Constructive Galois Connections

With applications to Abstracting Gradual Typing

Specification

Specification

```
succ : ℕ → ℕ
```

Specification

`succ : ℕ → ℕ`

"`succ(n)` flips the parity of `n`"

Specification

```
succ : ℕ → ℕ
```

“`succ(n)` flips the parity of `n`”

```
ℙ : Set
parity : ℕ → ℙ
flip   : ℙ → ℙ
```

Specification

```
succ : ℕ → ℕ
```

“`succ(n)` flips the parity of `n`”

```
ℙ : Set
parity : ℕ → ℙ
flip   : ℙ → ℙ

∀(n : ℕ),
parity(succ(n)) = flip(parity(n))
```

Verification

Verification

```
ℙ ≔ E | O

parity(0) ≔ E
parity(succ(n)) ≔ flip(parity(n))

flip(E) ≔ O ; flip(O) ≔ E
```

Verification

```
ℙ ≔ E | O

parity(0) ≔ E
parity(succ(n)) ≔ flip(parity(n))

flip(E) ≔ O ; flip(O) ≔ E

∀(n : ℕ),
parity(succ(n)) = flip(parity(n))
```

Verification

```
ℙ ≔ E | O

parity(0) ≔ E
parity(succ(n)) ≔ flip(parity(n))

flip(E) ≔ O ; flip(O) ≔ E

∀(n : ℕ),
parity(succ(n)) = flip(parity(n))
```

Proof is trivial by definition. ∎

Using Abstract Interpretation

Abstract Interpretation

Abstract Interpretation

$\gamma : \mathbb{P}^+ \rightarrow \wp(\mathbb{N})$

$\alpha : \wp(\mathbb{N}) \rightarrow \mathbb{P}^+$

Abstract Interpretation

$\gamma : \mathbb{P}^+ \rightarrow \wp(\mathbb{N})$

$\alpha : \wp(\mathbb{N}) \rightarrow \mathbb{P}^+$

$\mathbb{P}^+ \coloneqq E \mid 0 \mid \top \mid \bot$

Abstract Interpretation

$$\gamma : \mathbb{P}^+ \rightarrow \wp(\mathbb{N})$$
$$\alpha : \wp(\mathbb{N}) \rightarrow \mathbb{P}^+$$

$$\mathbb{P}^+ \coloneqq E \mid O \mid \top \mid \bot$$

$$\gamma(E) \coloneqq \{n \mid n \text{ is even}\}$$
$$\gamma(O) \coloneqq \{n \mid n \text{ is odd}\}$$
$$\gamma(\top) \coloneqq \{n \mid n \in \mathbb{N}\}$$
$$\gamma(\bot) \coloneqq \{\}$$

$$\alpha(N) \coloneqq \sqcup_{\llcorner} n \in N_{\lrcorner}\ \text{parity}^+(n)$$

AI Setup

AI Setup

```
gc-sound : ∀(N : ℘(ℕ)), N ⊆ γ(α(N))
gc-tight : ∀(p : ℙ⁺), α(γ(p)) ⊑ p
```

AI Setup

```
gc-sound : ∀(N : ℘(ℕ)), N ⊆ γ(α(N))
gc-tight : ∀(p : ℙ⁺), α(γ(p)) ⊑ p

γ(α({1,2})) = γ(⊤) = {n | n ∈ ℕ} ⊇ {1,2}
```

AI Setup

```
gc-sound : ∀(N : ℘(ℕ)), N ⊆ γ(α(N))
gc-tight : ∀(p : ℙ⁺), α(γ(p)) ⊑ p

γ(α({1,2})) = γ(⊤) = {n | n ∈ ℕ} ⊇ {1,2}

α(γ(E)) = α({n | n is even}) = E ⊑ E
```

AI Setup

$$\texttt{gc-sound} : \forall(N : \wp(\mathbb{N})),\ N \subseteq \gamma(\alpha(N))$$
$$\texttt{gc-tight} : \forall(p : \mathbb{P}^{+}),\ \alpha(\gamma(p)) \sqsubseteq p$$

$$\gamma(\alpha(\{1,2\})) = \gamma(\top) = \{n \mid n \in \mathbb{N}\} \supseteq \{1,2\}$$

$$\alpha(\gamma(E)) = \alpha(\{n \mid n \text{ is even}\}) = E \sqsubseteq E$$

alternatively: $\alpha(N) \sqsubseteq p$ *iff* $N \sqsubseteq \gamma(p)$

AI Specification (sound)

