# Form Corso Italia 40 to Implicit Program Analysis

Roberto Giacobazzi

UNIVERSITÀ DEGLI STUDI DI VERONA

institute iMdea software

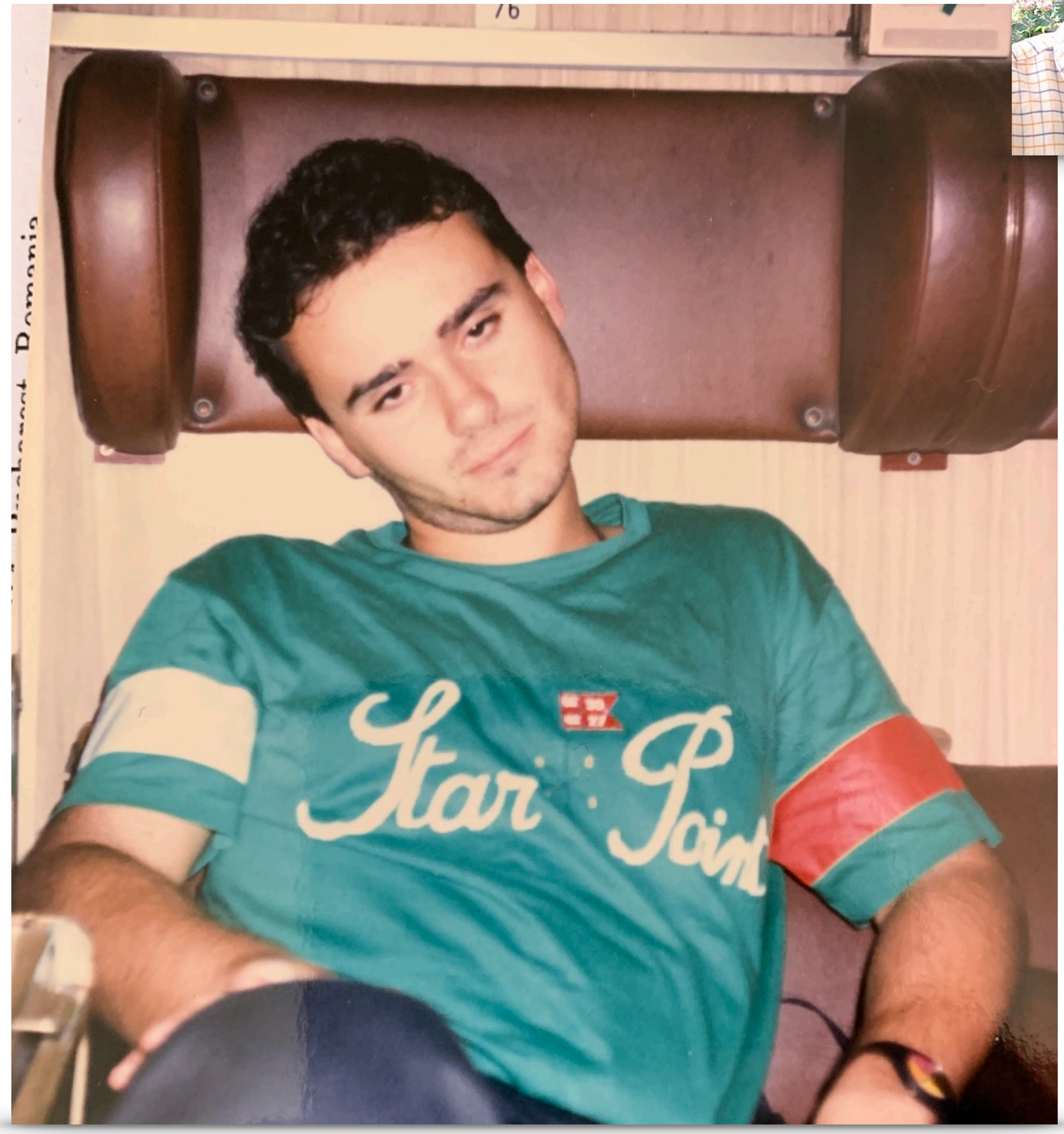
Simone's Fest — Bologna 2020

# 1982

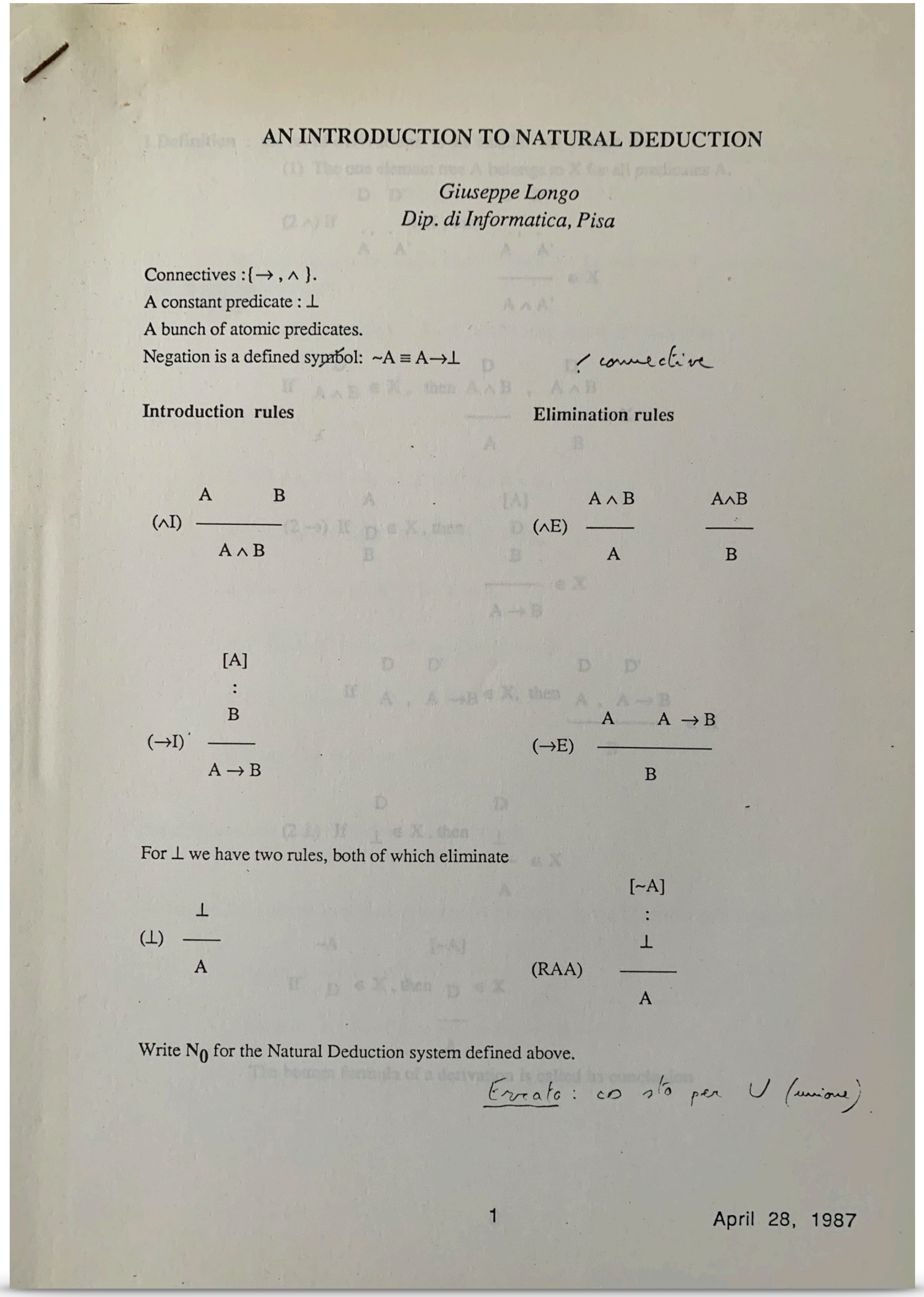

UNIVERSITA'
DEGLI STUDI DI
PISA
DIPARTIMENTO
DI SCIENZE
DELL' INFORMAZIONE

# 1982


UNIVERSITA'
DEGLI STUDI DI
PISA
DIPARTIMENTO
DI SCIENZE
DELL' INFORMAZIONE


Star Point

...1988

AN INTRODUCTION TO NATURAL DEDUCTION

*Giuseppe Longo*
*Dip. di Informatica, Pisa*

Connectives : $\{\to, \wedge\}$.
A constant predicate : $\bot$
A bunch of atomic predicates.
Negation is a defined ~~symbol~~: $\sim A \equiv A \to \bot$ connective

**Introduction rules** **Elimination rules**

$$(\wedge I)\ \frac{A \quad B}{A \wedge B} \qquad (\wedge E)\ \frac{A \wedge B}{A} \quad \frac{A \wedge B}{B}$$

$$(\to I)\ \frac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix}}{A \to B} \qquad (\to E)\ \frac{A \quad A \to B}{B}$$

For $\bot$ we have two rules, both of which eliminate

$$(\bot)\ \frac{\bot}{A} \qquad (RAA)\ \frac{\begin{matrix}[\sim A]\\ \vdots \\ \bot\end{matrix}}{A}$$

Write $N_0$ for the Natural Deduction system defined above.

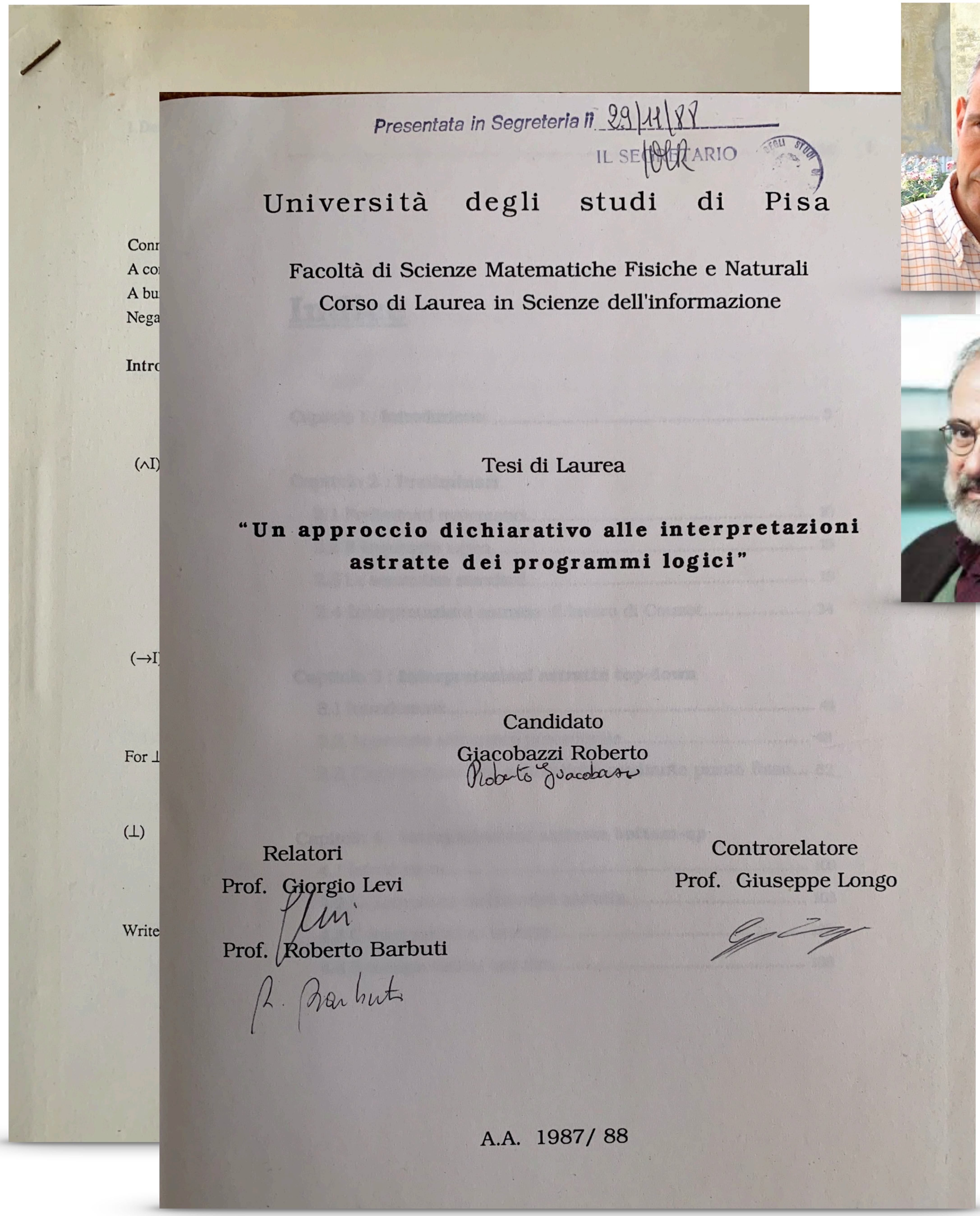
Errata: $\infty$ sta per $\cup$ (unione)

1 April 28, 1987

© Giacobazzi

# ...1988


UNIVERSITA'
DEGLI STUDI DI
PISA
DIPARTIMENTO
DI SCIENZE
DELL' INFORMAZIONE


Presentata in Segreteria il 29/11/88
IL SEGRETARIO
Università degli studi di Pisa
Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Scienze dell'informazione
Tesi di Laurea
"Un approccio dichiarativo alle interpretazioni astratte dei programmi logici"
Candidato
Giacobazzi Roberto
Roberto Giacobazzi
Relatori
Prof. Giorgio Levi
Prof. Roberto Barbuti
R. Barbuti
Controrelatore
Prof. Giuseppe Longo
A.A. 1987/ 88

1998 ... Prof !


UNIVERSITÀ · DEGLI · STUDI · DI · VERONA ·

1998 … Prof !


*Fondamenti dell'Informatica*

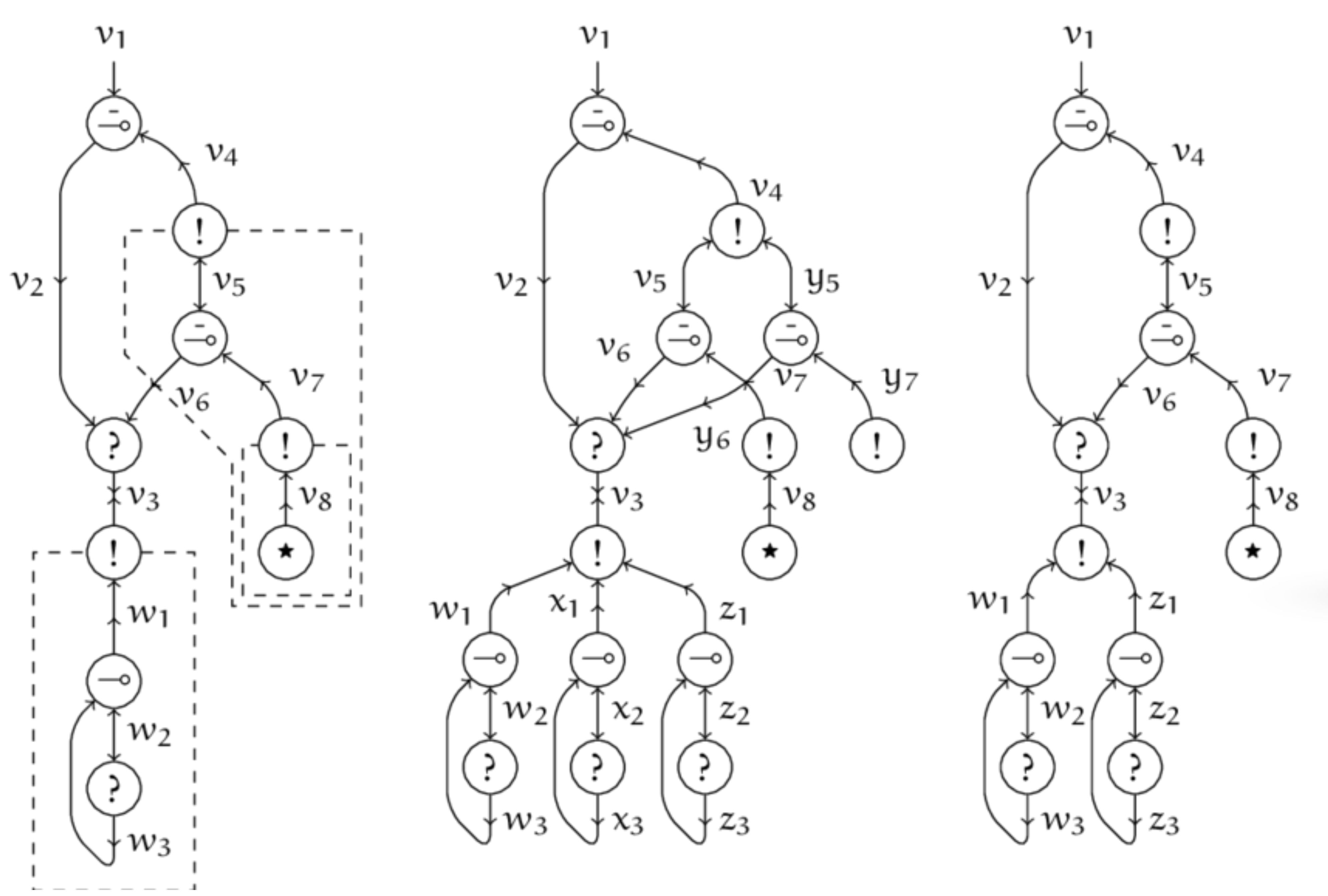
*Linguaggi Formali, Calcolabilità e Complessità*

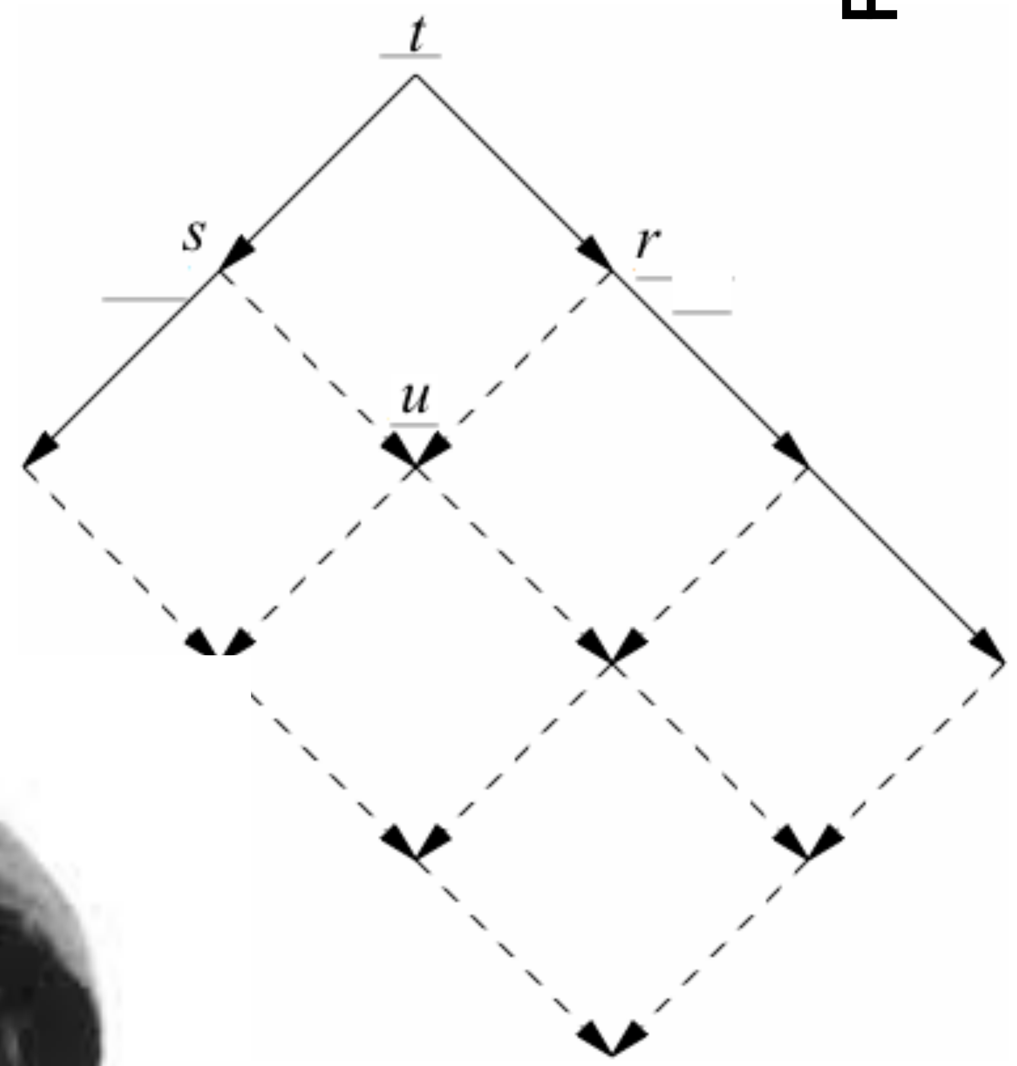
Agostino Dovier
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206, Loc. Rizzi
33100 Udine, Italy
`agostino.dovier@uniud.it`

Roberto Giacobazzi
Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15
37134 Verona, Italy
`roberto.giacobazzi@univr.it`

