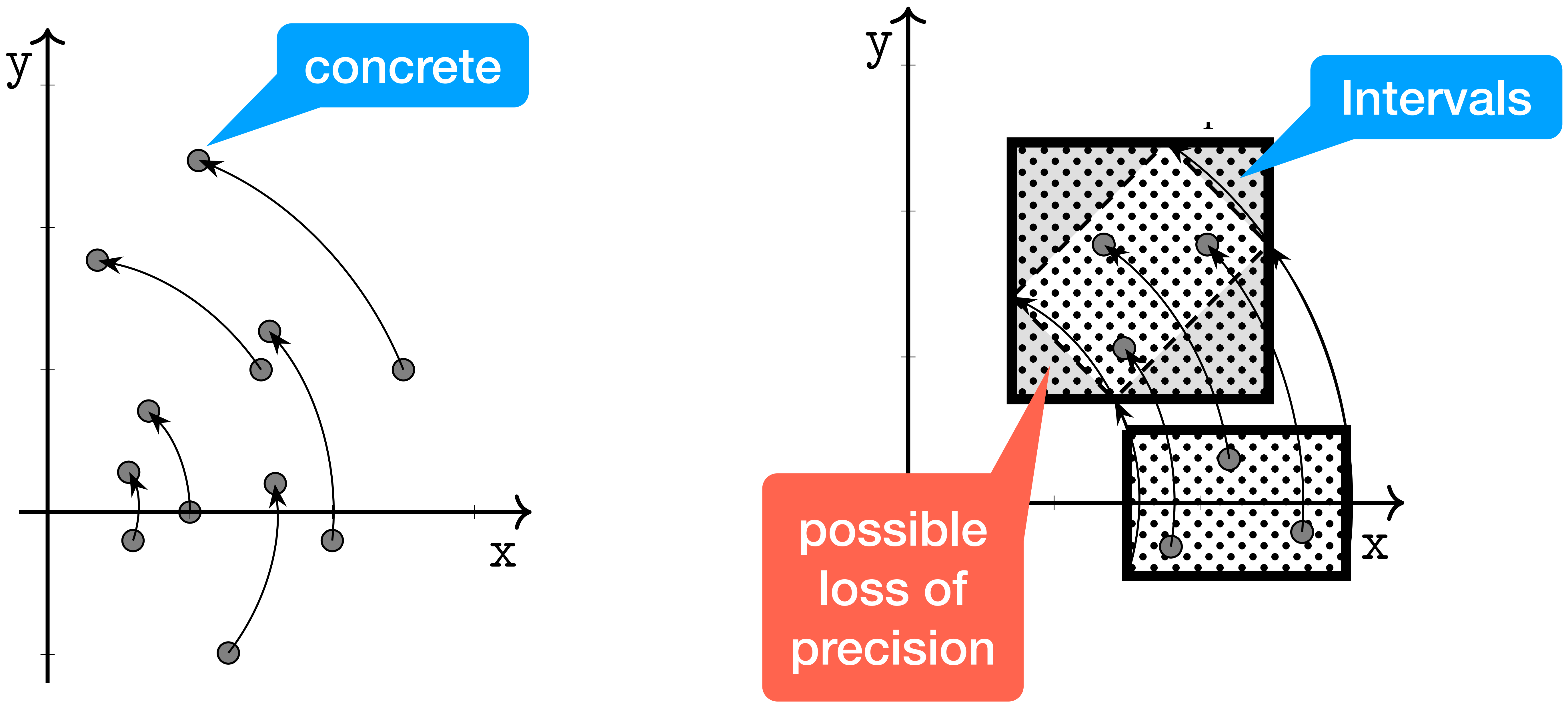
$P = \{-7,0\}$ → $(x < 0)$ → $\{-7\} \quad = [\![(x < 0)]\!]P$

# Example: translation



# Example: rotation

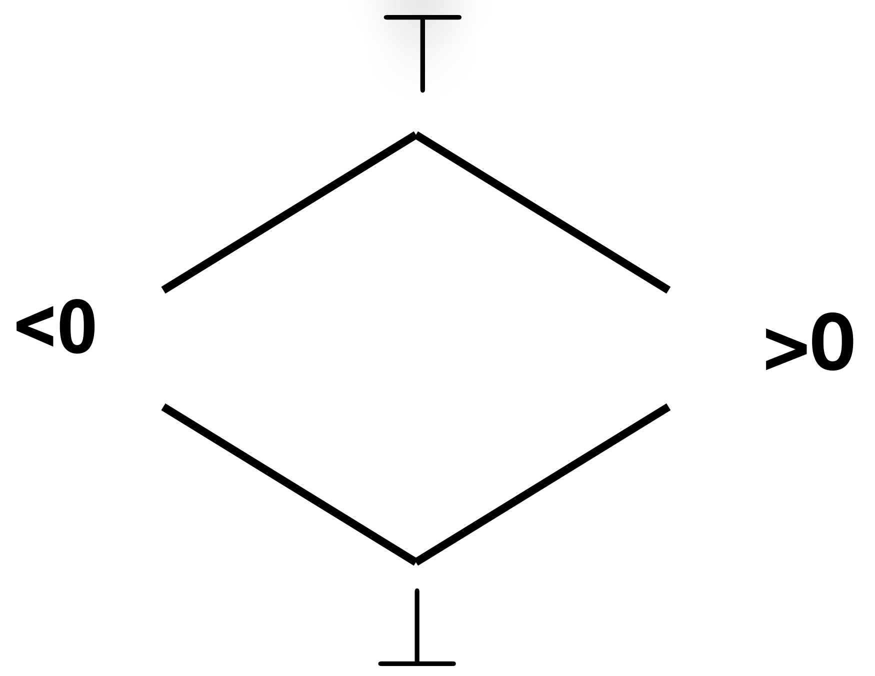


# Composition of bcas

The composition of bca is not always a bca

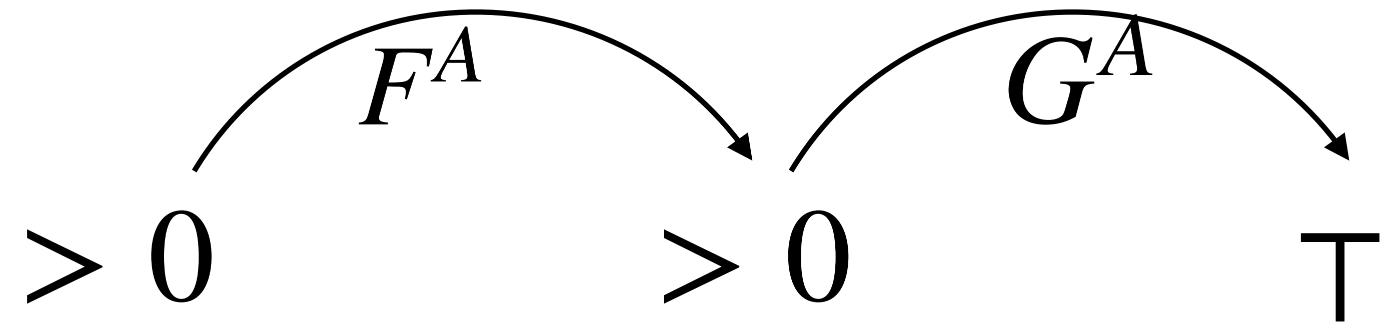
For $F^A$ and $G^A$ bca, in general

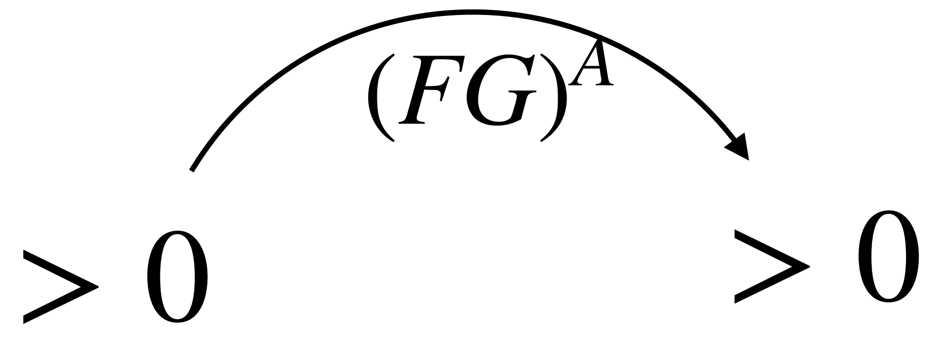
$$F^A G^A \neq (FG)^A \quad \text{Indeed} \quad \alpha F \gamma \alpha G \gamma \sqsupseteq \alpha F G \gamma \quad \text{because} \quad \gamma\alpha \sqsupseteq \text{id}$$




