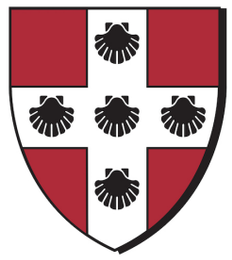
Modalities, Cohesion, and Information Flow

Alex Kavvos

Department of Mathematics and Computer Science, Wesleyan University



MIT (Applied) Categories Seminar, 30 Nov 2018

arXiv:1809.07897

Language-based Information Flow Control

- General idea:
  - **types** should include annotations concerning the **classification/sensitivity** of data
  - programs should type-check only if there is no **unsafe information flow** (e.g. from TOP SECRET to UNCLASSIFIED)
- Roots of idea go back to 1970s models of information flow.

  Leonard J. LaPadula and D. Elliott Bell. 1996. Secure Computer Systems: Mathematical Foundations. Journal of Computer Security 4, 2-3 (1996), 239–263. [first circulated in 1973]

  Dorothy E Denning. 1976. A lattice model of secure information flow. Commun. ACM 19, 5 (1976), 236–243.

- Great prominence in the late 1990s, also very popular today.
  See CSF conference: [Rajani and Garg 2018] won the distinguished paper award.

Language-based Information Flow Control

The Needham-Schroeder-Lowe protocol for authentication, given public keys:

1. $A \rightarrow B$: Alice, $E_{k_B}(\text{Alice} \,\|\, nonce_A)$
2. $B \rightarrow A$: $E_{k_A}(nonce_A \,\|\, \text{Bob} \,\|\, nonce_B)$
3. $A \rightarrow B$: $E_{k_B}(nonce_B \,\|\, k_s)$
4. $A \rightarrow B$: $E_{k_s}(m_1 \,\|\, \text{Alice} \,\|\, counter)$
5. etc.

Some code implementing Alice:

Alice's private key **Bob's public key**

```
proc Alice(pv : Key, pb : Key, Bob-I: ?[Msg], Bob-O : ![Msg]) {
  val nonce-a = generateNonce()
  val pk = generateSymmetricKey()
  Bob-O ! (Alice, encrypt(pb, (Alice, nonce-a)))
  val (nonce-a-b, name, nonce-b) = decrypt(pv, Bob-I ? ())
  if (nonce-a == nonce-a-b && name == Bob)) {
    Bob-O ! encrypt(pb, (nonce-b, pv))   <-- Message #3
    Alice-session(pk, Bob-I, Bob-O)
  }
}
```

Language-based Information Flow Control

The Needham-Schroeder-Lowe protocol for authentication, given public keys:

1. $A \rightarrow B$: Alice, $E_{k_B}(\text{Alice} \,\|\, nonce_A)$
2. $B \rightarrow A$: $E_{k_A}(nonce_A \,\|\, \text{Bob} \,\|\, nonce_B)$
3. $A \rightarrow B$: $E_{k_B}(nonce_B \,\|\, k_s)$
4. $A \rightarrow B$: $E_{k_s}(m_1 \,\|\, \text{Alice} \,\|\, counter)$
5. etc.

Some code implementing Alice:

Alice's private key **Bob's public key**

```
proc Alice(pv : Key, pb : Key, Bob-I: ?[Msg], Bob-O : ![Msg]) {
  val nonce-a = generateNonce()
  val pk = generateSymmetricKey()
  Bob-O ! (Alice, encrypt(pb, (Alice, nonce-a)))
  val (nonce-a-b, name, nonce-b) = decrypt(pv, Bob-I ? ())
  if (nonce-a == nonce-a-b && name == Bob)) {
    Bob-O ! encrypt(pb, (nonce-b, pv)) <-- Message #3
    Alice-session(pk, Bob-I, Bob-O)
  }
}
```

OH NO

Language-based Information Flow Control

The Needham-Schroeder-Lowe protocol for authentication, given public keys:

1. $A \rightarrow B$: Alice, $E_{k_B}(\text{Alice} \,\|\, nonce_A)$
2. $B \rightarrow A$: $E_{k_A}(nonce_A \,\|\, \text{Bob} \,\|\, nonce_B)$
3. $A \rightarrow B$: $E_{k_B}(nonce_B \,\|\, k_s)$
4. $A \rightarrow B$: $E_{k_s}(m_1 \,\|\, \text{Alice} \,\|\, counter)$
5. etc.

Some code implementing Alice:

Alice's private key **Bob's public key**

```
proc Alice(pv : ♦ Key, pb : Key, Bob-I: ?[Msg], Bob-O : ![Msg]) {
  val nonce-a = generateNonce()
  val pk = generateSymmetricKey()
  Bob-O ! (Alice, encrypt(pb, (Alice, nonce-a)))
  val (nonce-a-b, name, nonce-b) = decrypt(pv, Bob-I ? ())
  if (nonce-a == nonce-a-b && name == Bob)) {
    Bob-O ! encrypt(pb, (nonce-b, pv))   <-- Message #3
    Alice-session(pk, Bob-I, Bob-O)
  }
}
```

Language-based Information Flow Control

The Needham-Schroeder-Lowe protocol for authentication, given public keys:

1. $A \rightarrow B$: Alice, $E_{k_B}(\text{Alice} \,\|\, nonce_A)$
2. $B \rightarrow A$: $E_{k_A}(nonce_A \,\|\, \text{Bob} \,\|\, nonce_B)$
3. $A \rightarrow B$: $E_{k_B}(nonce_B \,\|\, k_s)$
4. $A \rightarrow B$: $E_{k_s}(m_1 \,\|\, \text{Alice} \,\|\, counter)$
5. etc.

Some code implementing Alice:

Alice's private key **Bob's public key**

```
proc Alice(pv : ♦ Key, pb : Key, Bob-I: ?[Msg], Bob-O : ![Msg]) {
  val nonce-a = generateNonce()
  val pk = generateSymmetricKey()
  Bob-O ! (Alice, encrypt(pb, (Alice, nonce-a)))
  val (nonce-a-b, name, nonce-b) = decrypt(pv, Bob-I ? ())
  if (nonce-a == nonce-a-b && name == Bob)) {
    Bob-O ! encrypt(pb, (nonce-b, pv))   <-- Message #3
    Alice-session(pk, Bob-I, Bob-O)
  }
}
```

Language-based Information Flow Control

The Needham-Schroeder-Lowe protocol for authentication, given public keys:

1. $A \rightarrow B$: Alice, $E_{k_B}(\text{Alice} \,\|\, nonce_A)$
2. $B \rightarrow A$: $E_{k_A}(nonce_A \,\|\, \text{Bob} \,\|\, nonce_B)$
3. $A \rightarrow B$: $E_{k_B}(nonce_B \,\|\, k_s)$
4. $A \rightarrow B$: $E_{k_s}(m_1 \,\|\, \text{Alice} \,\|\, counter)$
5. etc.

Some code implementing Alice:

Alice's private key **Bob's public key**

```
proc Alice(pv : ♦ Key, pb : Key, Bob-I: ?[Msg], Bob-O : ![Msg]) {
  val nonce-a = generateNonce()
  val pk = generateSymmetricKey()
  Bob-O ! (Alice, encrypt(pb, (Alice, nonce-a)))
  val (nonce-a-b, name, nonce-b) = decrypt(pv, Bob-I ? ())
  if (nonce-a == nonce-a-b && name == Bob)) {
    Bob-O ! encrypt(pb, (nonce-b, pv))
    Alice-session(pk, Bob-I, Bob-O)
  }
}
```

Message #3

TYPE ERROR

Modalities for Information Flow Control

✤ Modalities = unary operations on types. $\mathrm{T}(\mathrm{A})$ $\Box A$ $\blacklozenge A$ $||A||$

✤ They can be used to control information flow, as in previous example. One can copy techniques from the **proof theory of modal logic**.

✤ The hard part is proving **noninterference**:

[...] High-security data does not "interfere" with the calculation of low-security outputs [...]

✤ A notion due to [Goguen and Meseguer, 1982]. This definition: from seminal paper on **dependency core calculus** [Abadi et al 1999].

Modalities for Information Flow: an example

✤ An example: for each type A, a type $\blacklozenge A$ ⟵ **"high security A"**

✤ Can always get a $\blacklozenge A$: $$\frac{\Gamma \vdash M : A}{\Gamma \vdash [M] : \blacklozenge A}$$

✤ **I can use a high-security value when computing another high-security value:**

$$\frac{\Gamma \vdash M : \blacklozenge A \quad \Gamma, x : A \vdash N : \blacklozenge C}{\Gamma \vdash \text{ let x} = M \text{ in } N : \blacklozenge C}$$

✤ Reduction: let x $= [M]$ in $N \to N[M/x]$

✤ Noninterference:

> If $x : \blacklozenge A \vdash E : \text{Bool}$ and $\vdash M, N : \blacklozenge A$ then $E[M/x]$ and $E[N/x]$ compute the same boolean value.

How can one go about proving this?

Modalities for Information Flow: an example

✤ An example: for each type A, a type $\blacklozenge A$ ⟵ **"high security A"**

✤ Can always get a $\blacklozenge A$: $$\frac{\Gamma \vdash M : A}{\Gamma \vdash [M] : \blacklozenge A}$$

✤ **I can use a high-security value when computing another high-security value:**

$$\frac{\Gamma \vdash M : \blacklozenge A \quad \Gamma, x : A \vdash N : \blacklozenge C}{\Gamma \vdash \text{ let x} = M \text{ in } N : \blacklozenge C}$$

a.k.a.
"Moggi's monadic metalanguage"

✤ Reduction: let x $= [M]$ in $N \to N[M/x]$

✤ Noninterference:

If $x : \blacklozenge A \vdash E : \text{Bool}$ and $\vdash M, N : \blacklozenge A$ then
$E[M/x]$ and $E[N/x]$ compute the same boolean value.

How can one go about proving this?

Proving noninterference

- In the last ten to fifteen years: **logical relations**.

Proving noninterference

✤ In the last ten to fifteen years: **logical relations**.

Logical Methods in Computer Science
Vol. 4 (3:10) 2008, pp. 1–31
www.lmcs-online.org

Submitted Sep. 25, 2007
Published Sep. 20, 2008

PROVING NONINTERFERENCE BY A FULLY COMPLETE TRANSLATION TO THE SIMPLY TYPED λ-CALCULUS*

NAOKATA SHIKUMA AND ATSUSHI IGARASHI

Graduate School of Informatics, Kyoto University, Kyoto 606-8501 Japan
e-mail address: {naokata,igarashi}@kuis.kyoto-u.ac.jp

Abstract. Tse and Zdancewic have formalized the notion of noninterference for Abadi et al.'s DCC in terms of logical relations and given a proof of noninterference by reduction to parametricity of System F. Unfortunately, their proof contains errors in a key lemma that their translation from DCC to System F preserves the logical relations defined for both calculi. In fact, we have found a counterexample for it. In this article, instead of DCC, we prove noninterference for *sealing calculus*, a new variant of DCC, by reduction to the basic lemma of a logical relation for the simply typed λ-calculus, using a *fully complete* translation to the simply typed λ-calculus. Full completeness plays an important role in showing preservation of the two logical relations through the translation. Also, we investigate relationship among sealing calculus, DCC, and an extension of DCC by Tse and Zdancewic and show that the first and the last of the three are equivalent.

1. Introduction

Background. Dependency analysis is a family of static program analyses to trace dependencies between inputs and outputs of a given program. For example, information flow analysis [3], binding-time analysis [8], and call tracking [20] are its instances. One of the most important correctness criteria of the dependency analysis is called *noninterference* [5], which roughly means that, for any pair of program inputs that are equivalent from the

Proving noninterference

✤ In the last ten to fifteen years: **logical relations**.

Logical Methods in Computer Science
Vol. 4 (3:10) 2008, pp. 1–31
www.lmcs-online.org

Submitted Sep. 25, 2007
Published Sep. 20, 2008

PROVING NONINTERFERENCE BY A FULLY COMPLETE TRANSLATION TO THE SIMPLY TYPED λ-CALCULUS*

NAOKATA SHIKUMA AND ATSUSHI IGARASHI

Graduate School of Informatics, Kyoto University, Kyoto 606-8501 Japan
e-mail address: {naokata,igarashi}@kuis.kyoto-u.ac.jp

Abstract. Tse and Zdancewic have formalized the notion of noninterference for Abadi et al.'s DCC in terms of logical relations and given a proof of noninterference by reduction to parametricity of System F. Unfortunately, their proof contains errors in a key lemma that their translation from DCC to System F preserves the logical relations defined for both calculi. In fact, we have found a counterexample for it. In this article, instead of DCC, we prove noninterference for *sealing calculus*, a new variant of DCC, by reduction

Proving noninterference

✤ In the last ten to fifteen years: **logical relations**.

✤ Beautiful, but long and complicated syntactic proofs.

✤ This talk: using **categorical algebra** to simplify these proofs.

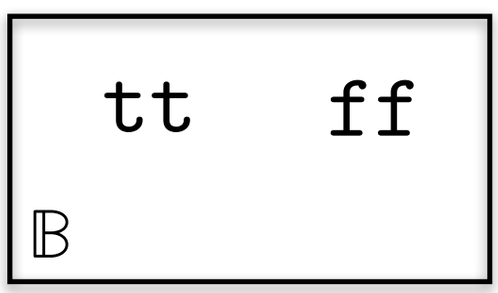
✤ Main result: one can use basic **axiomatic cohesion** to reason about information flow and prove noninterference results.

✤ **Axiomatic cohesion**: a theory developed by F. William Lawvere.
Aim: axiomatic desc. of all sorts of **geometric/topological spaces**.

Cohesion

Spaces **(= points + cohesion)**

Sets **(= points)**

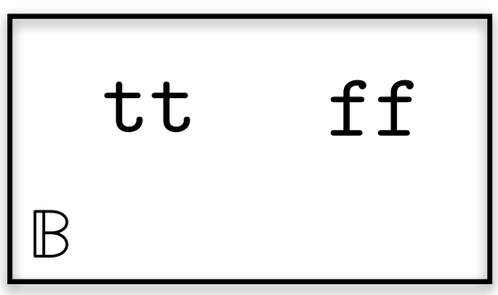


Cohesion

Spaces **(= points + cohesion)**

U

Sets **(= points)**



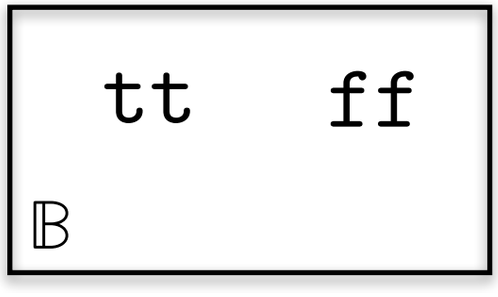
Cohesion

Spaces **(= points + cohesion)**

X = space
U(X) = points of space X (forget cohesion)