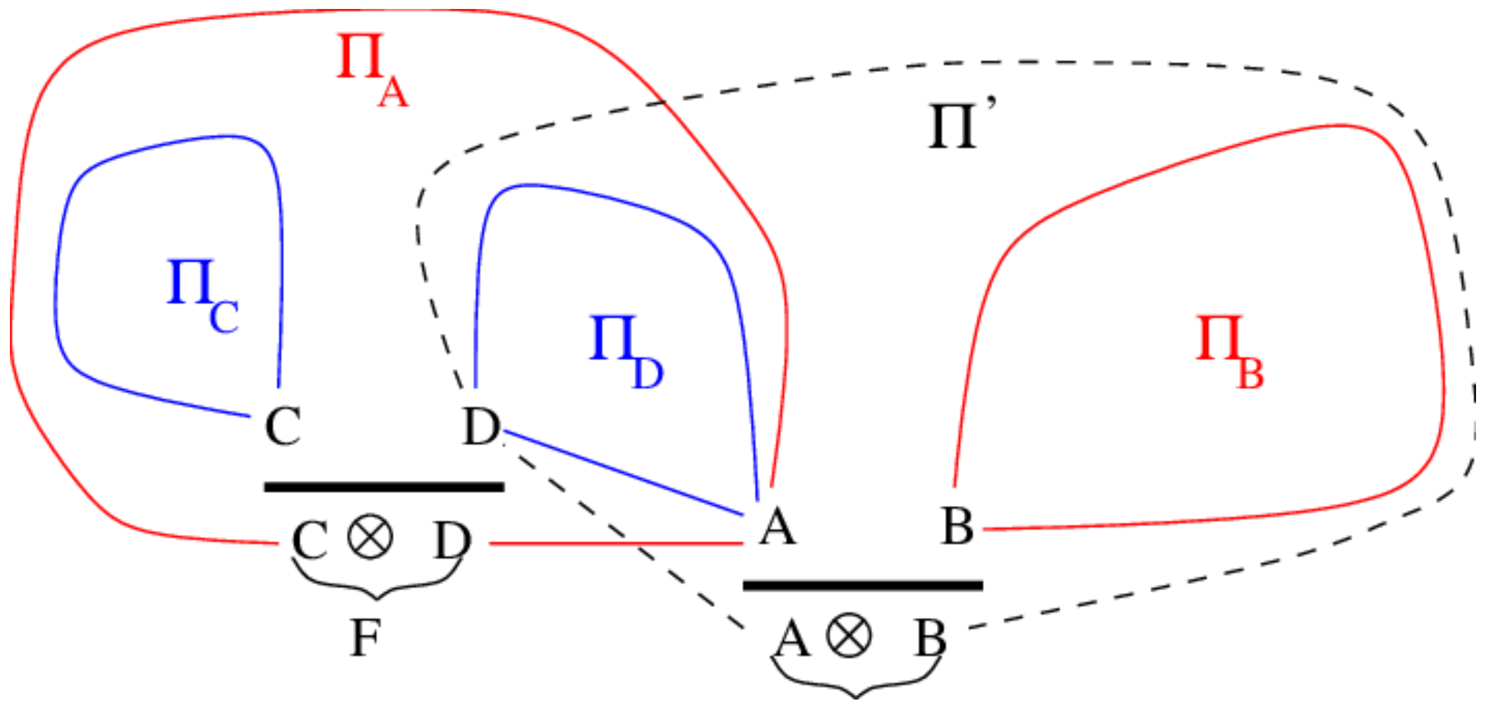
$$
\begin{aligned}
& (\lambda f.\lambda x.f\ (f\ x))(\lambda f.\lambda x.f\ (f\ x)) \\
\rightsquigarrow_\alpha\ & (\lambda f.\lambda x.f\ (f\ x))(\lambda g.\lambda y.g\ (g\ y)) \\
\rightsquigarrow_\beta\ & \lambda x.(\lambda g.\lambda y.g\ (g\ y))((\lambda g.\lambda y.g\ (g\ y))\ x) \\
\rightsquigarrow_\beta\ & \lambda x.(\lambda g.\lambda y.g\ (g\ y))(\lambda y.x\ (x\ y)) \\
\rightsquigarrow_\alpha\ & \lambda x.(\lambda g.\lambda y.g\ (g\ y))(\lambda z.x\ (x\ z)) \\
\rightsquigarrow_\beta\ & \lambda x.\lambda y.(\lambda z.x\ (x\ z))((\lambda z.x\ (x\ z))\ y) \\
\rightsquigarrow_\beta\ & \lambda x.\lambda y.(\lambda z.x\ (x\ z))(x\ (x\ y)) \\
\rightsquigarrow_\beta\ & \lambda x.\lambda y.x\ (x\ (x\ y))
\end{aligned}
$$

$$[\![P]\!] \models$$



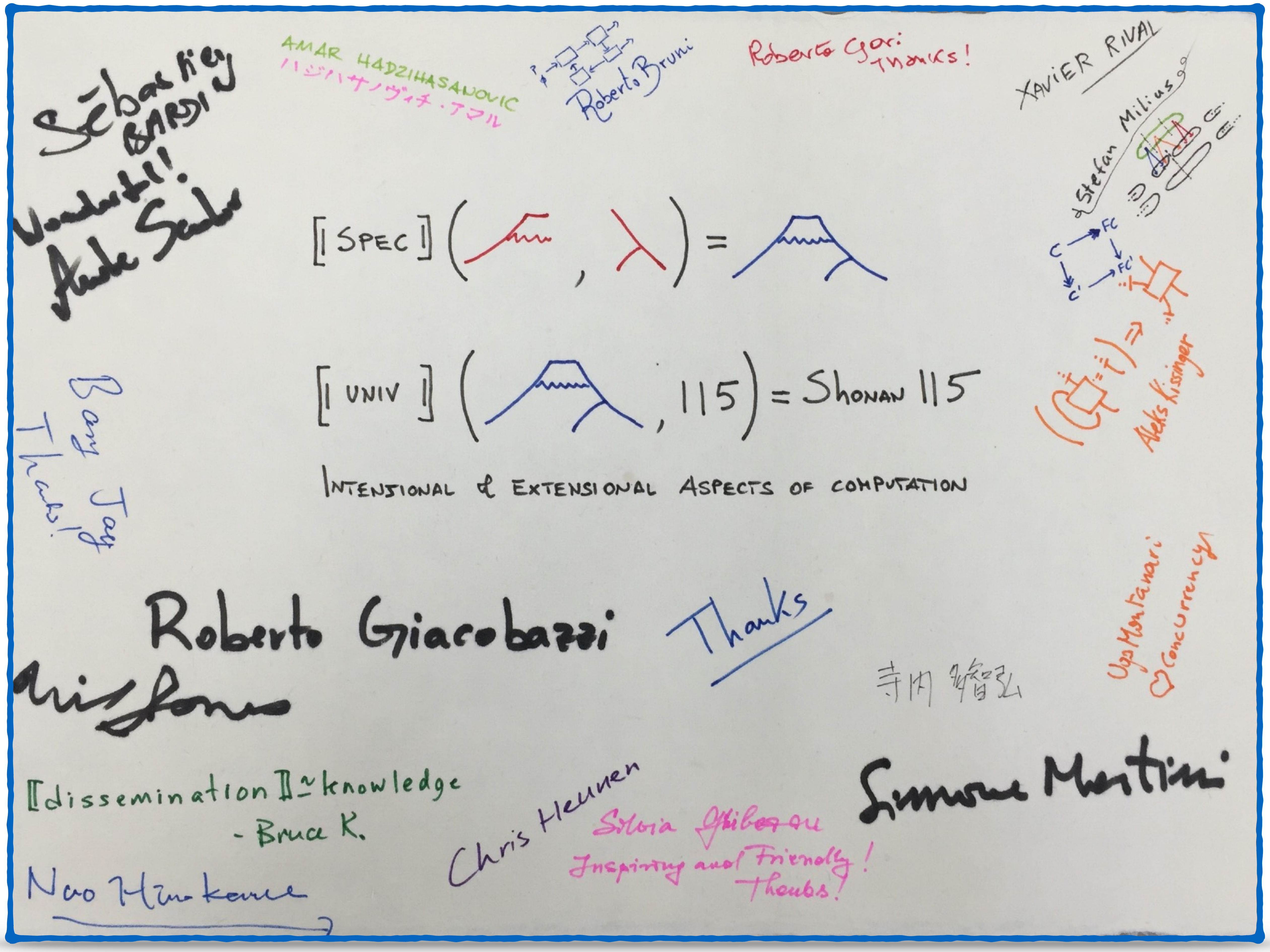

# Are we eventually working in the same field?

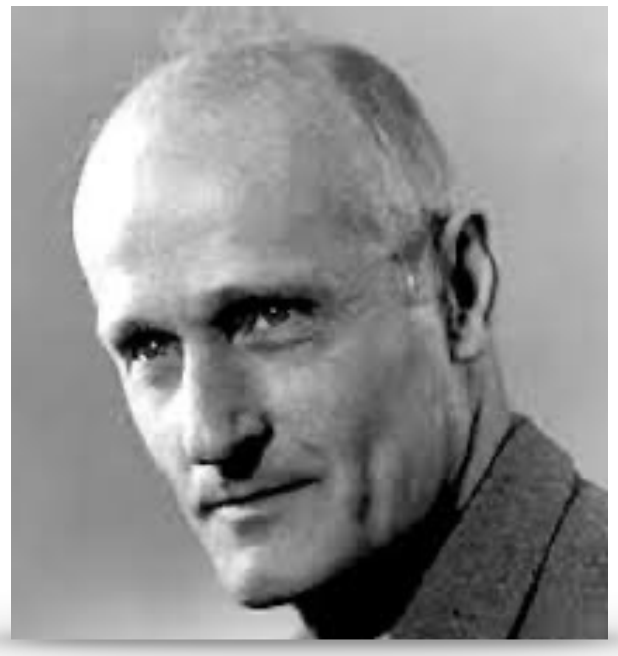



$$\frac{\Gamma \vdash A : \mathrm{sProp}_i \qquad \Gamma \vdash 🍎 : A \qquad \Gamma \vdash 🍊 : A}{\Gamma \vdash 🍎 \equiv 🍊 : A}$$

# The Standard Model is to PL what movement without friction is to mechanics.

[S. Martini]

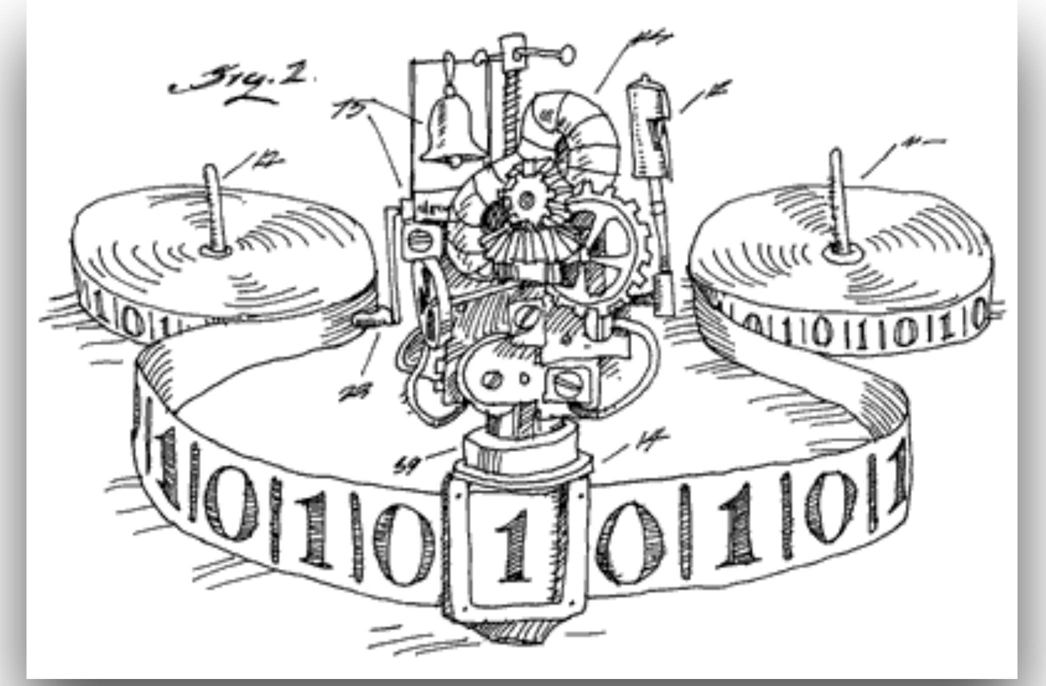


How much of the Standard Model holds in Program Analysis?

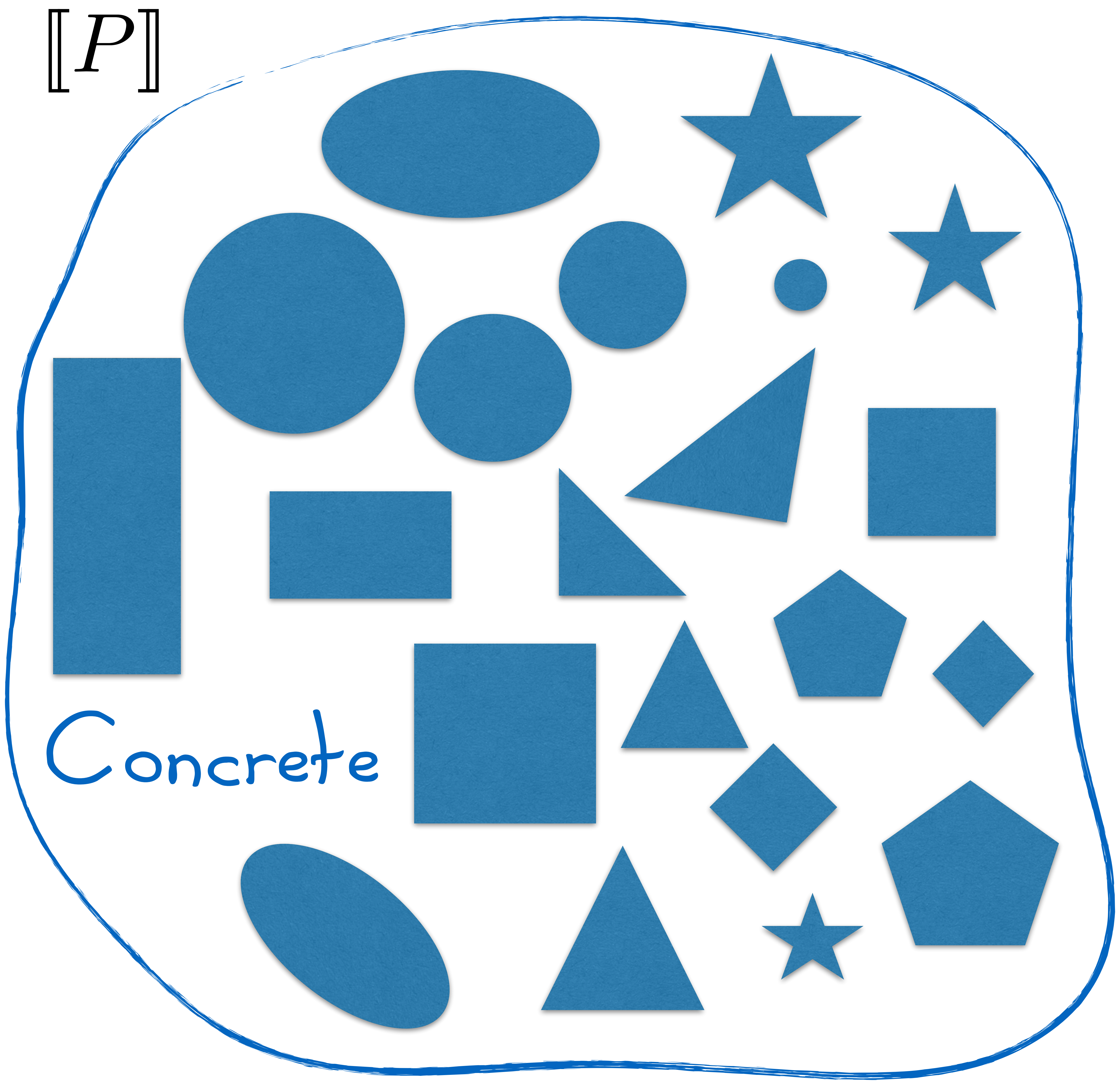
# What is Program Analysis?

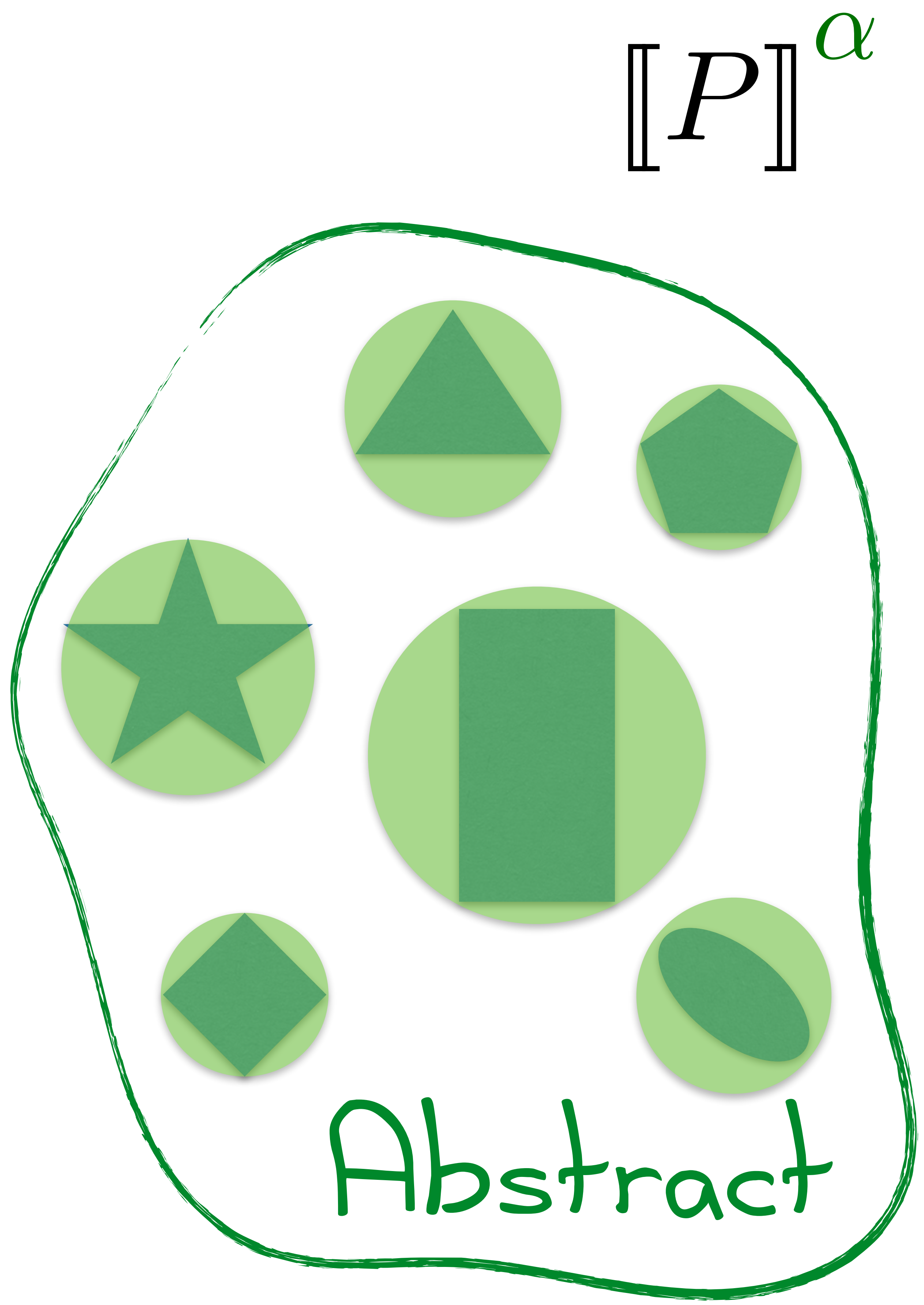
```
n := n0;
i := n;
while (i <> 0 ) do
      j := 0;
      while (j <> i) do
            j := j + 1
      od;
      i := i - 1
od
```



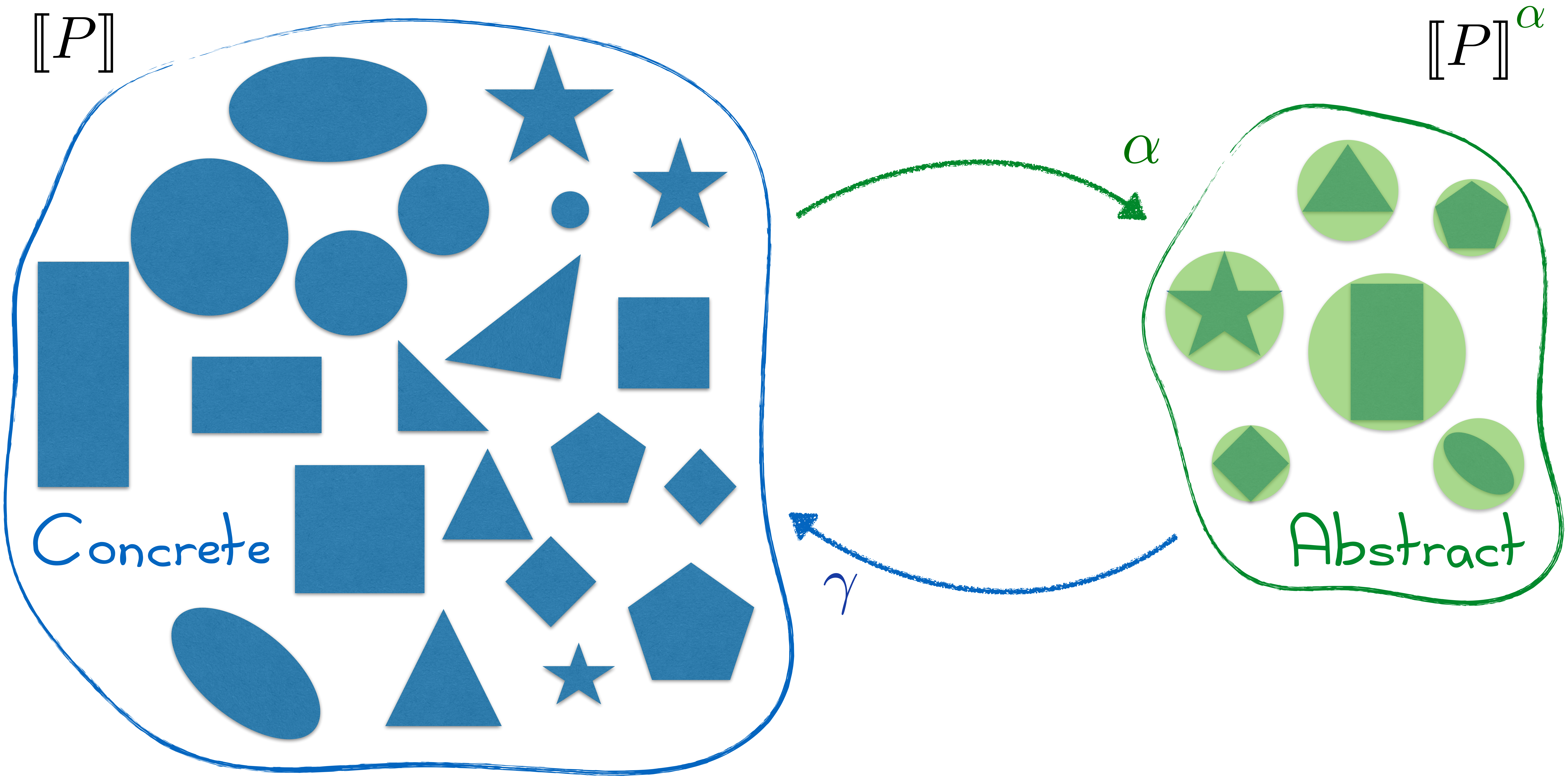
```
{n0>=0}
   n := n0;
{n0=n,n0>=0}
   i := n;
{n0=i,n0=n,n0>=0}
   while (i <> 0 ) do
      {n0=n,i>=1,n0>=i}
         j := 0;
      {n0=n,j=0,i>=1,n0>=i}
         while (j <> i) do
            {n0=n,j>=0,i>=j+1,n0>=i}
               j := j + 1
            {n0=n,j>=1,i>=j,n0>=i}
         od;
      {n0=n,i=j,i>=1,n0>=i}
         i := i - 1
      {i+1=j,n0=n,i>=0,n0>=i+1}
   od
{n0=n,i=0,n0>=0}
```