Example

$$F = \_ + 1 \quad G = \_ - 1 \quad FG = \text{id}$$



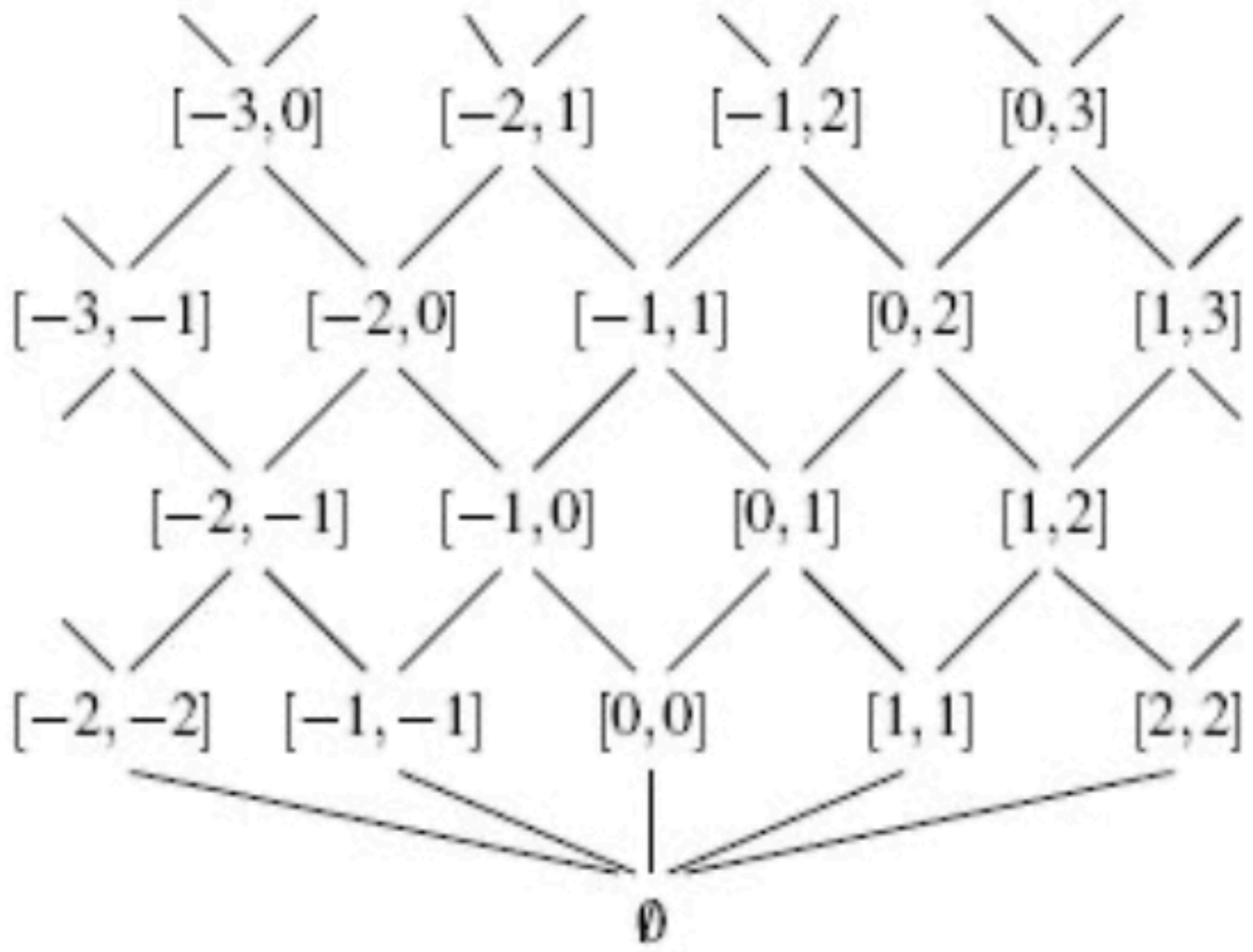



# Composition of complete abstractions

The composition of complete abstractions is always complete

For $F^\#$ and $G^\#$ complete abstractions

$$F^\# G^\# \alpha = F^\# \alpha G = \alpha F G$$




Example

$$F = \_ + 1 \quad G = \_ - 1 \quad FG = \text{id}$$

$[2,3] \xrightarrow{F^\#} [3,4] \xrightarrow{G^\#} [2,3]$

$[2,3] \xrightarrow{(FG)^A} [2,3]$

# Non-relational domains

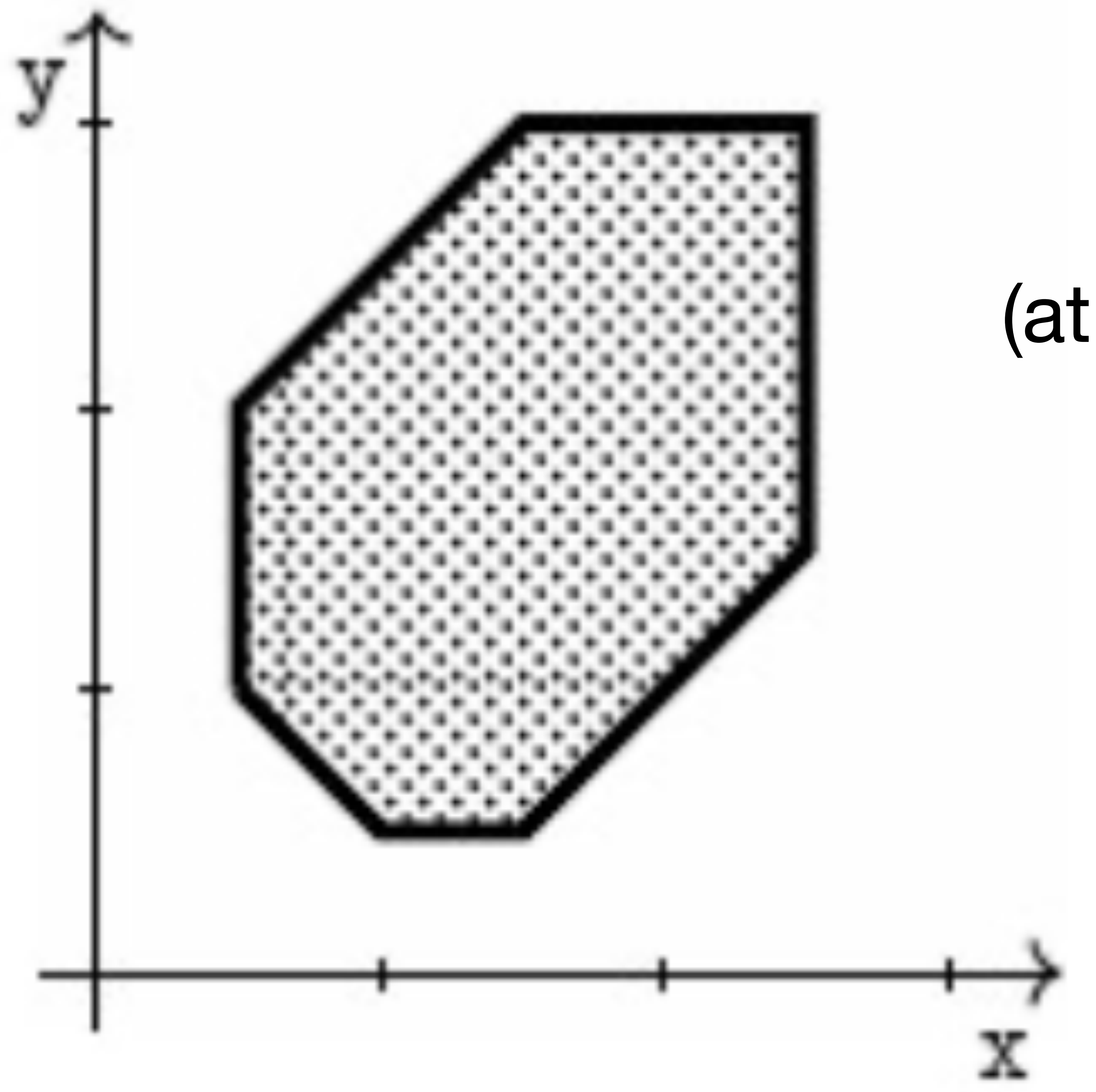
The domains of Sign and Interval are non-relational domains

They cannot track relations between variables values

The set of states

$$\{[x \mapsto 1, y \mapsto 6]\ [x \mapsto 3, y \mapsto 8]\ [x \mapsto 10, y \mapsto 15]\} \xrightarrow{\alpha} [x \mapsto [1,10], y \mapsto [6,15]] \xrightarrow{\gamma} \{[x \mapsto 1, y \mapsto 6]\ [x \mapsto 1, y \mapsto 7]\ \dots\}$$

# Relational domain
# Octagon domain




sets of numerical constraints of the form

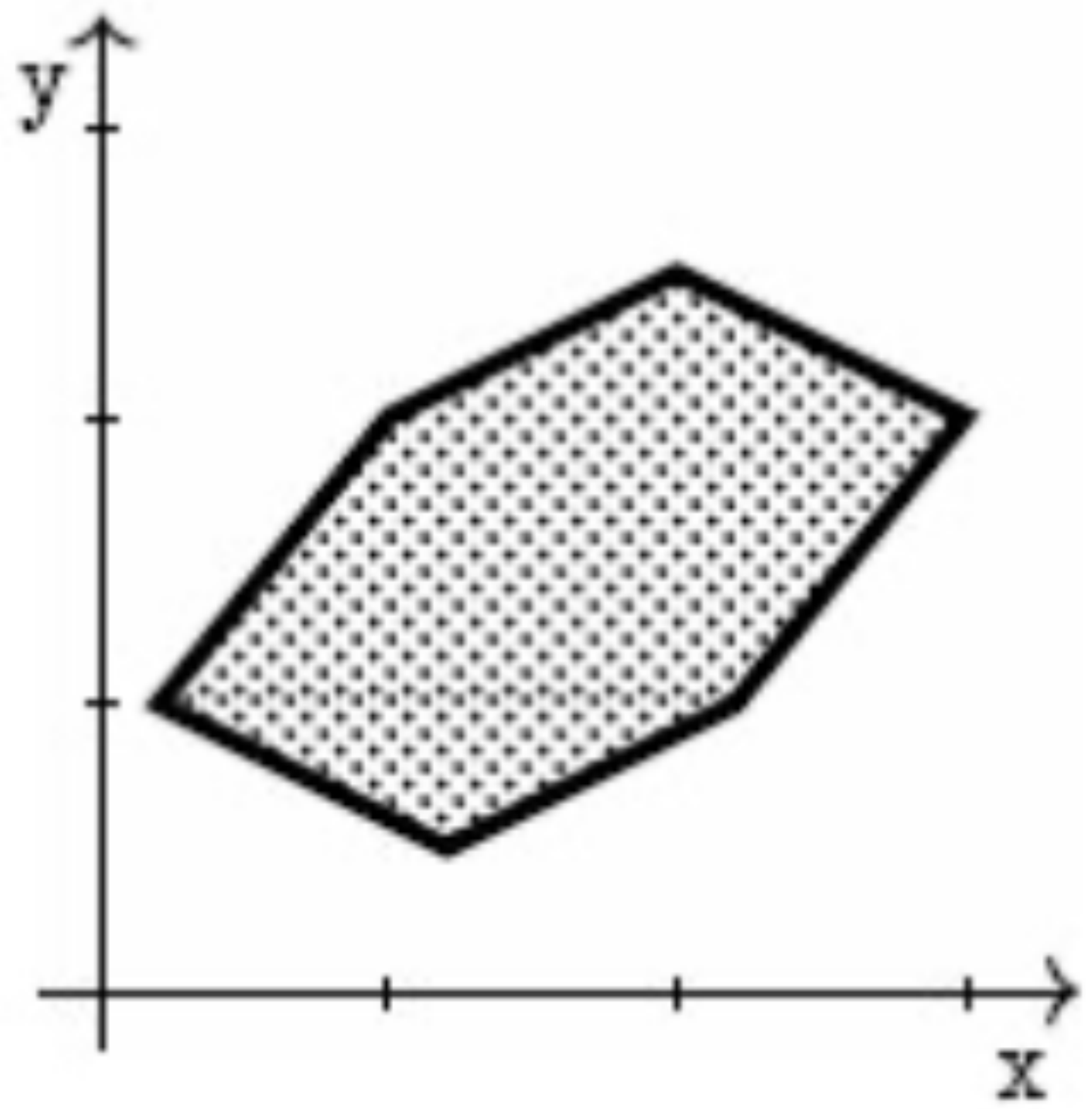
$$\pm x \pm y \leq c$$

(at most two variables per constraint, with unit coefficients)