U

Sets **(= points)**

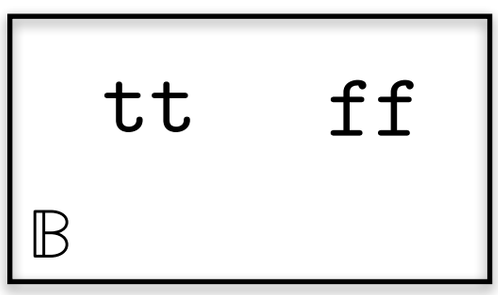


Cohesion

Spaces **(= points + cohesion)**

U

Sets **(= points)**

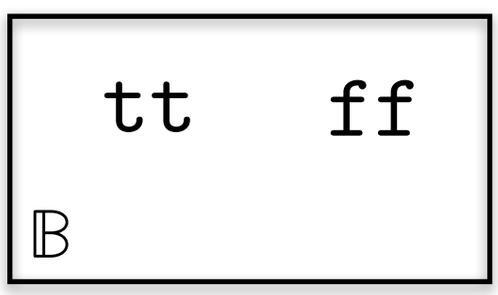


Cohesion

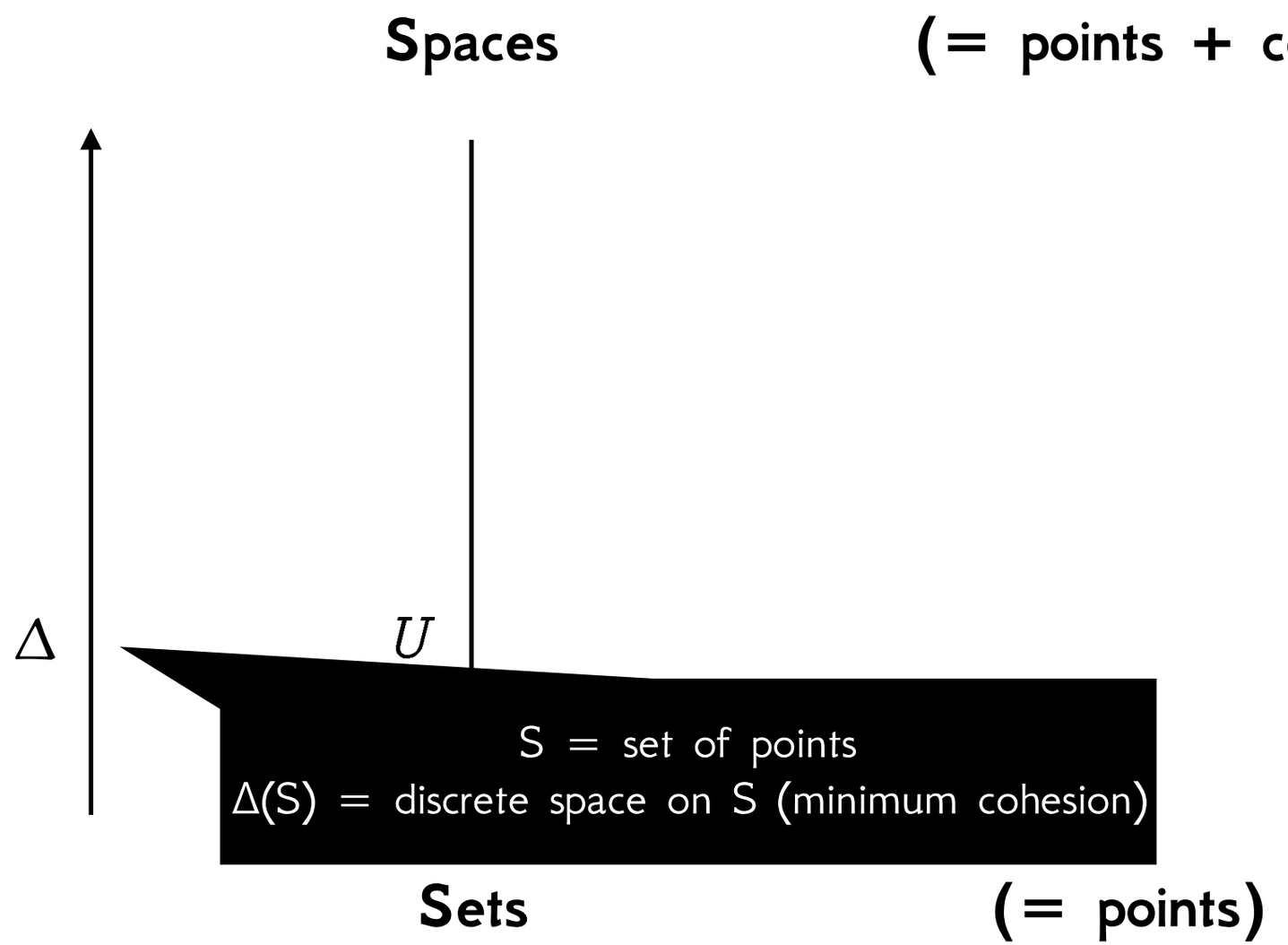
Spaces **(= points + cohesion)**

Δ U

Sets **(= points)**



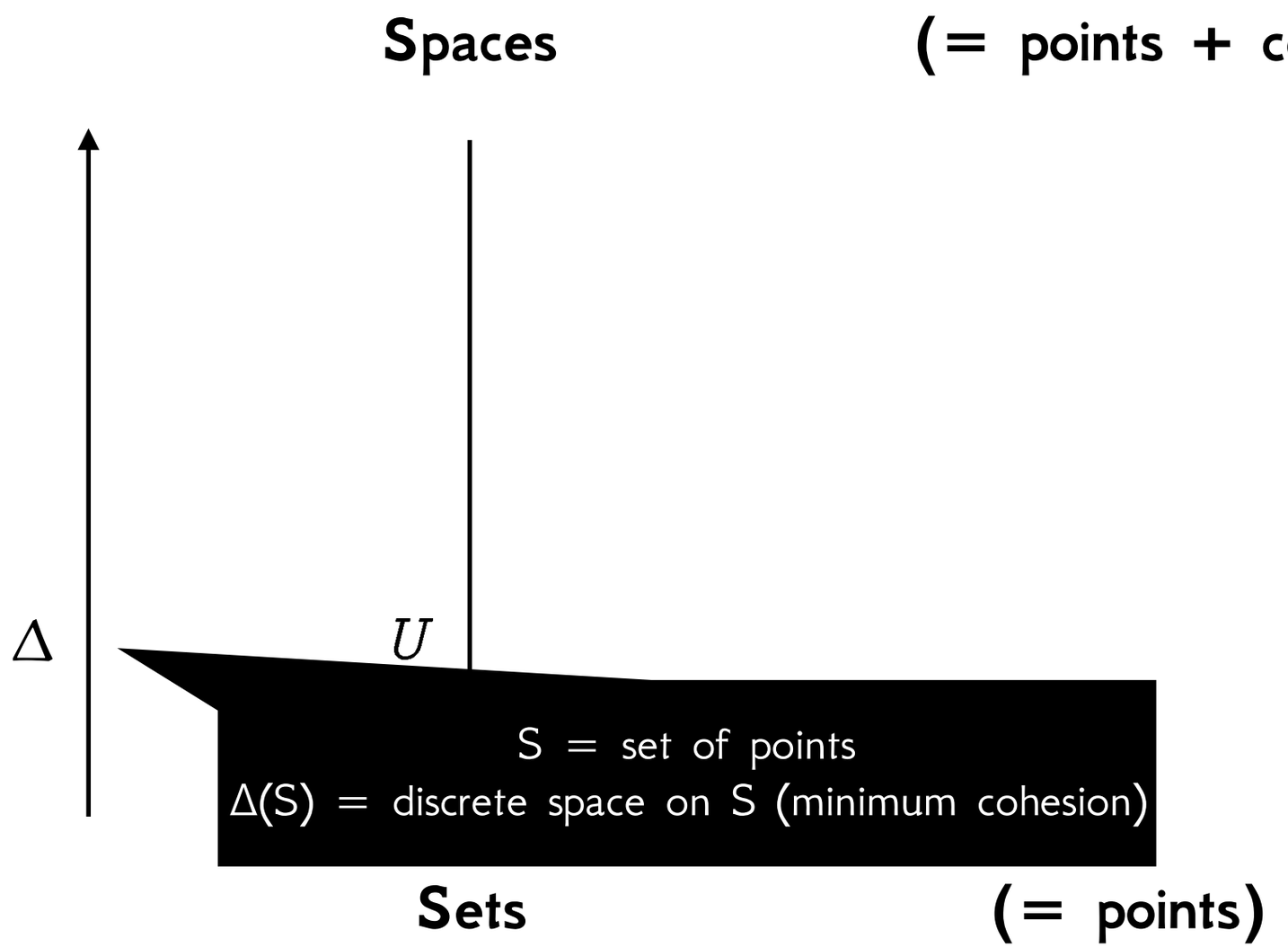
Cohesion

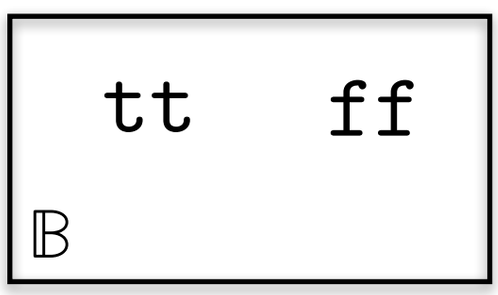


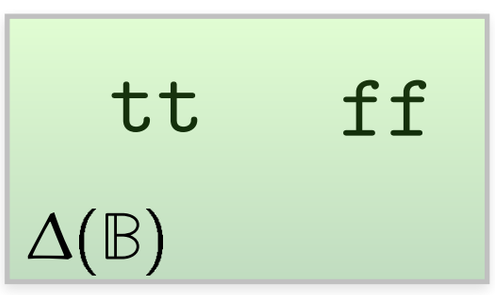

```
  tt   ff

𝔹
```

Cohesion


Spaces
(= points + cohesion)
Δ
U
S = set of points
Δ(S) = discrete space on S (minimum cohesion)
Sets
(= points)


tt ff
𝔹


tt ff
Δ(𝔹)