# What is Program Analysis?

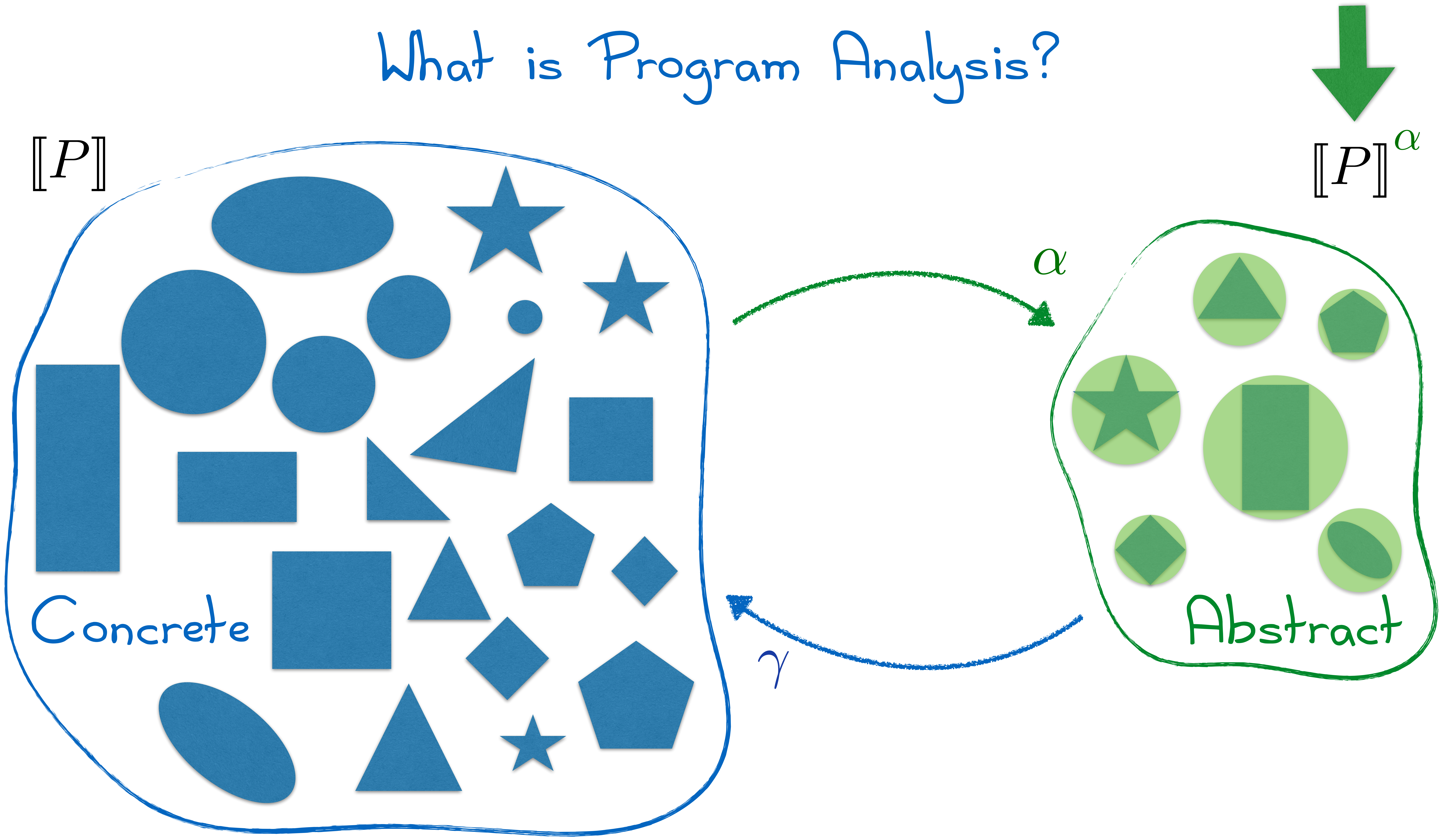


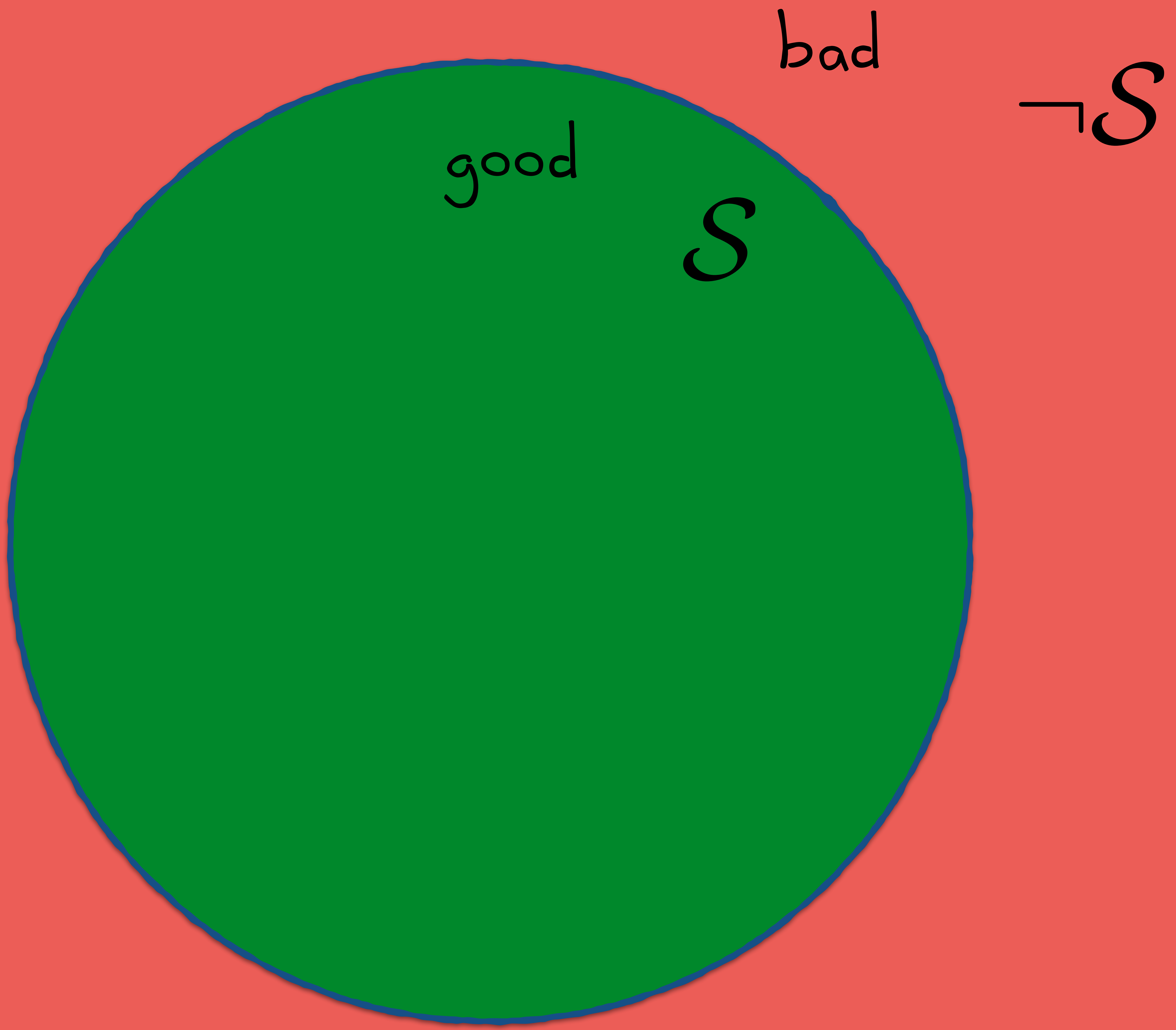



# What is Program Analysis?

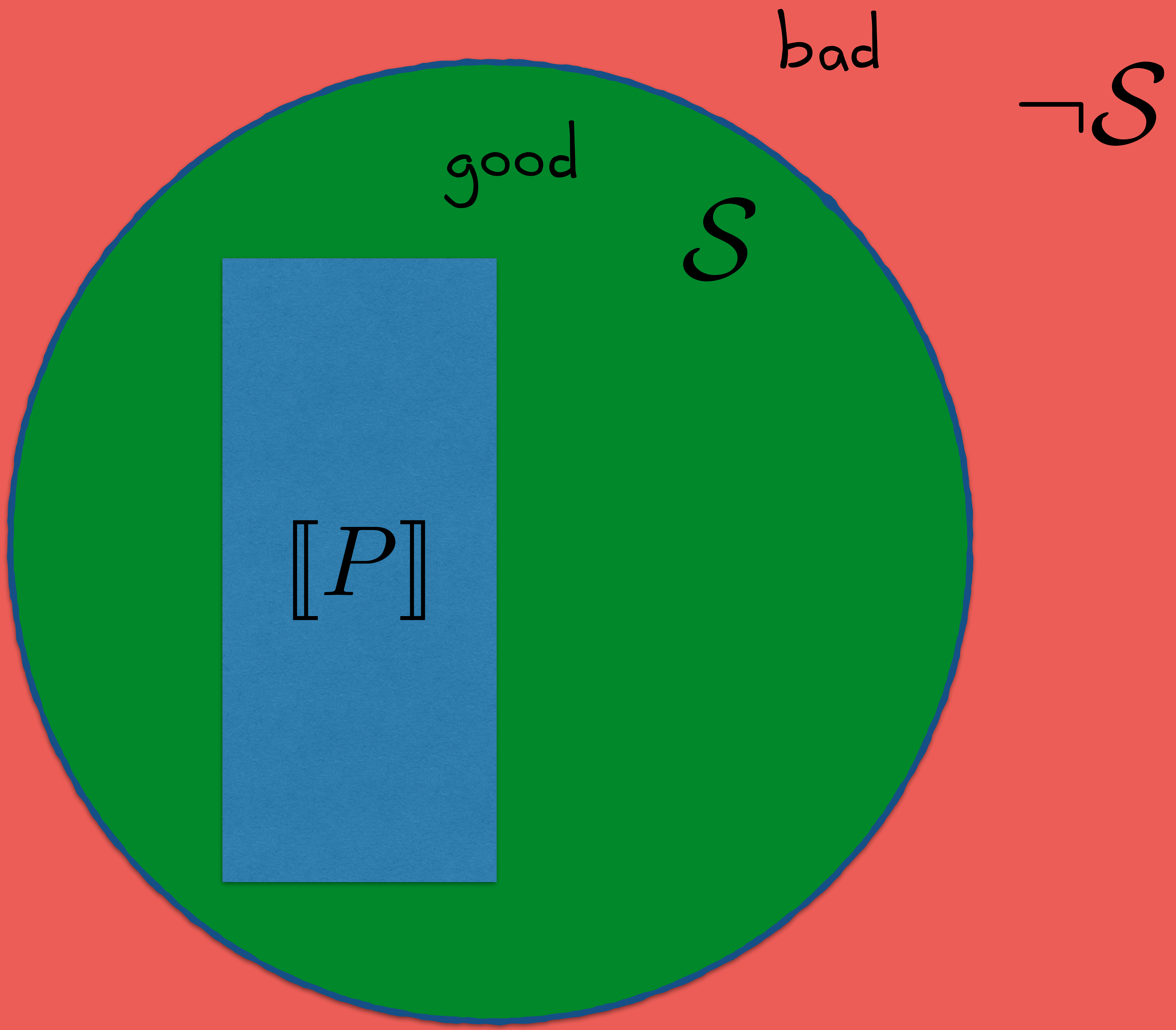


# What is Program Analysis?



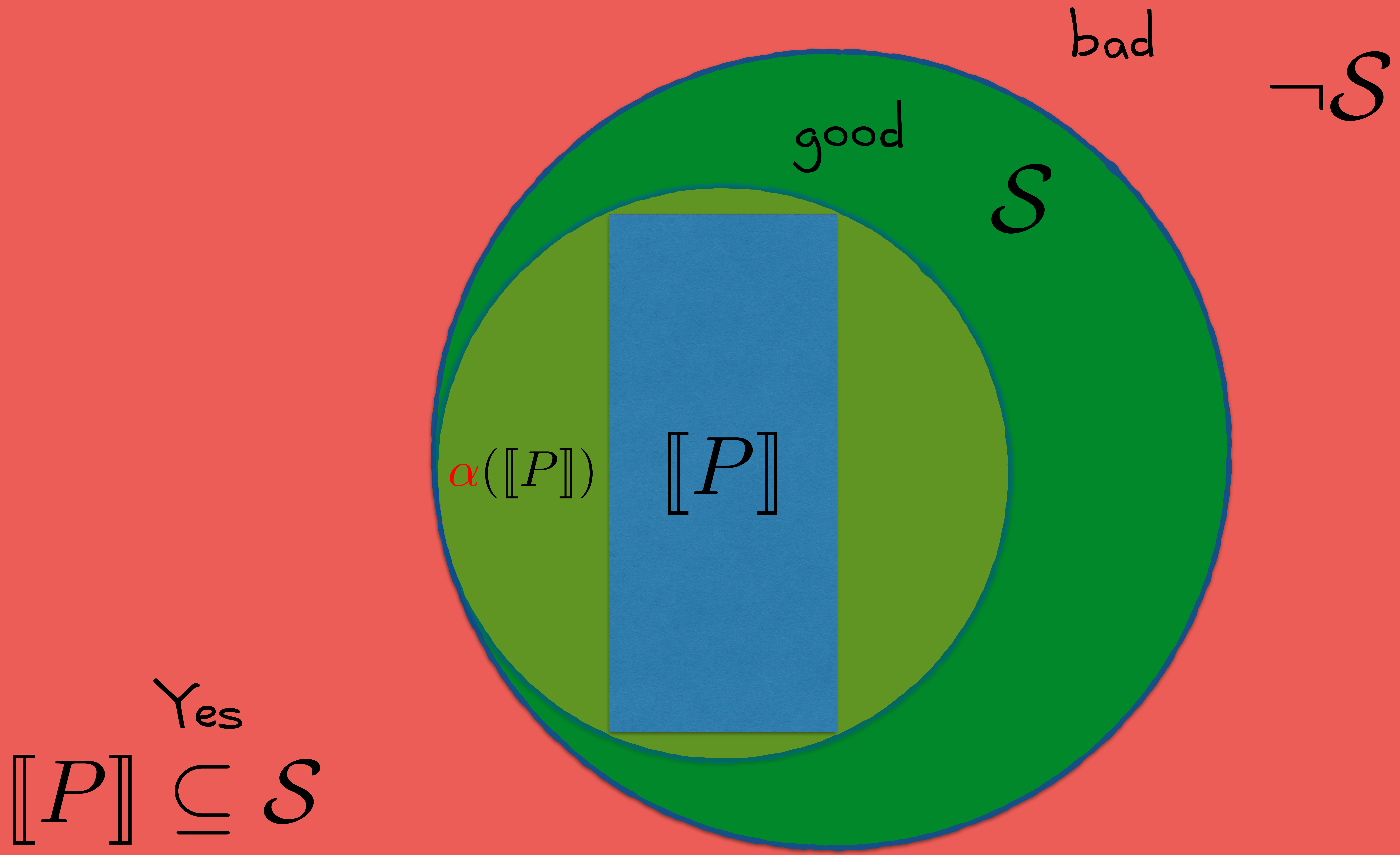



$$[\![P]\!] \subseteq \mathcal{S}$$



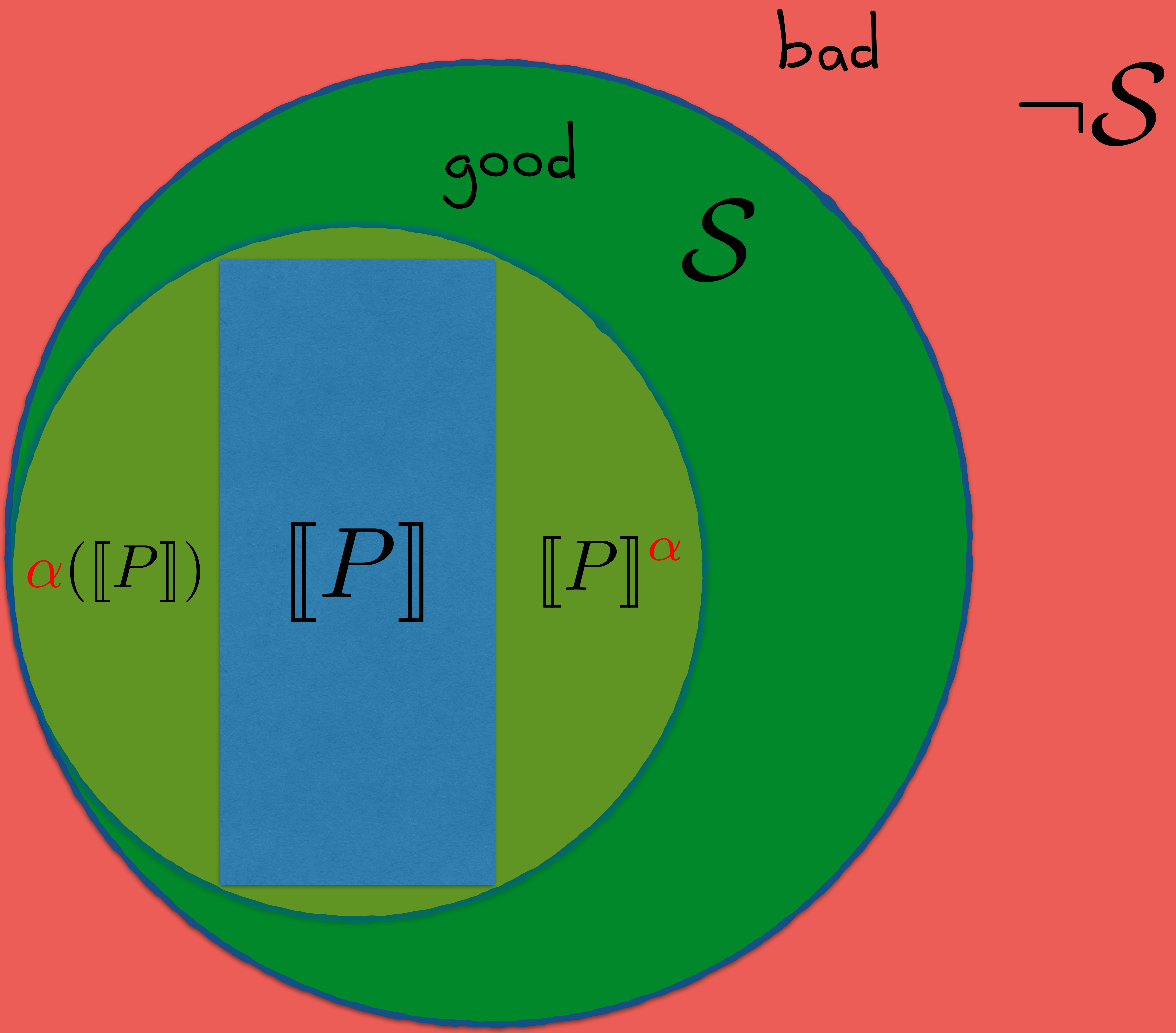

Yes

$$[\![P]\!] \subseteq \mathcal{S}$$



bad
$\neg\mathcal{S}$
good
$\mathcal{S}$
$\alpha([\![P]\!])$
$[\![P]\!]$
Yes
$[\![P]\!] \subseteq \mathcal{S}$

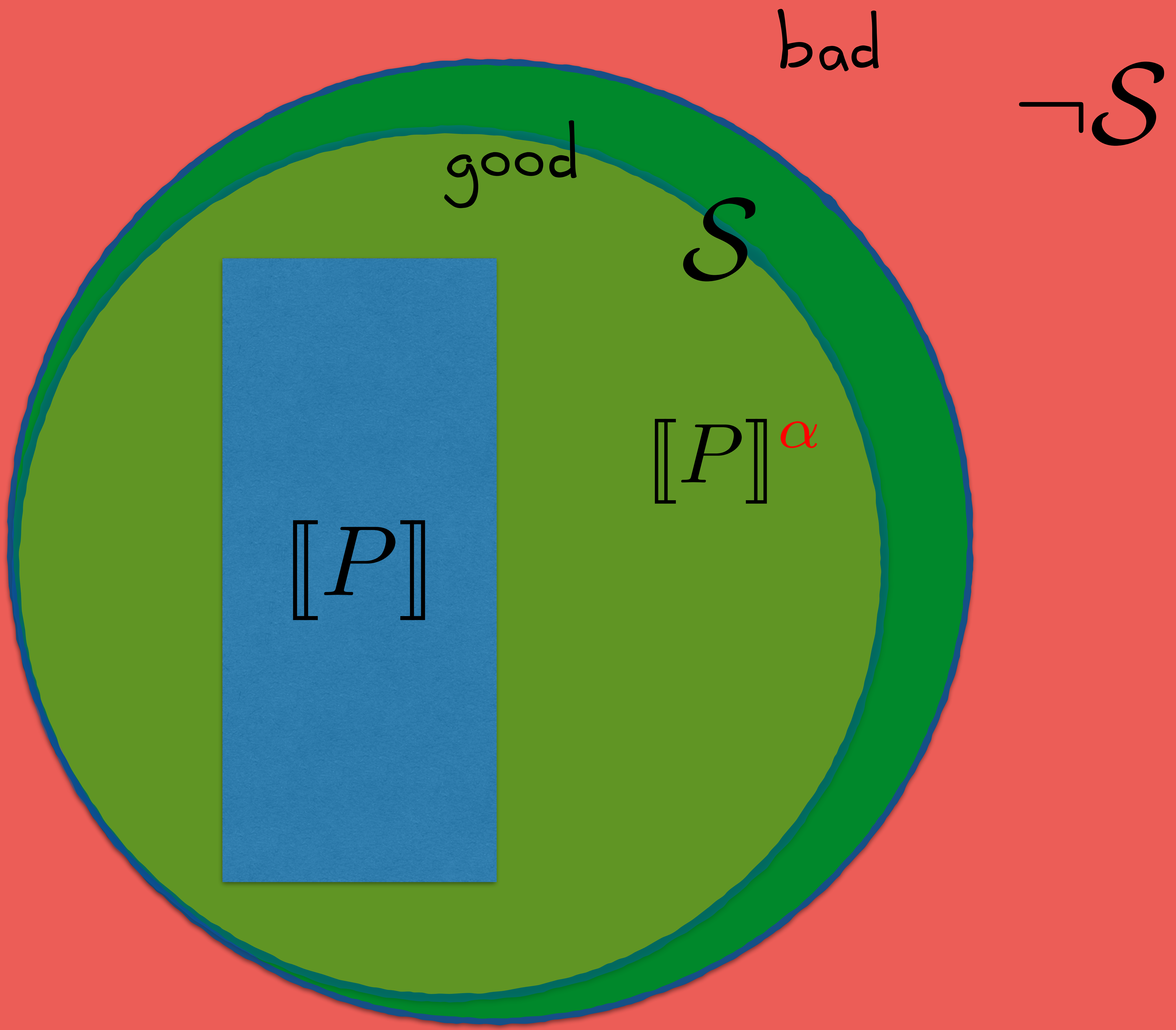
$$\alpha(\llbracket P \rrbracket) = \llbracket P \rrbracket^{\alpha}$$




Yes

$$\llbracket P \rrbracket \subseteq \mathcal{S}$$

$$\alpha([\![P]\!]) \subseteq [\![P]\!]^{\alpha}$$

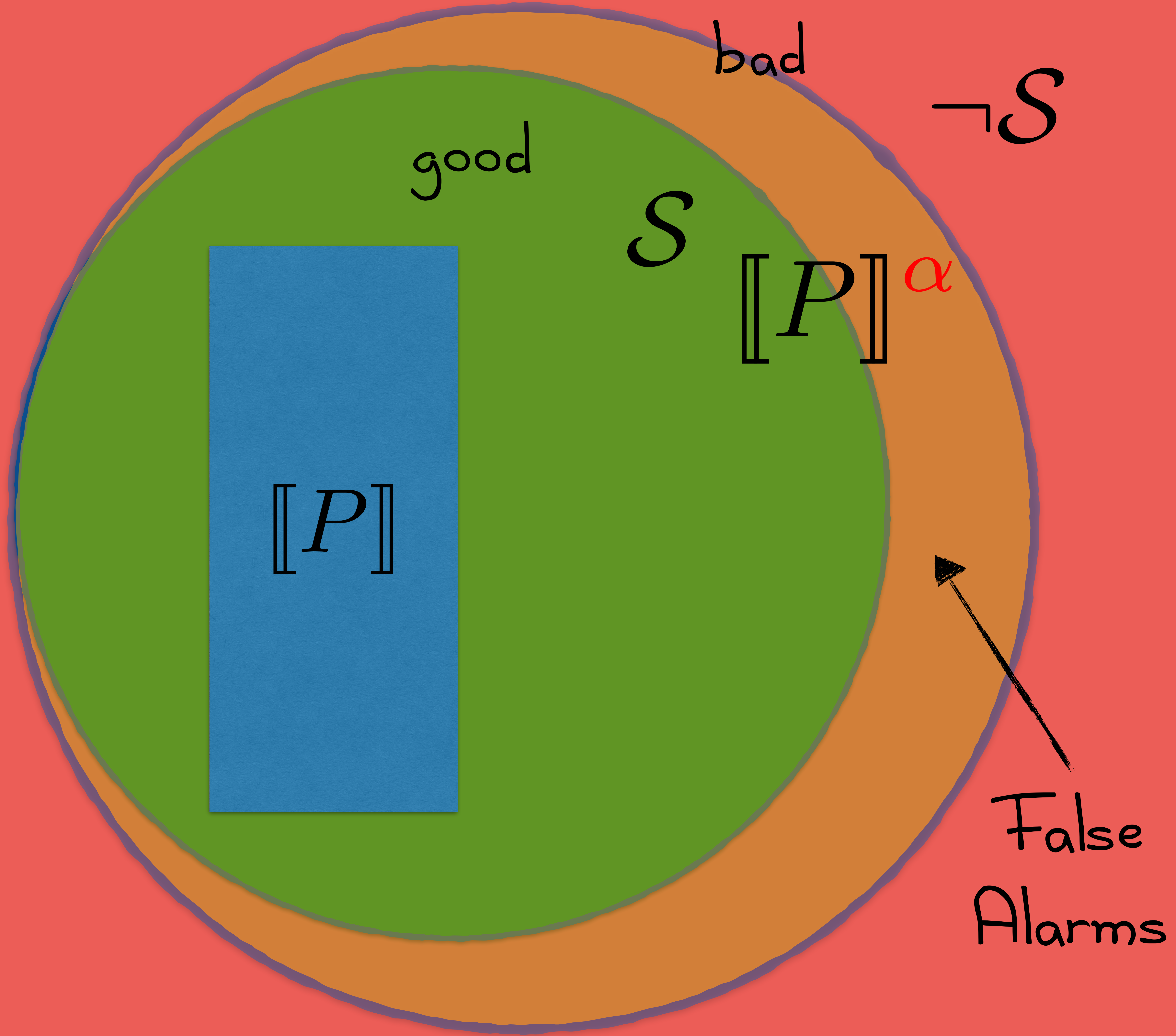



Yes

$$[\![P]\!] \subseteq \mathcal{S}$$

$$\alpha(\llbracket P \rrbracket) \subseteq \llbracket P \rrbracket^{\alpha}$$

$$\llbracket P \rrbracket \overset{?}{\subseteq} \mathcal{S}$$



$$P \sim Q$$

Program equivalence

$$P \sim Q \iff [\![P]\!] = [\![Q]\!]$$

Program equivalence

$$P \overset{?}{\sim} Q \iff [\![P]\!]^{\alpha} = [\![Q]\!]^{\alpha}$$

Program equivalence
by
Abstract Interpreters?