The set of states

$$\{[x \mapsto 1, y \mapsto 6] \quad [x \mapsto 3, y \mapsto 8] \quad [x \mapsto 10, y \mapsto 15]\} \xrightarrow{\alpha} \begin{array}{l} x \leq 10 \\ x \geq 1 \\ y \leq 15 \\ y \geq 6 \\ y - x = 5 \end{array}$$

# Relational domain
# Convex Polyhedra domain

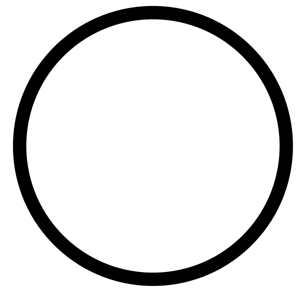



sets of numerical constraints of the form

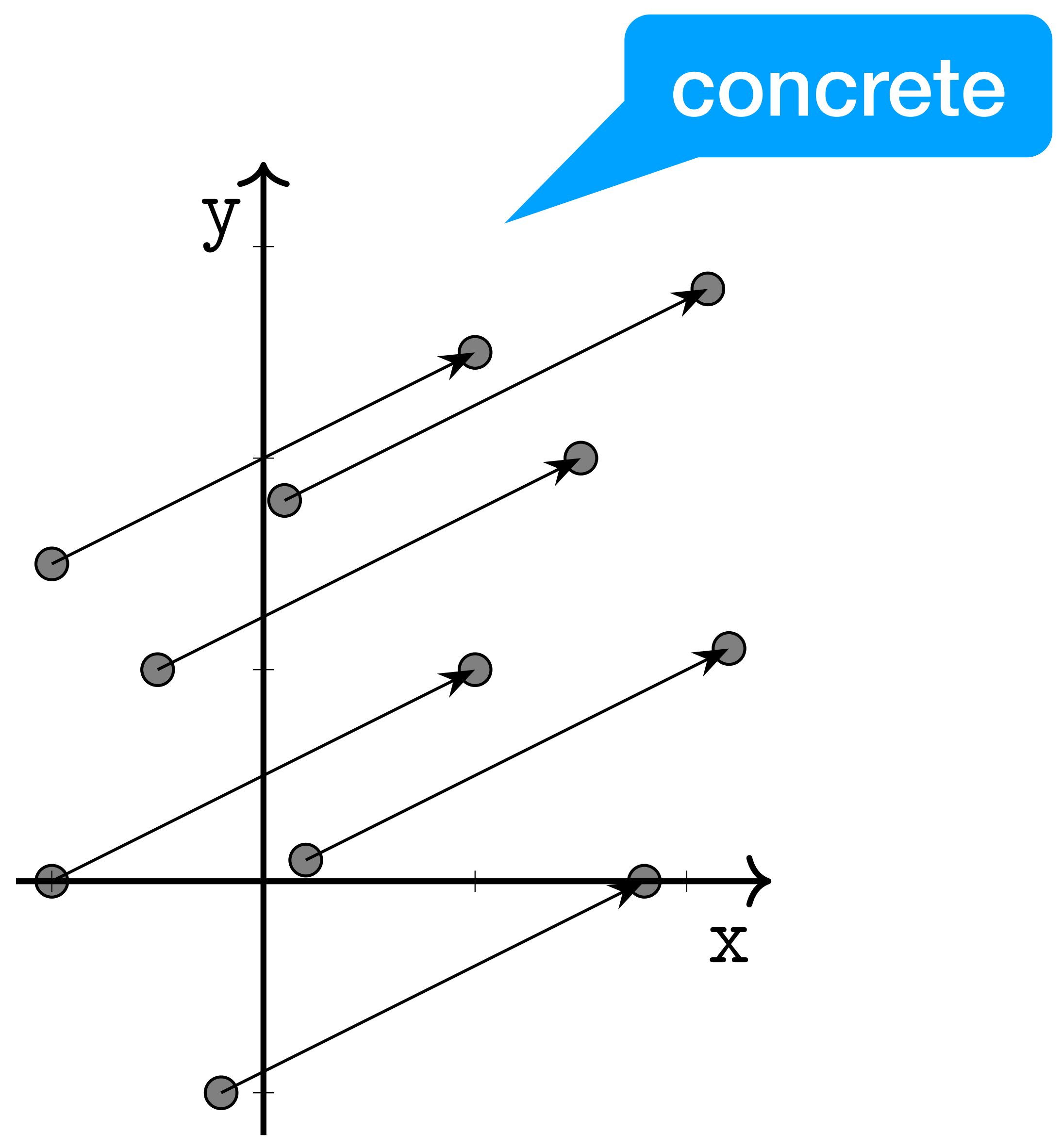
$$c_1x + c_2y \leq c$$

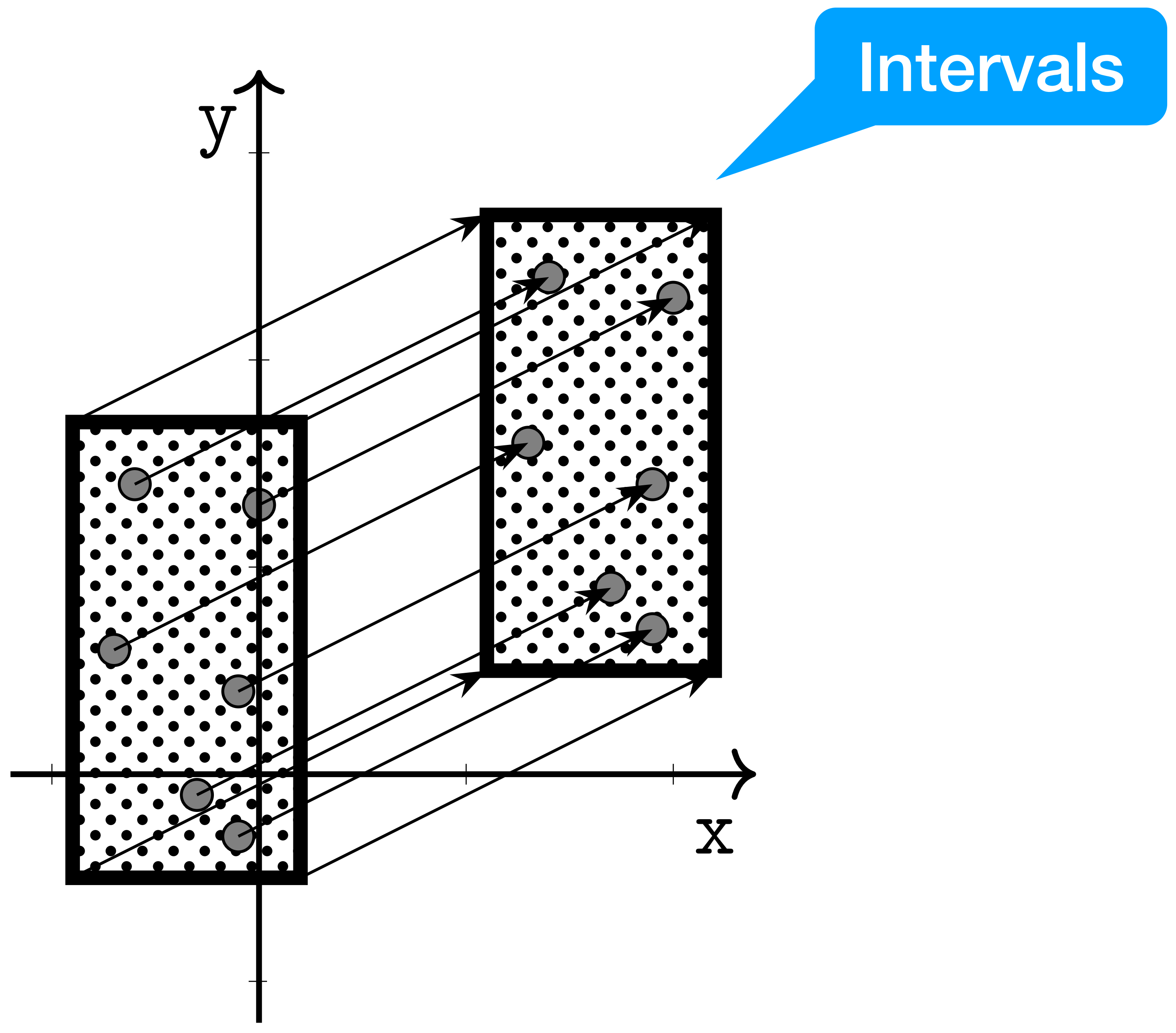
(at most two variables per constraint,
with unit coefficients)