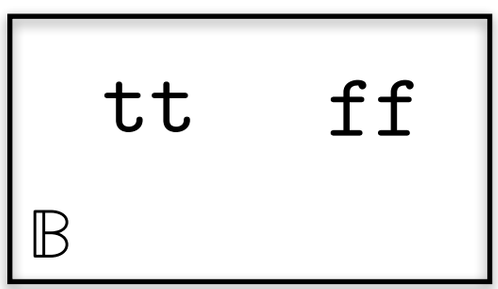
Cohesion

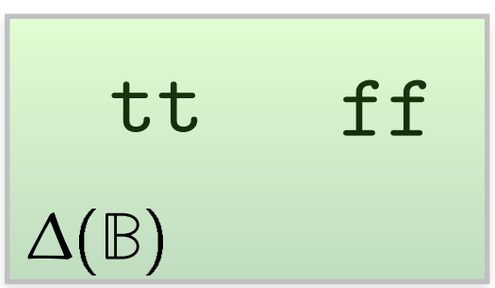
Spaces **(= points + cohesion)**

Δ

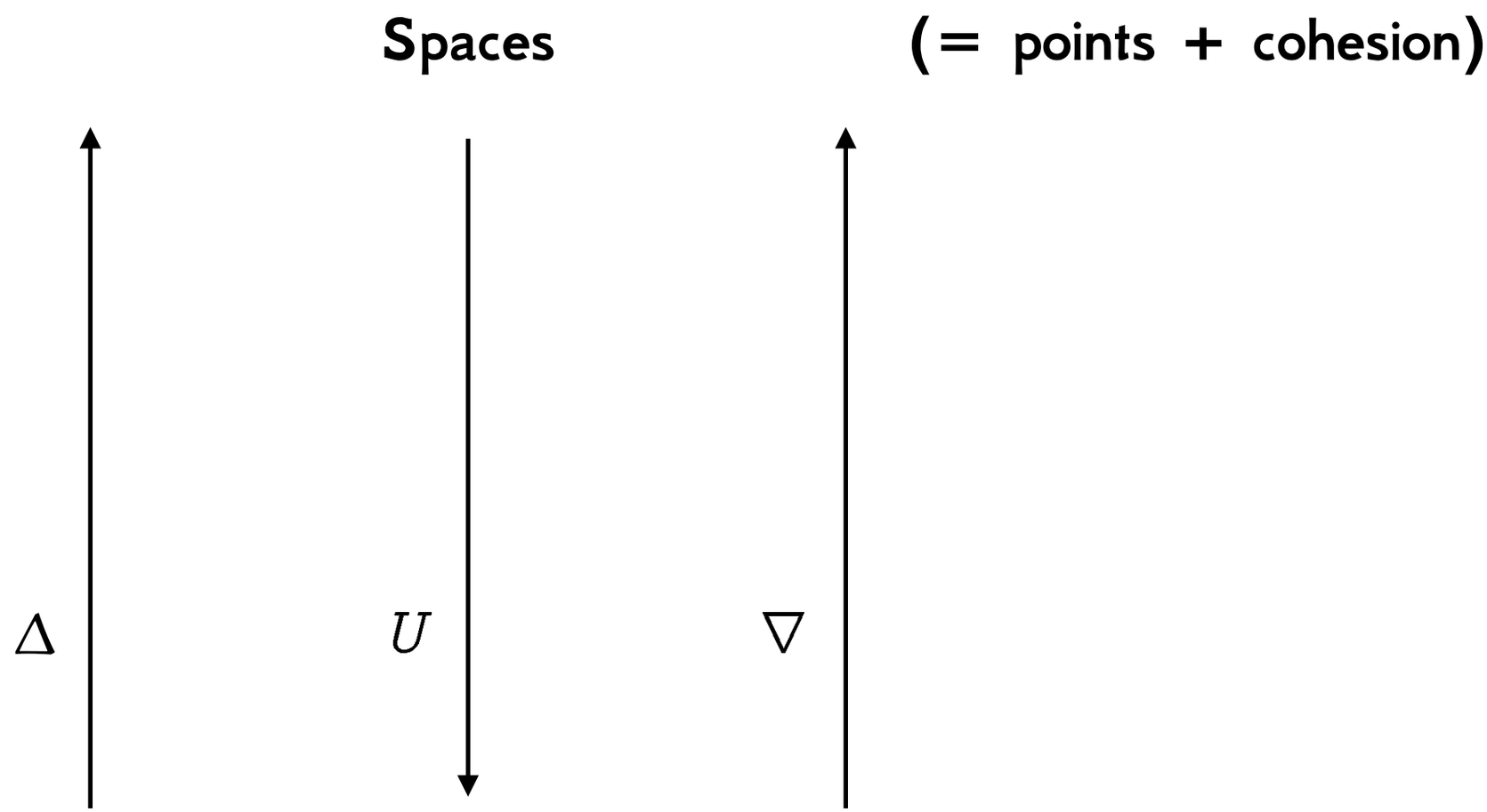
U

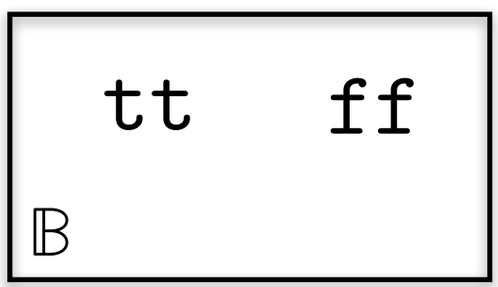
Sets **(= points)**

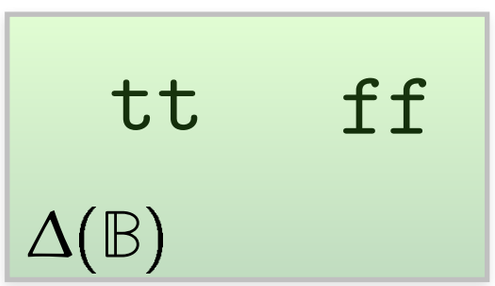




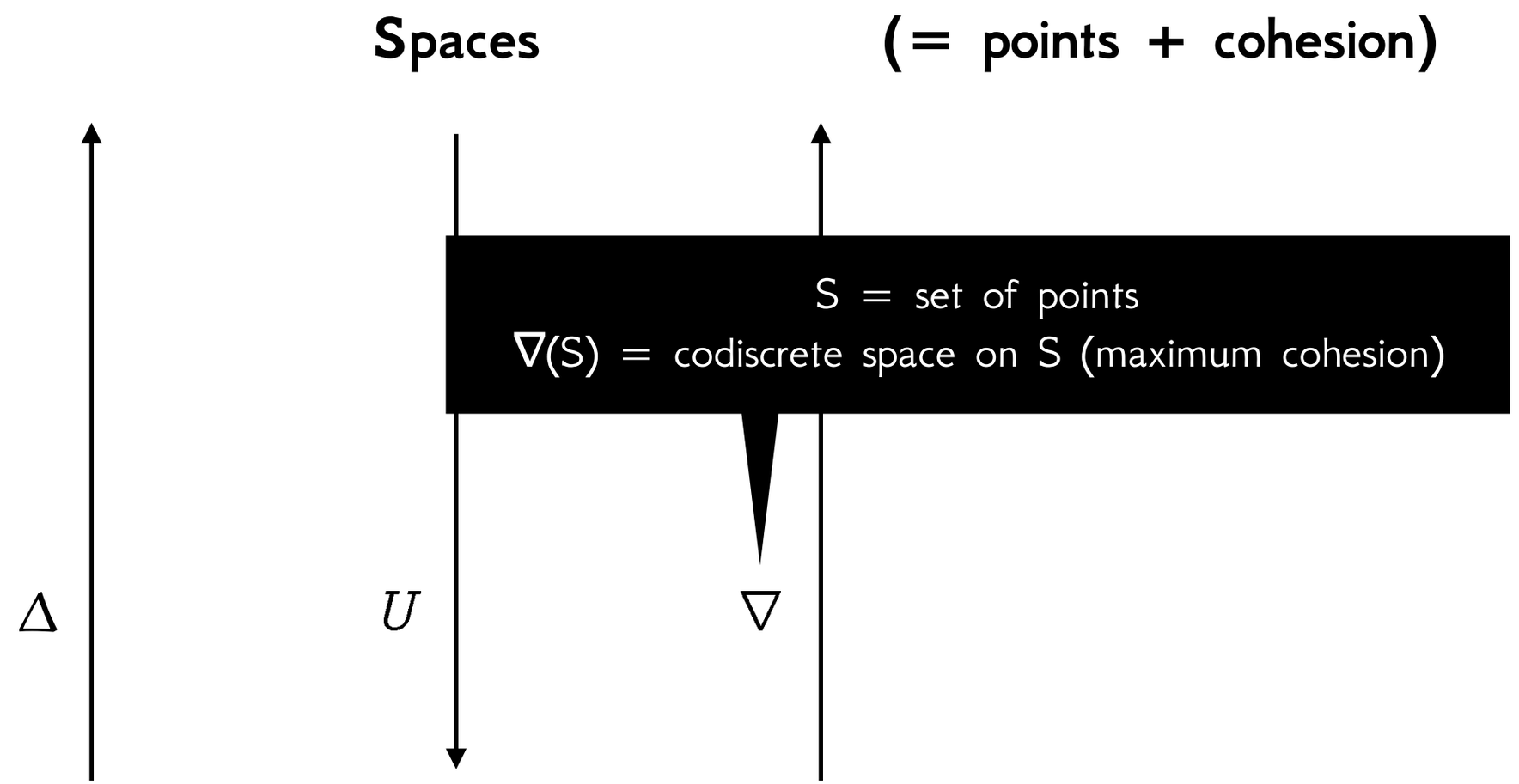
Cohesion

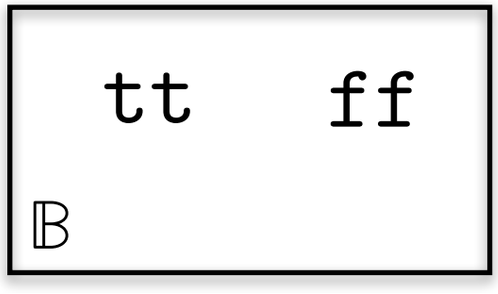


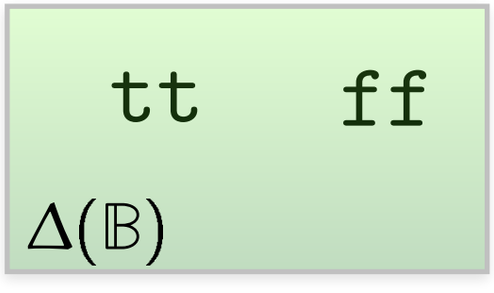




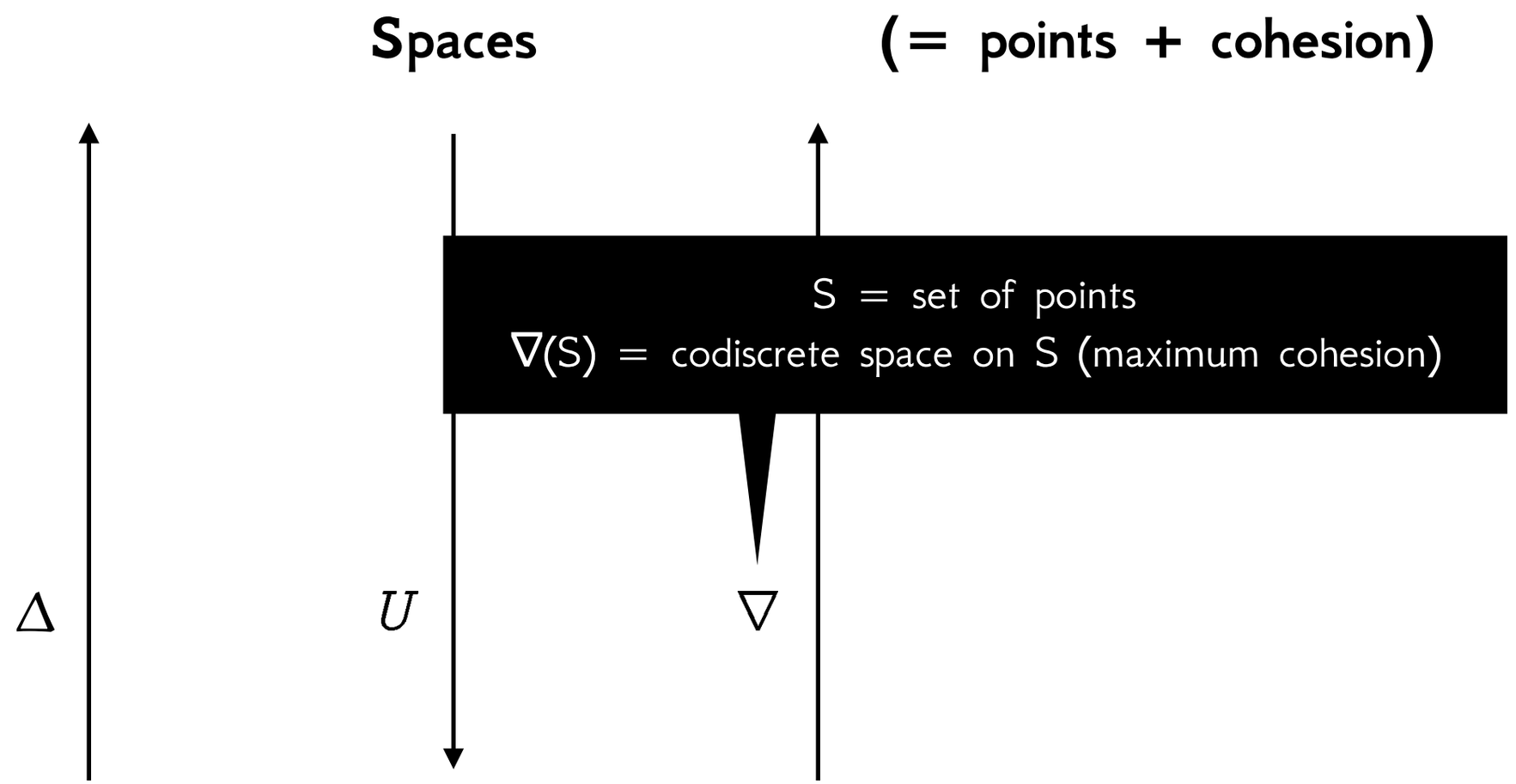
Cohesion

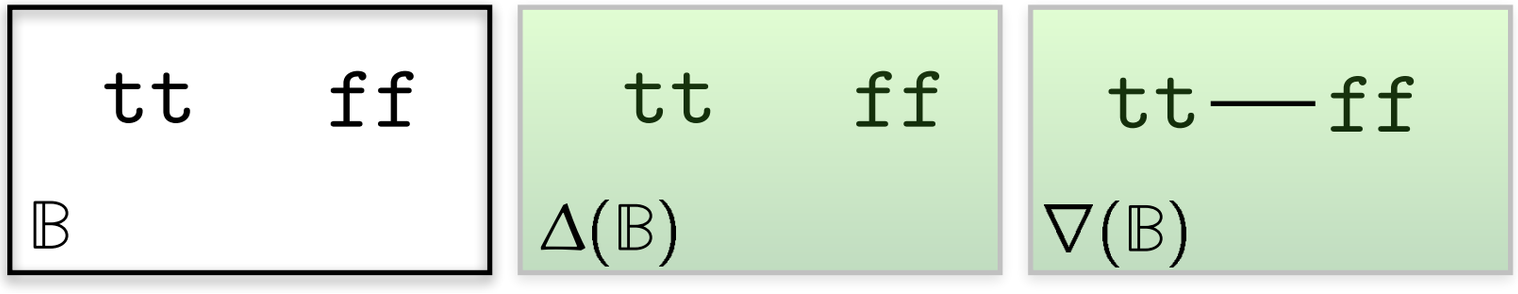

Spaces
(= points + cohesion)
S = set of points
∇(S) = codiscrete space on S (maximum cohesion)
Δ
U
∇
Sets
(= points)


tt ff
𝔹

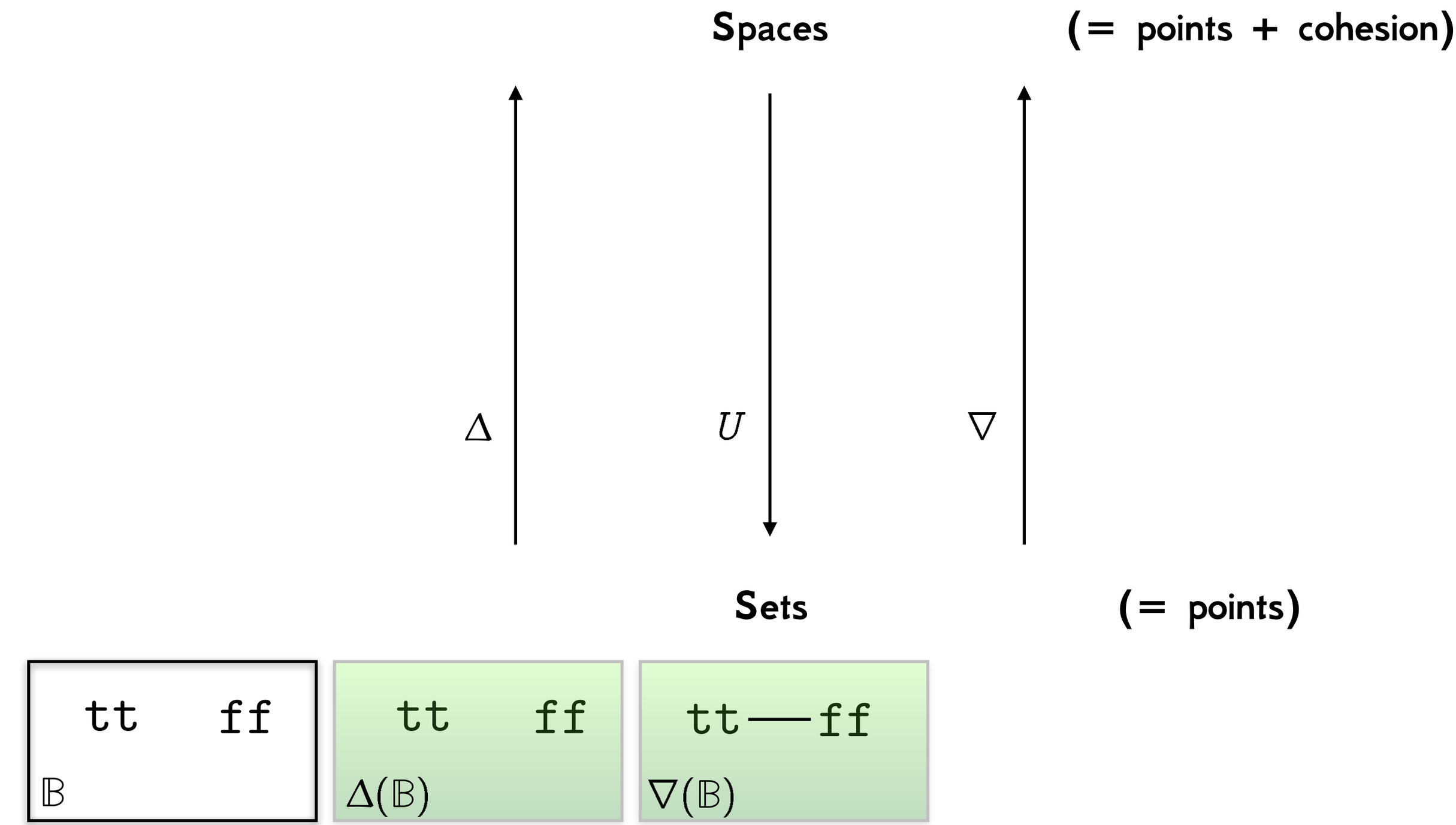

tt ff
Δ(𝔹)

Cohesion

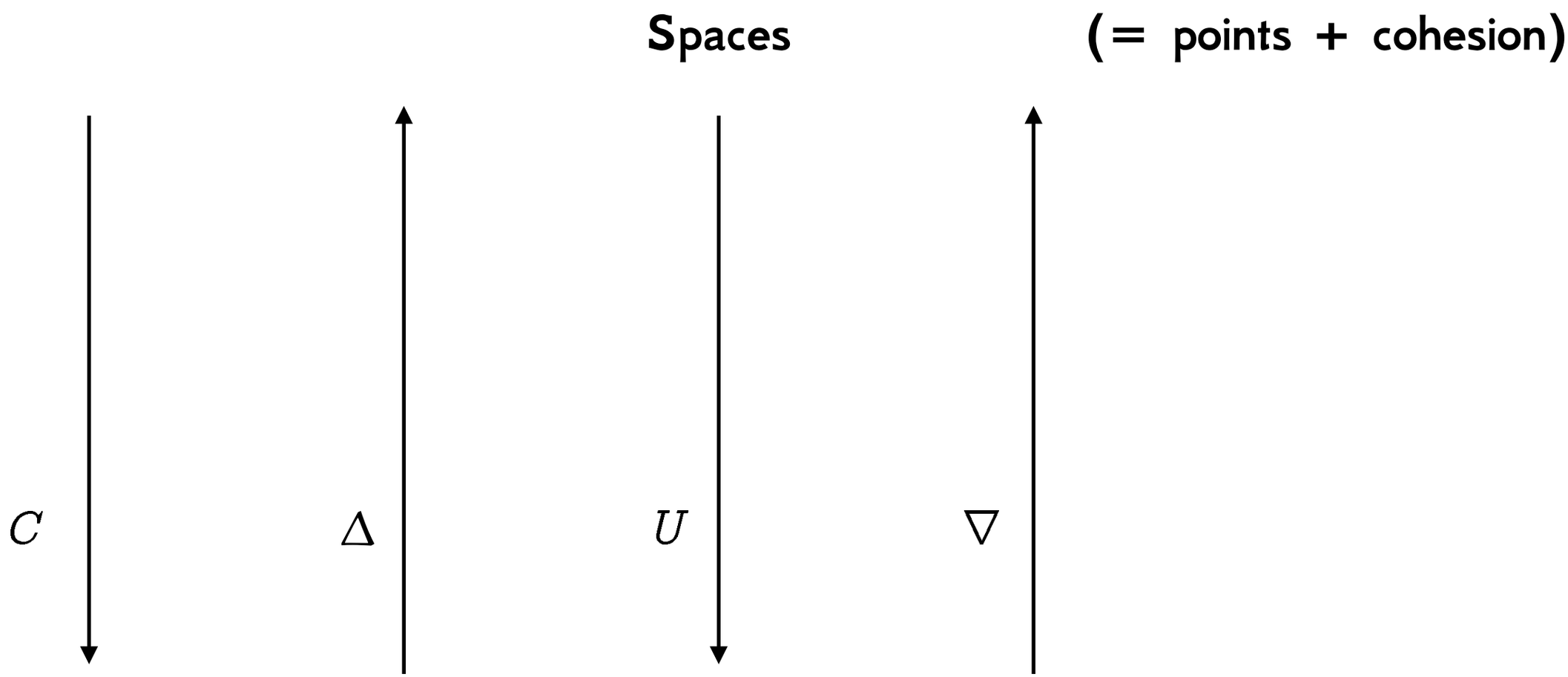

Spaces
(= points + cohesion)
S = set of points
∇(S) = codiscrete space on S (maximum cohesion)
Δ
U
∇
Sets
(= points)

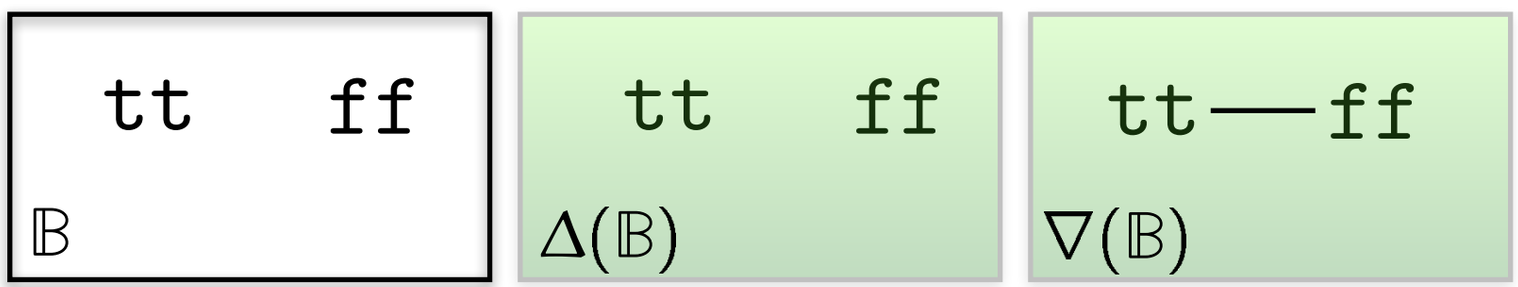

tt ff
𝔹
tt ff
Δ(𝔹)
tt—ff
∇(𝔹)