AI Specification (sound)

```
↑succ : ℘(ℕ) → ℘(ℕ)
↑succ(N) ≔ {succ(n) | n ∈ N}
```

AI Specification (sound)

```
↑succ : ℘(ℕ) → ℘(ℕ)
↑succ(N) ≔ {succ(n) | n ∈ N}

α ∘ ↑succ ∘ γ ⊑ flip    (αγ)
↑succ ∘ γ ⊆ γ ∘ flip    (γγ)
α ∘ ↑succ ⊑ flip ∘ α    (αα)
↑succ ⊆ γ ∘ flip ∘ α    (γα)
```

AI Specification (sound)

```
↑succ : ℘(ℕ) → ℘(ℕ)
↑succ(N) ≔ {succ(n) | n ∈ N}

α ∘ ↑succ ∘ γ ⊑ flip     (αγ)
↑succ ∘ γ ⊆ γ ∘ flip     (γγ)
α ∘ ↑succ ⊑ flip ∘ α     (αα)
↑succ ⊆ γ ∘ flip ∘ α     (γα)
```

All statements are equivalent.

AI Verification (sound)

AI Verification (sound)

```
(αγ): ∀(p : ℙ⁺), α(↑succ(γ(p))) ⊑ flip(p)
```

AI Verification (sound)

```
(αγ): ∀(p : ℙ⁺), α(↑succ(γ(p))) ⊑ flip(p)

Proof by case analysis on p:
  Case E:
    α(↑succ(γ(E)))
    = α(↑succ({n | n is even}))
    = α({succ(n) | n is even})
    = ⊔⌞n | n is even⌟parity(succ(n))
    = ⊔⌞n | n is even⌟flip(parity(n))
    = ⊔⌞n | n is even⌟flip(E)
    = ⊔⌞n | n is even⌟O
    = O
    = flip(E)
  ...
```

AI Verification (sound)

AI Verification (sound)

```
(αα) : ∀(N : ℘(ℕ)), α(↑succ(N)) ⊑ flip(α(N))
```

AI Verification (sound)

```
(αα) : ∀(N : ℘(ℕ)), α(↑succ(N)) ⊑ flip(α(N))

Proof by case analysis:
  Case ∃ n ∈ N st n is even
    ∧ ¬∃ n ∈ N st n is odd:
      α(↑succ(N))
      = α({succ(n) | n ∈ N})
      = ⊔⸤n | n ∈ N⸥parity(succ(n))
      = ⊔⸤n | n ∈ N⸥flip(parity(n))
      = ⊔⸤n | n ∈ N⸥O
      = O
      = flip(E)
      = flip(⊔⸤n | n ∈ N⸥parity(n))
      = flip(α(N))
  ...
```

AI Specification (complete)

AI Specification (complete)

```
↑succ : ℘(ℕ) → ℘(ℕ)
↑succ(N) ≔ {succ(n) | n ∈ N}

α ∘ ↑succ ∘ γ ⊑ flip       (αγ)
↑succ ∘ γ ⊆ γ ∘ flip       (γγ)
α ∘ ↑succ ⊑ flip ∘ α       (αα)
↑succ ⊆ γ ∘ flip ∘ α       (γα)
```

All statements are equivalent.

AI Specification (complete)

$$\uparrow\mathrm{succ} : \wp(\mathbb{N}) \to \wp(\mathbb{N})$$
$$\uparrow\mathrm{succ}(N) \coloneqq \{\mathrm{succ}(n) \mid n \in N\}$$

$$\alpha \circ \uparrow\mathrm{succ} \circ \gamma = \mathrm{flip} \qquad (\alpha\gamma)$$
$$\uparrow\mathrm{succ} \circ \gamma = \gamma \circ \mathrm{flip} \qquad (\gamma\gamma)$$
$$\alpha \circ \uparrow\mathrm{succ} = \mathrm{flip} \circ \alpha \qquad (\alpha\alpha)$$
$$\uparrow\mathrm{succ} = \gamma \circ \mathrm{flip} \circ \alpha \qquad (\gamma\alpha)$$

All statements are *not* equivalent.