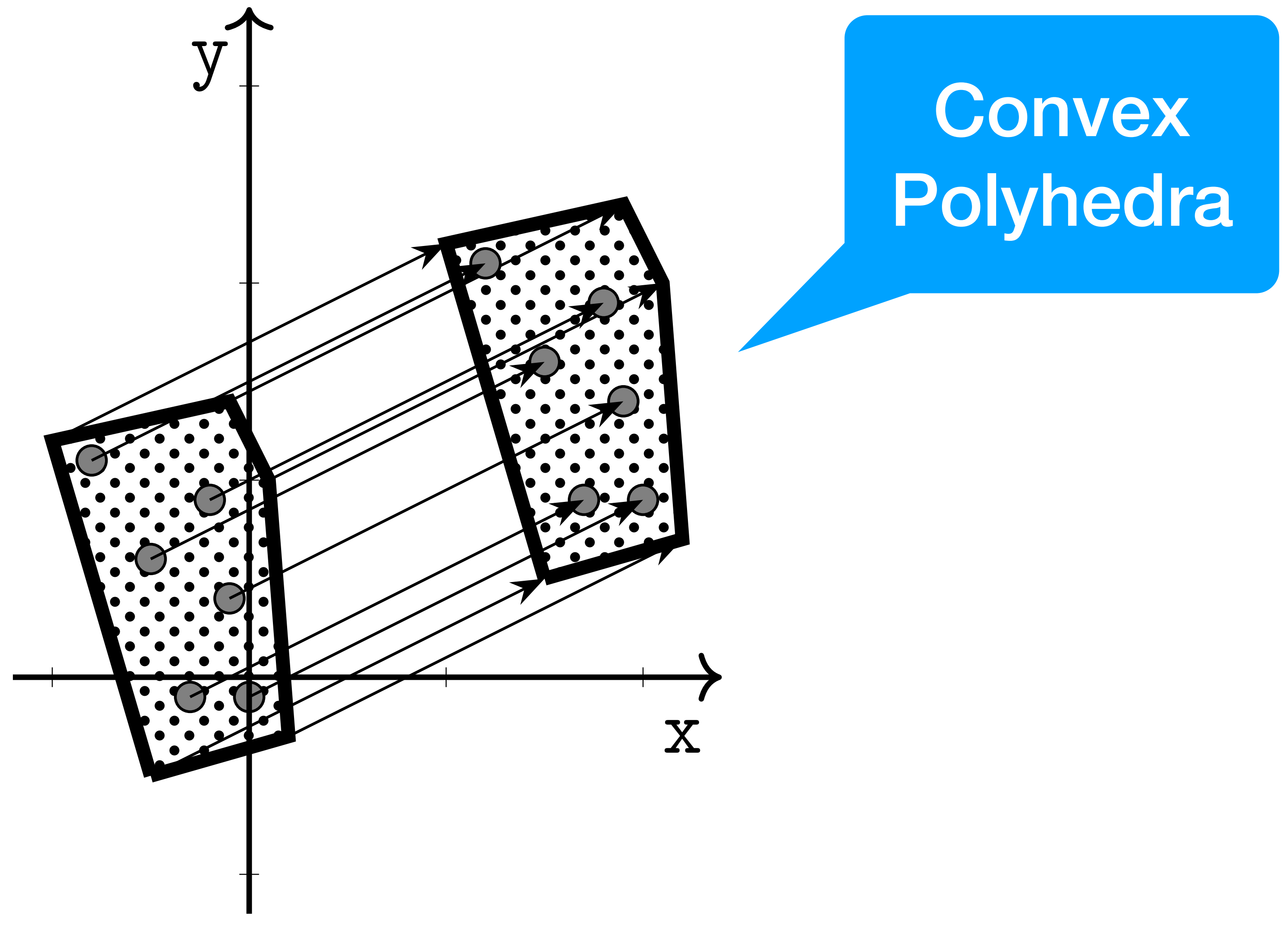
does not admit an abstraction map

best abstraction of  ?

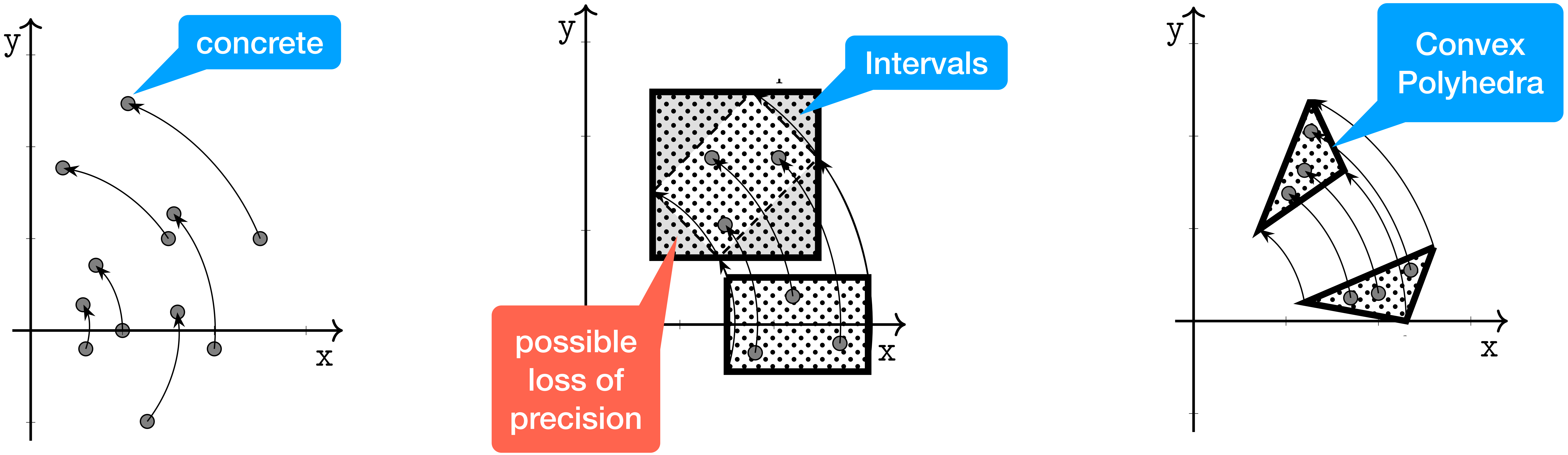
# Example: translation



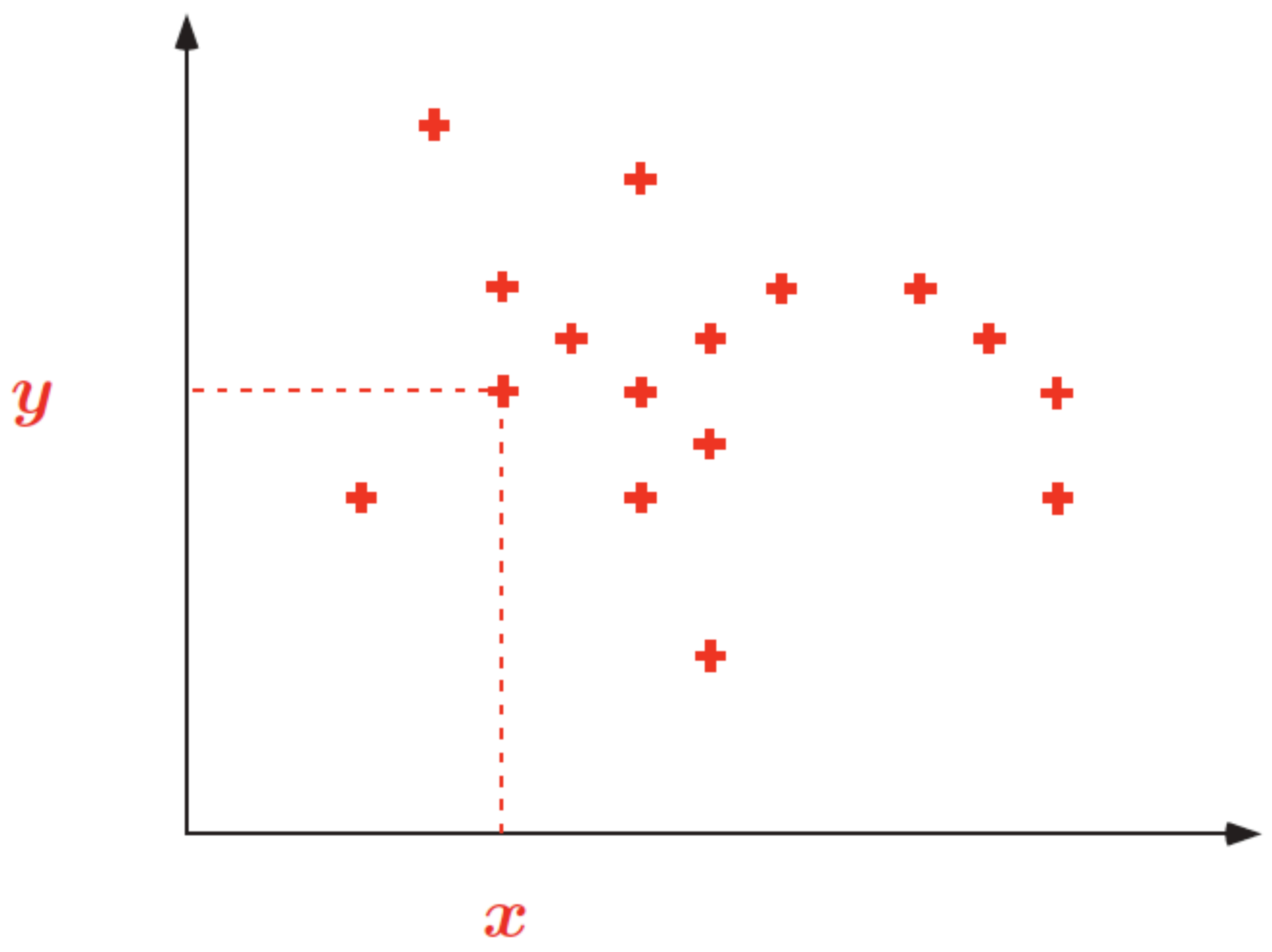
# Example: rotation



# Refinements of abstraction

An (in)-finite set of points :

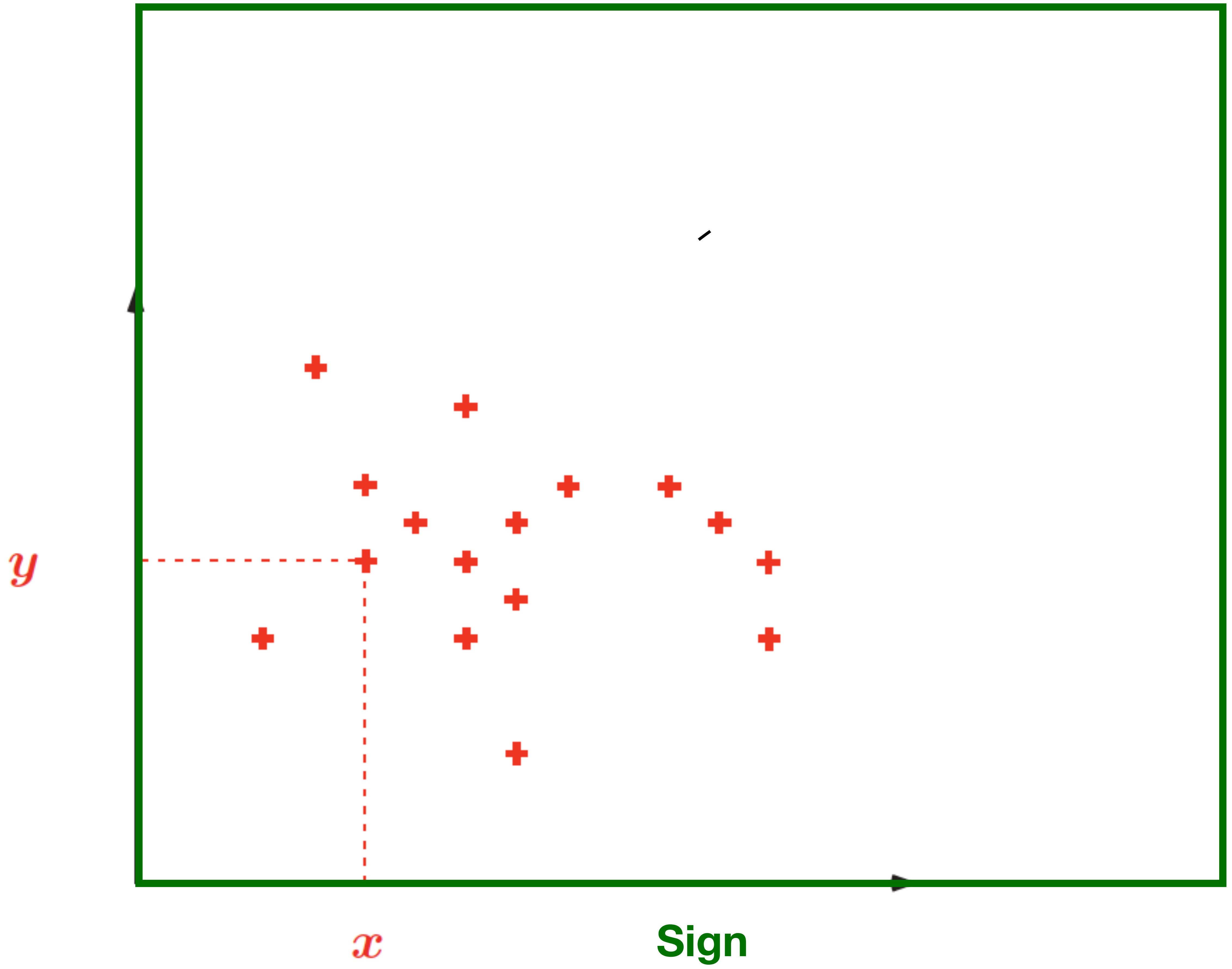
$$\{ \ldots (19{,}77) \ldots (20{,}03) \ldots . \}$$