Cohesion

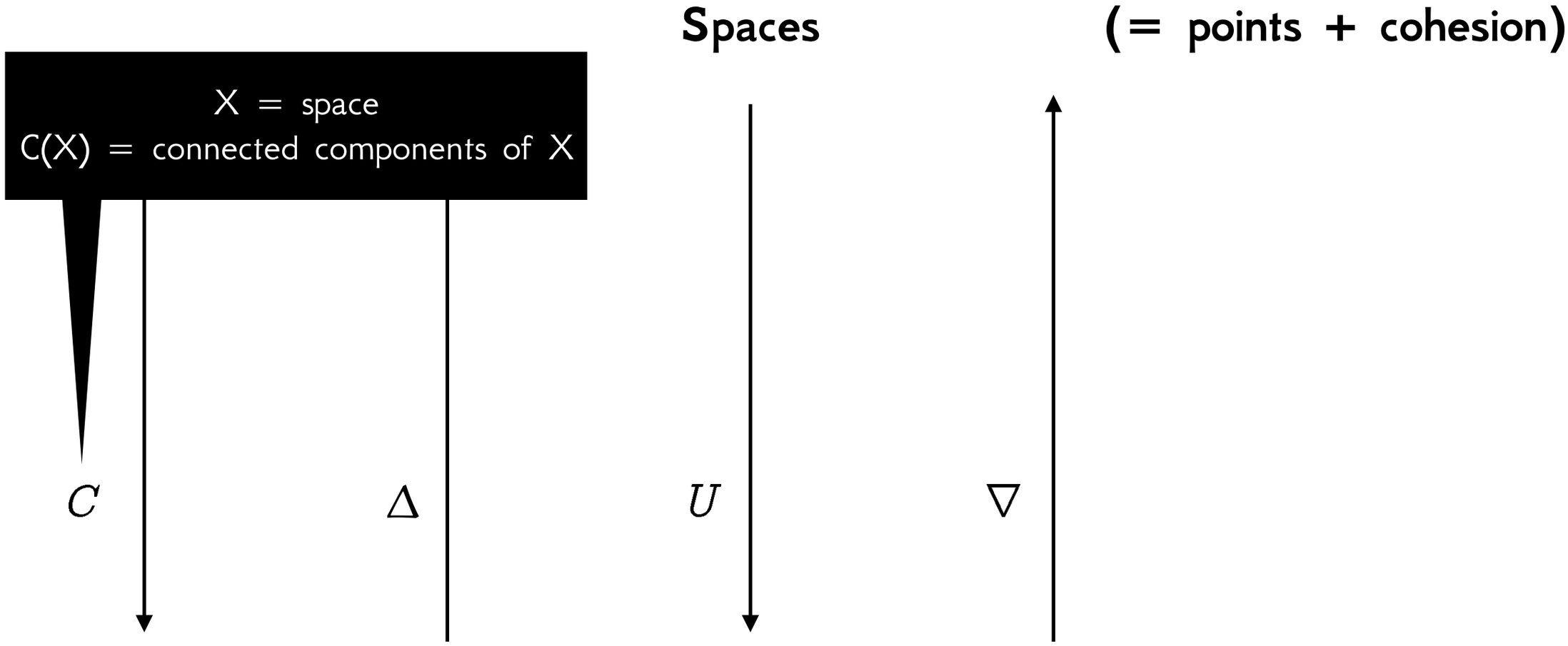


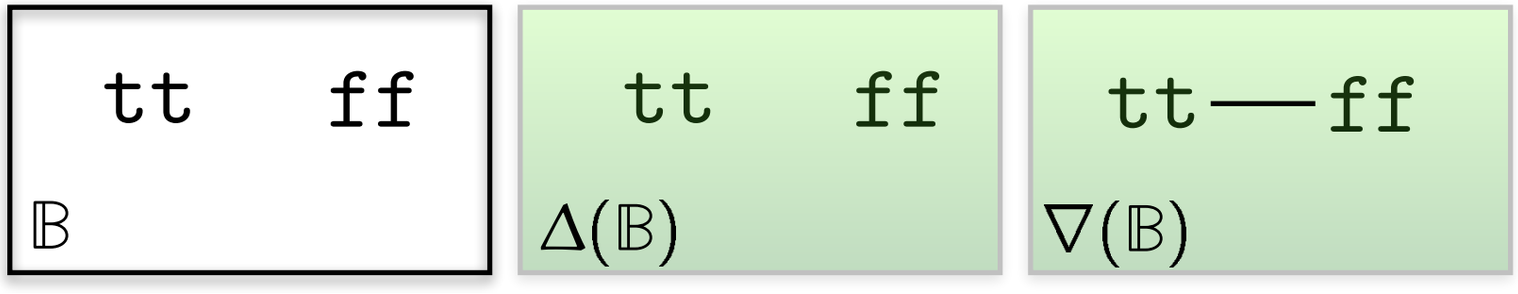
Cohesion



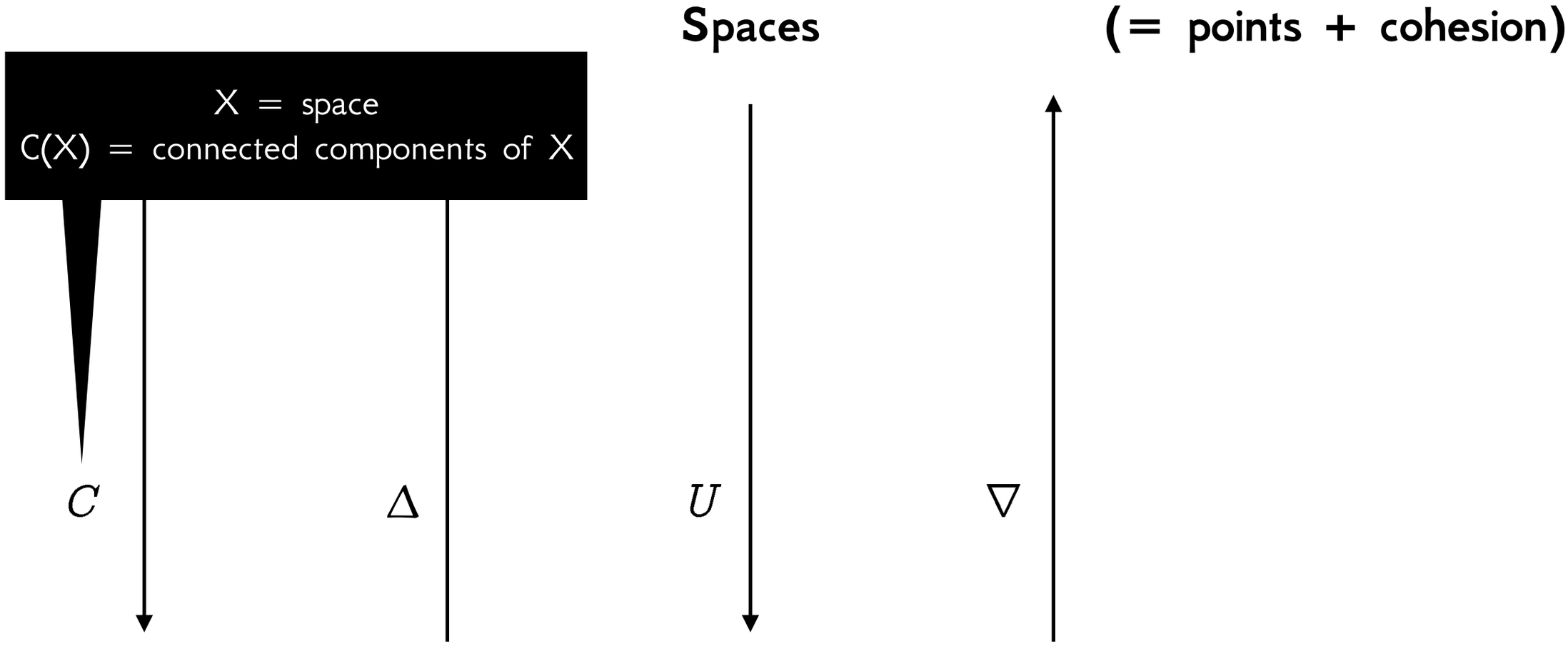


Cohesion

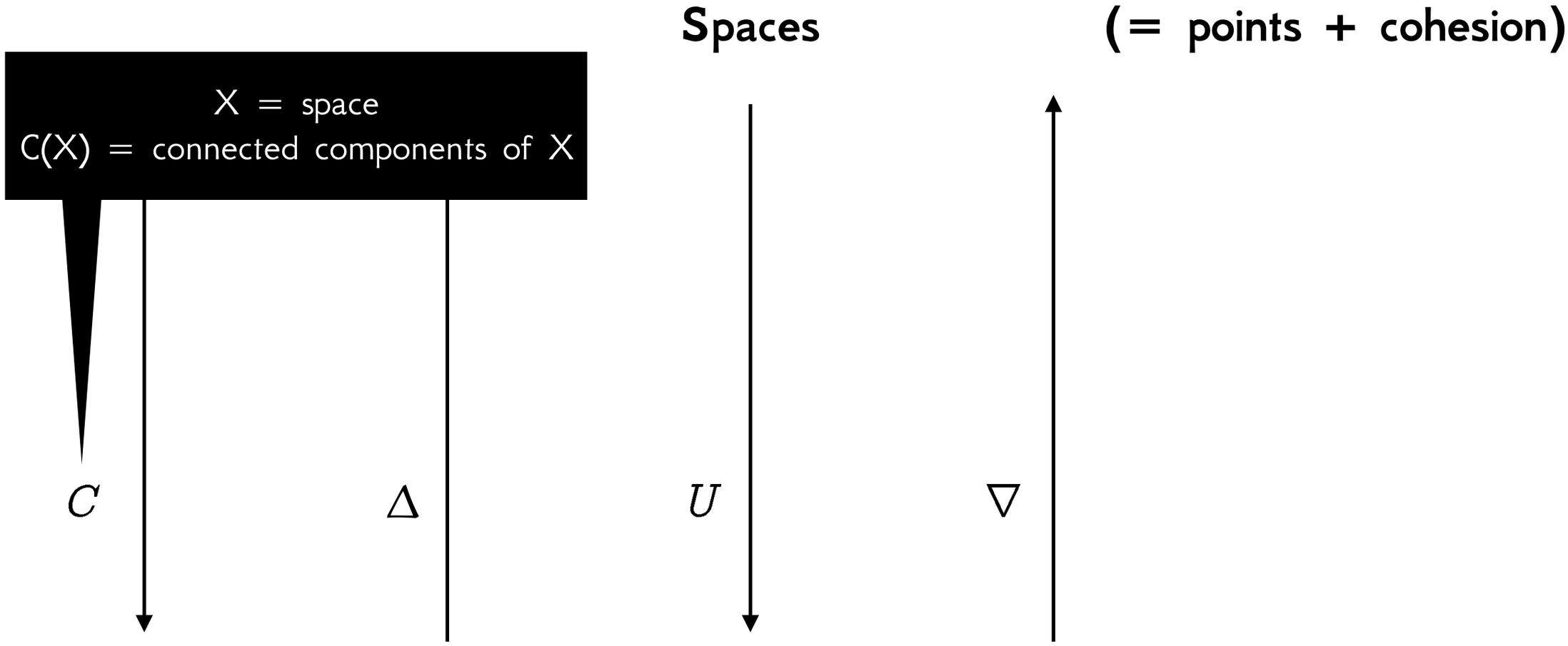

X = space
C(X) = connected components of X
Spaces
(= points + cohesion)
C
Δ
U
∇
Sets
(= points)


tt ff
𝔹
tt ff
Δ(𝔹)
tt—ff
∇(𝔹)

Cohesion


X = space
C(X) = connected components of X
C
Δ
Spaces
(= points + cohesion)
U
∇
Sets
(= points)
tt ff
𝔹
tt ff
Δ(𝔹)
tt—ff
∇(𝔹)
{tt} {ff}
C(Δ(𝔹))

Cohesion


X = space
C(X) = connected components of X
C
Spaces
Δ
U
∇
(= points + cohesion)
Sets
(= points)
tt ff
𝔹
tt ff
Δ(𝔹)
tt—ff
∇(𝔹)
{tt} {ff}
C(Δ(𝔹))
{tt, ff}
C(∇(𝔹))

CRIB

$U(X)$ = points of space X (forget cohesion)

$\Delta(S)$ = discrete space on S (min. cohesion)

$\nabla(S)$ = codiscrete space on S (max. cohesion)

$C(X)$ = connected components of X

Cohesion

Spaces **(= points + cohesion)**

C Δ U ∇

Sets **(= points)**

$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$
`tt` `ff`	`tt` `ff`	`tt`—`ff`	`{tt}` `{ff}`	`{tt, ff}`

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**

$$\frac{S \to U(X)}{\Delta(S) \to X}$$

C Δ U ∇

Sets **(= points)**

tt ff	tt ff	tt—ff	{tt} {ff}	{tt, ff}
$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**

$$\frac{S \to U(X)}{\Delta(S) \to X}$$

C ⊣ Δ ⊣ U ⊣ ∇

Sets **(= points)**

tt ff	tt ff	tt—ff	{tt} {ff}	{tt, ff}
$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X (forget cohesion)

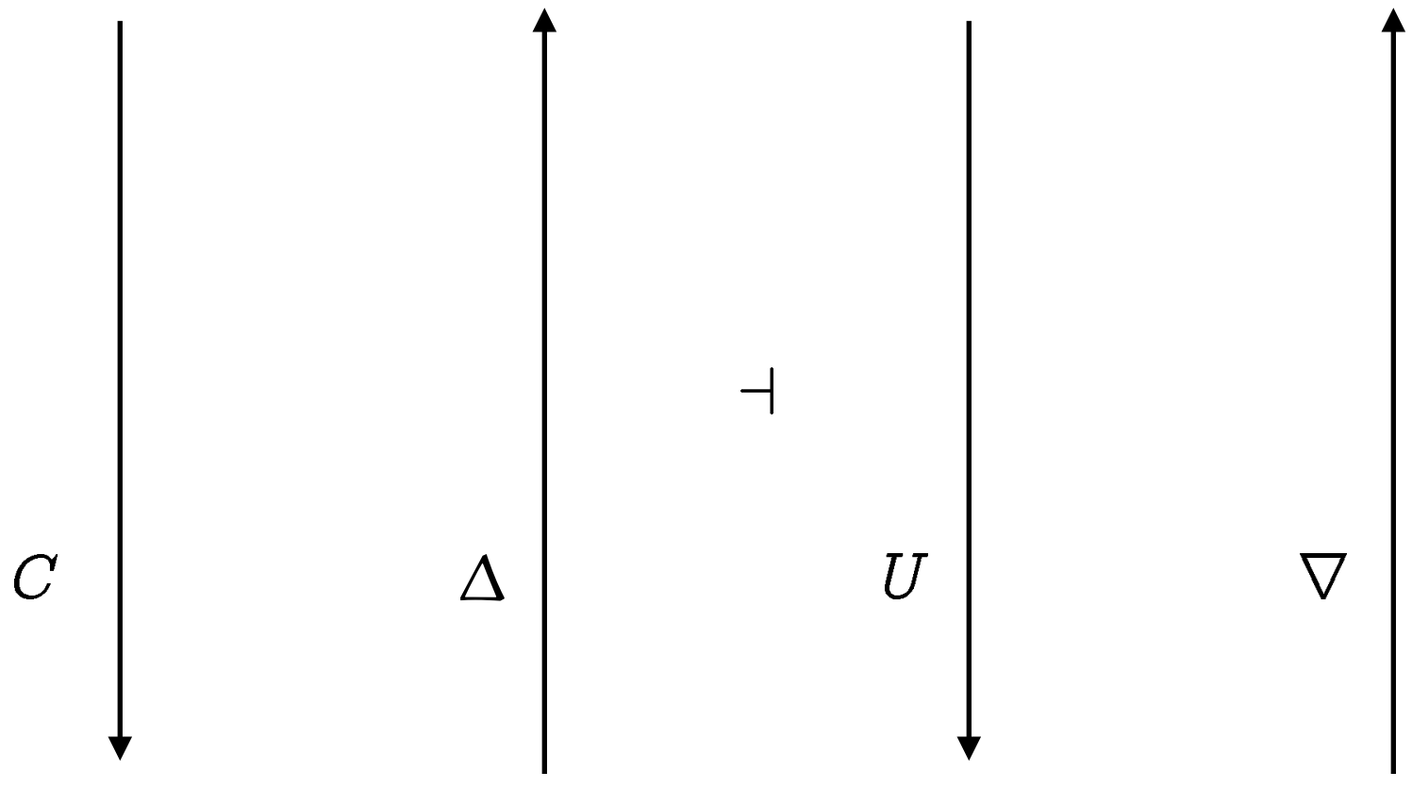
Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**



$$\frac{S \to U(X)}{\Delta(S) \to X}$$

$$\frac{U(X) \to S}{X \to \nabla(S)}$$

Sets **(= points)**

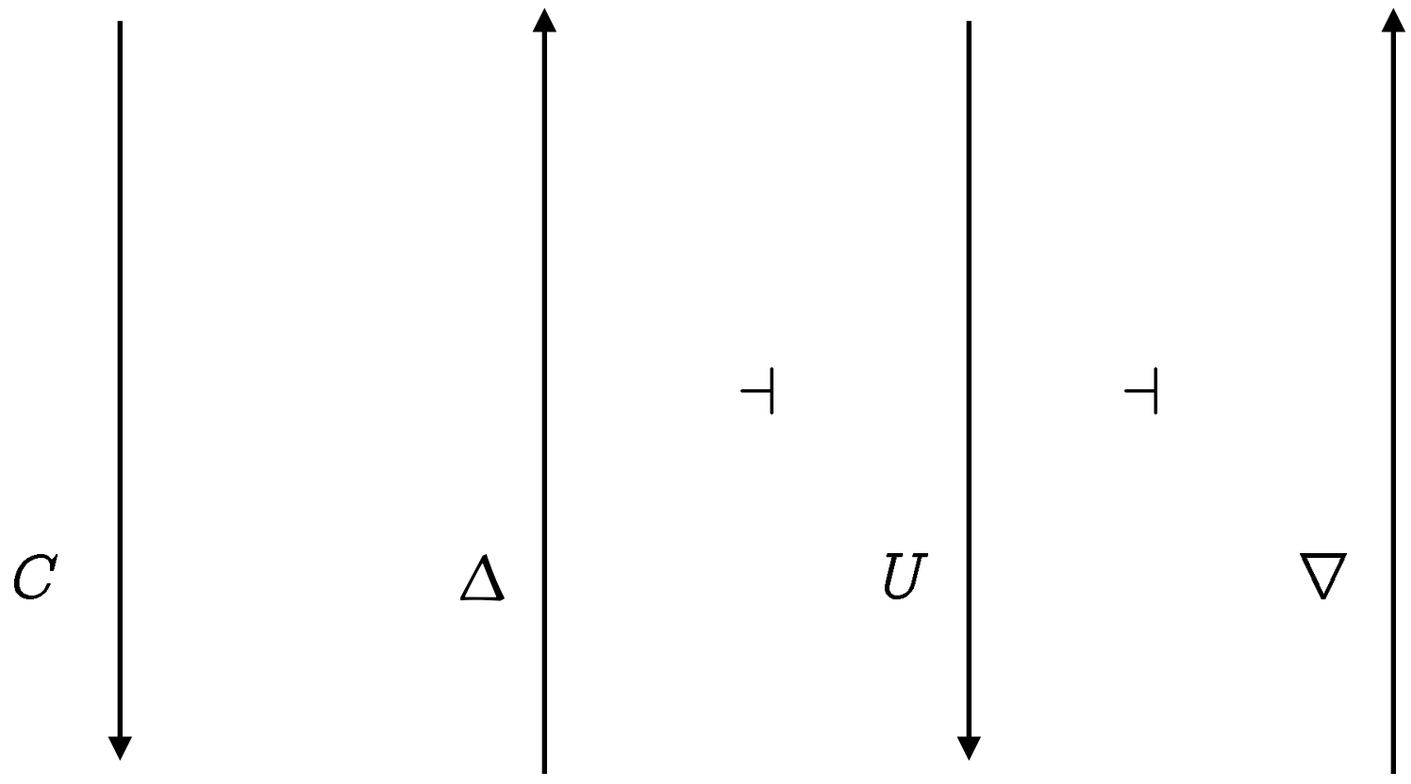
tt ff	tt ff	tt—ff	{tt} {ff}	{tt, ff}
$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**