{ x int } `x := 10;`

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

{ x = 0 }

$\sim$

{ x int } `x := 10;`

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

{ x = 0 }

{ x int } `x := 10;`

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

$\sim$

{ x int } `x := 10;`

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

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

$$\alpha(\llbracket P \rrbracket) = \llbracket P \rrbracket^{\alpha}$$

[0,10] ∧ (x ≤ 0) = { x ∈ [0,0] }

$$\sim \quad \not\sim$$

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

$$\alpha(\llbracket P \rrbracket) \subset \llbracket P \rrbracket^{\alpha}$$

[0,10] ∧ (x ≤ 1) = { x ∈ [0,1] }

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

$$\alpha([\![P]\!]) = [\![P]\!]^\alpha$$

[0,10] $\wedge$ (x $\leq$ 0) = { x $\in$ [0,0] }

Complete!

$\sim$ $\not\sim$

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

$$\alpha([\![P]\!]) \subset [\![P]\!]^\alpha$$

[0,10] $\wedge$ (x $\leq$ 1) = { x $\in$ [0,1] }

Incomplete!

```
{ x int } x := -9;
          while
             (x<0)
             {
               x := x+2
             }; { x = 1 }
```

$$\not\sim$$

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

{ x int }

```
x := -9;
while
    (x<0)
    {
      x := x+2
    };
```

[-9,1]

{ x = 1 }

$$\not\sim$$

{ x int }

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

[0,10]

{ x = 0 }

```
{ x int } x := -9;
          while
             (x<0)
             {
               x := x+2
             }; { x = 1 }
```

$$\alpha(\llbracket P \rrbracket) \subset \llbracket P \rrbracket^{\alpha}$$

$[-9,1] \wedge (x \geq 0) = \{ x \in [0,1] \}$

$$\not\sim \quad \sim$$

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

$$\alpha(\llbracket P \rrbracket) \subset \llbracket P \rrbracket^{\alpha}$$

$[0,10] \wedge (x \leq 1) = \{ x \in [0,1] \}$

{ x int }
```
x := -9;
while
    (x<0)
    {
      x := x+2
    };
```
{ x = 1 }

$$\alpha([\![P]\!]) \subset [\![P]\!]^{\alpha}$$

[-9,1] $\wedge$ (x $\geq$ 0) = { x $\in$ [0,1] }

Incomplete!

$\not\sim$ $\sim$

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

$$\alpha([\![P]\!]) \subset [\![P]\!]^{\alpha}$$

[0,10] $\wedge$ (x $\leq$ 1) = { x $\in$ [0,1] }

Incomplete!

# On Completeness Classes

POPL2015

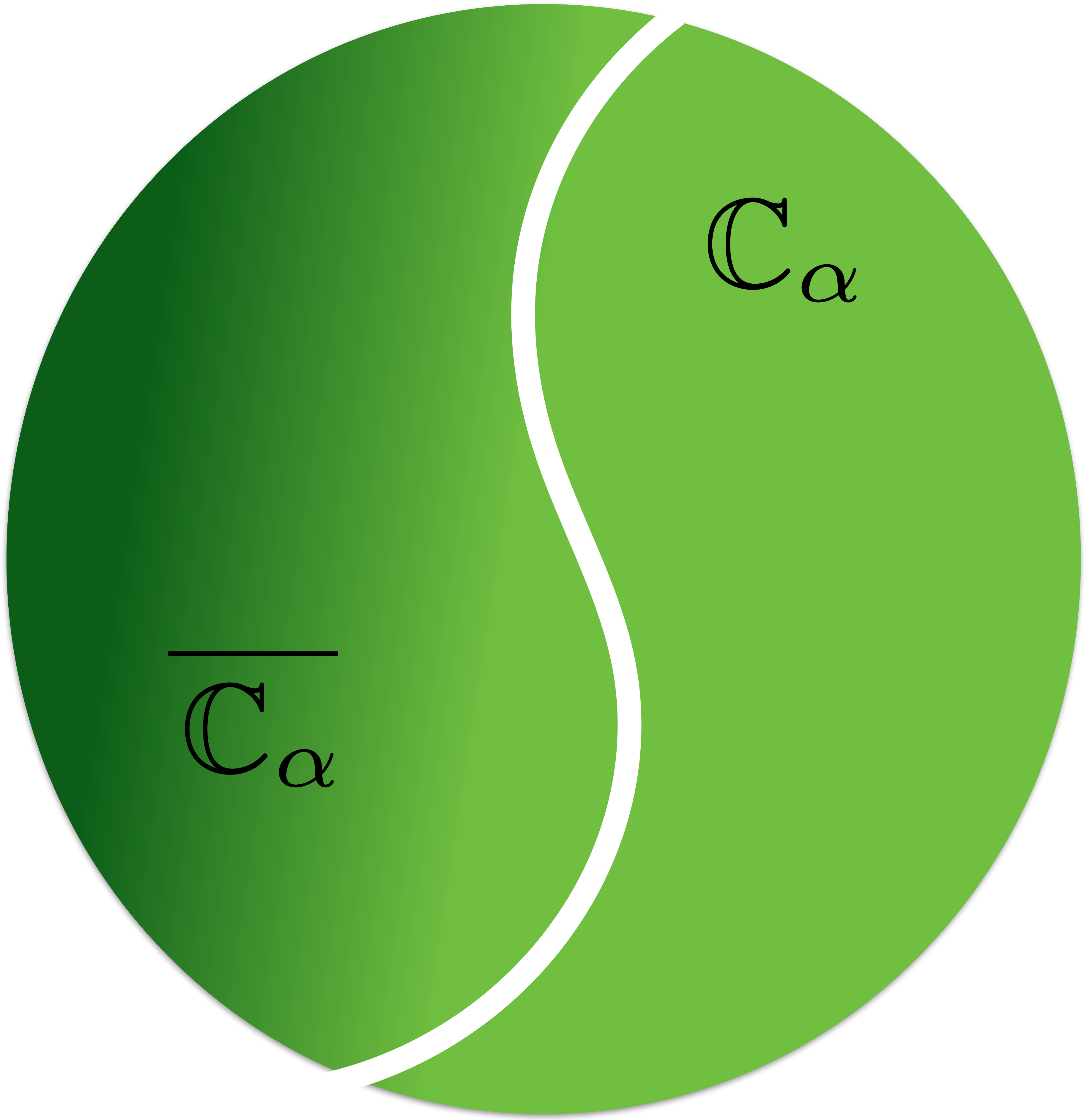
$$\mathbb{C}_\alpha = \{P \in \text{Programs} \mid \alpha(\llbracket P \rrbracket) = \llbracket P \rrbracket^\alpha\}$$

$$\begin{tikzcd} \text{Incomplete} \quad \overline{\mathbb{C}_\alpha} \arrow[r, bend right] & \mathbb{C}_\alpha \quad \text{Complete} \arrow[l, bend right] \end{tikzcd}$$

$$\overline{\mathbb{C}_\alpha} = \{P \in \text{Programs} \mid \alpha(\llbracket P \rrbracket) \neq \llbracket P \rrbracket^\alpha\}$$

# On Completeness Classes

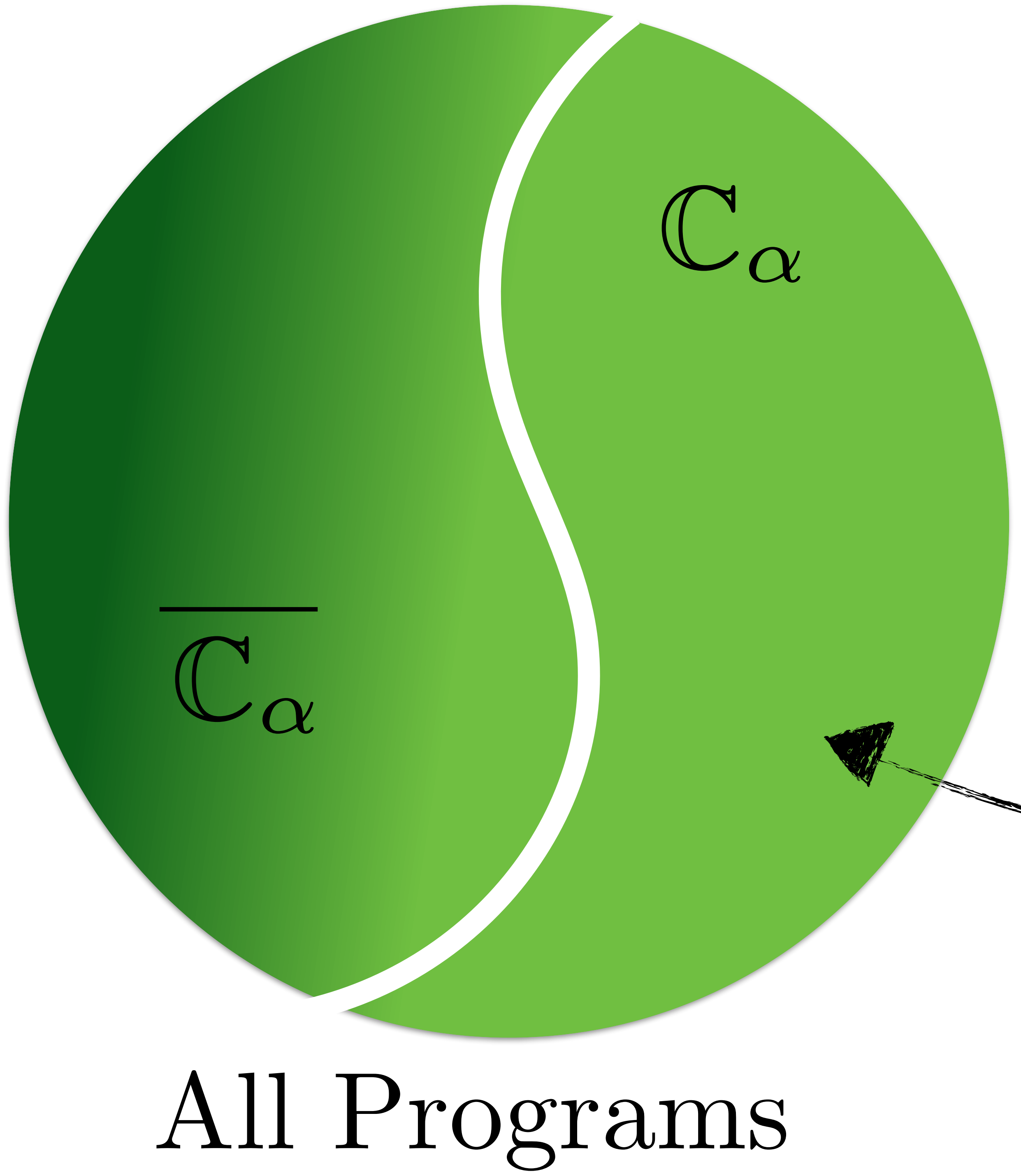
POPL2015




All Programs

# On Completeness Classes

POPL2015



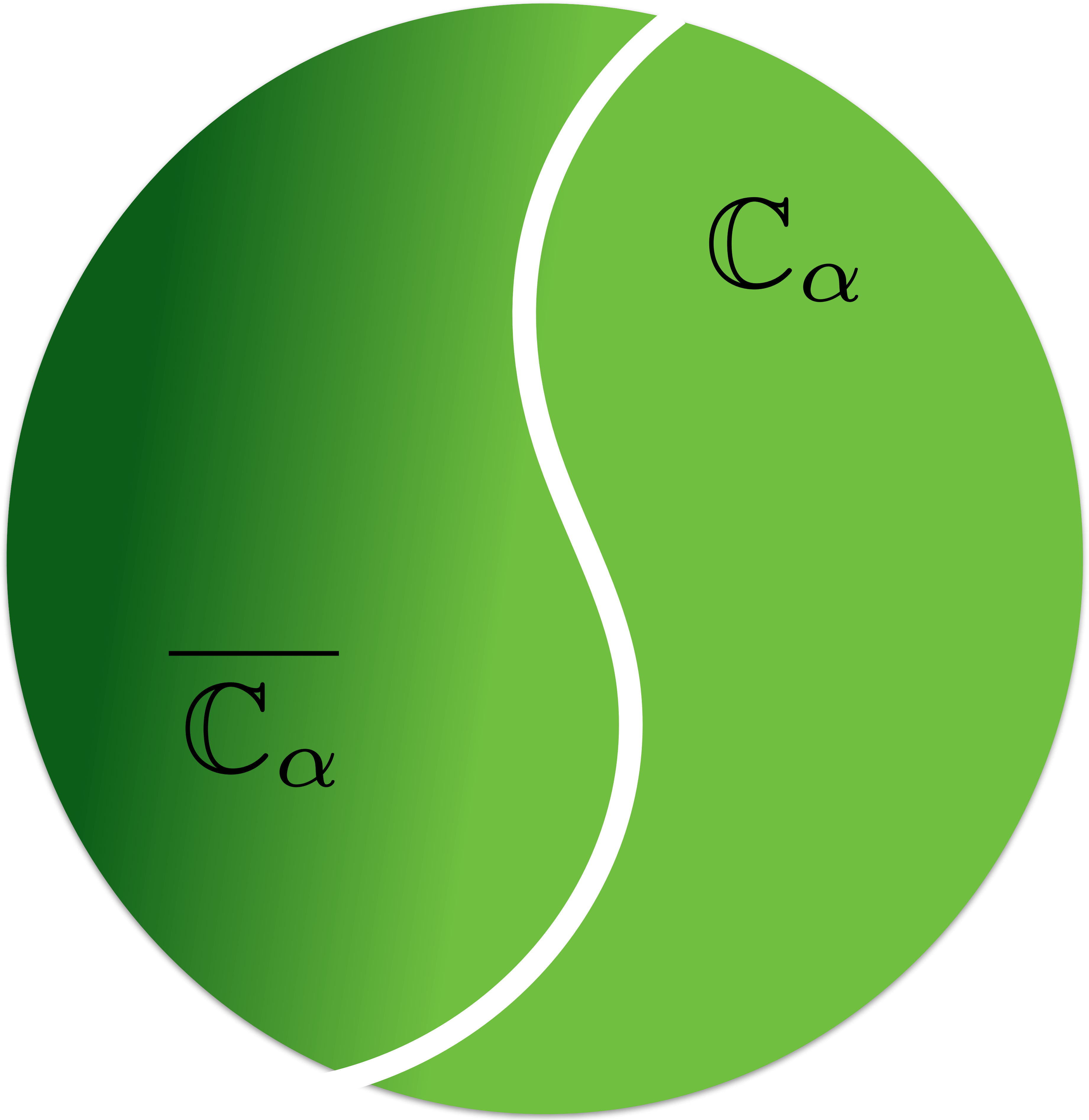

Infinite

**skip;**
**skip; skip;**
**skip; skip; skip;**
**skip; skip; skip; skip;**
......

Standard padding!

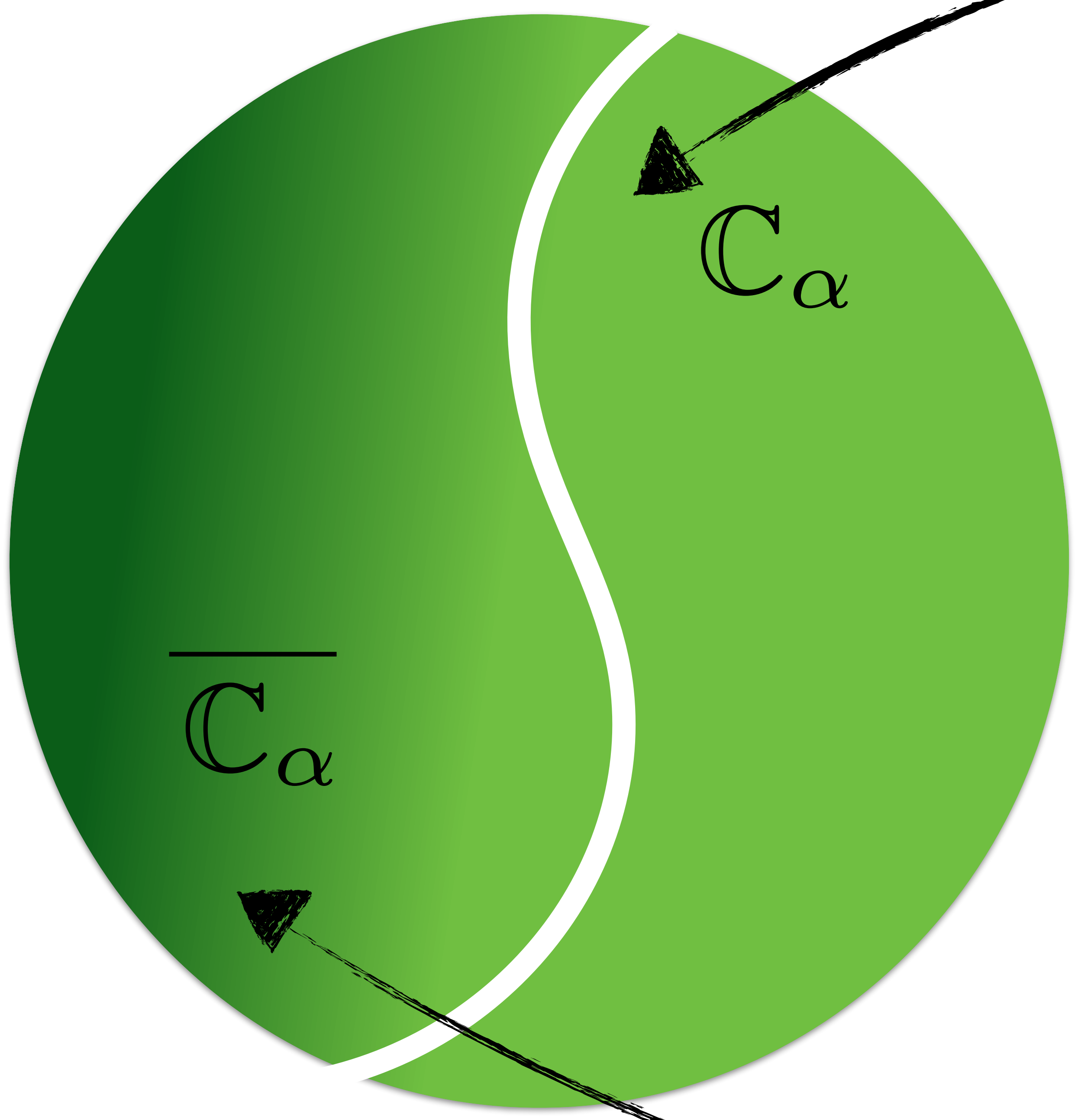
# On Completeness Classes

POPL2015



# On Completeness Classes

POPL2015




Non Extensional

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

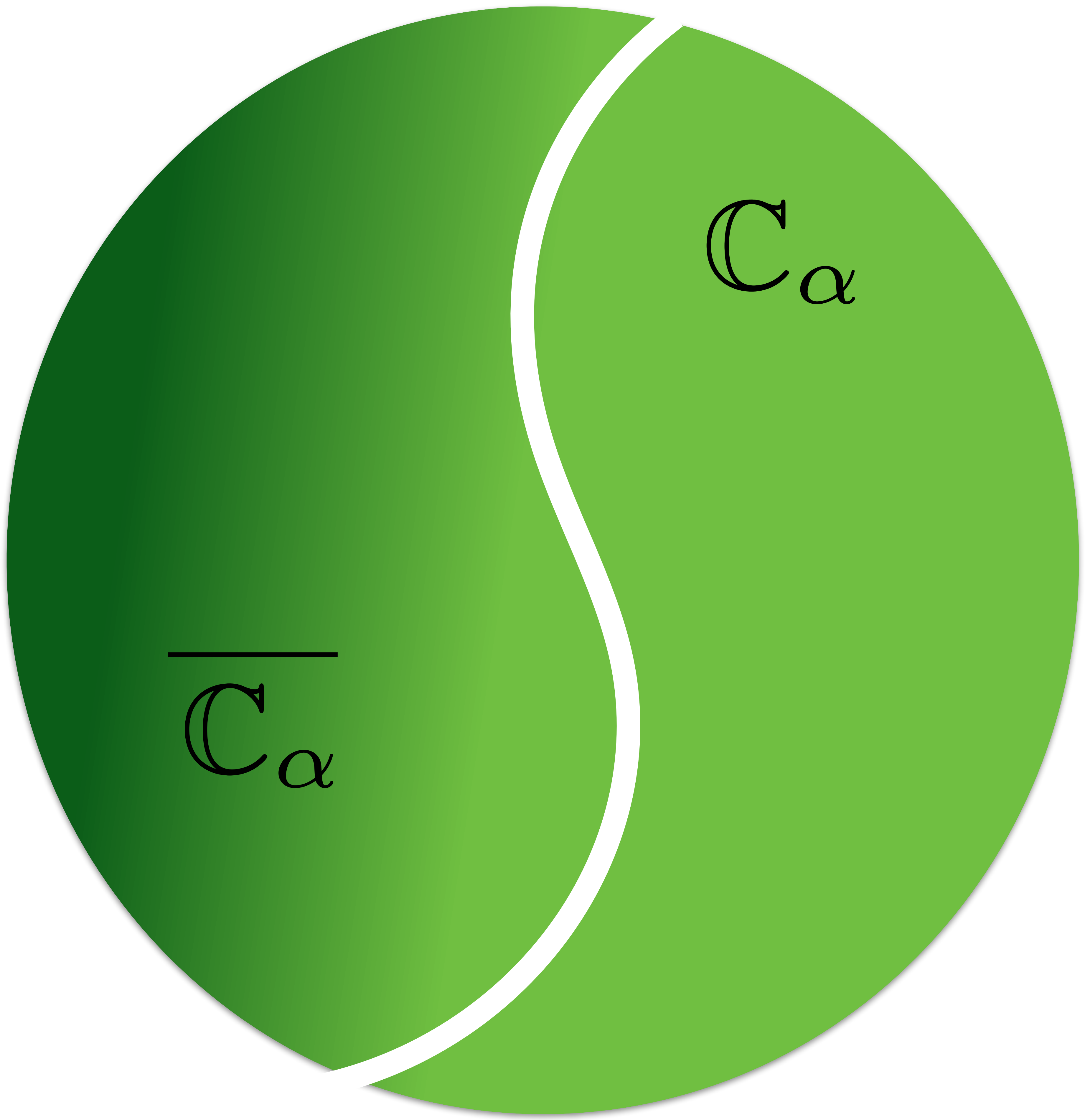
$x \in [0,0]$

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

$x \in [0,1]$

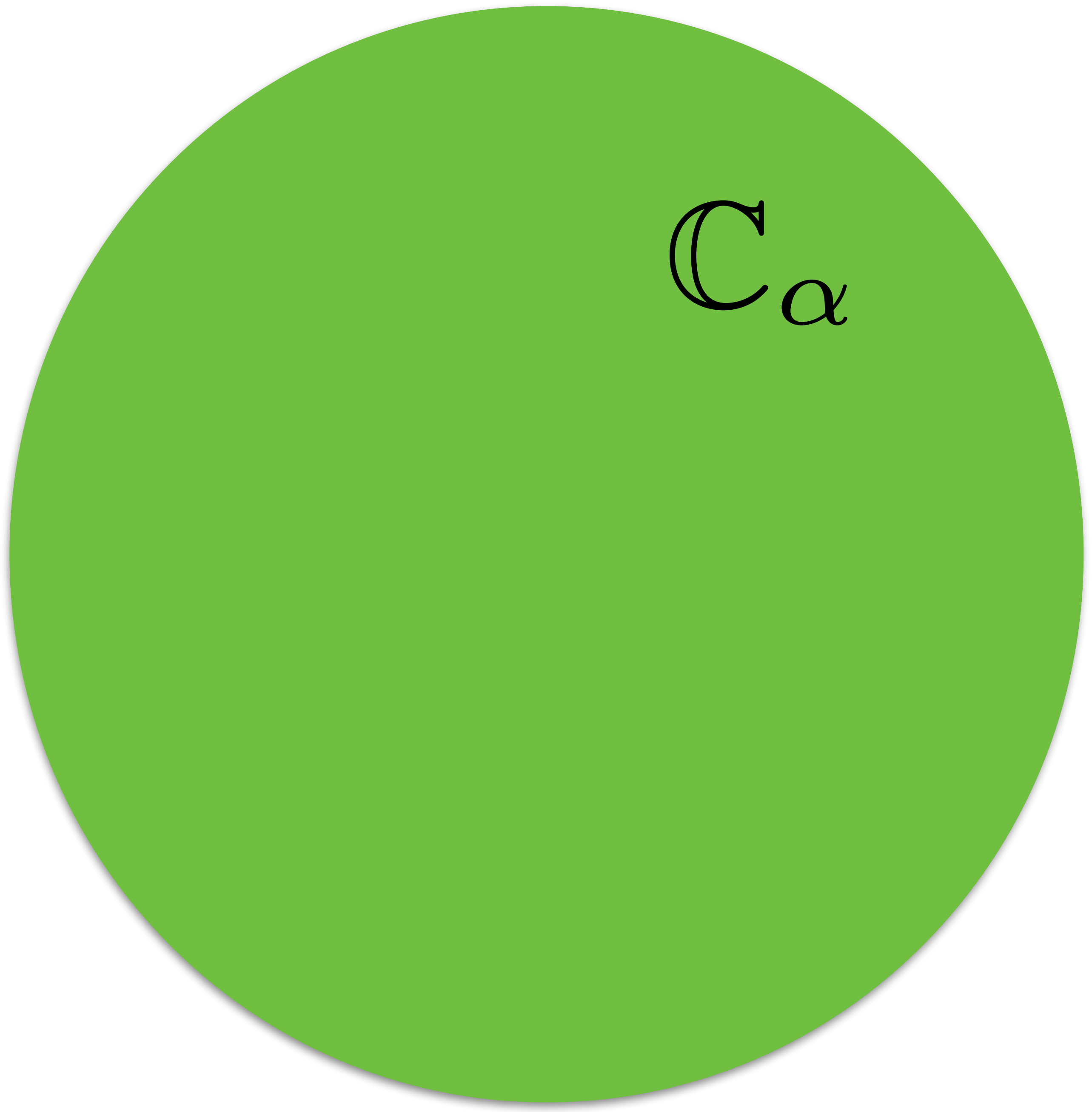
# On Completeness Classes

POPL2015



# On Completeness Classes

POPL2015




$$\mathbb{C}_\alpha = \text{All Programs}$$

$$\Leftrightarrow \ \alpha \in \{\lambda x.x,\ \lambda x.\top\}$$

Similar to Rice Theorem!!