# Refinements of abstraction

An (in)-finite set of points :

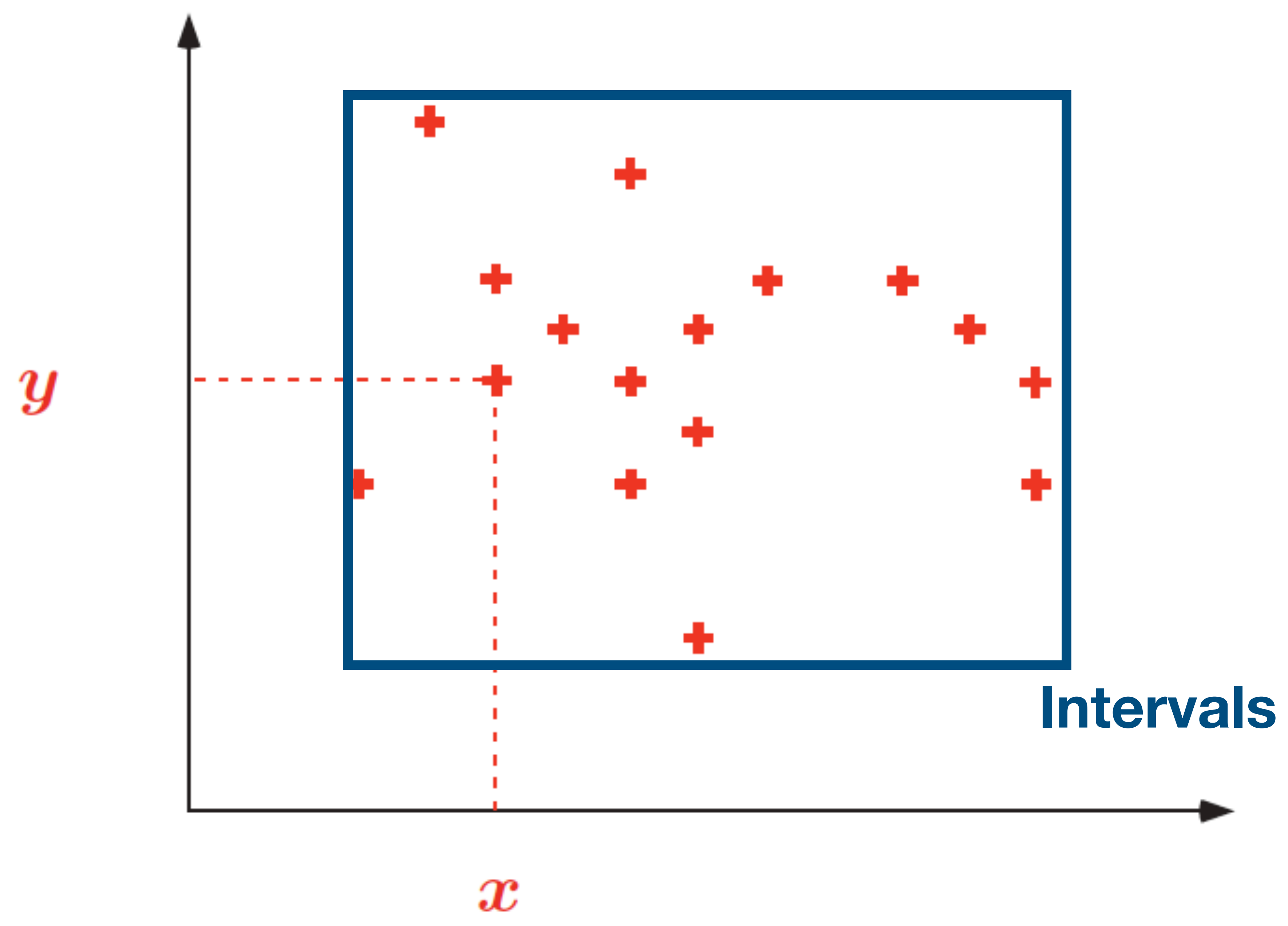
$$\{\ldots(19{,}77)\ldots(20{,}03)\ldots.\}$$



# Refinements of abstraction

An (in)-finite set of points :

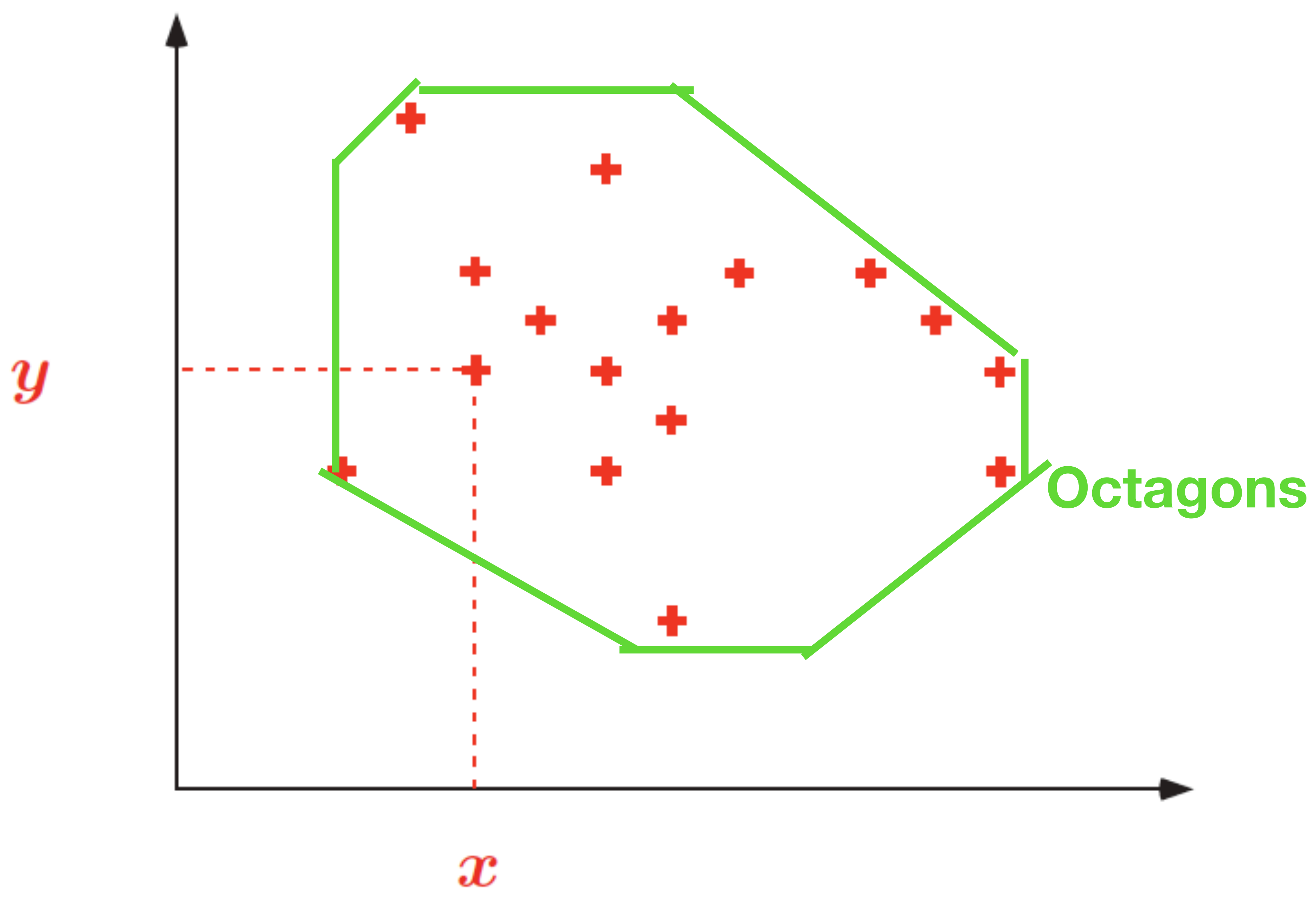
$$\{\ldots(19{,}77)\ldots(20{,}03)\ldots.\}$$



# Refinements of abstraction

An (in)-finite set of points :