$$\frac{S \to U(X)}{\Delta(S) \to X}$$

$$\frac{U(X) \to S}{X \to \nabla(S)}$$

Sets **(= points)**

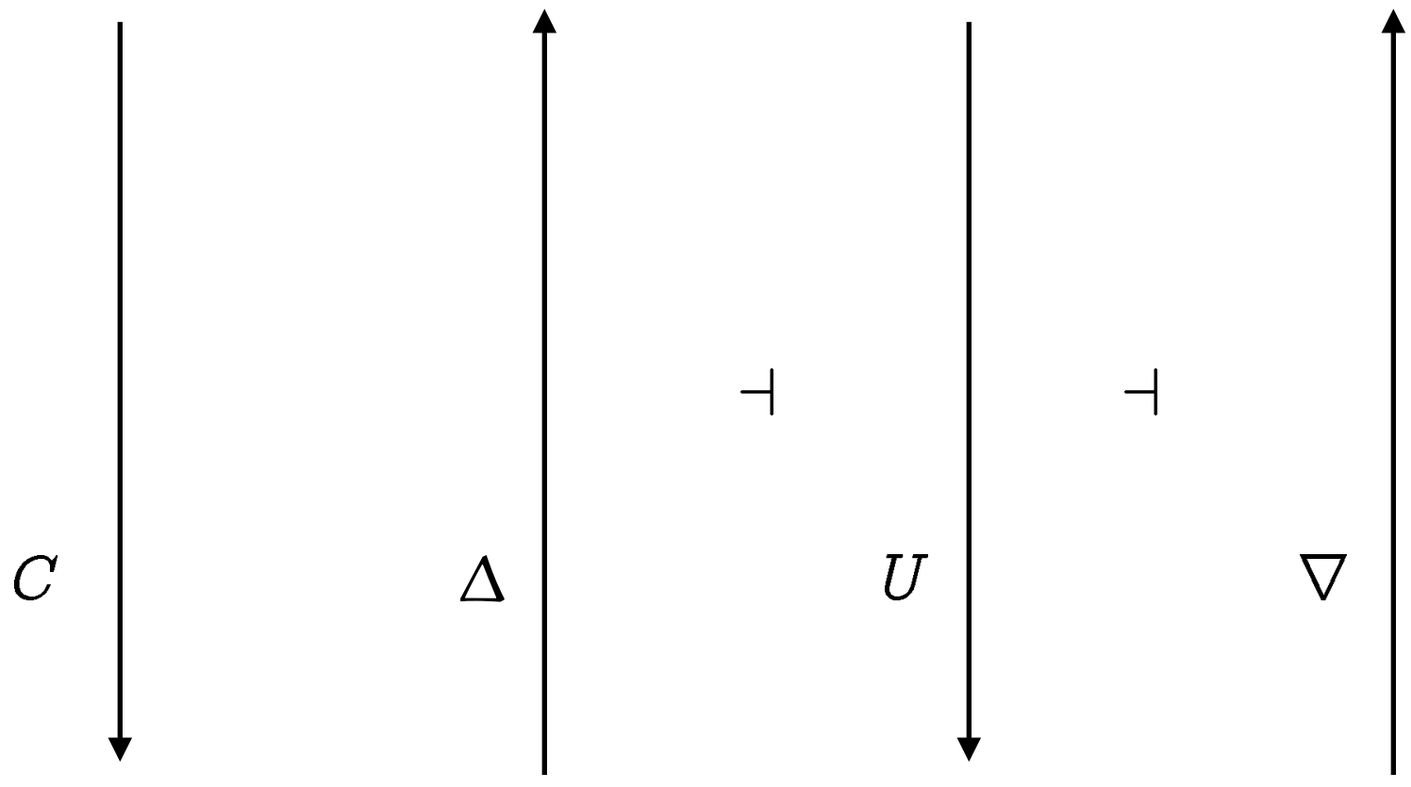
tt ff	tt ff	tt—ff	{tt} {ff}	{tt, ff}
$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**



$$\frac{S \to U(X)}{\Delta(S) \to X}$$

$$\frac{U(X) \to S}{X \to \nabla(S)}$$

$$\frac{X \to \Delta(S)}{C(X) \to S}$$

Sets **(= points)**

tt ff $\mathbb{B}$	tt ff $\Delta(\mathbb{B})$	tt—ff $\nabla(\mathbb{B})$	{tt} {ff} $C(\Delta(\mathbb{B}))$	{tt, ff} $C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X (forget cohesion)

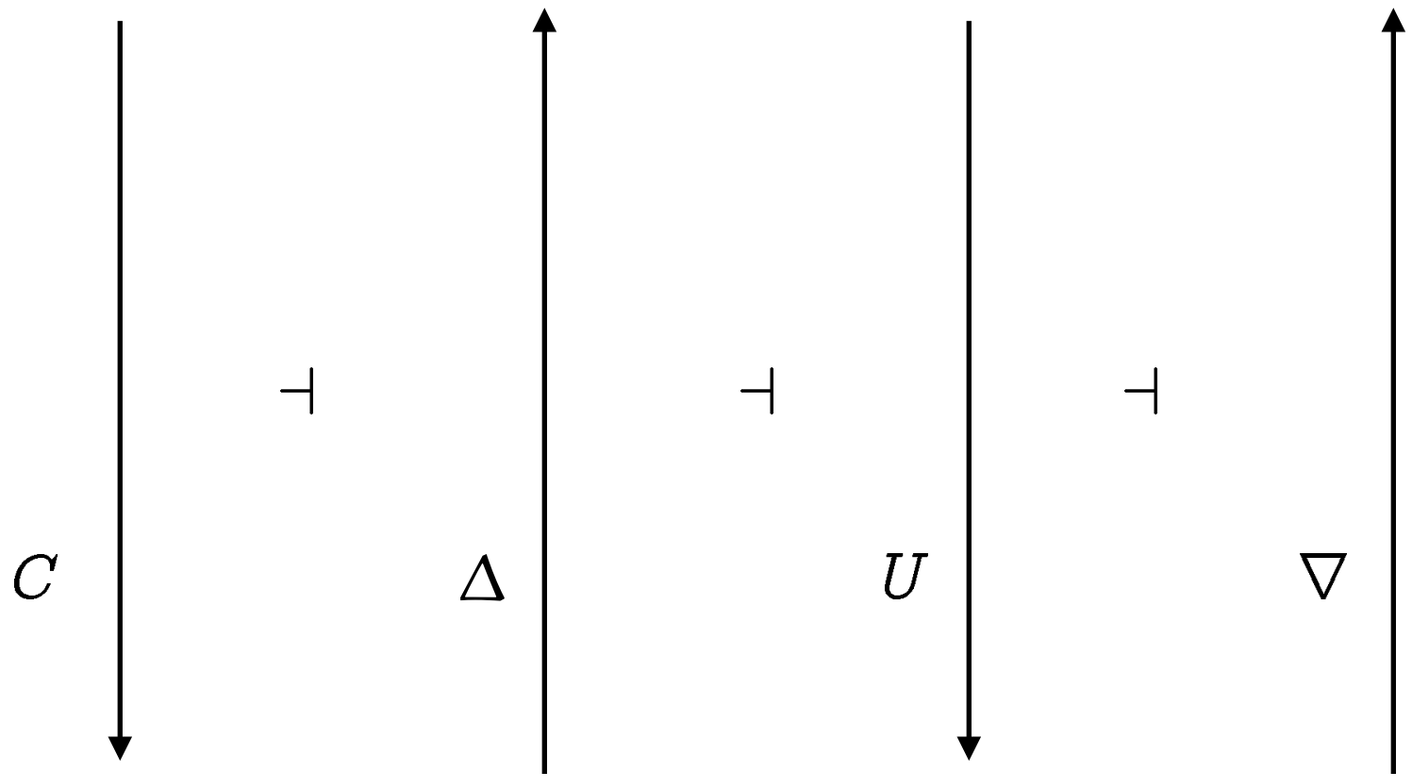
Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

Spaces **(= points + cohesion)**



$$\frac{S \to U(X)}{\Delta(S) \to X}$$

$$\frac{U(X) \to S}{X \to \nabla(S)}$$

$$\frac{X \to \Delta(S)}{C(X) \to S}$$

Sets **(= points)**

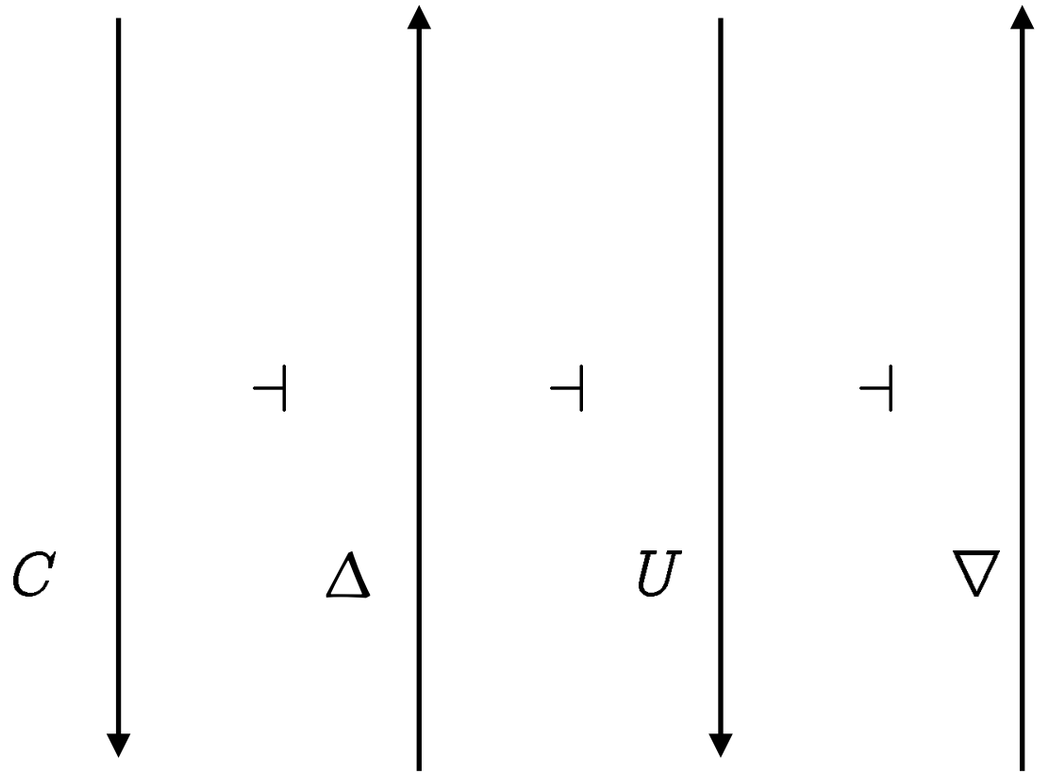
tt ff	tt ff	tt—ff	{tt} {ff}	{tt, ff}
$\mathbb{B}$	$\Delta(\mathbb{B})$	$\nabla(\mathbb{B})$	$C(\Delta(\mathbb{B}))$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces



Sets

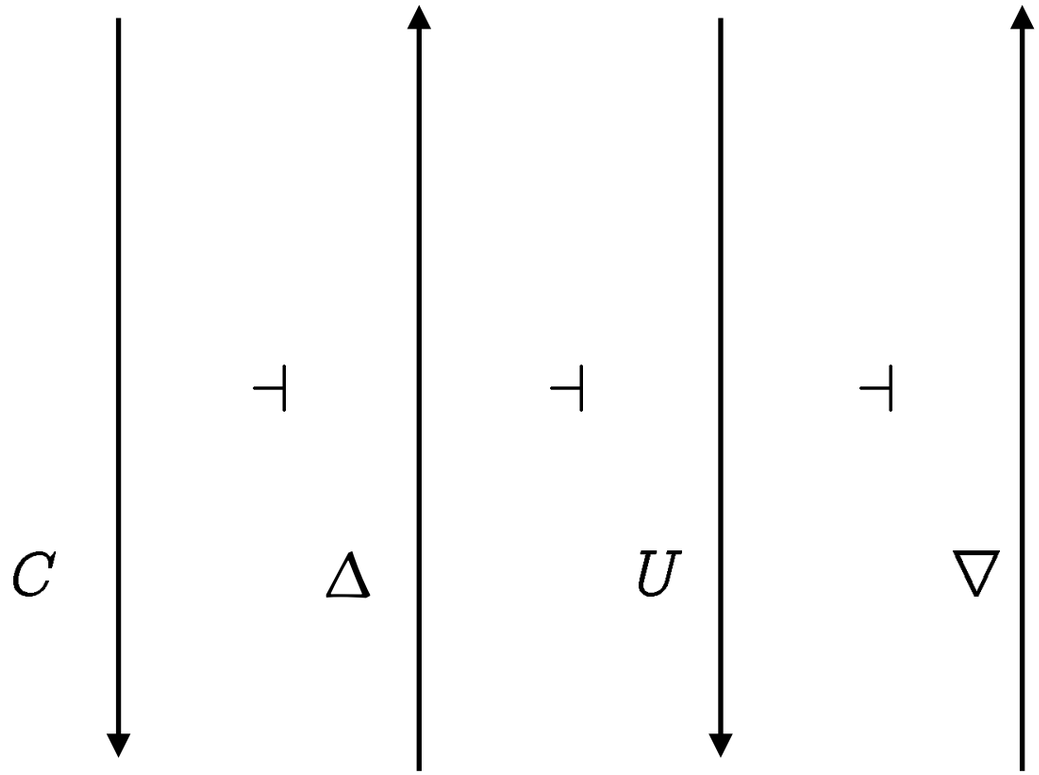
tt ff $\mathbb{B}$	tt——ff $\nabla(\mathbb{B})$	{tt, ff} $C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces



Sets

tt ff

$\mathbb{B}$

tt—ff

$\nabla(\mathbb{B})$

{tt, ff}

$C(\nabla(\mathbb{B}))$

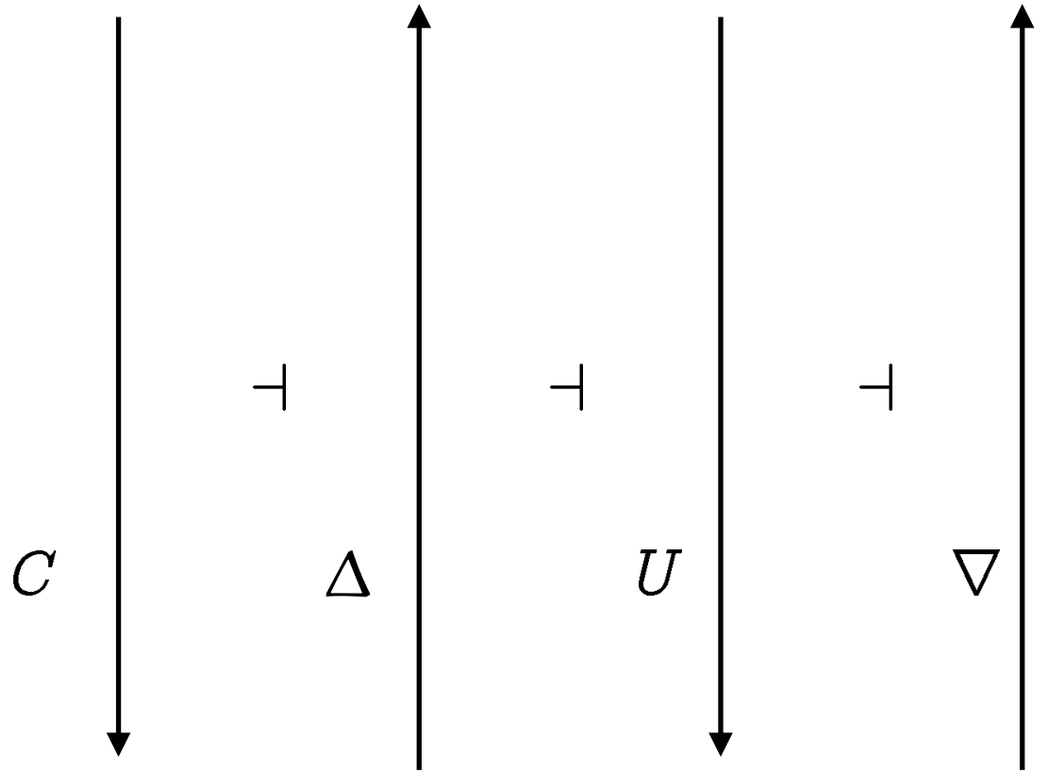
in the codiscrete space ∇(S) on S everything is "stuck together"
⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