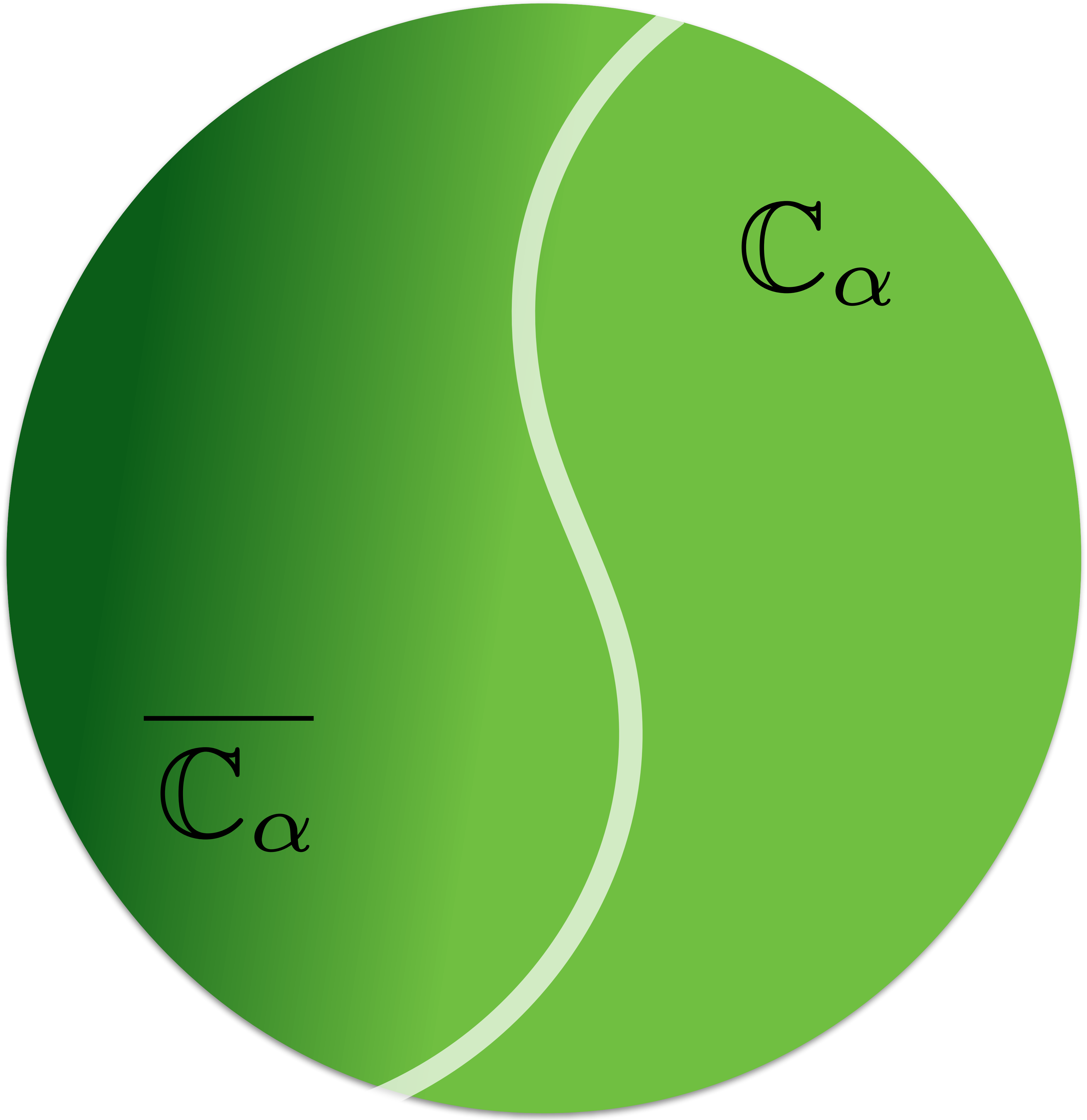
Whenever we can see something but not all

there exists a program for which you cannot be precise!

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

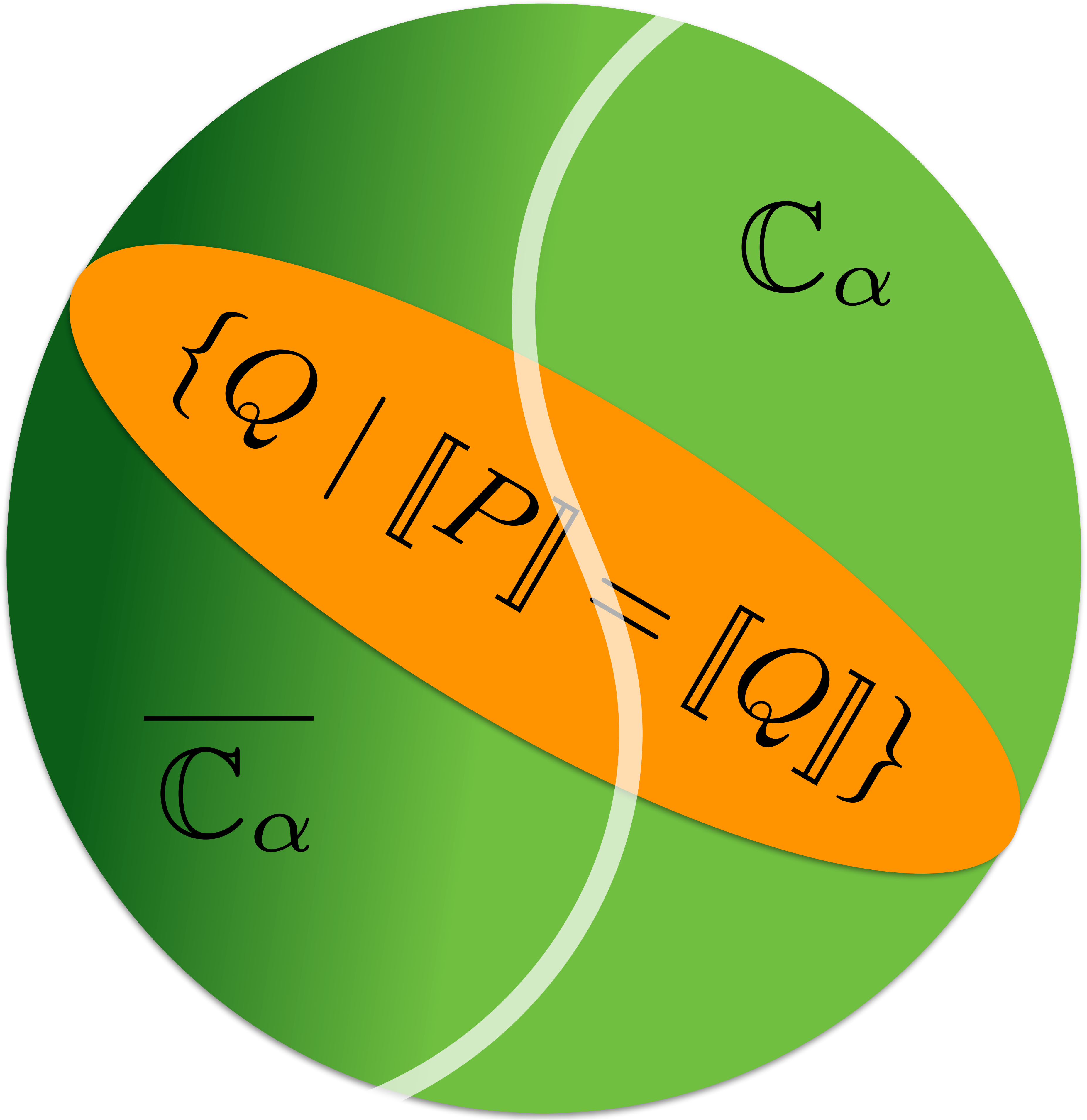
POPL2020



# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \mathrm{Programs}$

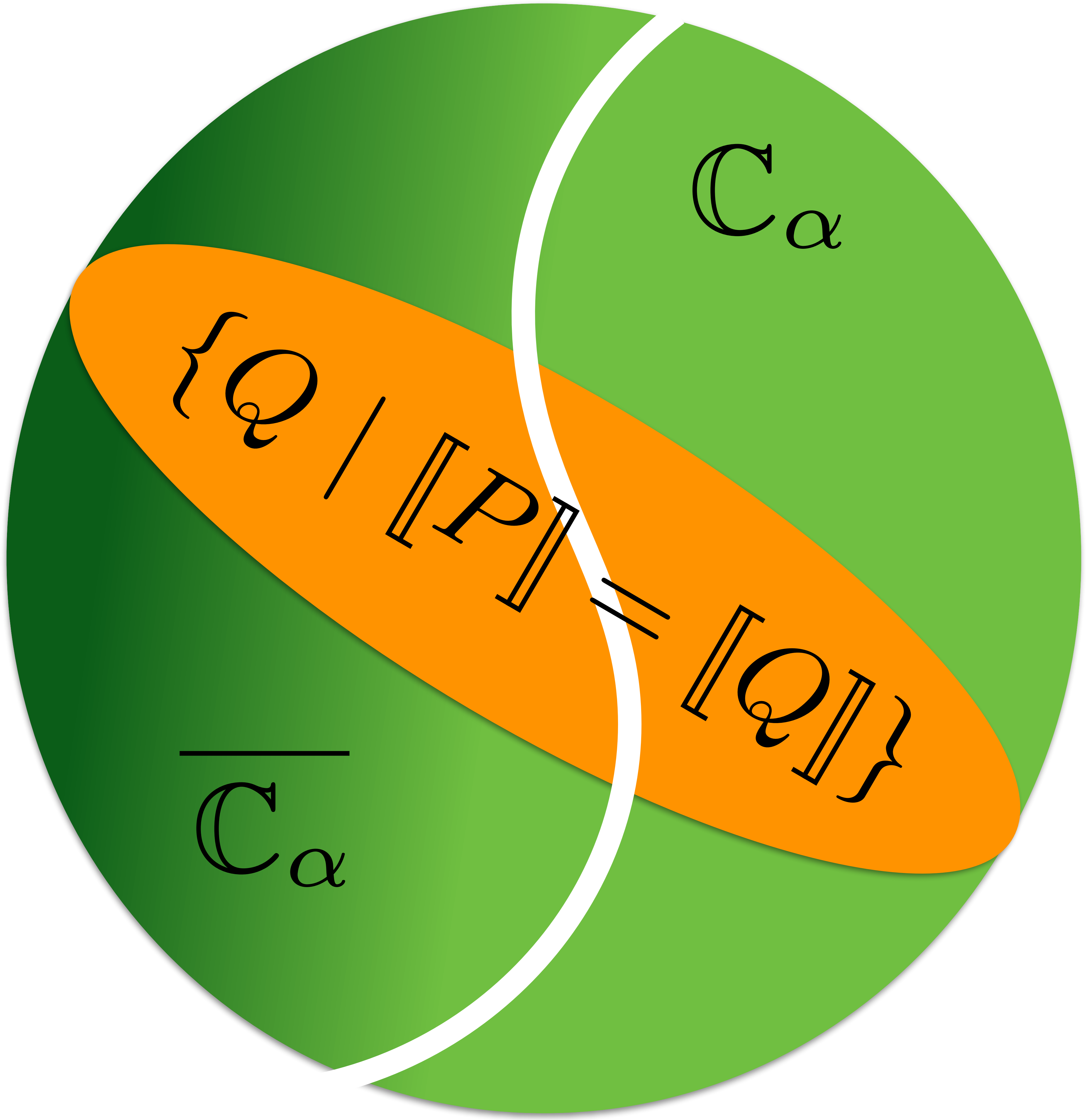
POPL2020



# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020

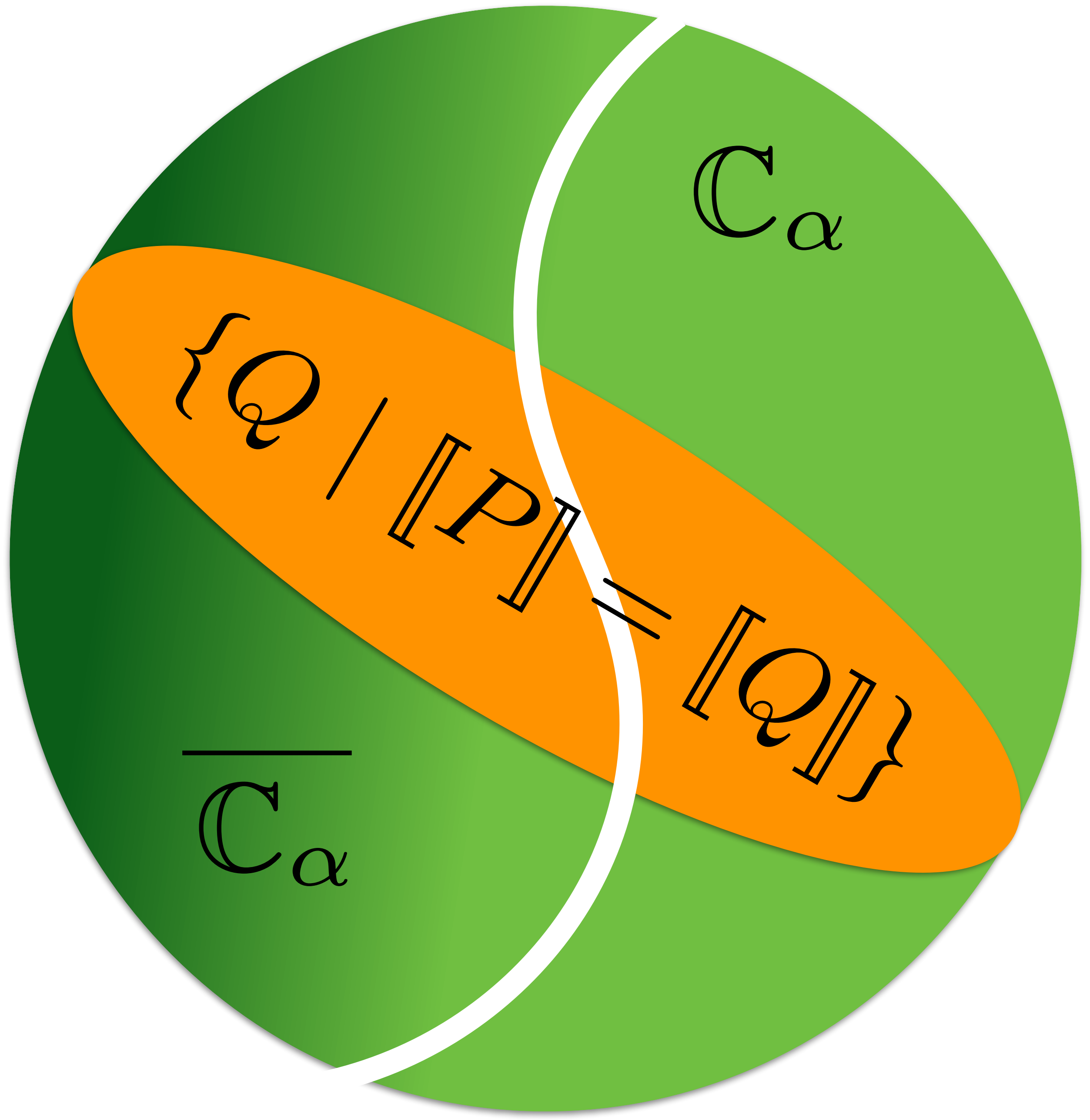



$$Q \in \overline{\mathbb{C}_\alpha(P)} \overset{?}{\Rightarrow} f(Q) \in \mathbb{C}_\alpha(P)$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020

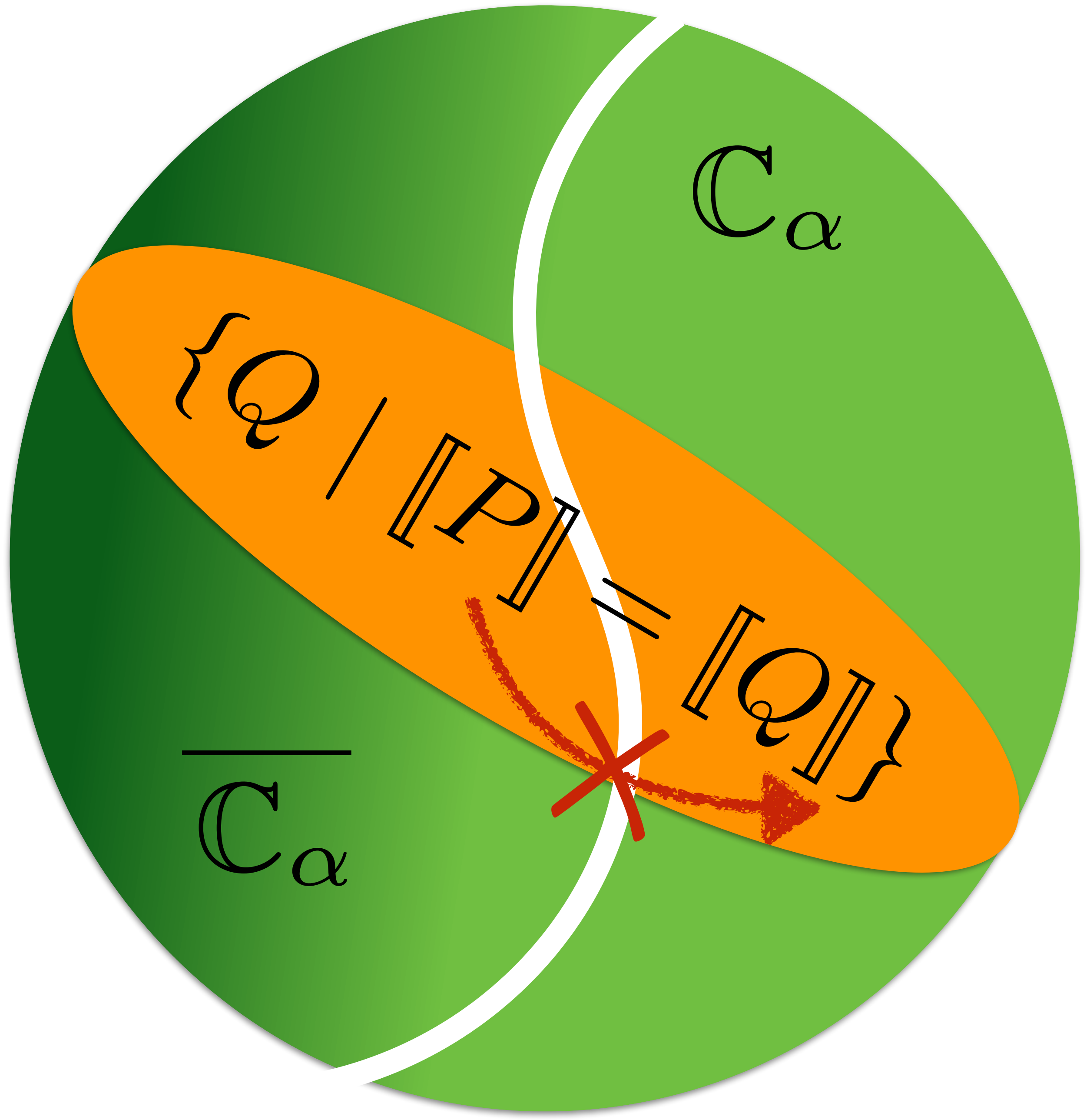



$$Q \in \overline{\mathbb{C}_\alpha(P)} \overset{?}{\Rightarrow} f(Q) \in \mathbb{C}_\alpha(P)$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020




non-termination

$$\alpha = \{\top, \bot\}$$

$$f(Q)$$

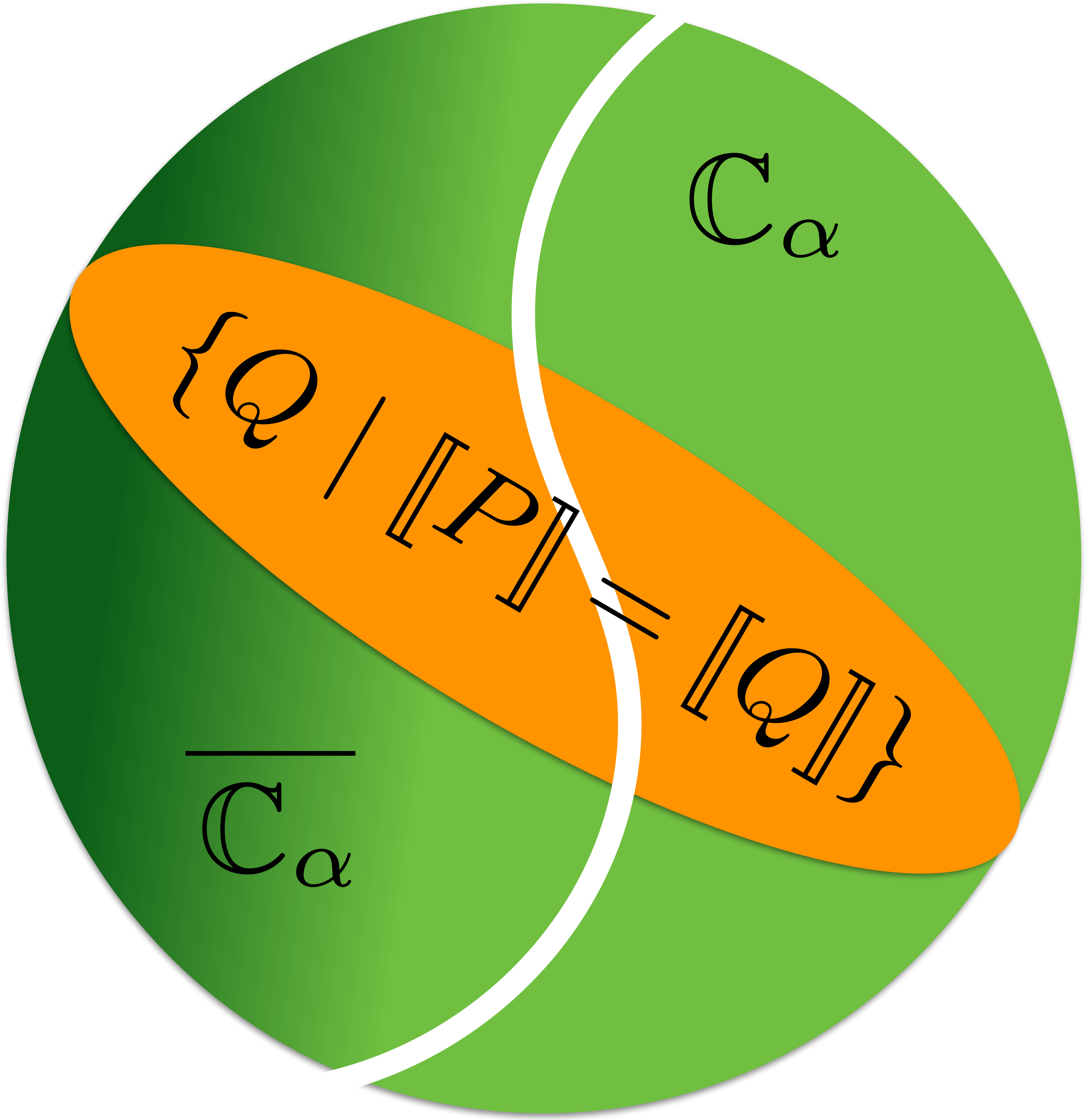
Would decide termination!