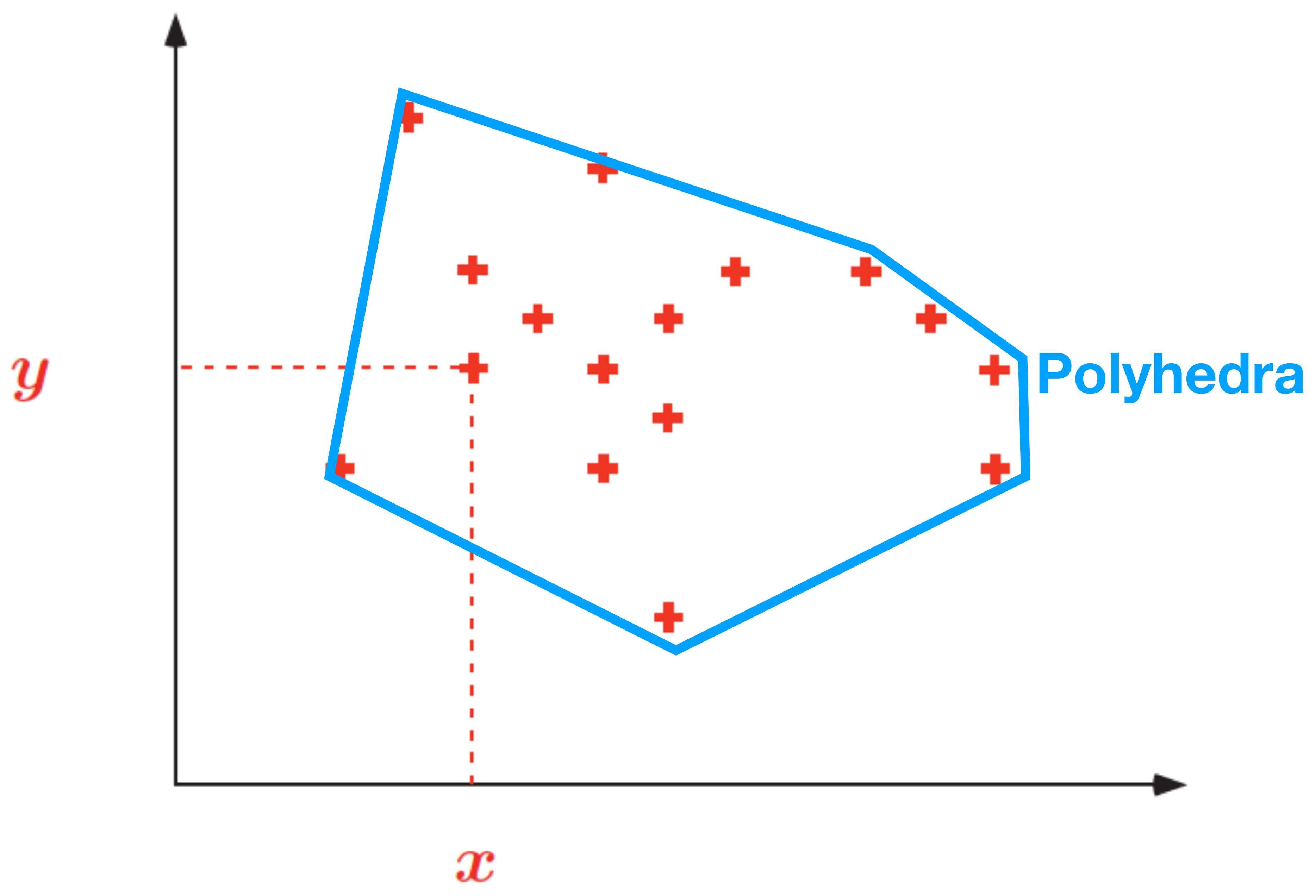
$$\{\ldots(19{,}77)\ldots(20{,}03)\ldots.\}$$



# Refinements of abstraction

An (in)-finite set of points :

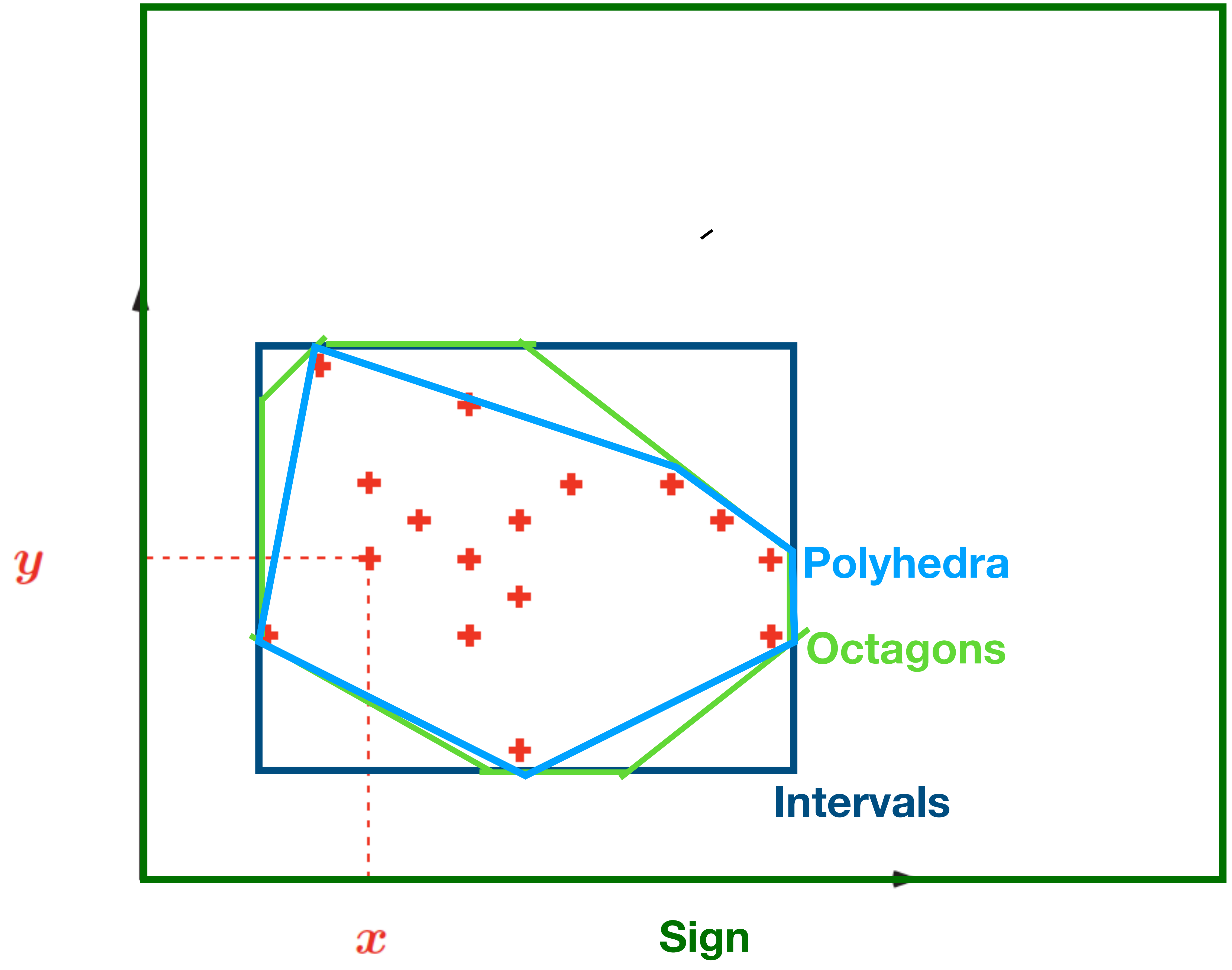
$$\{\ldots(19,77)\ldots(20,03)\ldots.\}$$



# Refinements of abstraction

An (in)-finite set of points :

$\{\ldots(19{,}77)\ldots(20{,}03)\ldots.\}$



# Order on abstract domains

We say that the abstract domain $A_1$ refines $A_2$,

written $A_1 \preceq A_2$, iff

$$\forall c \in C \,.\, \gamma_{A_1}(\alpha_{A_1}(c)) \subseteq \gamma_{A_2}(\alpha_{A_2}(c))$$

intuitively, $A_1$ is more precise than $A_2$

$$Octagons \sqsubseteq Int \sqsubseteq Sign$$

# Conjunctive properties

program verification often requires the use of the conjunction of several basic predicates

concrete states = stores with two variables $x, y$

intervals abstraction for each variable

abstract state = an interval for each variable

[0,∞] [3,8]

# Product domain

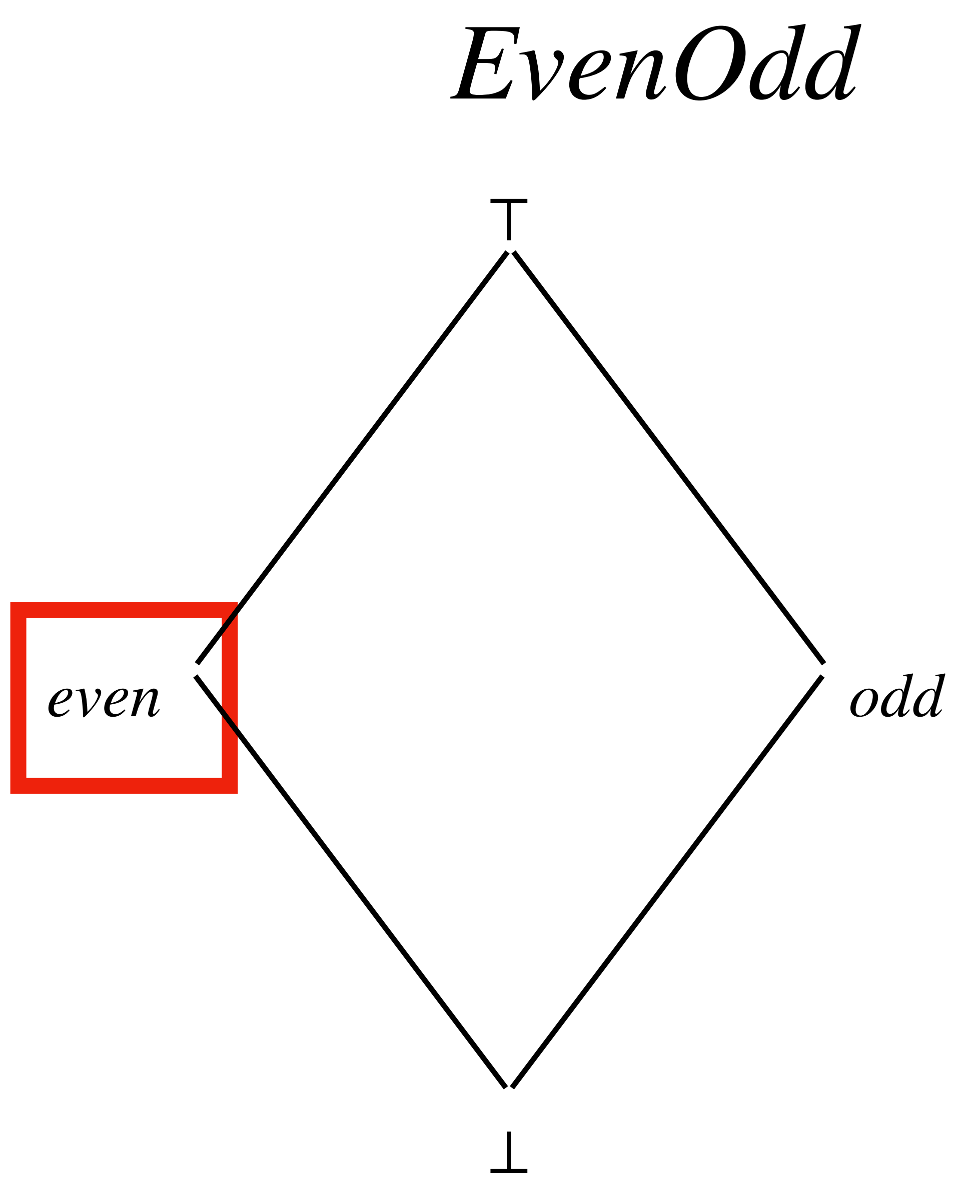
$$C \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} A_0$$

$$C \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} A_1$$

$$C \underset{\alpha_\times}{\overset{\gamma_\times}{\leftrightarrows}} A_0 \times A_1$$

$$\gamma_\times(a_0, a_1) = \gamma_0(a_0) \cap \gamma_1(a_1)$$

# Problem




concrete stores = stores with one variable $x$

Int × EvenOdd

e.g. an abstract state $([2,10], even)$

describes **even** values between $2$ and $10$

but also $([1,11], even)$ represents the same

concrete set $\{2,4,6,8,10\}$!