Spaces



Sets

Axiom of
CONTRACTIBLE CODISCRETENESS:

$$\forall S.\ |C(\nabla S)| \leq 1$$

(For category theorists: the canonical $C(\nabla S) \xrightarrow{!} 1$ is a monic arrow.)

tt ff 𝔹	tt—ff ∇(𝔹)	{tt, ff} $C(\nabla(\mathbb{B}))$

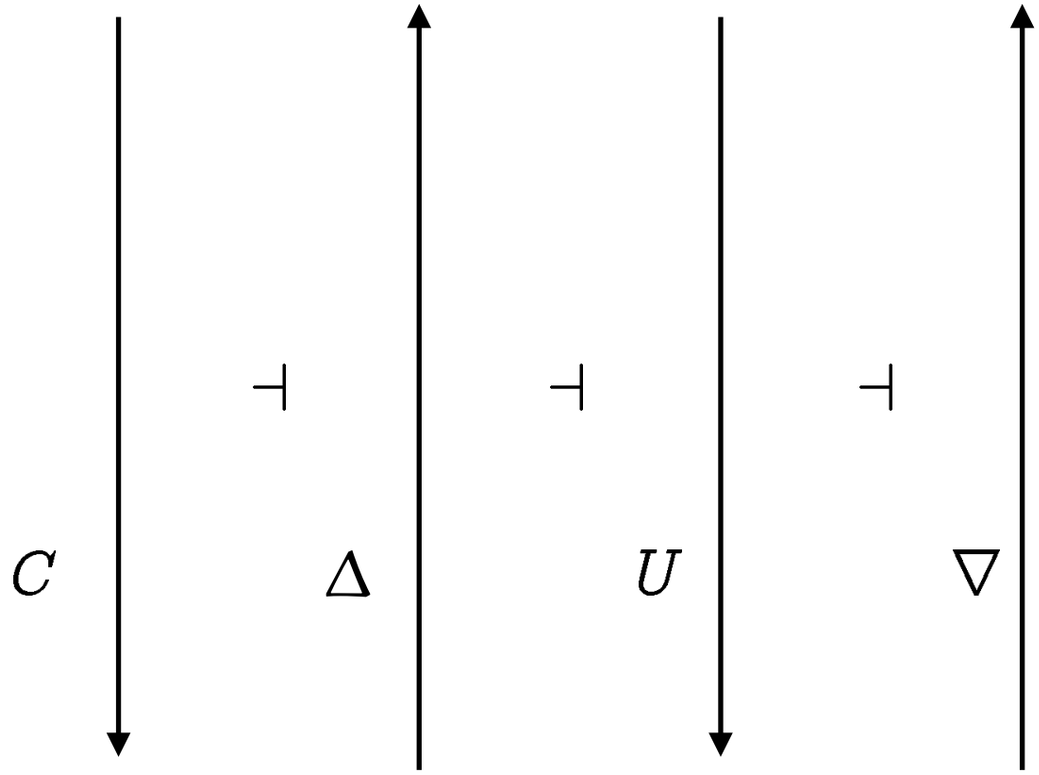
in the codiscrete space ∇(S) on S everything is "stuck together"
⇒ there is ≤ 1 connected component

Cohesion

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Spaces



Sets

CLAIM: This is all one needs to reason about information flow.

Axiom of
CONTRACTIBLE CODISCRETENESS:

$$\forall S.\ |C(\nabla S)| \leq 1$$

(For category theorists: the canonical $C(\nabla S) \xrightarrow{!} 1$ is a monic arrow.)

tt ff 𝔹	tt—ff ∇(𝔹)	{tt, ff} $C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together"
⇒ there is ≤ 1 connected component

CRIB

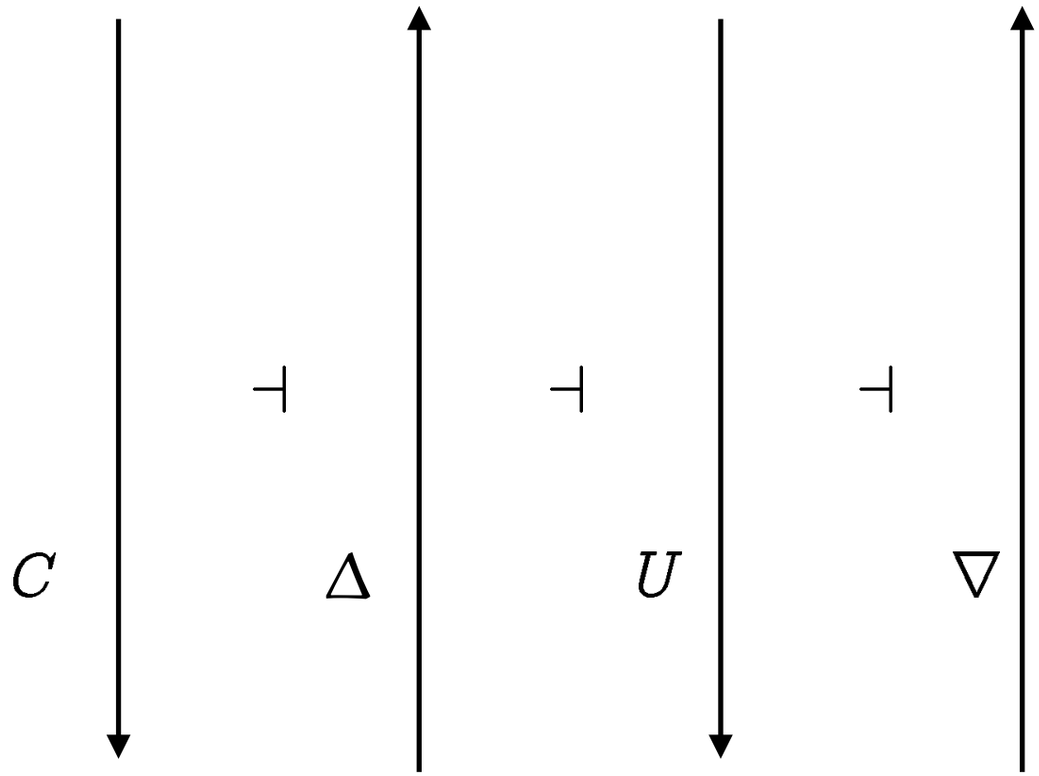
U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

Spaces



Sets

tt ff $\mathbb{B}$	tt—ff $\nabla(\mathbb{B})$	{tt, ff} $C(\nabla(\mathbb{B}))$

in the codiscrete space **∇(S)** on **S** everything is "stuck together"
⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X
(forget cohesion)
Δ(S) = discrete space on S
(min. cohesion)
∇(S) = codiscrete space on S
(max. cohesion)
C(X) = connected components of X

Cohesion

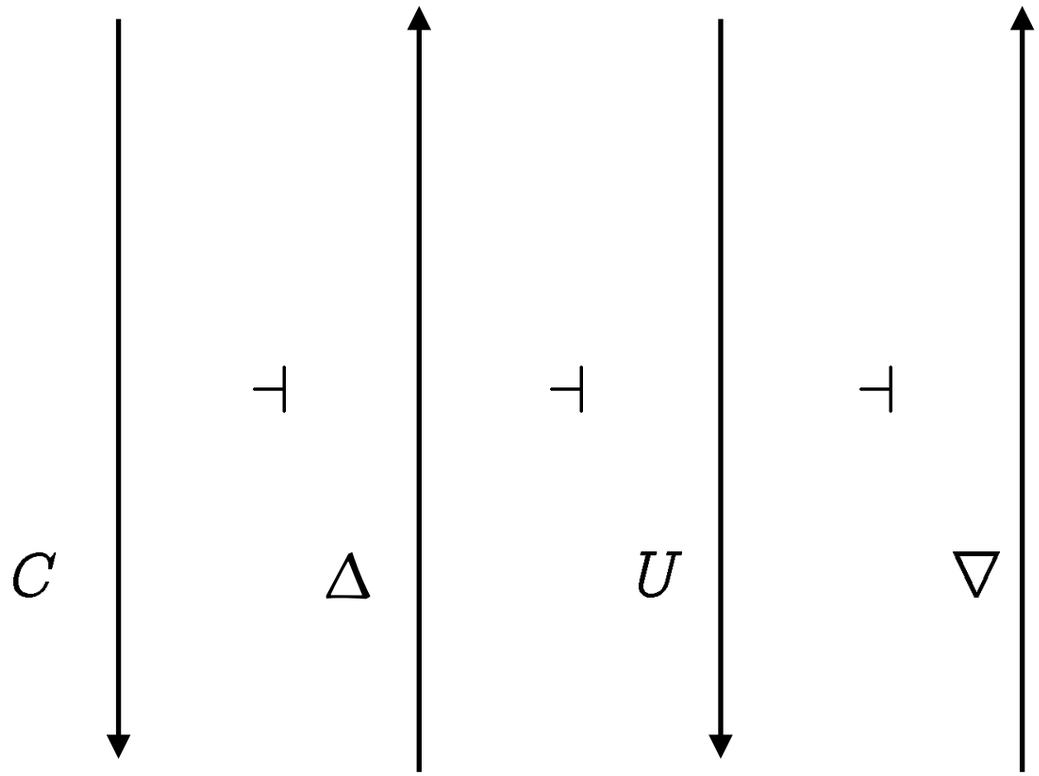
CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(U X)$

Spaces



Sets

tt ff $\mathbb{B}$	tt—ff $\nabla(\mathbb{B})$	{tt, ff} $C(\nabla(\mathbb{B}))$

in the codiscrete space **∇(S)** on **S** everything is "stuck together"
⇒ there is ≤ 1 connected component

Cohesion

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$C \dashv \Delta \dashv U \dashv \nabla$

Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space **∇(S)** on **S** everything is "stuck together"

⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$C \dashv \Delta \dashv U \dashv \nabla$

Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

Proof: every such f is by def. a continuous function

$$f : \nabla(UX) \to \Delta S$$

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together" ⇒ there is ≤ 1 connected component

Cohesion

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$$\frac{X \to \Delta(S)}{C(X) \to S}$$

$$C \dashv \Delta \dashv U \dashv \nabla$$

Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

Proof: every such f is by def. a continuous function $f : \nabla(UX) \to \Delta S$

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together"

⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$$\frac{X \to \Delta(S)}{C(X) \to S}$$

$$C \dashv \Delta \dashv U \dashv \nabla$$

Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

Proof: every such f is by def. a continuous function

$$f : \nabla(UX) \to \Delta S$$

which is just a set function

$$f : C(\nabla(UX)) \to S$$

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together"

⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$$\frac{X \to \Delta(S)}{C(X) \to S}$$

$$C \dashv \Delta \dashv U \dashv \nabla$$

Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

Proof: every such f is by def. a continuous function

$$f : \nabla(UX) \to \Delta S$$

which is just a set function

$$f : C(\nabla(UX)) \to S$$

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together"

⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

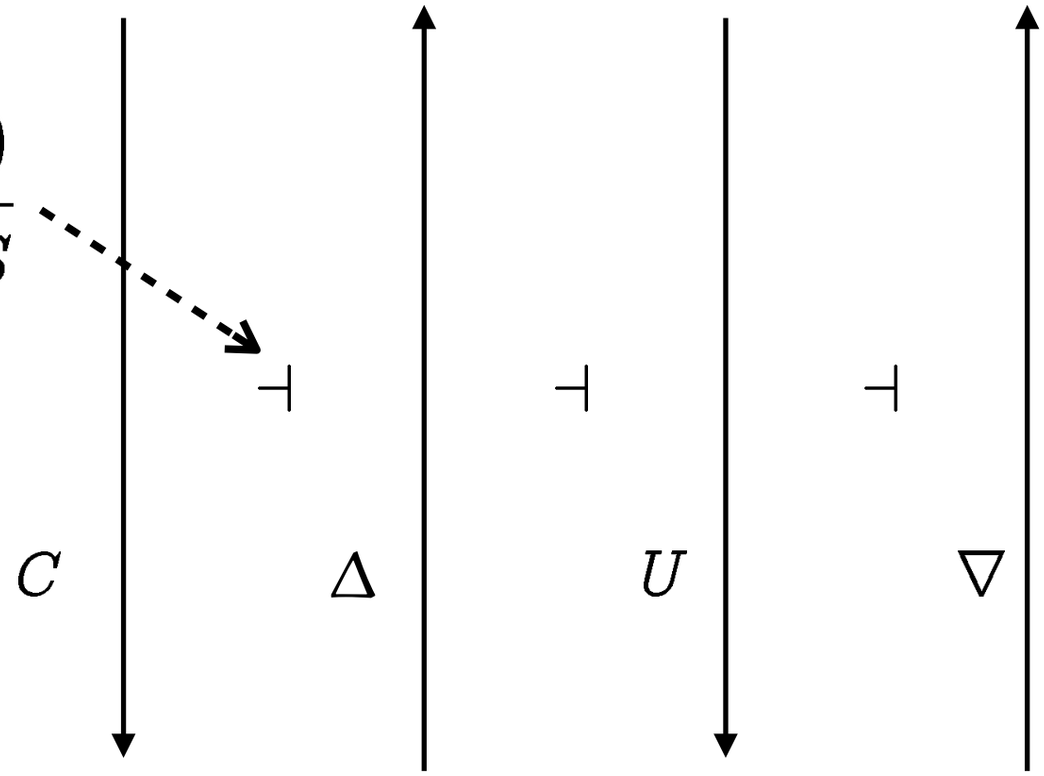
$$\forall S.\ |C(\nabla S)| \leq 1$$

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

Spaces

$$\frac{X \to \Delta(S)}{C(X) \to S}$$



Sets

Theorem: every $f : \blacklozenge X \to \Delta S$ continuous is a point of S (maybe)

Proof: every such f is by def. a continuous function

$$f : \nabla(UX) \to \Delta S$$

which is just a set function

$$f : C(\underbrace{\nabla(UX)}) \to S$$

a set of ≤ 1 element!

tt ff	tt——ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

in the codiscrete space ∇(S) on S everything is "stuck together"

⇒ there is ≤ 1 connected component

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

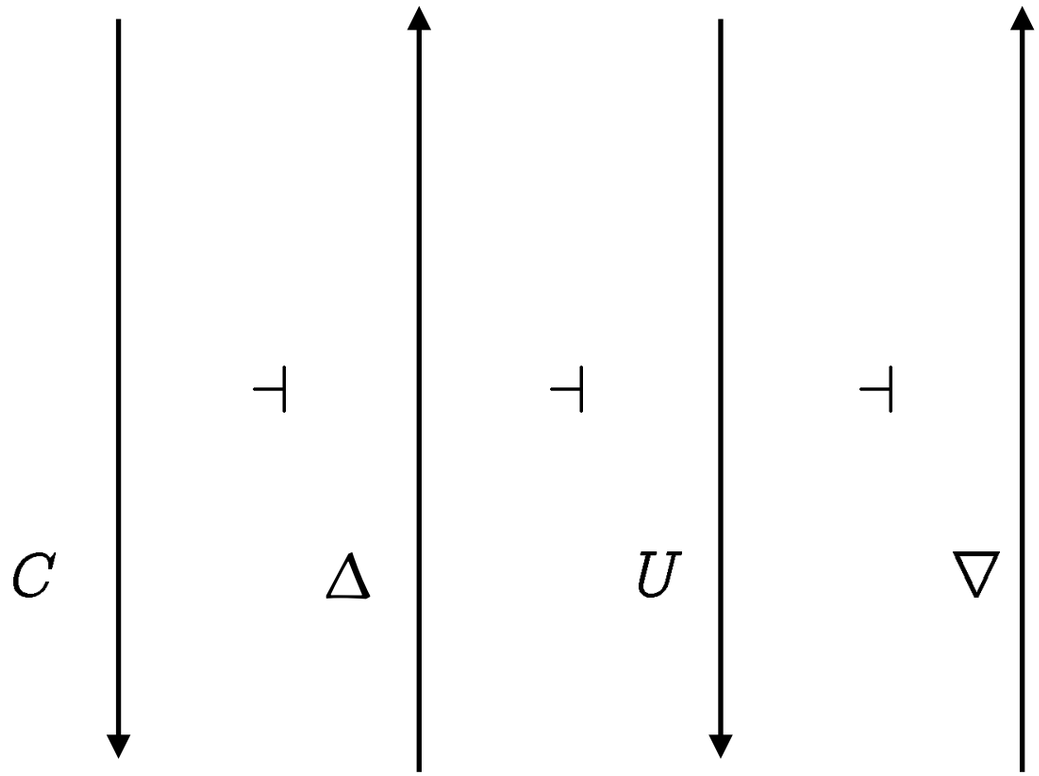
C(X) = connected components of X

Cohesion

CLAIM: This is all one needs to reason about information flow.