$$Q \in \overline{\mathbb{C}_\alpha(P)} \overset{?}{\Rightarrow} f(Q) \in \mathbb{C}_\alpha(P)$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \mathrm{Programs}$

POPL2020

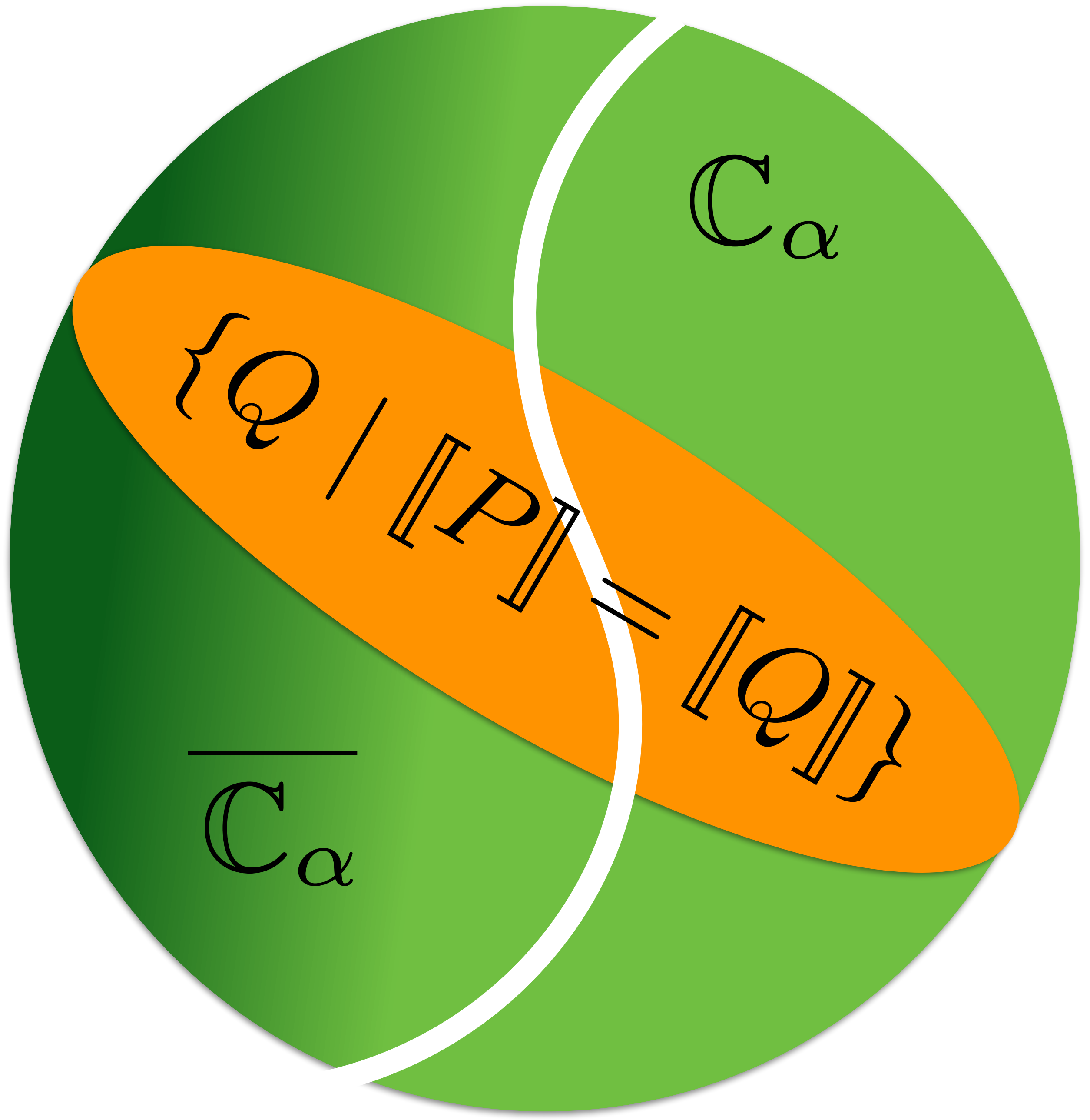



$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020

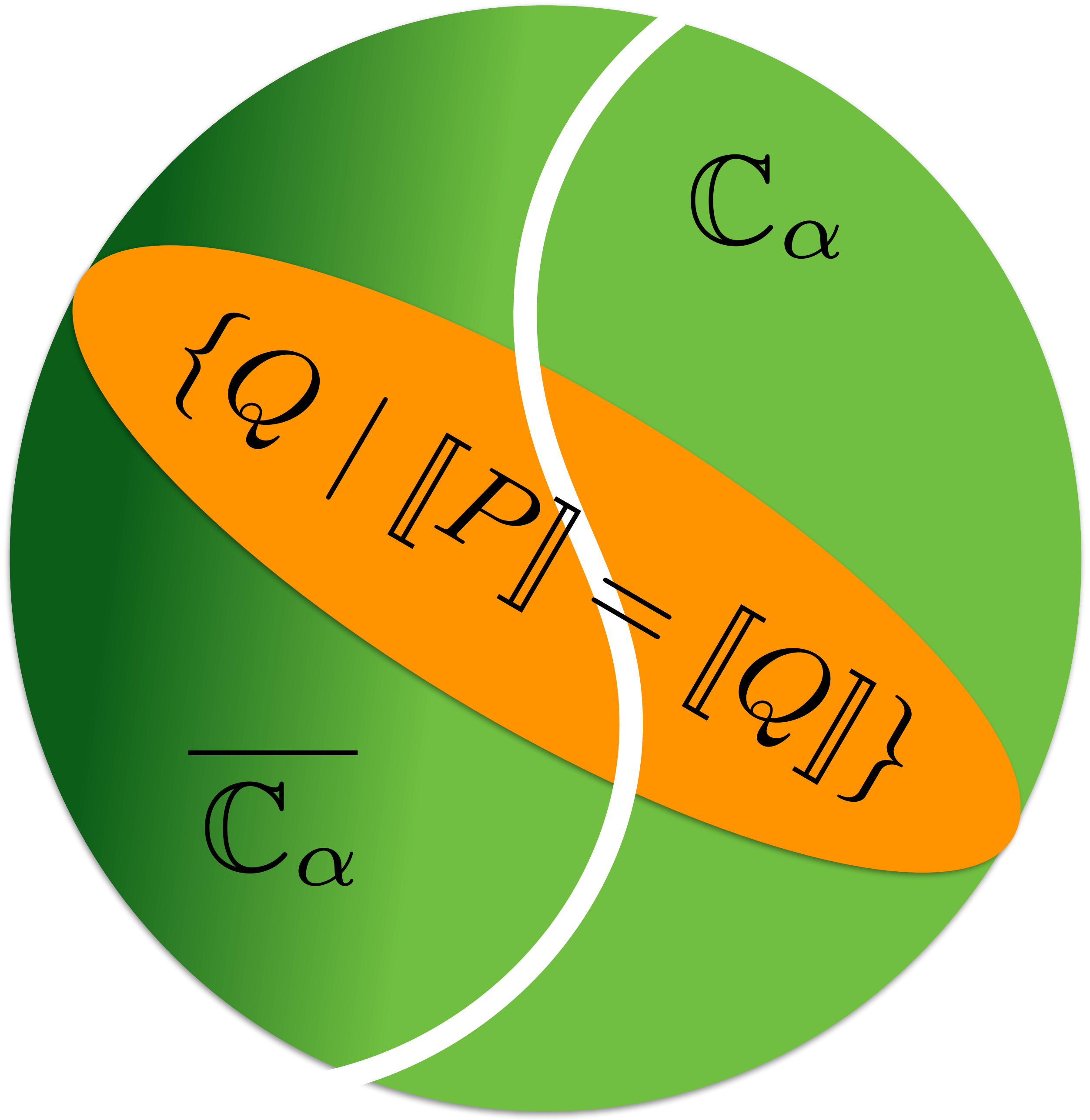



$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020



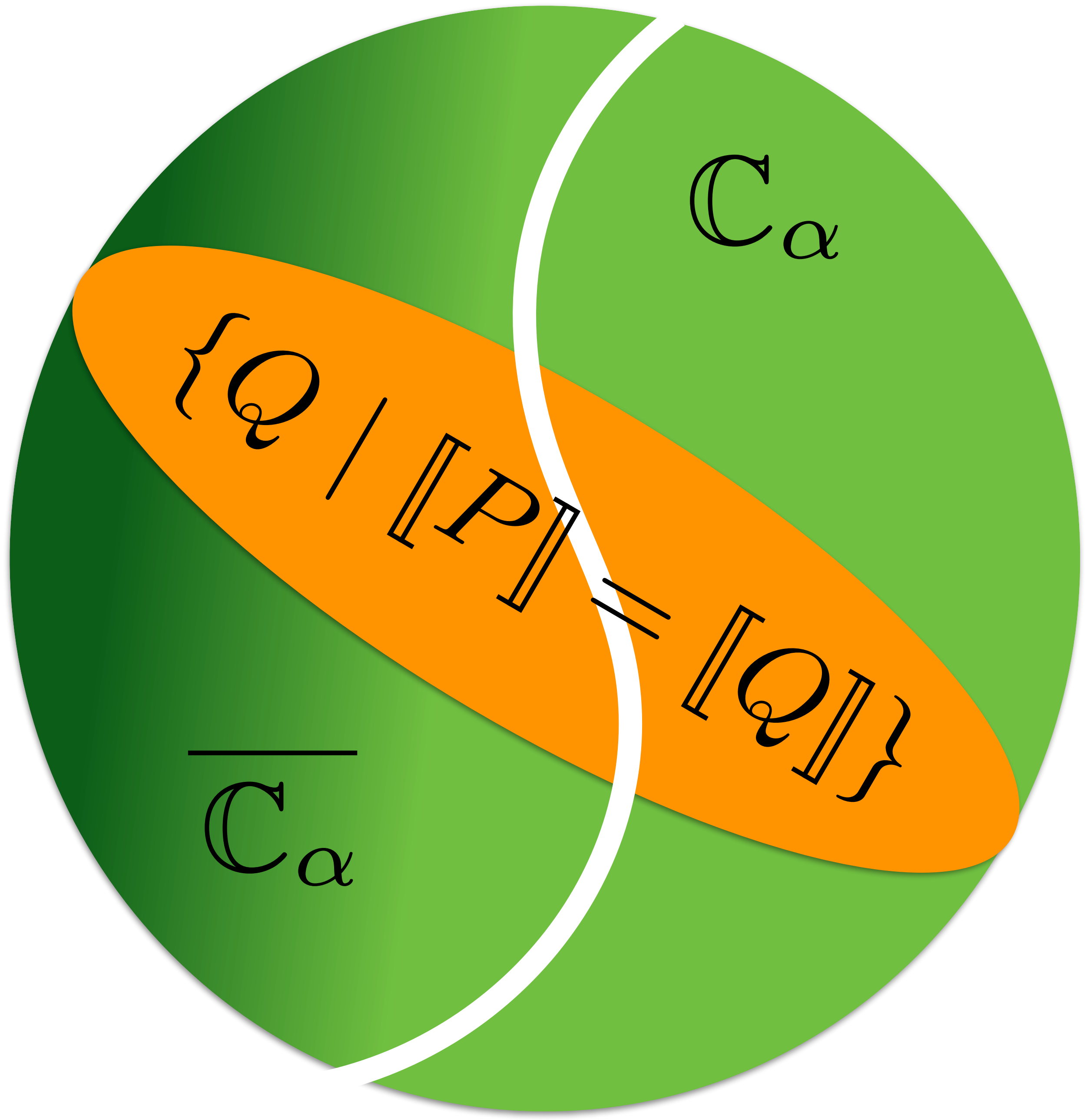

S

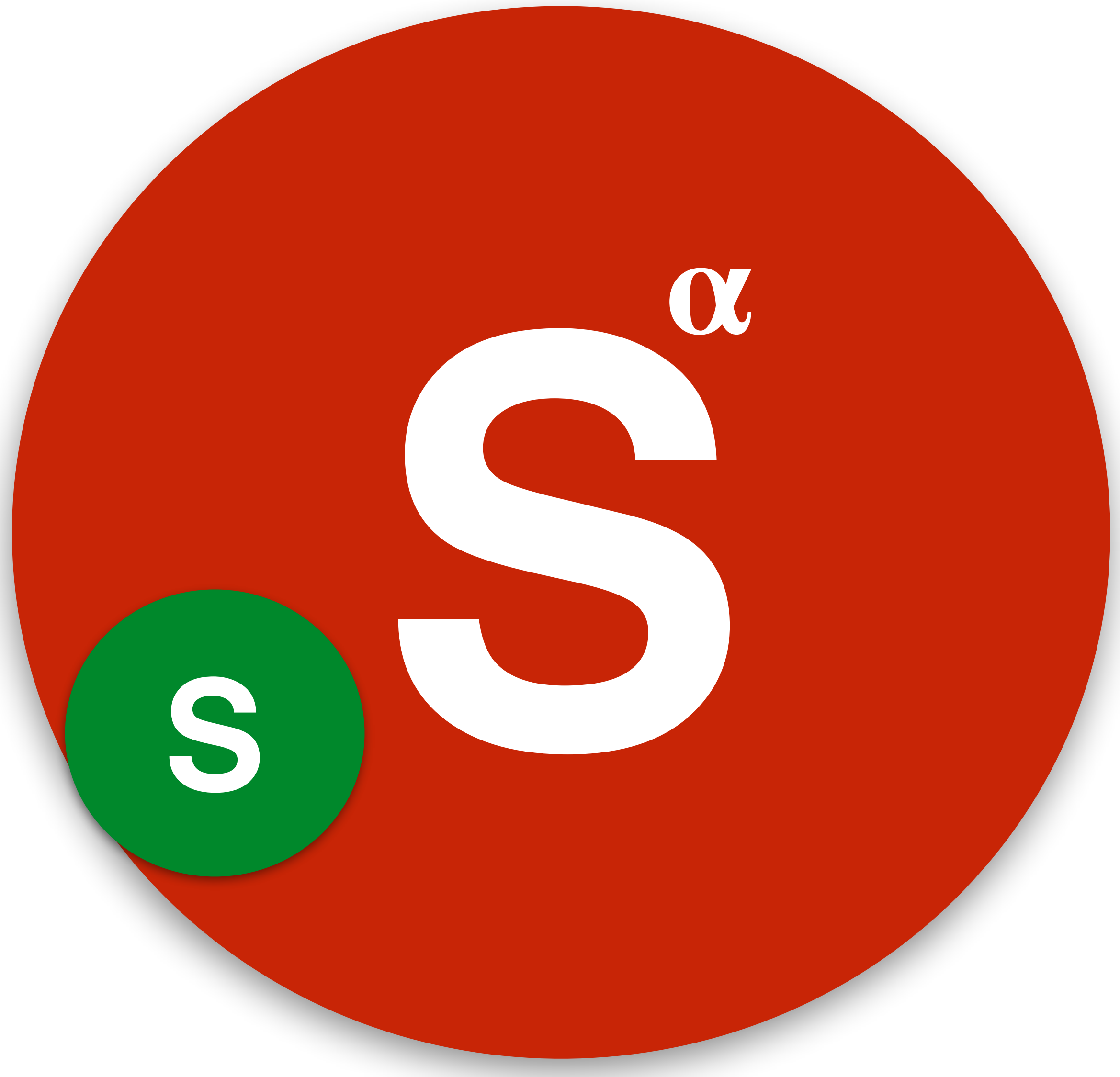
$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020