# Reduced product $A_0 \sqcap A_1$

$$C \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} A_0 \qquad\qquad C \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} A_1$$

$$C \underset{\alpha_\sqcap}{\overset{\gamma_\sqcap}{\leftrightarrows}} (A_0 \times A_1)_\equiv \quad A_0 \sqcap A_1$$

take the equivalence classes

$$(a_0, a_1) \equiv (a_0', a_1') \Leftrightarrow \gamma_\times(a_0, a_1) = \gamma_\times(a_0', a_1')$$

$$\gamma_\sqcap([a_0, a_1]_\equiv) = \gamma_0(a_0) \cap \gamma_1(a_1)$$

# Abstract program analysis

# Regular commands

regular command

atomic command

choice

Kleene star

$$
\begin{aligned}
r ::= \;& e \\
\mid \;& r_1 ; r_2 \\
\mid \;& r_1 + r_2 \\
\mid \;& r^\star
\end{aligned}
$$

$$e ::= \mathsf{skip} \mid x := a \mid b? \mid \ldots$$

$$a ::= n \mid x \mid a_1 + a_2 \mid \ldots$$

$$b ::= a_1 \leq a_2 \mid b_1 \wedge b_2 \mid \ldots$$

# Collecting semantics

$$[\![\mathsf{skip}]\!]P \triangleq P$$

$$[\![x := a]\!]P \triangleq \{\sigma[x \mapsto [\![a]\!]\sigma] \mid \sigma \in P\}$$

$$[\![b?]\!]P \triangleq [\![b]\!]P$$

$$[\![r_1; r_2]\!]P \triangleq [\![r_2]\!]([\![r_1]\!]P)$$

$$[\![r_1 + r_2]\!]P \triangleq [\![r_1]\!]P \cup [\![r_2]\!]P$$

$$[\![r^\star]\!]P \triangleq \bigcup_{k=0}^{\infty} [\![r]\!]^k P$$

# Abstract semantics

$$[\![e]\!]^{\#}_{A} a \triangleq [\![e]\!]^{A} \triangleq (\alpha \circ [\![e]\!] \circ \gamma) a$$

$$[\![r_1 ; r_2]\!]^{\#}_{A} a \triangleq [\![r_2]\!]^{\#}_{A}([\![r_1]\!]^{\#}_{A} a)$$

Just a composition of bcas!

$$[\![r_1 + r_2]\!]^{\#}_{A} a \triangleq [\![r_1]\!]^{\#}_{A} a \vee [\![r_2]\!]^{\#}_{A} a$$

$$[\![r^{\star}]\!]^{\#}_{A} a \triangleq \bigvee_{k=0}^{\infty} ([\![r]\!]^{\#}_{A})^{k} a$$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$$
\begin{aligned}
&x := 10;\\
&(\\
&\quad (x > 0)?;\\
&\quad x := x - 1;\\
&)^*;\\
&(x \leq 0)?
\end{aligned}
$$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; 
```
**{ x = 0 }?**

$[\, x \mapsto \top \,]$

$x := 10;$

$($

$(x > 0)?;$

$x := x - 1;$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

x := 10;
while (x>0) {
    x := x-1
};  **{ x = 0 }?**

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [10,10] \,]$

$\quad (x > 0)?;$

$\quad x := x - 1;$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [10,10] \,]$

$\quad (x > 0)?;$

$\quad\quad [\, x \mapsto [10,10] \,]$

$\quad x := x - 1;$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    };
```
**{ x = 0 }?**

[ x ↦ ⊤ ]

$x := 10;$

$($ [ x ↦ [10,10] ]

$(x > 0)?;$

[ x ↦ [10,10] ]

$x := x - 1;$

[ x ↦ [ 9 ,9] ]

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    };
```

**{ x = 0 }?**

$[\, \mathtt{x} \mapsto \top \,]$

$x := 10;$

$( \quad [\, \mathtt{x} \mapsto [9{,}10] \,]$

$\quad (x > 0)?;$

$\qquad [\, \mathtt{x} \mapsto [10{,}10] \,]$

$\quad x := x - 1;$

$\qquad [\, \mathtt{x} \mapsto [9{,}9] \,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    };
```

**{ x = 0 }?**

[ x ↦ ⊤ ]

$x := 10;$

( [ x ↦ [9,10] ]

$(x > 0)?;$

[ x ↦ [9,10] ]

$x := x - 1;$

[ x ↦ [9,9] ]

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    };
```
**{ x = 0 }?**

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [9,10] \,]$

$\quad (x > 0)?;$

$\quad\quad [\, x \mapsto [9,10] \,]$

$\quad x := x - 1;$

$\quad\quad [\, x \mapsto [8,9] \,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    };
```
**{ x = 0 }?**

[ x ↦ ⊤ ]

$x := 10;$

$($ [ x ↦ [8,10] ]

$(x > 0)?;$

[ x ↦ [9,10] ]

$x := x - 1;$

[ x ↦ [8,9] ]

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

x := 10;
while (x>0) {
        x := x-1
    }; **{ x = 0 }?**

$[\mathbf{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathbf{x} \mapsto [8,10]]$

$\quad (x > 0)?;$

$\quad\quad [\mathbf{x} \mapsto [8,10]]$

$\quad x := x - 1;$

$\quad\quad [\mathbf{x} \mapsto [8,9]]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$[\,\mathtt{x} \mapsto \top\,]$

$x := 10;$

$(\quad [\,\mathtt{x} \mapsto [8,10]\,]$

$\quad (x > 0)?;$

$\qquad [\,\mathtt{x} \mapsto [8,10]\,]$

$\quad x := x - 1;$

$\qquad [\,\mathtt{x} \mapsto [7,9]\,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [1,10] \,]$

$\quad (x > 0)?;$

$\qquad [\, x \mapsto [8,10] \,]$

$\quad x := x - 1;$

$\qquad [\, x \mapsto [7,9] \,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

[ x ↦ ⊤ ]

$x := 10;$

( [ x ↦ [1,10] ]

$(x > 0)?;$

[ x ↦ [ 1 ,10] ] ← Abstract loop invariant

$x := x - 1;$

[ x ↦ [ 7 ,9] ]

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

x := 10;
while (x>0) {
x := x-1
}; **{ x = 0 }?**

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [1,10] \,]$

$(x > 0)?;$

$[\, x \mapsto [1,10] \,]$ ← Abstract loop invariant

$x := x - 1;$

$[\, x \mapsto [0,9] \,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$[\,x \mapsto \top\,]$

$x := 10;$

$(\quad [\,x \mapsto [0,10]\,]$

$\quad (x > 0)?;$

$\quad\quad [\,x \mapsto [1,10]\,]$ ← Abstract loop invariant

$\quad x := x - 1;$

$\quad\quad [\,x \mapsto [0,9]\,]$

$)^*;$

$(x \leq 0)?$

# Example on Interval

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }?
```

$[\, x \mapsto \top \,]$

$x := 10;$

$( \;\; [\, x \mapsto [0, 10] \,]$

$\quad (x > 0)?;$

$\quad [\, x \mapsto [1, 10] \,]$ ← Abstract loop invariant

$\quad x := x - 1;$

$\quad [\, x \mapsto [0, 9] \,]$

$)^*; [\, x \mapsto [0, 10] \,]$

$(x \leq 0)?$

# Example on Interval

$c_1$

x := 10;
while (x>0) {
x := x-1
}; **{ x = 0 }?**

$[\,x \mapsto \top\,]$

$x := 10;$

$(\ [\,x \mapsto [0,10]\,]$

$(x > 0)?;$

$[\,x \mapsto [1,10]\,]$ ← Abstract loop invariant

$x := x - 1;$

$[\,x \mapsto [0,9]\,]$

$)^*; [\,x \mapsto [0,10]\,]$

$(x \leq 0)?$

$[\,x \mapsto [0,0]\,]$

# Example on Interval

$c_1$

x := 10;
while (x>0) {
    x := x-1
}; **{ x = 0 }?**

Complete!

$[\, x \mapsto \top \,]$

$x := 10;$

$( \quad [\, x \mapsto [0,10] \,]$

$(x > 0)?;$

$[\, x \mapsto [1,10] \,]$ ← Abstract loop invariant

$x := x - 1;$

$[\, x \mapsto [0,9] \,]$

$)^*; [\, x \mapsto [0,10] \,]$

$(x \leq 0)?$

$[\, x \mapsto [0,0] \,]$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
      };
```

$x := 10;$

(

$(x > 1)?;$

$x := x - 2;$

)*;

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    };
```

**{ x = 0 }?**

$$
\begin{aligned}
& x := 10; \\
& ( \\
& \quad (x > 1)?; \\
& \quad x := x - 2; \\
& )^*; \\
& (x \leq 1)?
\end{aligned}
$$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    }; 
```

**{ x = 0 }?**

[ x ↦ ⊤ ]

$x := 10;$

$($

$\quad (x > 1)?;$

$\quad x := x - 2;$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    };
```

**{ x = 0 }?**

$[x \mapsto \top]$

$x := 10;$

$($ $[x \mapsto [10, 10]]$

$(x > 1)?;$

$x := x - 2;$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
    x := x-2
};  **{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [10,10]]$

$\quad (x > 1)?;$

$\quad\quad [\mathtt{x} \mapsto [10,10]]$

$\quad x := x - 2;$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    }; { x = 0 }?
```