$$\forall S.\ |C(\nabla S)| \leq 1$$

Spaces



Sets

"X redacted"

Define: $\blacklozenge X = \nabla(UX)$

$\square X = \Delta(UX)$

$\int X = \Delta(CX)$

"X declassified"

"X as viewed by a low security user"

$\int \dashv \square \dashv \blacklozenge$

tt ff	tt—ff	{tt, ff}
$\mathbb{B}$	$\nabla(\mathbb{B})$	$C(\nabla(\mathbb{B}))$

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Classified sets

Set of classifications/labels: $\ell \in \mathcal{L}$

must be **reflexive**

Classified set: $X = (|X|, (R_\ell \subseteq |X| \times |X|)_{\ell \in \mathcal{L}})$

Cont. function: $f : X \to Y$ **s.t.** $\forall \ell.\ a R_\ell b \Rightarrow f(a) R_\ell f(b)$

"f is continuous when it maps inputs indistinguishable at $\ell \in \mathcal{L}$ to outputs indistinguishable at $\ell \in \mathcal{L}$ "

$$\begin{array}{ccccc} X & & (S, (\{(s,s) \mid s \in S\})_{\ell \in \mathcal{L}}) & & X & & (S, (S \times S)_{\ell \in \mathcal{L}}) \\ \downarrow C & \dashv & \downarrow \Delta & \dashv & \downarrow U & \dashv & \downarrow \nabla \\ \text{(equivalence classes)} & & S & & |X| & & S \end{array}$$

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Classified sets

Set of classifications/labels: $\ell \in \mathcal{L}$

must be **reflexive**

Classified set: $X = (|X|, (R_\ell \subseteq |X| \times |X|)_{\ell \in \mathcal{L}})$

Cont. function: $f : X \to Y$ **s.t.** $\forall \ell.\ aR_\ell b \Rightarrow f(a)R_\ell f(b)$

"f is continuous when it maps inputs indistinguishable at $\ell \in \mathcal{L}$ to outputs indistinguishable at $\ell \in \mathcal{L}$ "

$$
\begin{array}{ccccccc}
X & & (S, (\{(s,s) \mid s \in S\})_{\ell \in \mathcal{L}}) & & X & & (S, (S \times S)_{\ell \in \mathcal{L}}) \\
\Big\downarrow C & \dashv & \Big\downarrow \Delta & \dashv & \Big\downarrow U & \dashv & \Big\downarrow \nabla \\
\text{(equivalence classes)} & & S & & |X| & & S
\end{array}
$$

Theorem: the category of classified sets is **cartesian closed** and **cohesive over Sets**, and it satisfies **contractible codiscreteness.**

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Classified sets

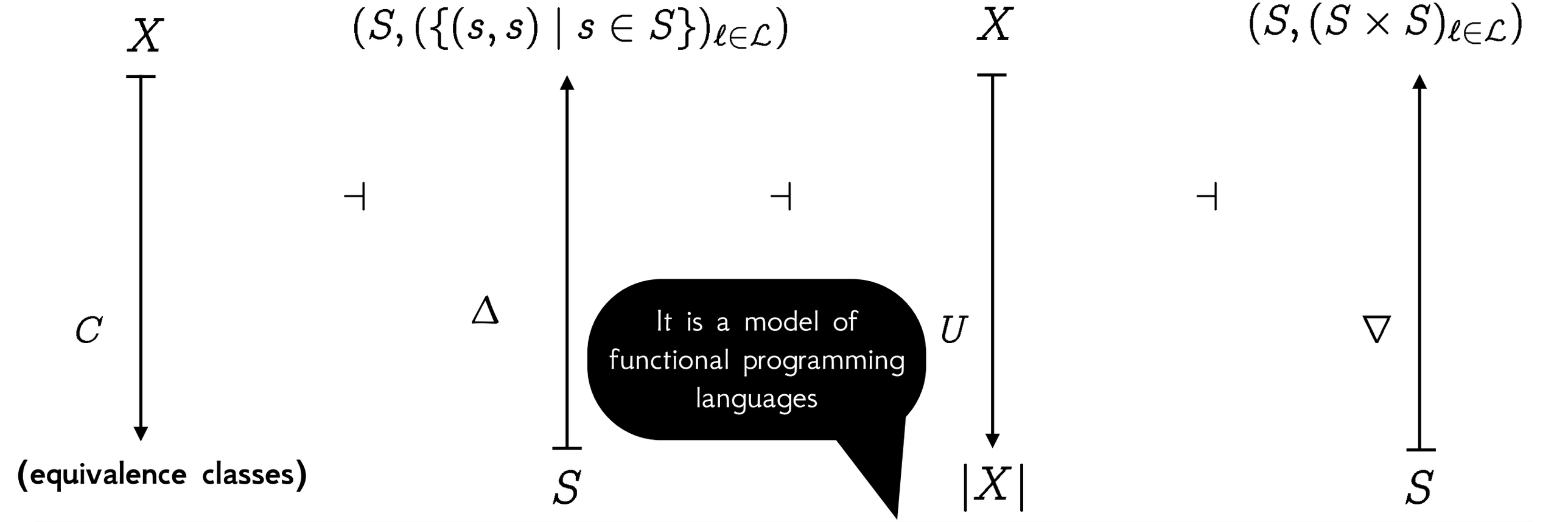
Set of classifications/labels: $\ell \in \mathcal{L}$

must be **reflexive**

Classified set: $X = (|X|, (R_\ell \subseteq |X| \times |X|)_{\ell \in \mathcal{L}})$

Cont. function: $f : X \to Y$ **s.t.** $\forall \ell.\ a R_\ell b \Rightarrow f(a) R_\ell f(b)$

"f is continuous when it maps inputs indistinguishable at $\ell \in \mathcal{L}$ to outputs indistinguishable at $\ell \in \mathcal{L}$ "



Theorem: the category of classified sets is **cartesian closed** and **cohesive over Sets**, and it satisfies **contractible codiscreteness.**

CRIB

U(X) = points of space X (forget cohesion)

Δ(S) = discrete space on S (min. cohesion)

∇(S) = codiscrete space on S (max. cohesion)

C(X) = connected components of X

Classified sets

Set of classifications/labels: $\ell \in \mathcal{L}$

must be **reflexive**

Classified set: $X = (|X|, (R_\ell \subseteq |X| \times |X|)_{\ell \in \mathcal{L}})$

Cont. function: $f : X \to Y$ **s.t.** $\forall \ell.\ a R_\ell b \Rightarrow f(a) R_\ell f(b)$

"f is continuous when it maps inputs indistinguishable at $\ell \in \mathcal{L}$ to outputs indistinguishable at $\ell \in \mathcal{L}$ "

$$X \qquad (S, (\{(s,s) \mid s \in S\})_{\ell \in \mathcal{L}}) \qquad X \qquad (S, (S \times S)_{\ell \in \mathcal{L}})$$

$$C \dashv \Delta \dashv U \dashv \nabla$$

(equivalence classes)

S $\qquad$ $|X|$ $\qquad$ S

It is a model of functional programming languages

It is a model of information flow

Theorem: the category of classified sets is **cartesian closed** and **cohesive over Sets**, and it satisfies **contractible codiscreteness.**

Cohesion and non-interference

✤ Recall what we were trying to prove:

If $x : \blacklozenge A \vdash E : \mathsf{Bool}$ and $\vdash M, N : \blacklozenge A$ then
$E[M/x]$ and $E[N/x]$ compute the same boolean value.

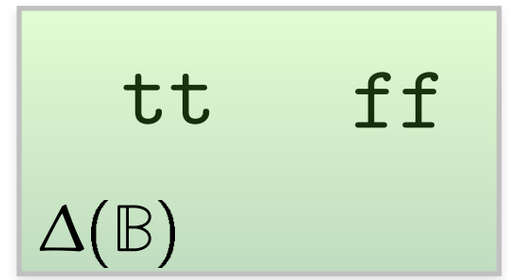
✤ There is a way to map every term to a continuous function between classified sets — a **categorical semantics**:

$$x : \blacklozenge A \vdash E : \mathsf{Bool} \qquad \longmapsto \qquad [\![E]\!] : \blacklozenge [\![A]\!] \to \Delta\mathbb{B}$$

✤ By the Theorem, this corresponds to an element of $\mathbb{B}$
So it is essentially a constant function.

$\blacklozenge X = \nabla(UX)$

✤ If only this could tell us something about the language...



Cohesion and non-interference

✤ Recall what we were trying to prove:

If $x : \blacklozenge A \vdash E : \mathsf{Bool}$ and $\vdash M, N : \blacklozenge A$ then
$E[M/x]$ and $E[N/x]$ compute the same boolean value.

✤ There is a way to map every term to a continuous function between classified sets — a **categorical semantics**:

$$x : \blacklozenge A \vdash E : \mathsf{Bool} \qquad \longmapsto \qquad [\![E]\!] : \blacklozenge [\![A]\!] \to \Delta\mathbb{B}$$

✤ By the Theorem, this corresponds to an element of $\mathbb{B}$
So it is essentially a constant function.

✤ If only this could tell us something about the language…

ADEQUACY, a.k.a completeness at base types (automatically holds when the language has no recursion)

Cohesion and non-interference

✤ Recall what we were trying to prove:

If $x : \blacklozenge A \vdash E : \mathsf{Bool}$ and $\vdash M, N : \blacklozenge A$ then
$E[M/x]$ and $E[N/x]$ compute the same boolean value.

✤ There is a way to map every term to a continuous function between classified sets — a **categorical semantics**:

$$x : \blacklozenge A \vdash E : \mathsf{Bool} \qquad \longmapsto \qquad [\![E]\!] : \blacklozenge [\![A]\!] \to \Delta\mathbb{B}$$

✤ By the Theorem, this corresponds to an element of $\mathbb{B}$
So it is essentially a constant function.

✤ If only this could tell us something about the language…

ADEQUACY, a.k.a completeness at base types (automatically holds when the language has no recursion)

Cohesion and non-interference

- This approach can be leveraged to prove noninterference for multiple type theories for secure information flow:
  - Moggi's monadic metalanguage [Moggi 1991]
  - Davies-Pfenning calculus (S4 modality) [D&Pf 2001]
  - Dependency Core Calculus [Abadi et al. 1999]

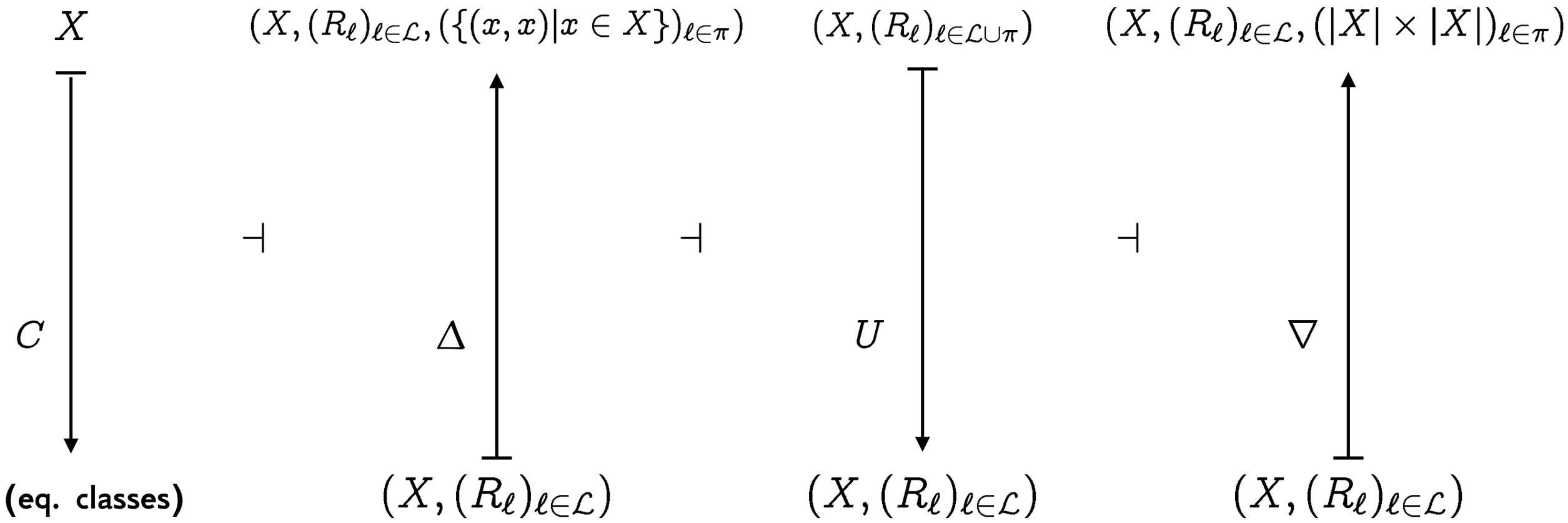
  } (A little bit of care is required here w.r.t. adequacy)

  - Sealing Calculus [Shikuma & Igarashi 2008]

- The last two are **multi-modal** type theories.

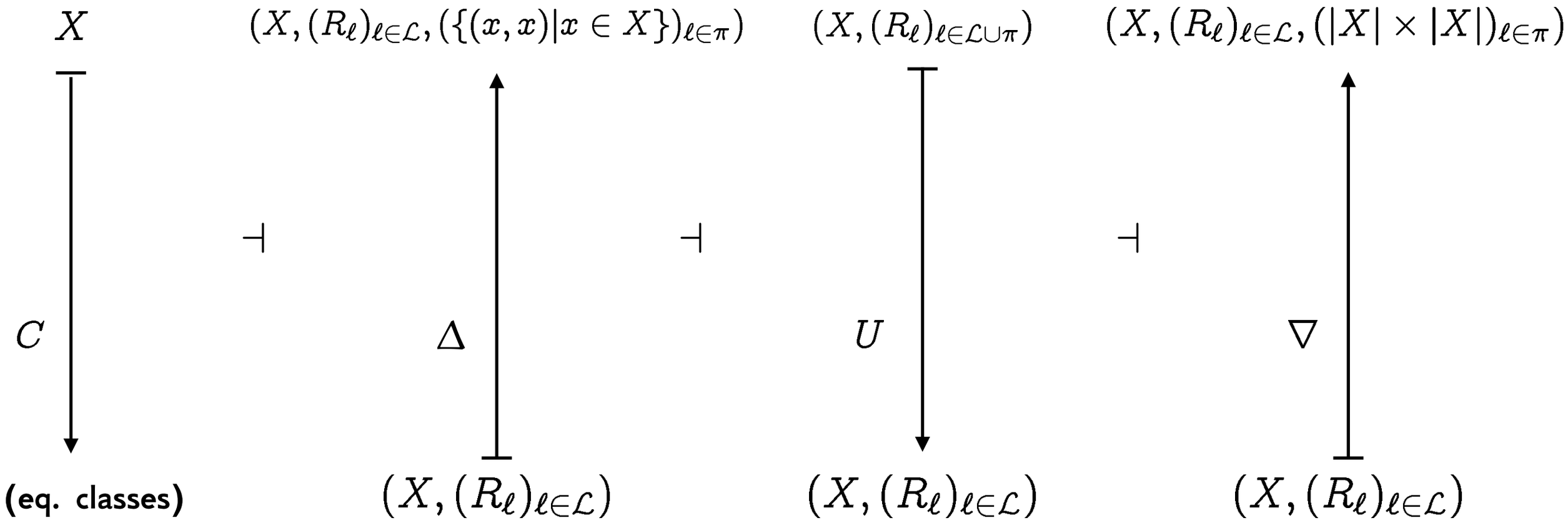
Cohesion and multi-modal type theories for information flow

Let π be a fresh set of labels, disjoint from $\mathcal{L}$.



Cohesion and multi-modal type theories for information flow

Let π be a fresh set of labels, disjoint from $\mathcal{L}$.