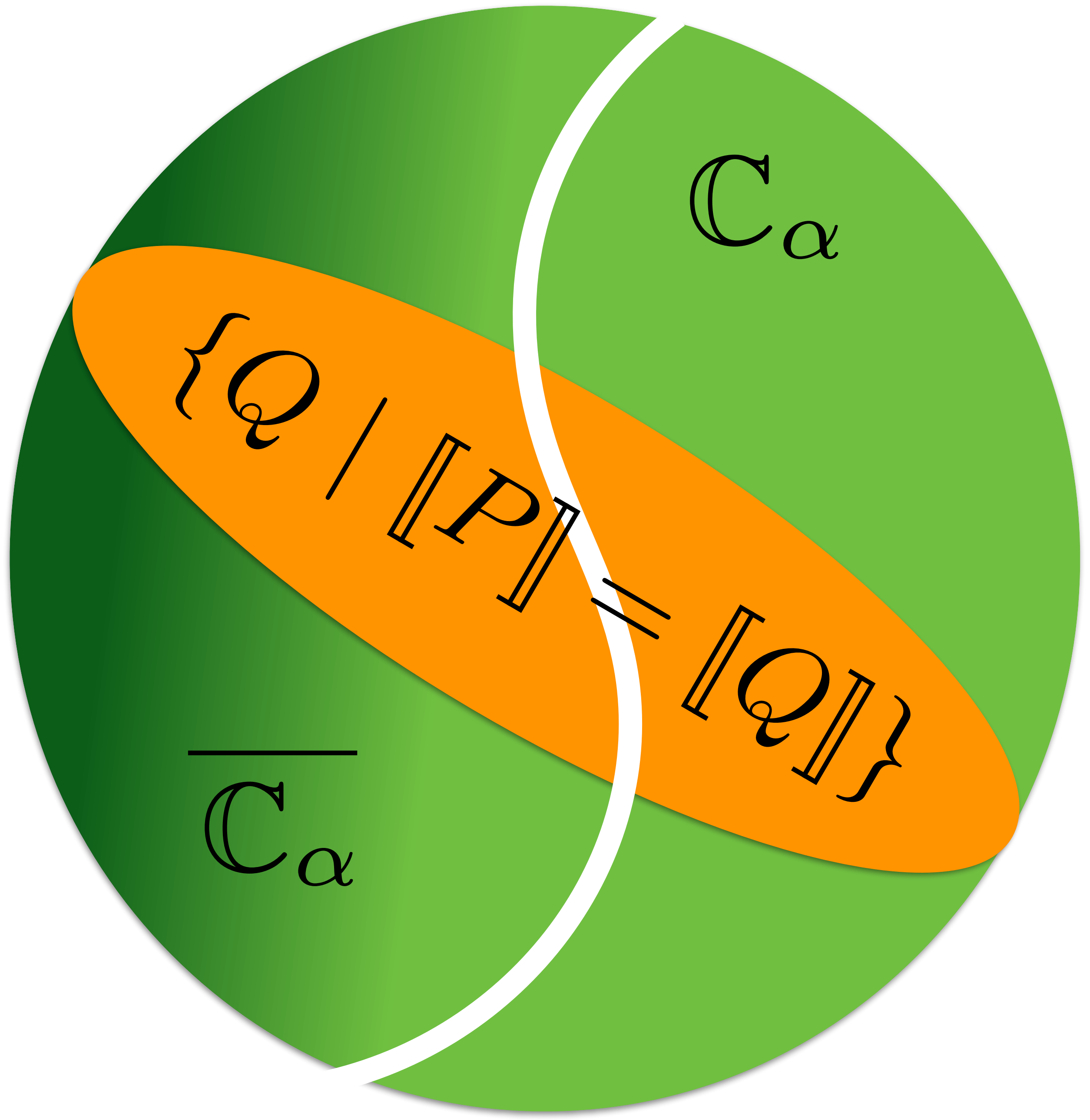



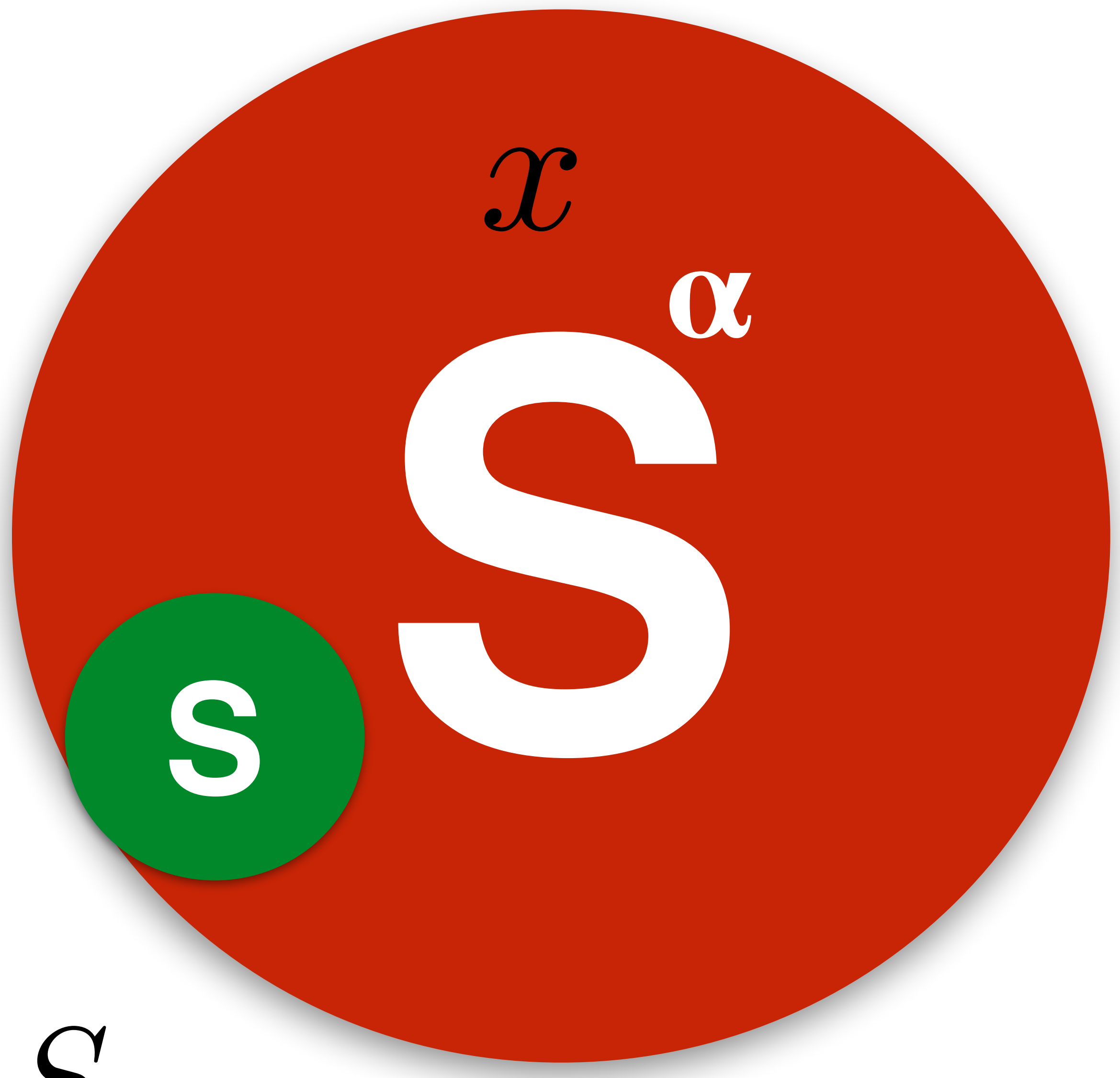

$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

Given $P \in \text{Programs}$

POPL2020



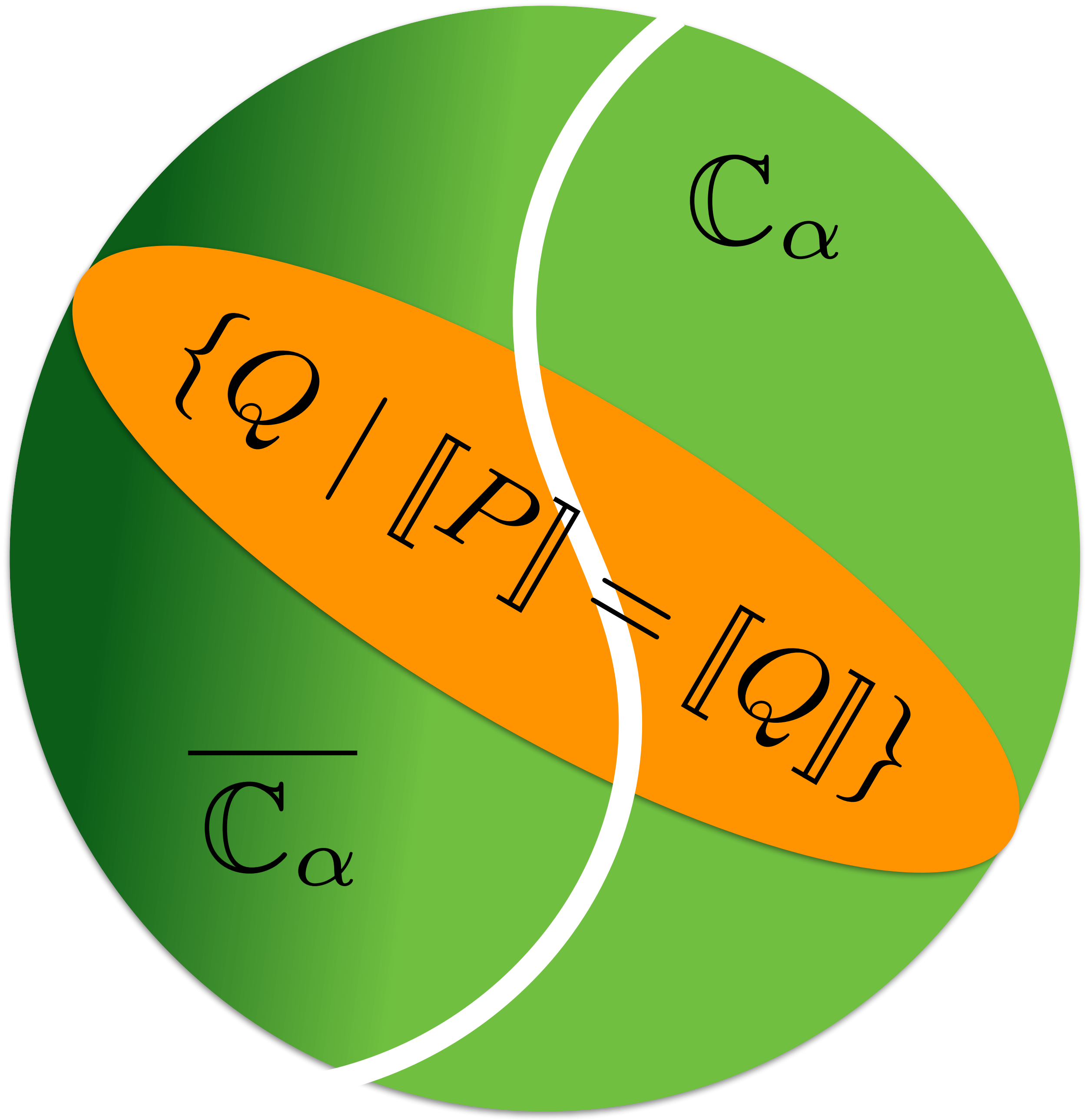

$$\mathtt{In}(S) \equiv x \in^{?} S$$

$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

# On Completeness Cliques $\mathbb{C}_\alpha(P)$

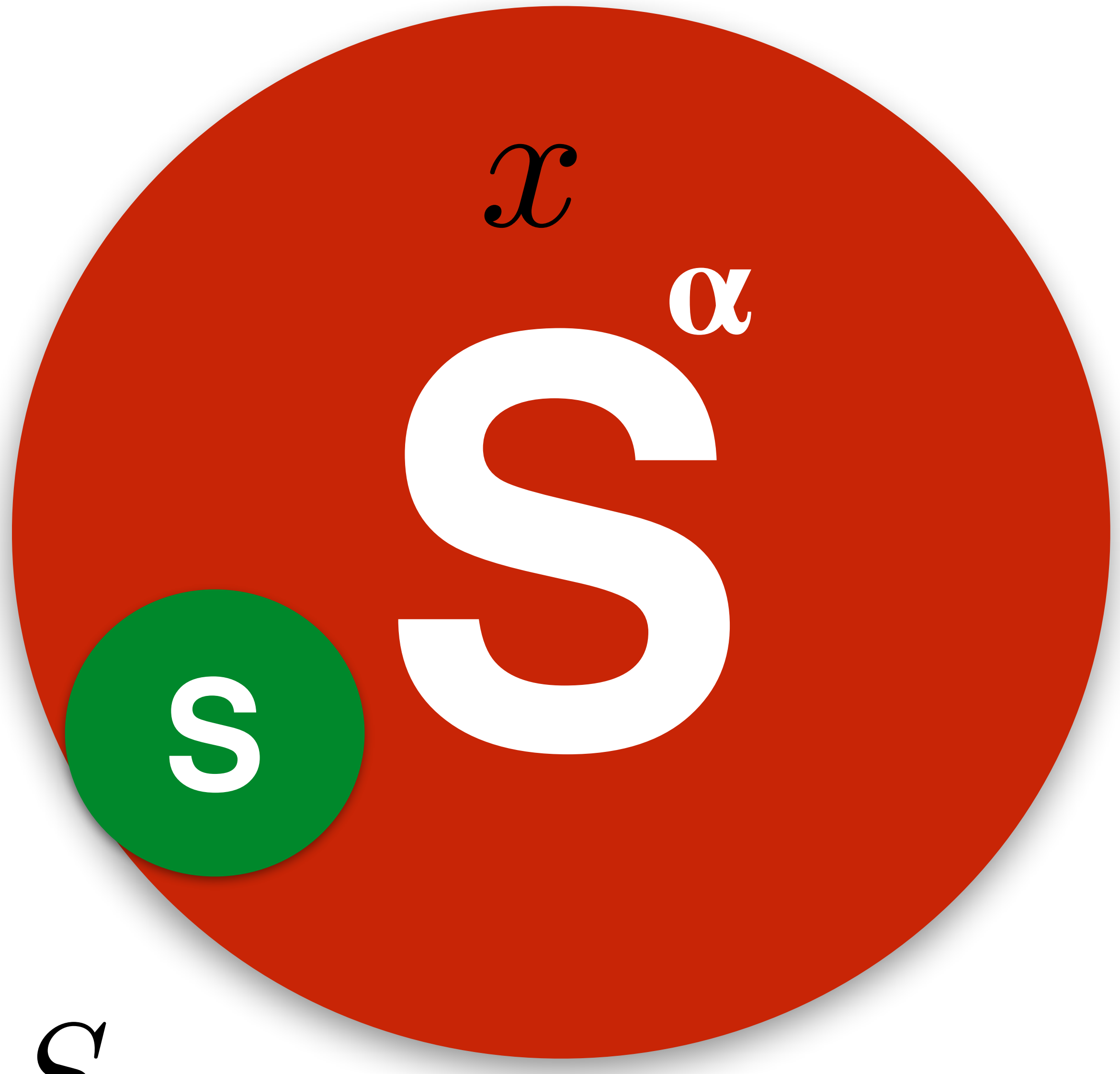
Given $P \in$ Programs

POPL2020




**if** $\texttt{In}(S)$ **then**
  **if** $\neg\texttt{In}(S)$ **then** $\texttt{Set}(\texttt{m}', S)$ $\approx$ $P$ ;
  **else** $P$
**else** $P$

dead code




$$\texttt{In}(S) \equiv x \in^? S$$

$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

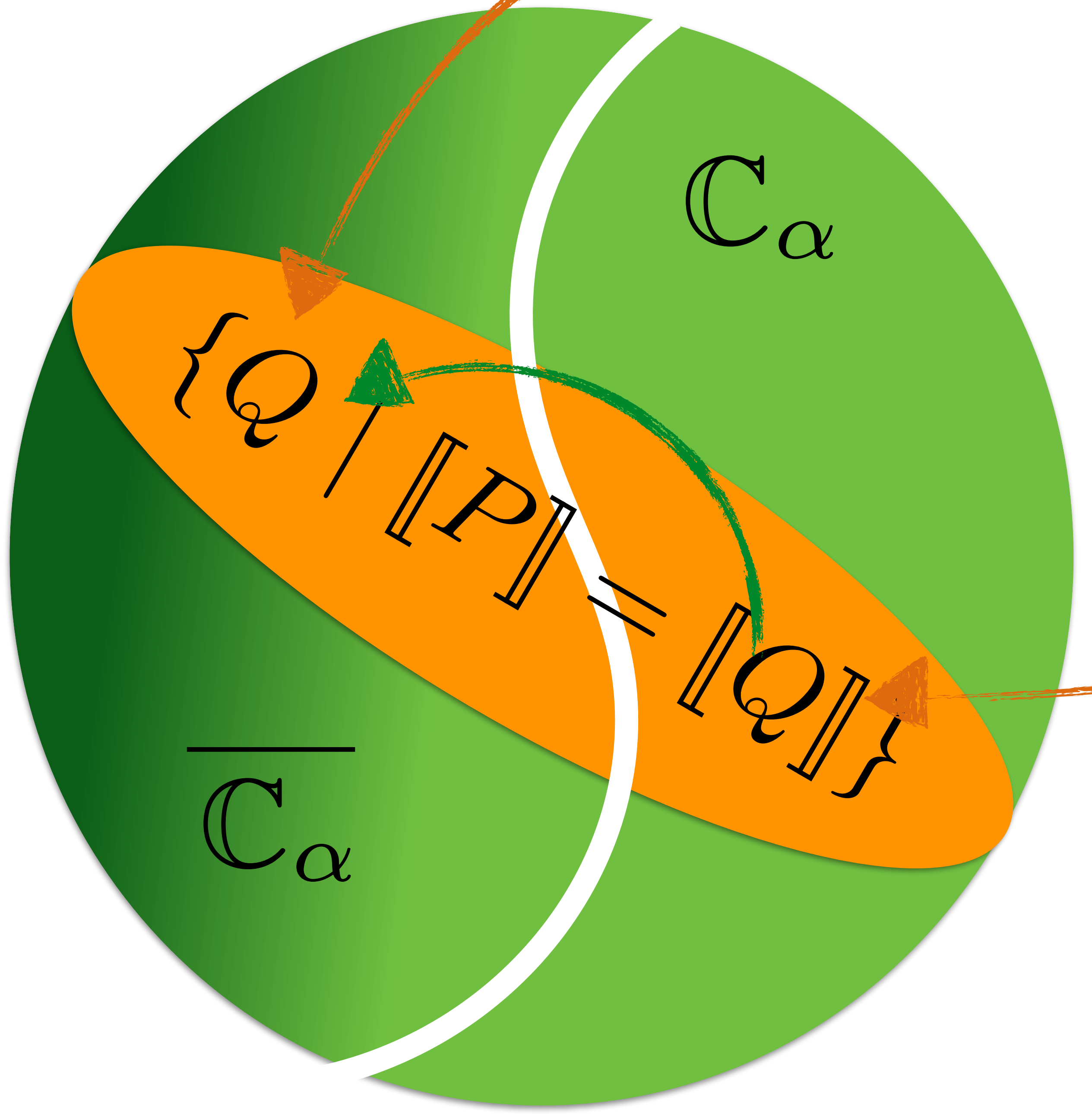
# On Completeness Cliques $\mathbb{C}_\alpha(P)$

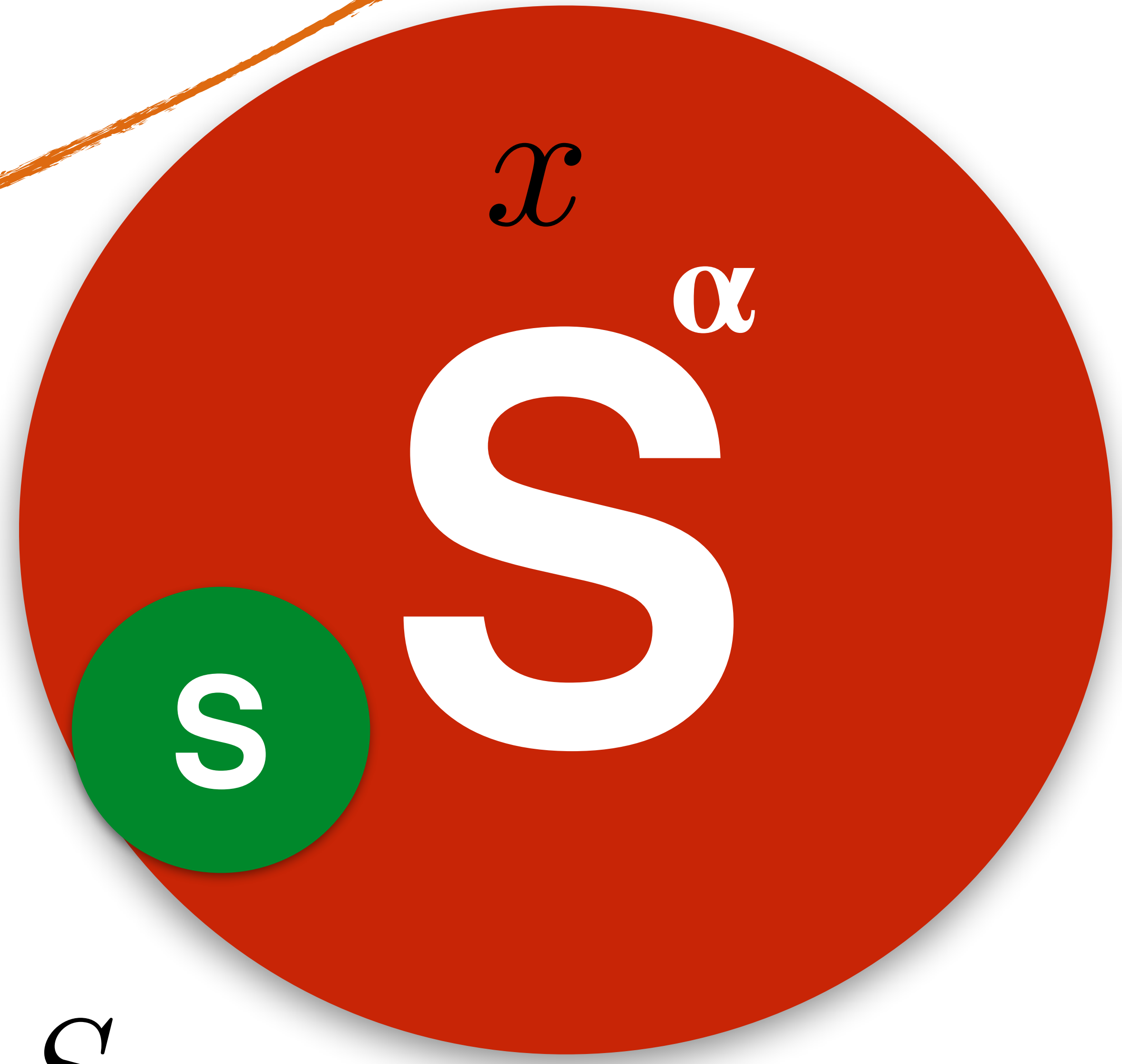
Given $P \in$ Programs

POPL2020

**if** $\texttt{In}(S)$ **then**
  **if** $\neg\texttt{In}(S)$ **then** $\texttt{Set}(\texttt{m}', S)$ $\approx$ $P$ ;
  **else** $P$
**else** $P$

dead code




$$\texttt{In}(S) \equiv x \in^{?} S$$

$$Q \in \mathbb{C}_\alpha(P) \overset{?}{\Rightarrow} f(Q) \in \overline{\mathbb{C}_\alpha(P)}$$