$[x \mapsto \top]$

$x := 10;$

$( \quad [x \mapsto [10,10]]$

$\quad (x > 1)?;$

$\quad\quad [x \mapsto [10,10]]$

$\quad x := x - 2;$

$\quad\quad [x \mapsto [8,8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
    x := x-2
};  **{ x = 0 }?**

$[x \mapsto \top]$

$x := 10;$

$( \quad [x \mapsto [8, 10]]$

$(x > 1)?;$

$[x \mapsto [10, 10]]$

$x := x - 2;$

$[x \mapsto [8, 8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
x := x-2
}; **{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [8,10]]$

$(x > 1)?;$

$[\mathtt{x} \mapsto [8\ ,10]]$

$x := x - 2;$

$[\mathtt{x} \mapsto [8\ ,8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
        x := x-2
    }; **{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [8,10]]$

$\quad (x > 1)?;$

$\quad [\mathtt{x} \mapsto [8,10]]$

$\quad x := x - 2;$

$\quad [\mathtt{x} \mapsto [6,8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    };
```
**{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$($ $[\mathtt{x} \mapsto [6, 10]]$

$(x > 1)?;$

$[\mathtt{x} \mapsto [8, 10]]$

$x := x - 2;$

$[\mathtt{x} \mapsto [6, 8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    }; { x = 0 }?
```

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [6, 10]]$

$\quad (x > 1)?;$

$\qquad [\mathtt{x} \mapsto [6, 10]]$

$\quad x := x - 2;$

$\qquad [\mathtt{x} \mapsto [6, 8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
    x := x-2
}; **{ x = 0 }?**

$[\mathtt{x} \mapsto \top\ ]$

$x := 10;$

$(\quad [\mathtt{x} \mapsto [\,6, 10]\,]$

$(x > 1)?;$

$[\mathtt{x} \mapsto [6\ ,10]\,]$

$x := x - 2;$

$[\mathtt{x} \mapsto [4\ ,8]\,]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
  x := x-2
};  **{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [2,10]]$

$\quad (x > 1)?;$

$\quad\quad [\mathtt{x} \mapsto [6,10]]$

$\quad x := x - 2;$

$\quad\quad [\mathtt{x} \mapsto [4,8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
x := x-2
}; **{ x = 0 }?**

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [2, 10]]$

$(x > 1)?;$

$[\mathtt{x} \mapsto [2, 10]]$ ← Abstract loop invariant

$x := x - 2;$

$[\mathtt{x} \mapsto [4, 8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
x := x-2
}; **{ x = 0 }?**

$[\mathtt{x} \mapsto \top\ ]$

$x := 10;$

$(\quad [\mathtt{x} \mapsto [2,10]]$

$(x > 1)?;$

$[\mathtt{x} \mapsto [2\ ,10]]$ ← Abstract loop invariant

$x := x - 2;$

$[\mathtt{x} \mapsto [0\ ,8]]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    }; 
```

**{ x = 0 }?**

$[\,x \mapsto \top\,]$

$x := 10;$

$( \quad [\,x \mapsto [0, 10]\,]$

$\quad (x > 1)?;$

$\quad\quad [\,x \mapsto [2, 10]\,]$ ← Abstract loop invariant

$\quad x := x - 2;$

$\quad\quad [\,x \mapsto [0, 8]\,]$

$)^*;$

$(x \leq 1)?$

# Example on Interval

$c_2$

x := 10;
while (x>1) {
x := x-2
}; **{ x = 0 }?**

$[x \mapsto \top]$

$x := 10;$

$( \quad [x \mapsto [0,10]]$

$(x > 1)?;$

$[x \mapsto [2,10]]$ ← Abstract loop invariant

$x := x - 2;$

$[x \mapsto [0,8]]$

$)^*; \quad [x \mapsto [0,10]]$

$(x \leq 1)?$

# Example on Interval

$c_2$

```
x := 10;
while (x>1) {
      x := x-2
   };
```

**{ x = 0 }?**

Incomplete!

$[\mathtt{x} \mapsto \top]$

$x := 10;$

$( \quad [\mathtt{x} \mapsto [0,10]]$

$\quad (x > 1)?;$

$\quad\quad [\mathtt{x} \mapsto [2,10]]$ ← Abstract loop invariant

$\quad x := x - 2;$

$\quad\quad [\mathtt{x} \mapsto [0,8]]$

$)^*; \quad [\mathtt{x} \mapsto [0,10]]$

$(x \leq 1)?$

$\quad [\mathtt{x} \mapsto [0,1]]$

# The precision of the analysis depends on how the program is written!!!

Complete!

Incomplete!

$c_1$

```
x := 10;
while (x>0) {
        x := x-1
    }; { x = 0 }
```

x ∈ [0,0]

Like complexity is a property of the program not of the computed function!!

$c_2$

```
x := 10;
while (x>1) {
        x := x-2
    }; { x = 0 }
```

x ∈ [0,1]

# Questions

# Question 1

Let $P \triangleq (x \in \{-7,5\})$ and $r \triangleq (x < 0)?; x := -x$.

1. Compute the abstract semantics $[\![c]\!]^{\#}_{\mathsf{Sign}}$ on $\alpha_{\mathsf{Sign}}(P)$
2. Check is the result is the same as $\alpha_{\mathsf{Sign}}([\![c]\!]P)$

# Question 2

What is the bca for the test (=0?) in the Interval domain?

$$(=0?)^A[n,m] = \begin{cases} [0,0] & \text{If } n \leq 0 \leq m \\ \bot & \text{Otherwise} \end{cases}$$

# * Exam 5

Consider the abstract domain $\mathsf{Sign}'$ in the figure

1. Define the corresponding $\alpha$ and $\gamma$.

2. Does it admit a complete abstract multiplication?

3. If not, can you add some abstract elements to $\mathsf{Sign}'$ so that a complete abstract multiplication can be designed?

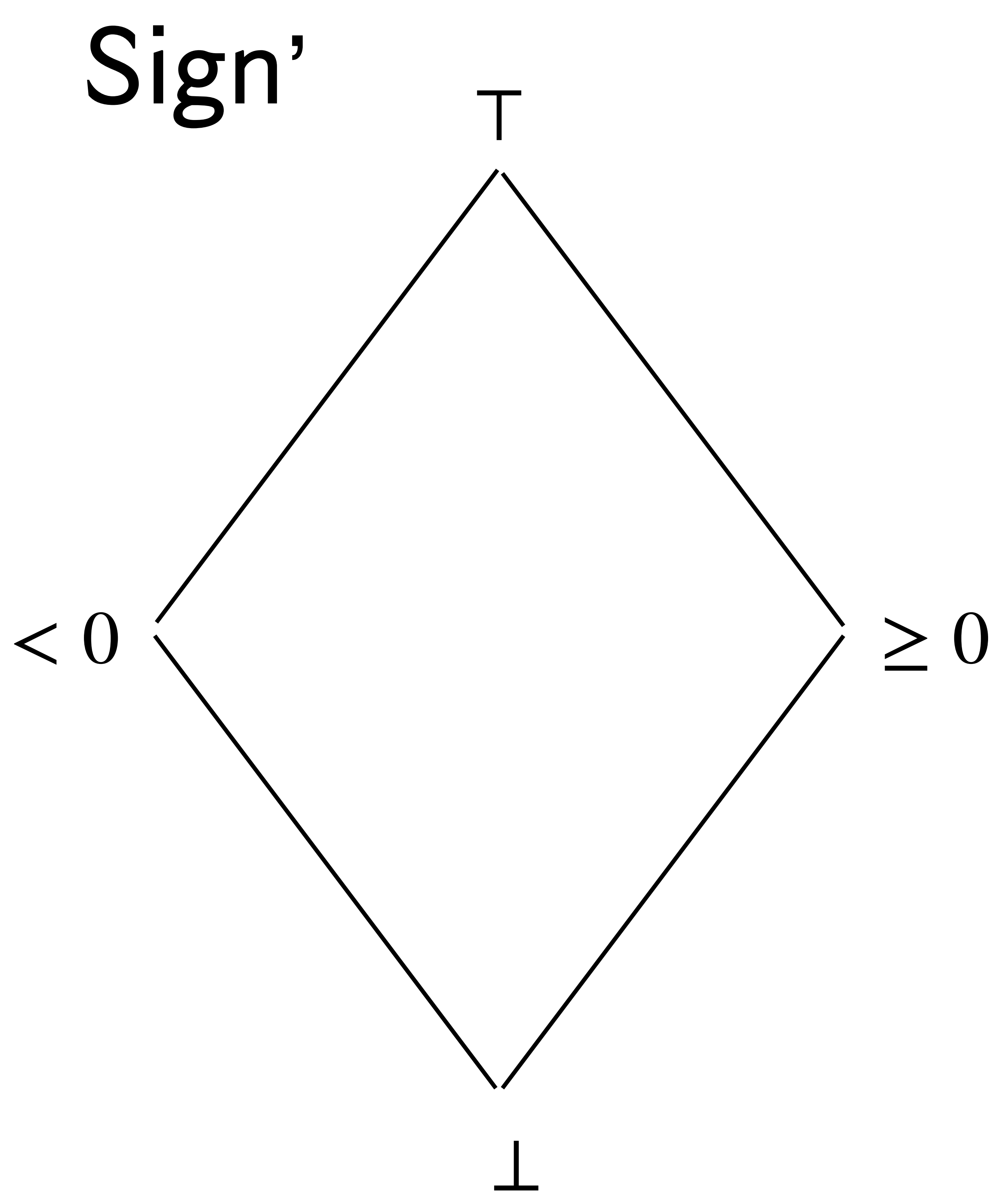
# * Exam 5

Consider the abstract domain Sign' in the figure

1. Define the corresponding $\alpha$ and $\gamma$.

2. Does it admit a complete abstract multiplication?

3. If not, can you add some abstract elements to Sign' so that a complete abstract multiplication can be designed?



# * Exam 6

Is the bca of $f : \mathbb{Z} \to \mathbb{Z}$ below complete on the Interval domain?

$$f(x) = \begin{cases} x & \text{if } x \leq 10 \\ 10 & \text{Otherwise} \end{cases}$$

# * Exam 7

Let $C \triangleq \wp(\Sigma^*)$ be the domain of sets of strings over a (finite) alphabet $\Sigma$.
Let the abstract domain be $A \triangleq \wp(\Sigma)$. Assuming $|\Sigma| \geq 2$:

1. Define suitable $\alpha$ and $\gamma$ and prove that they form a Galois Insertion.

2. Lift the concrete operation $\cdot$ of string concatenation to sets of string.

3. Define its best correct approximation.

4. Prove whether the previously defined abstract operation is complete.