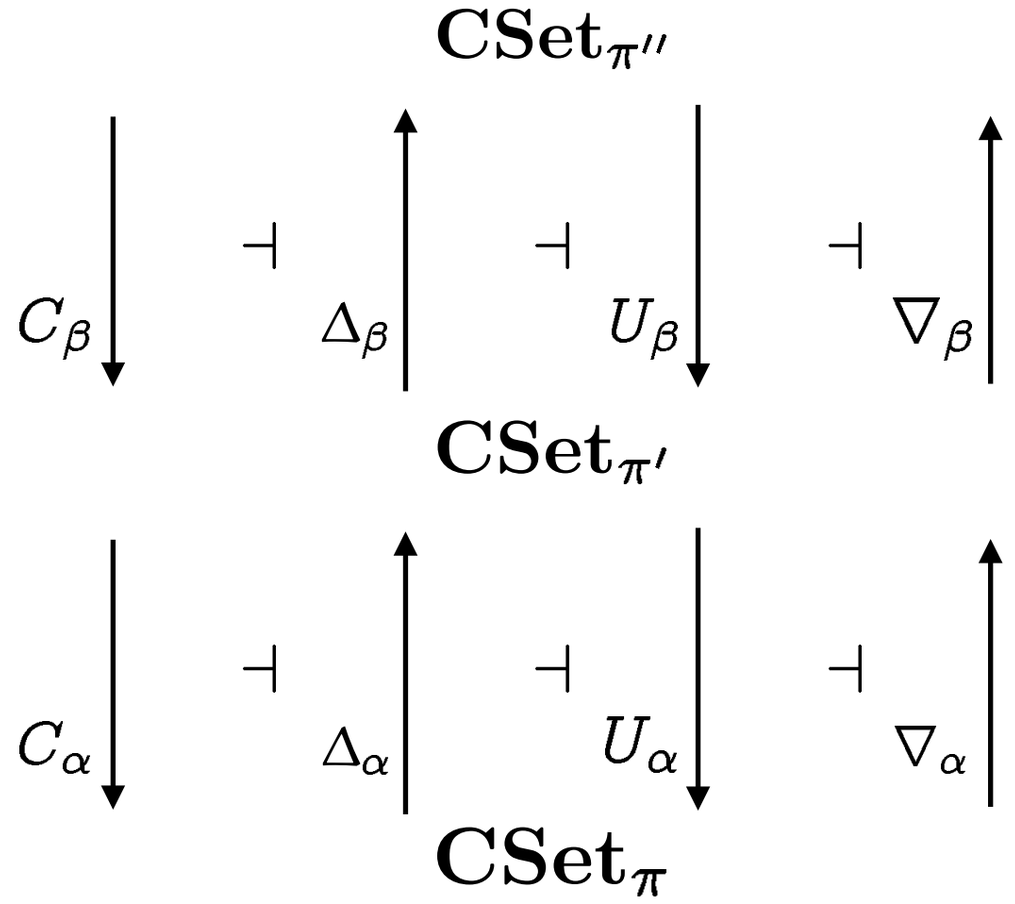


Theorem: the category of classified sets over $\mathcal{L}\cup\pi$ is **cohesive** over the category of classified sets over $\mathcal{L}$ and satisfies **contractible codiscreteness**.

Cohesion and multi-modal type theories for information flow

Writing $\mathbf{CSet}_\pi$ for the category of classified sets over $\pi \subseteq \mathcal{L}$

and $\begin{array}{l} \alpha : \pi \subseteq \pi' \\ \beta : \pi' \subseteq \pi'' \end{array}$ for the unique morphisms in $\mathcal{P}(\mathcal{L})$ we have the two cohesive situation on the right.



It's a functor

$$\mathcal{P}(\mathcal{L})^{\mathrm{op}} \longrightarrow \mathbf{Coh}$$

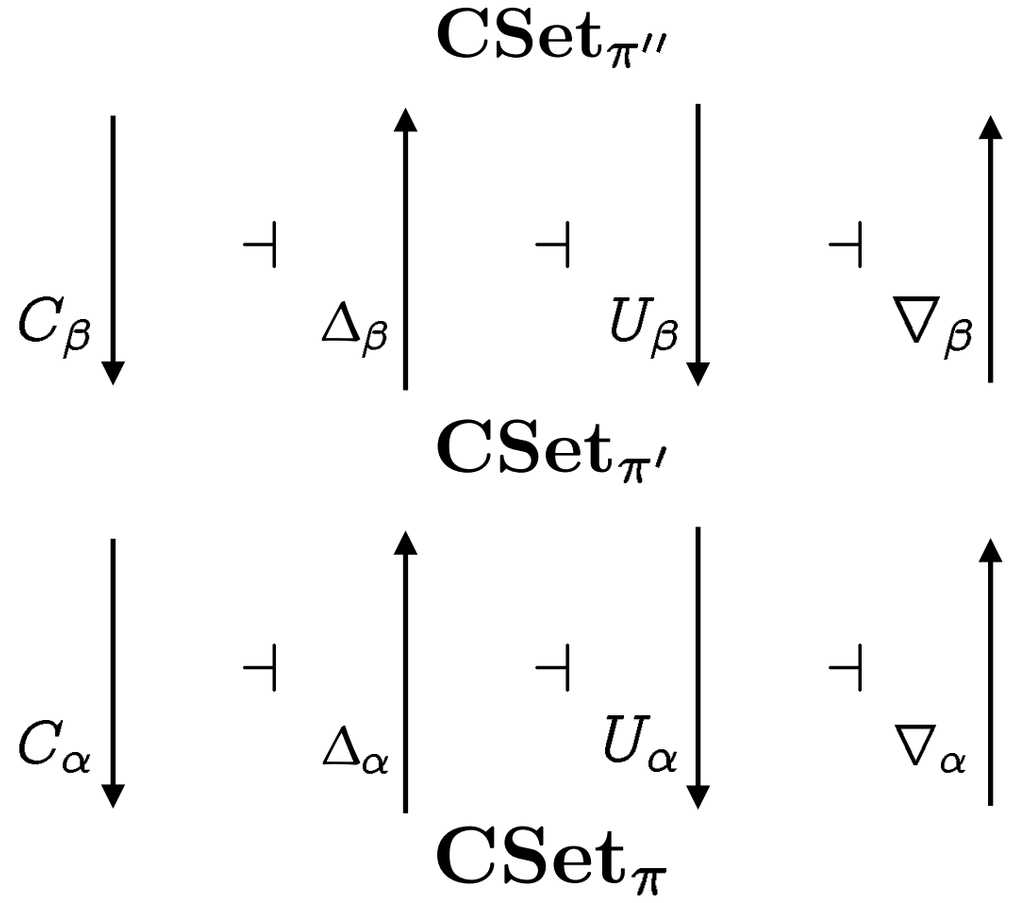
$\mathcal{L} \cup \pi$

$\mathcal{L}$

Cohesion and multi-modal type theories for information flow

Writing $\mathbf{CSet}_\pi$ for the category of classified sets over $\pi \subseteq \mathcal{L}$

and $\begin{array}{l} \alpha : \pi \subseteq \pi' \\ \beta : \pi' \subseteq \pi'' \end{array}$ for the unique morphisms in $\mathcal{P}(\mathcal{L})$ we have the two cohesive situation on the right.

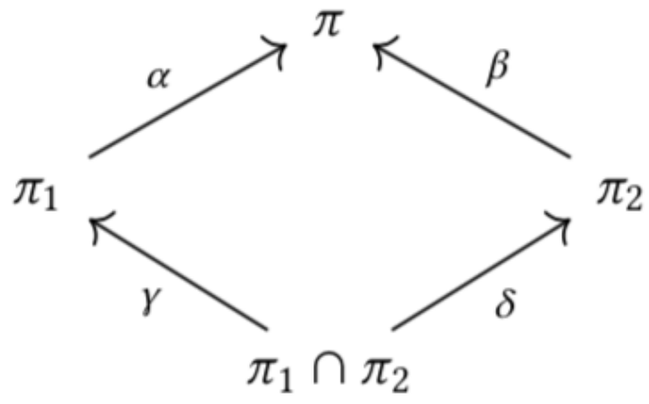


It's a functor

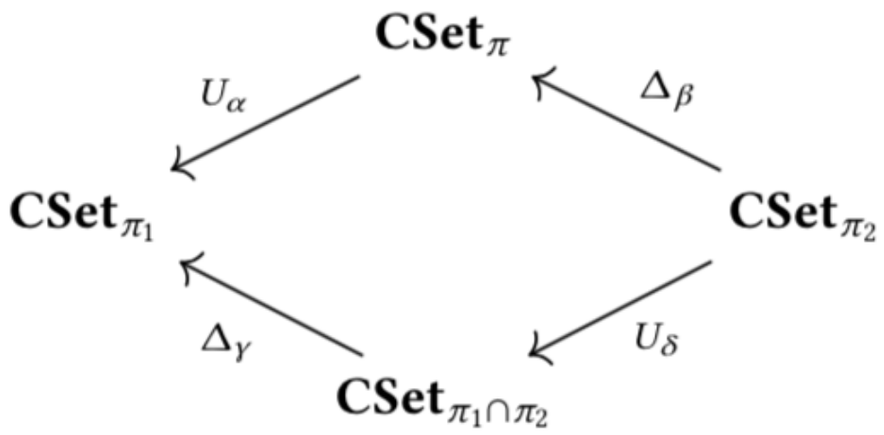
$$\mathcal{P}(\mathcal{L})^{\mathrm{op}} \longrightarrow \mathbf{Coh}$$

Theorem: the category of classified sets over $\mathcal{L} \cup \pi$ is **cohesive** over the category of classified sets over $\mathcal{L}$ and satisfies **contractible codiscreteness**.

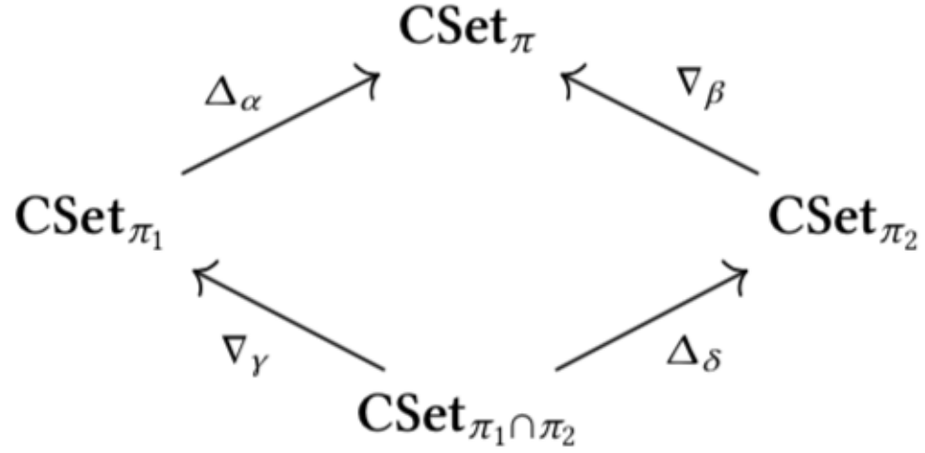
Three fundamental equations

✤ Given 

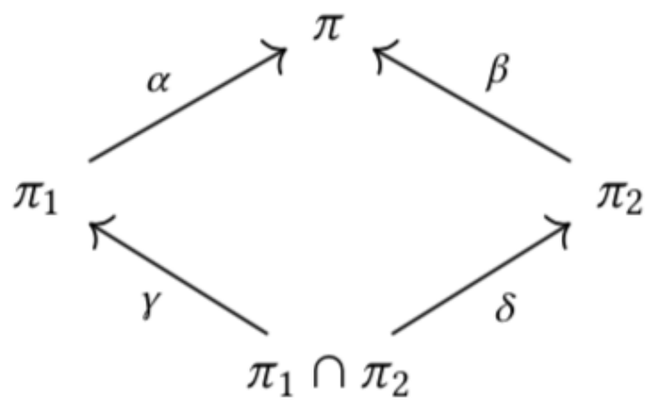
we want:

(1) 

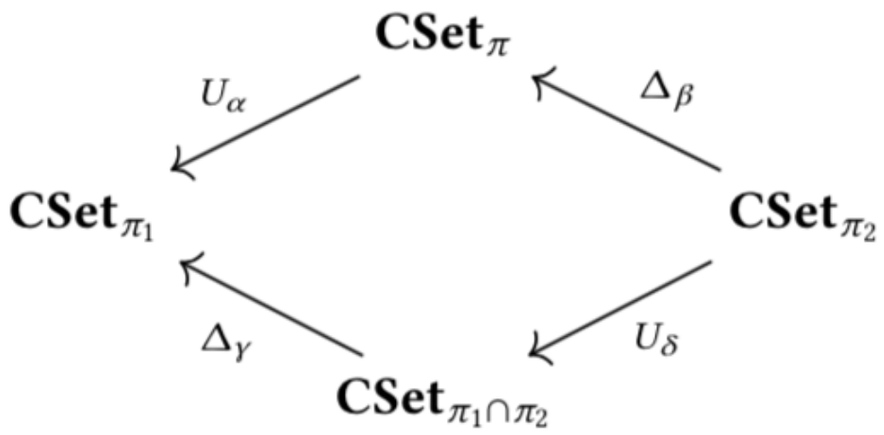
(2) same as (1) but for ∇

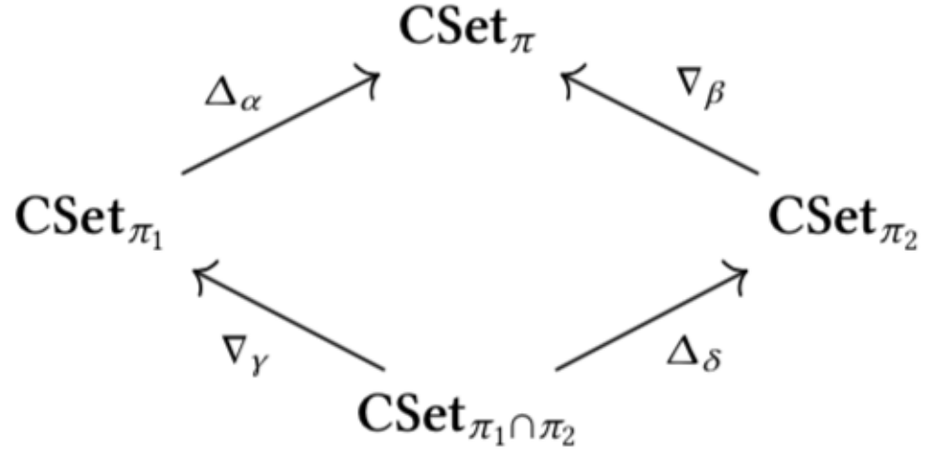
(3) 

Three fundamental equations

✤ Given 

we want:

(1) 

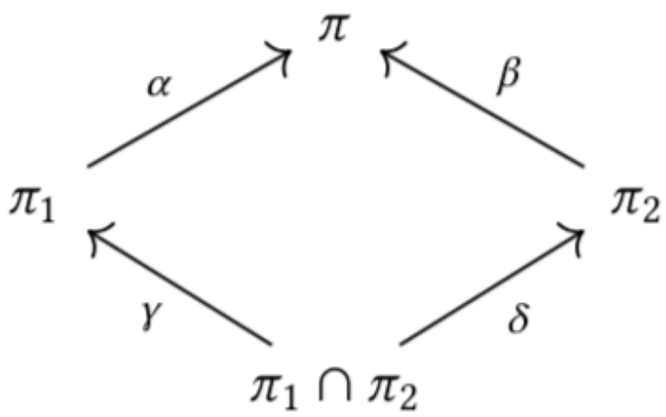
(3) 

(2) same as (1) but for ∇

> **Observation**: These suffice to prove all the laws I have needed so far.

The laws for $\int \dashv \Box \dashv \blacklozenge$

PROPOSITION 20. *Let*



be a pullback diagram. Then

(1) $\Box_\gamma U_\alpha = U_\alpha \Box_\beta$
(2) $\blacklozenge_\gamma U_\alpha = U_\alpha \blacklozenge_\beta$
(3) $\Box_{\alpha\circ\gamma} = \Box_\alpha \Box_\beta$
(4) $\blacklozenge_{\alpha\circ\gamma} = \blacklozenge_\alpha \blacklozenge_\beta$
(5) $\Box_\alpha \Box_\beta = \Box_\beta \Box_\alpha$
(6) $\blacklozenge_\alpha \blacklozenge_\beta = \blacklozenge_\beta \blacklozenge_\alpha$

PROPOSITION 21.

(1) *If* $\pi \cap \pi' = \emptyset$, *then* $\Box_\pi \Box_{\pi'} = \Box_{\pi\cup\pi'}$.
(2) *If* $\pi \cap \pi' = \emptyset$, *then* $\blacklozenge_\pi \blacklozenge_{\pi'} = \blacklozenge_{\pi\cup\pi'}$.
(3) $\Box_\pi \Box_{\pi'} = \Box_{\pi\cup\pi'}$
(4) $\blacklozenge_\pi \blacklozenge_{\pi'} = \blacklozenge_{\pi\cup\pi'}$
(5) *If* $\pi \subseteq \pi'$, *then* $\Box_{\pi'} \blacklozenge_\pi = \Box_{\pi'}$.
(6) *If* $\pi \subseteq \pi'$, *then* $\blacklozenge_{\pi'} \Box_\pi = \blacklozenge_{\pi'}$.
(7) *If* $\pi \cap \pi' = \emptyset$, *then* $\Box_\pi \blacklozenge_{\pi'} = \blacklozenge_{\pi'} \Box_\pi$.
(8) $\Box_\pi \blacklozenge_{\pi'} = \blacklozenge_{\pi'-\pi} \Box_\pi$.
(9) $\blacklozenge_\pi \Box_{\pi'} = \Box_{\pi'-\pi} \blacklozenge_\pi$.

Conclusions

- ✤ One of the most abstract/philosophical parts of categorical algebra, namely **axiomatic cohesion,** is a practical theory of information flow.

- ✤ It can be used to prove properties of LBIFC…

- ✤ **…** and, hopefully, it can inspire new languages for LBIFC.

- ✤ Despite the looks of it, **categorical algebra** as a way to reason about programming is still largely unexplored territory.

- ✤ **Multi-modal type theories** have intuitive categorical semantics.