POPL2020

An Abstract property of programs is (Rice)-extensional

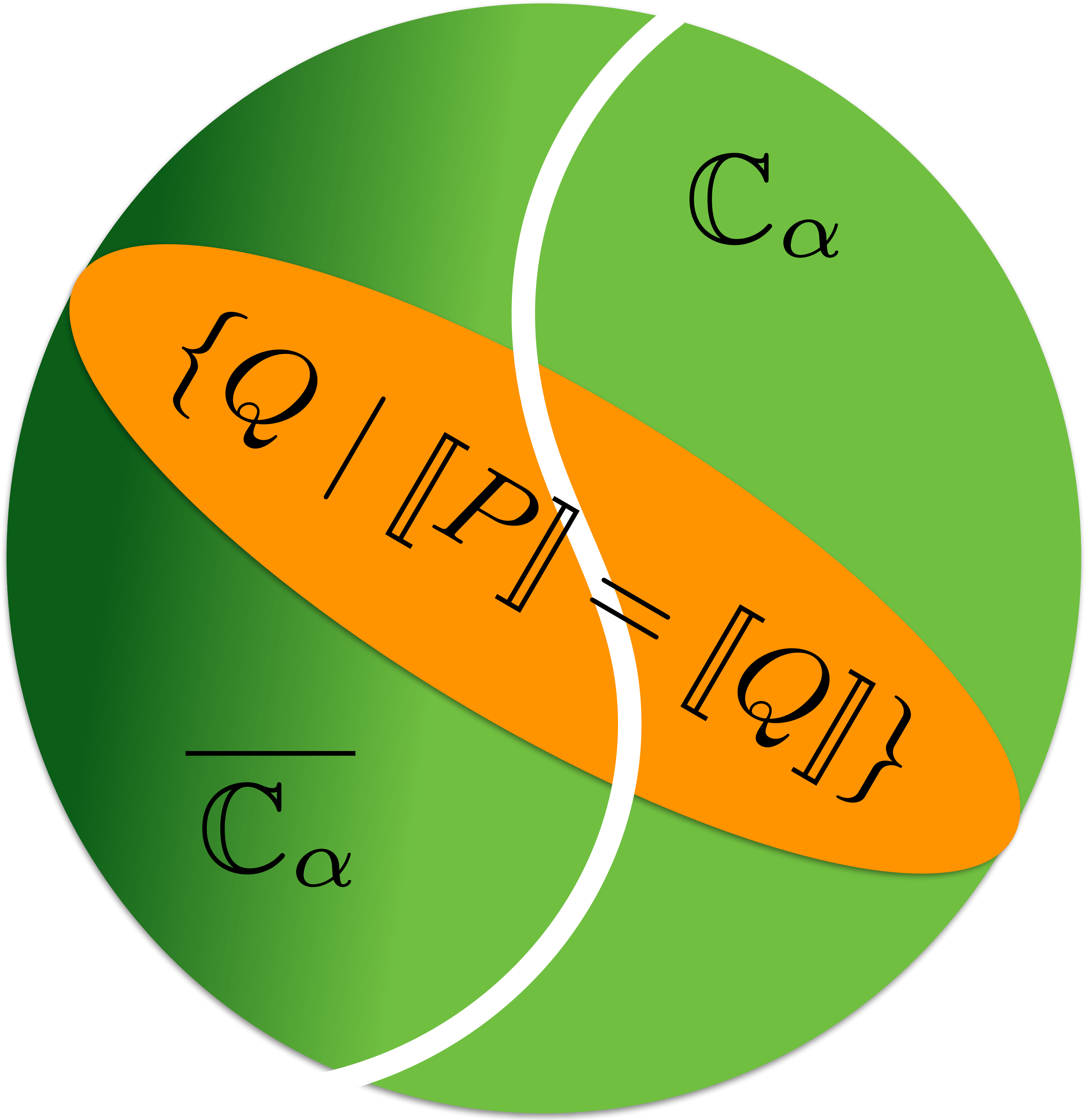
iff

$\alpha$ is trivial

# On Completeness Classes again

Given $P \in \text{Programs}$

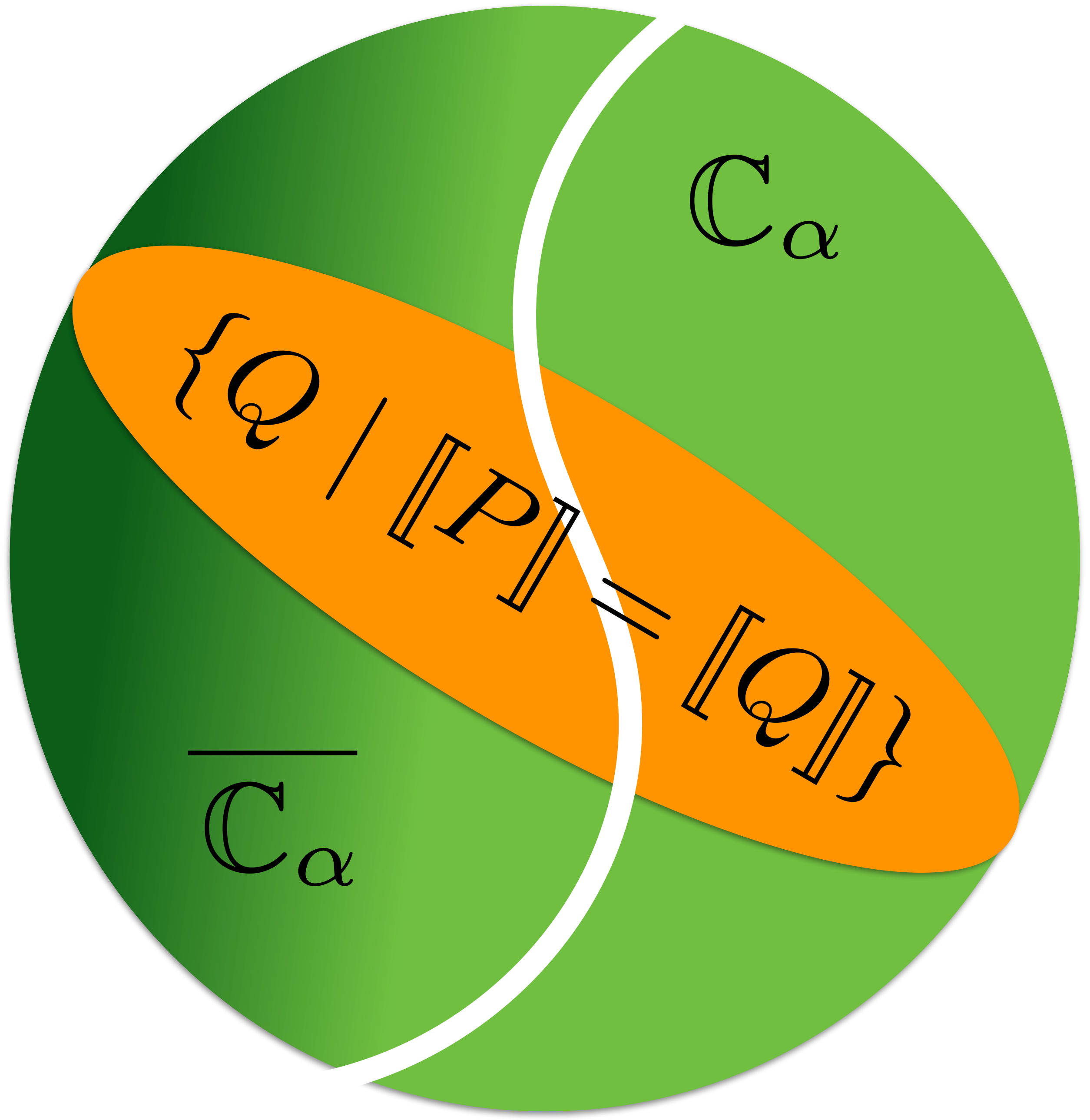
POPL2020



# On Completeness Classes again

POPL2020

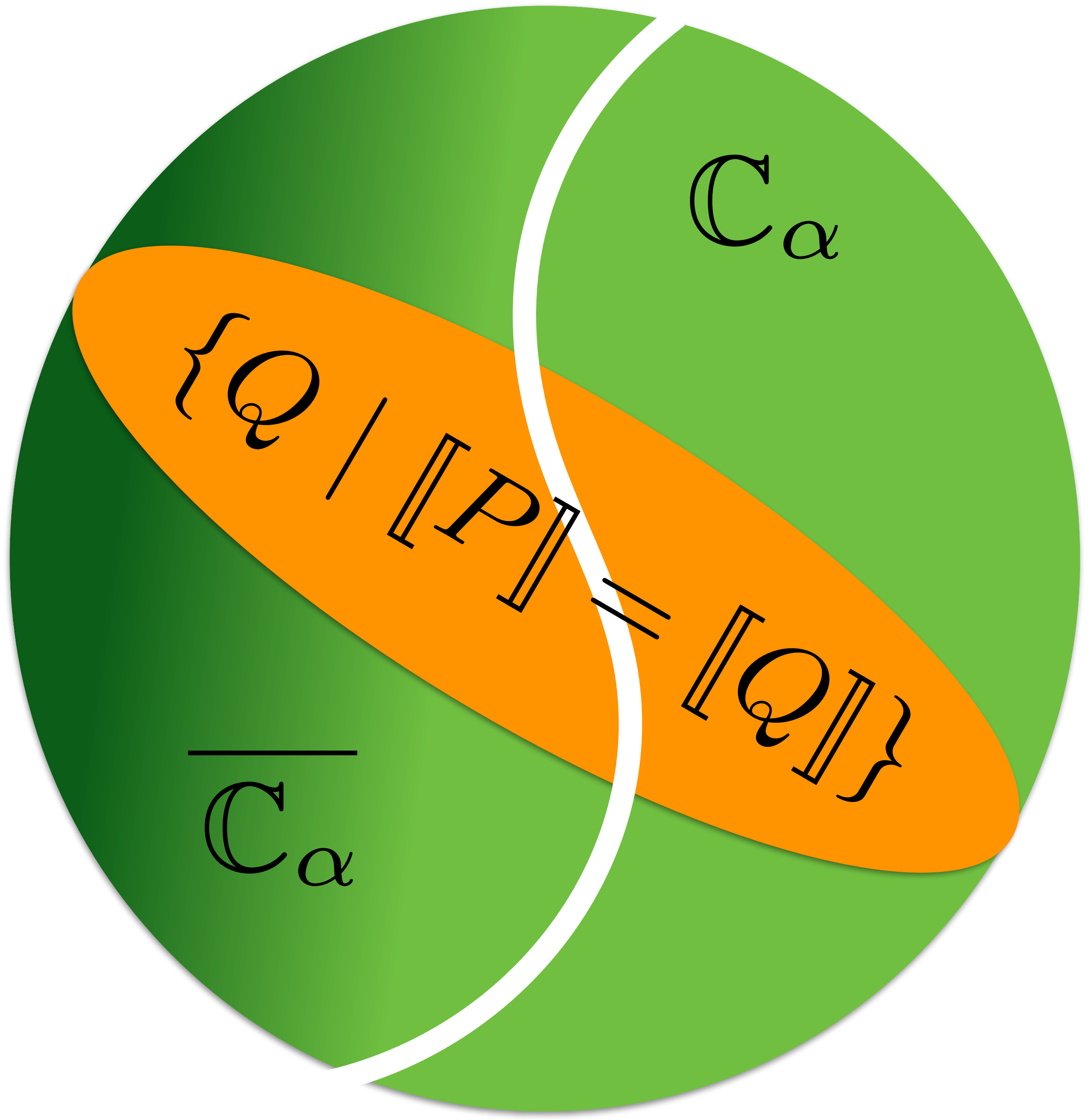
Given $P \in \text{Programs}$



# On Completeness Classes again

Given $P \in \text{Programs}$

POPL2020

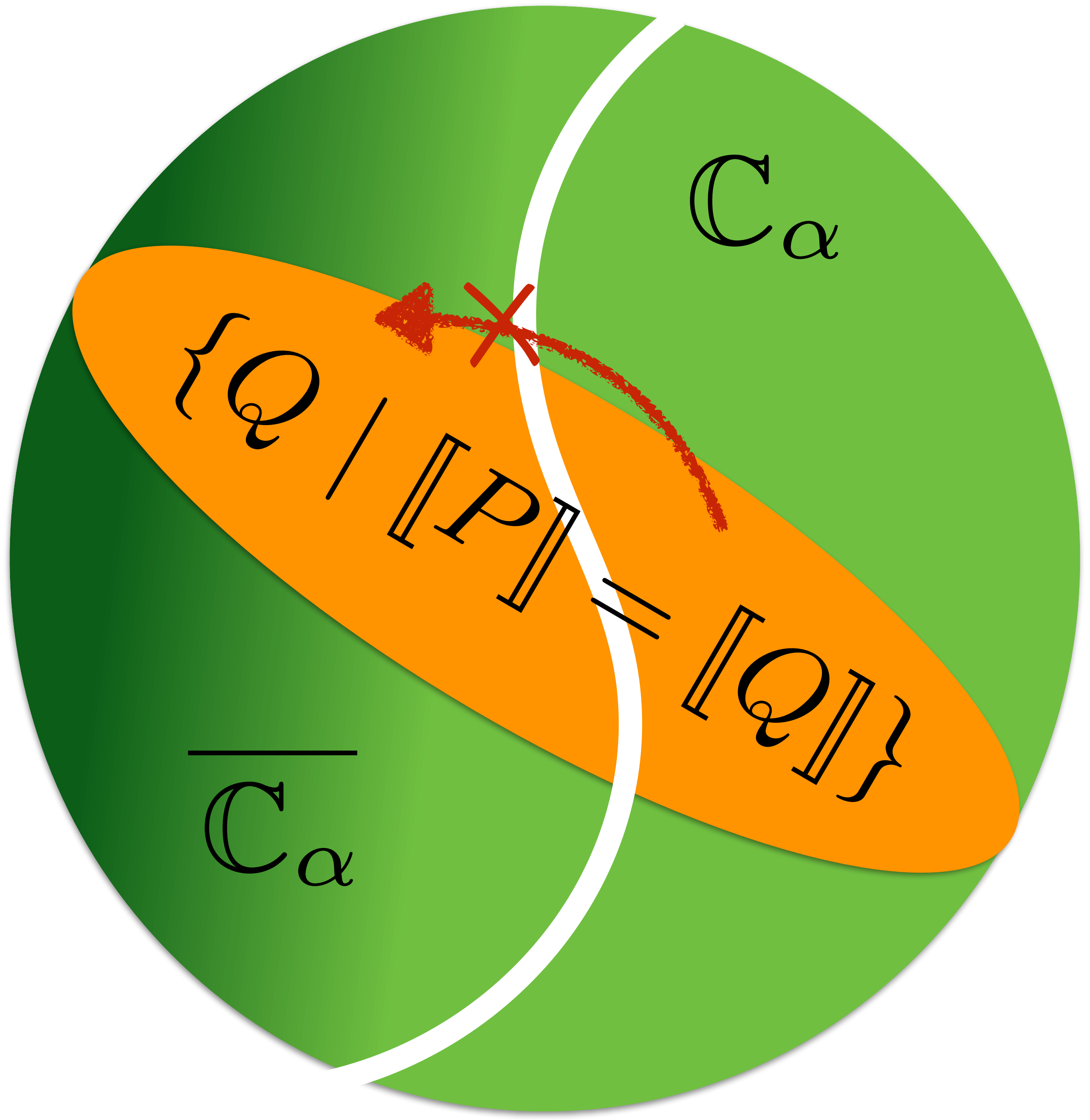



Can $\mathbb{C}_\alpha$
be Turing complete?

# On Completeness Classes again

Given $P \in \text{Programs}$

POPL2020




Can $\mathbb{C}_\alpha$
be Turing complete?

NO

$$\texttt{spec}, \texttt{int} \in \mathbb{C}_\alpha$$

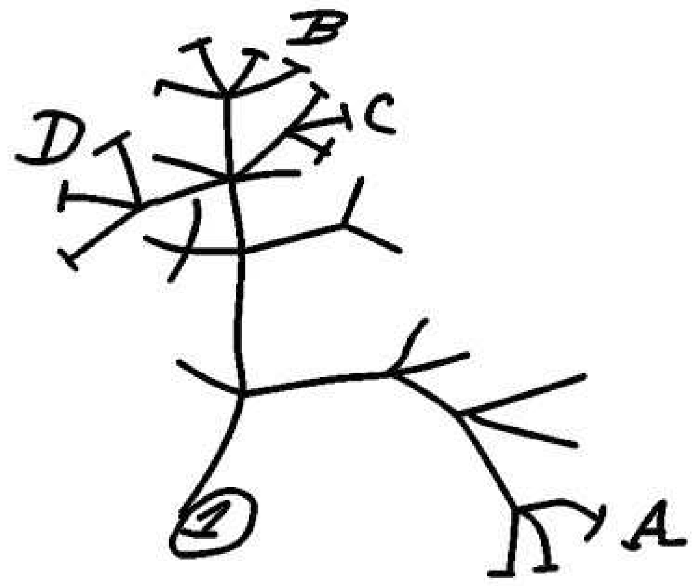
$$\Downarrow$$

$$\forall P : \llbracket \texttt{spec} \rrbracket(\texttt{int}, P) \in \mathbb{C}_\alpha$$

POPL2020

Any non-trivial abstract property of programs is intensional!

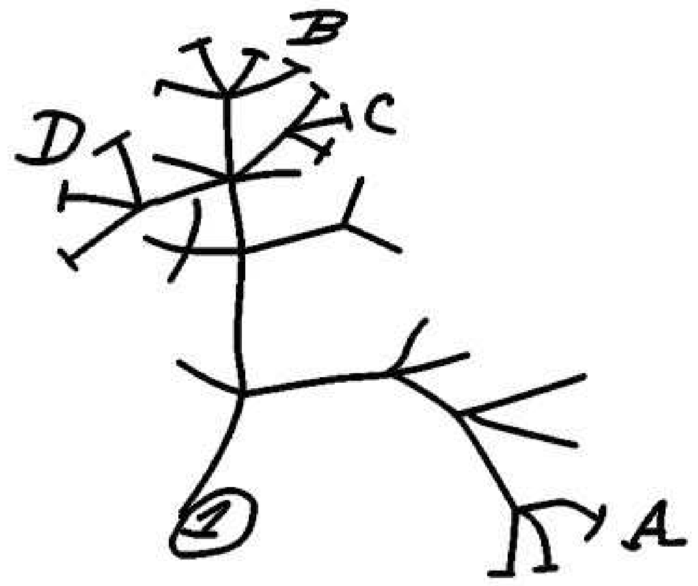
Non-trivial abstract interpretations always reveal properties about the way the code is written!



POPL2020

Program Analysis is like Computational Complexity

Can we build an implicit program analysis theory?

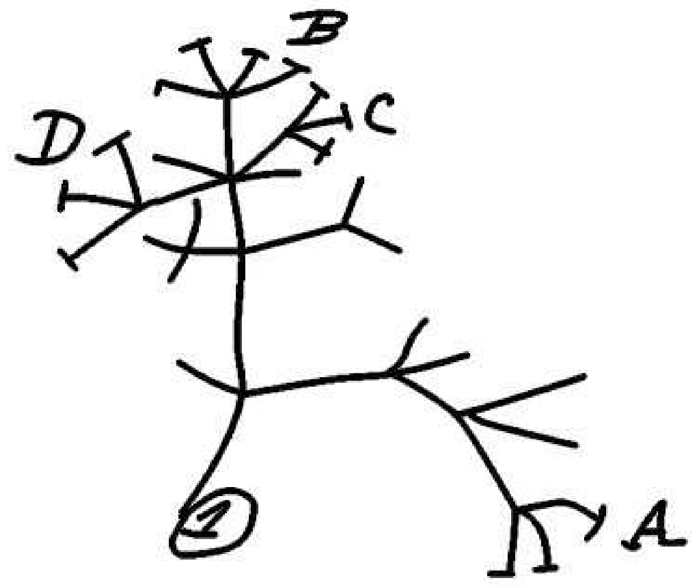


# In the Future?

## PRIN — Analysis of Program Analyses (ASPRA)

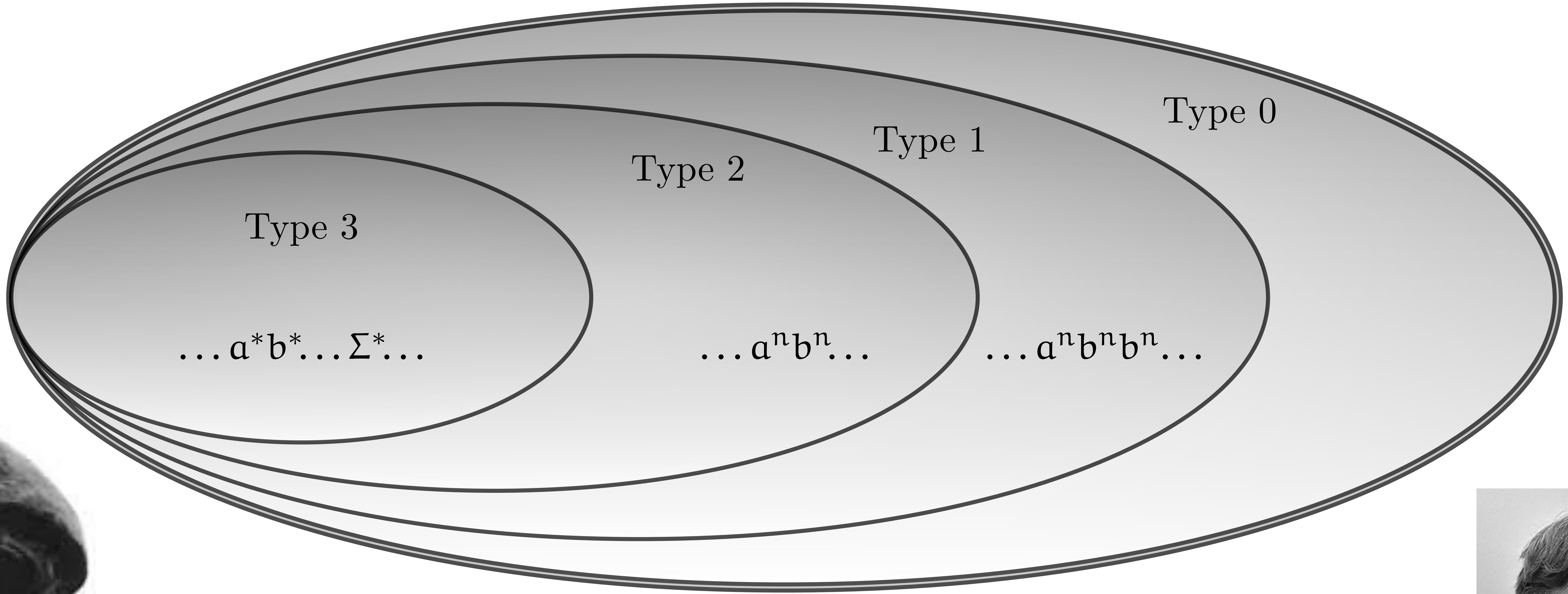
If $\alpha$ is non-trivial which programs can I build in $\mathbb{C}_\alpha$?

Hack the code to increase precision!

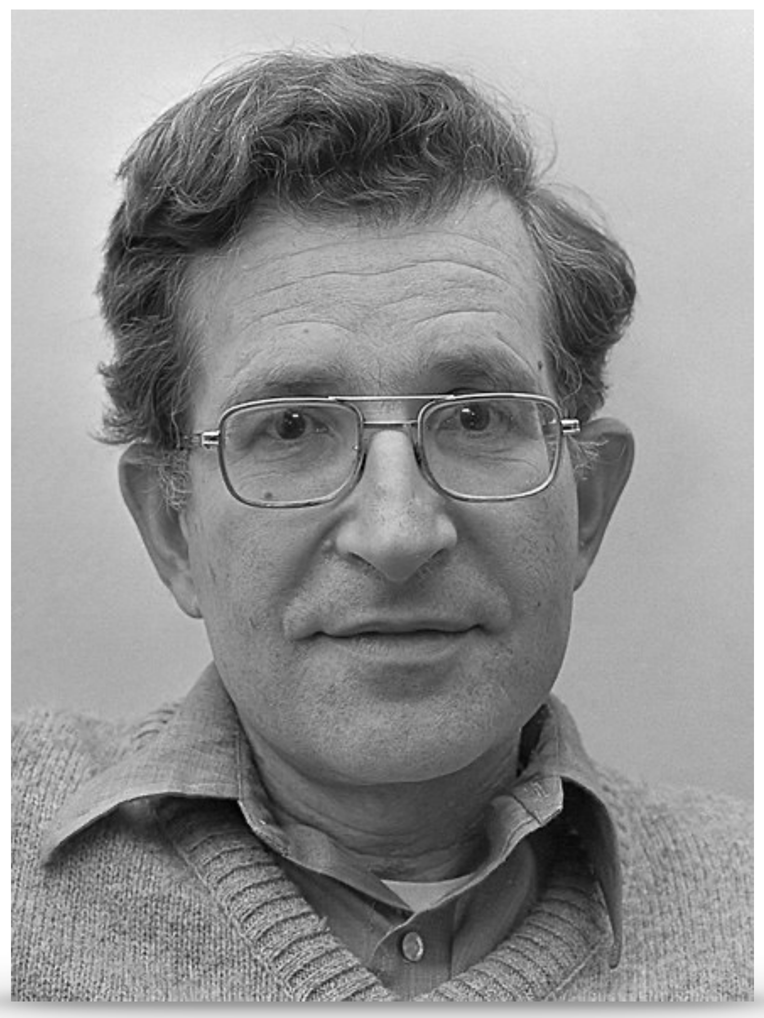


# In the Future?

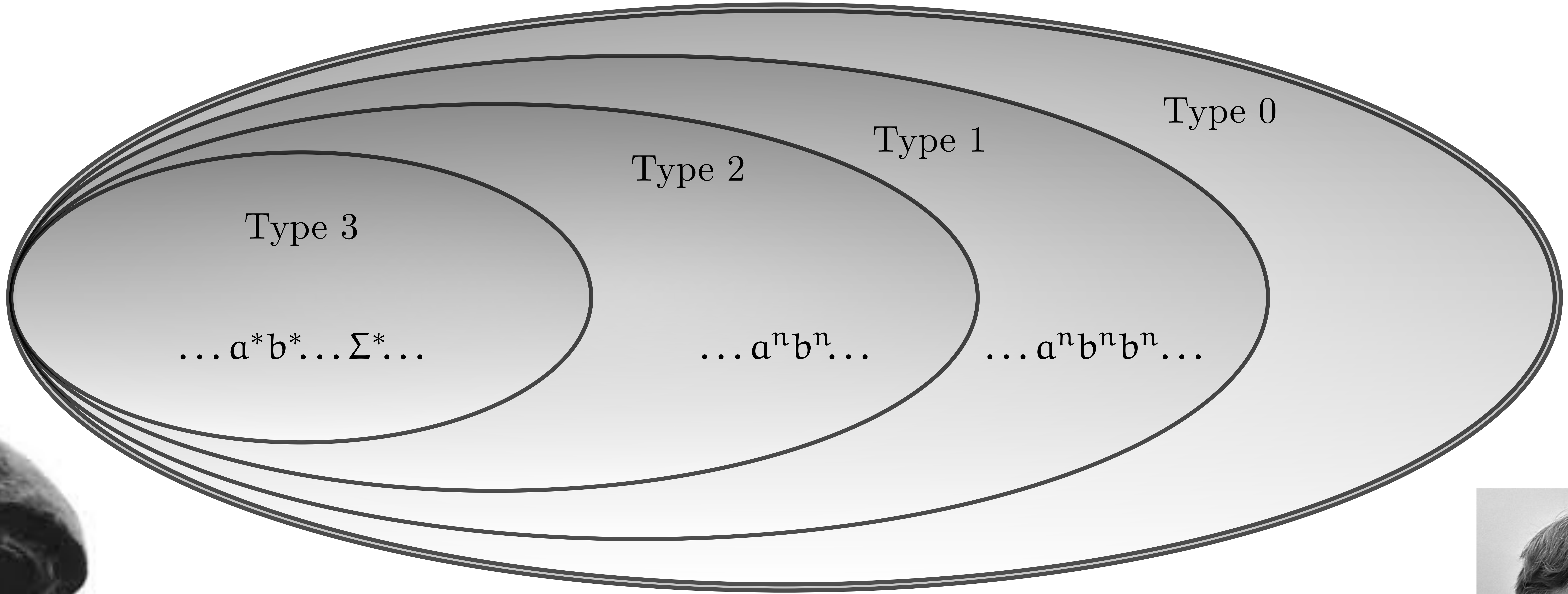
## PRIN – Analysis of Program Analyses (ASPRA)



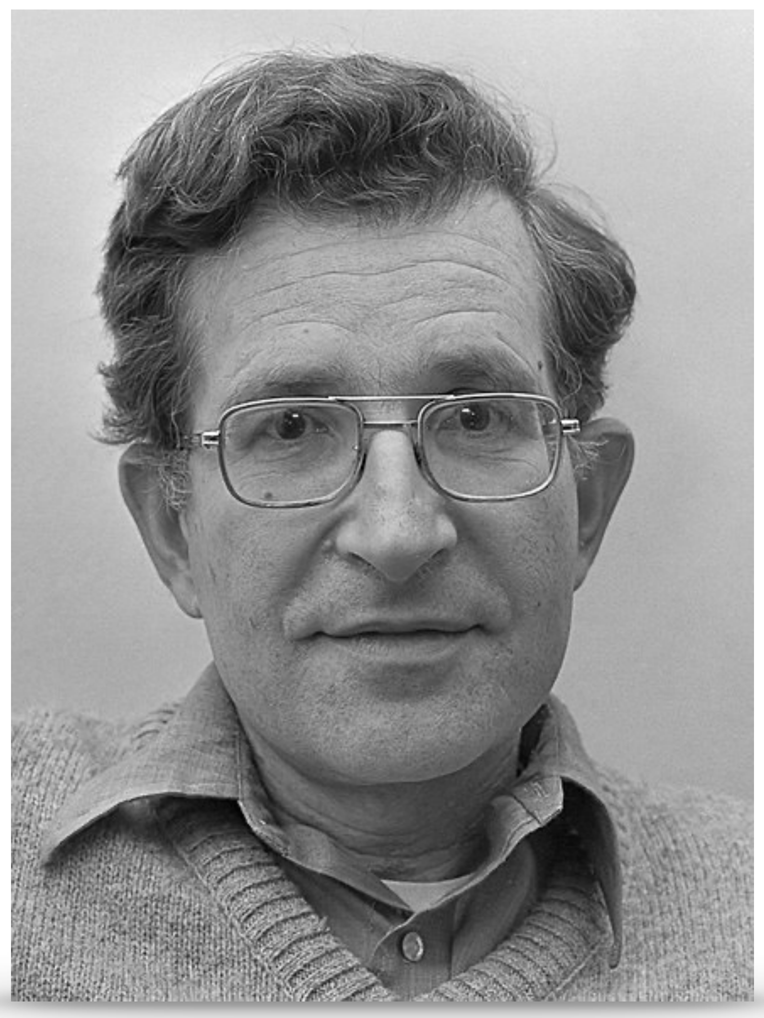

$\mathbb{C}_\alpha$ ?



# In the Future?
# PRIN – Analysis of Program Analyses (ASPRA)




$$\mathbb{C}_\alpha\,?$$



Thanks Simone