AI Specification (complete)

$$\uparrow succ : \wp(\mathbb{N}) \to \wp(\mathbb{N})$$
$$\uparrow succ(N) \coloneqq \{succ(n) \mid n \in N\}$$

$\alpha \circ \uparrow succ \circ \gamma = flip$	$(\alpha\gamma)$	✓
$\uparrow succ \circ \gamma = \gamma \circ flip$	$(\gamma\gamma)$	✓
$\alpha \circ \uparrow succ = flip \circ \alpha$	$(\alpha\alpha)$	✓
$\uparrow succ = \gamma \circ flip \circ \alpha$	$(\gamma\alpha)$	✗

All statements are *not* equivalent.

Issues with Abstract Interpretation

Unwanted Complexity

Unwanted Complexity

$\alpha \circ \uparrow\mathtt{succ} \sqsubseteq \mathtt{flip} \circ \alpha$

vs

$\eta \circ \mathtt{succ} \sqsubseteq \mathtt{flip} \circ \eta$ (where $\eta = \mathtt{parity}^{+}$)

Unwanted Complexity

`α ∘ ↑succ ⊑ flip ∘ α`

vs

`η ∘ succ ⊑ flip ∘ η` (where `η = parity⁺`)

Are these equivalent? Yes.

Mechanization Problems

Mechanization Problems

```
α ∘ ↑succ ∘ γ ⊑ flip      (αγ)
↑succ ∘ γ ⊆ γ ∘ flip      (γγ)
α ∘ ↑succ ⊑ flip ∘ α      (αα)
↑succ ⊆ γ ∘ flip ∘ α      (γα)
```

Mechanization Problems

```
α ∘ ↑succ ∘ γ ⊑ flip      (αγ)
↑succ ∘ γ ⊆ γ ∘ flip      (γγ)
α ∘ ↑succ ⊑ flip ∘ α      (αα)
↑succ ⊆ γ ∘ flip ∘ α      (γα)

α : ℘(ℕ) → ℙ⁺
α(N) ≔ ⊔⌞n ∈ N⌟ parity⁺(n)
```

Mechanization Problems

```
α ∘ ↑succ ∘ γ ⊑ flip    (αγ)
↑succ ∘ γ ⊆ γ ∘ flip    (γγ)
α ∘ ↑succ ⊑ flip ∘ α    (αα)
↑succ ⊆ γ ∘ flip ∘ α    (γα)

α : ℘(ℕ) → ℙ⁺
α(N) ≔ ⊔⌞n ∈ N⌟ parity⁺(n)
```

α is nonconstructive and poses problems in mechanization (particularly extraction)

Mechanization Problems

$$\alpha \circ {\uparrow}succ \circ \gamma \sqsubseteq flip \qquad (\alpha\gamma)$$

$$\uparrow succ \circ \gamma \subseteq \gamma \circ flip \qquad (\gamma\gamma)$$

$$\alpha \circ {\uparrow}succ \sqsubseteq flip \circ \alpha \qquad (\alpha\alpha)$$

$$\uparrow succ \subseteq \gamma \circ flip \circ \alpha \qquad (\gamma\alpha)$$

$$\alpha : \wp(\mathbb{N}) \rightarrow \mathbb{P}^{+}$$

$$\alpha(N) \coloneqq \bigsqcup_{\llcorner n \in N \lrcorner} parity^{+}(n)$$

α is nonconstructive and poses problems in mechanization (particularly extraction)

State of the art approaches to mechanizing abstract interpreters use $\gamma\gamma$ exclusively and do not formalize Galois connections in their full generality

Constructive Galois Connections

CGCs Religion

CGCs Religion

`℘(A) ≔ A → prop`

constructive encoding for powersets (Coq/Agda)

CGCs Religion

$\wp$`(A) ≔ A → prop`
constructive encoding for powersets (Coq/Agda)

$\wp$ is a monad, and forms a Kleisli category

`A → B` vs `A →` $\wp$`(B)`

“returns a value” vs "returns a specification”

CGCs Religion

`℘(A) ≔ A → prop`
constructive encoding for powersets (Coq/Agda)

℘ is a monad, and forms a Kleisli category
`A → B` vs `A → ℘(B)`
“returns a value” vs "returns a specification”

℘ is a "specification effect" or "nonconstructive effect"

CGCs Religion

$\wp$`(A) ≔ A → prop`
constructive encoding for powersets (Coq/Agda)

$\wp$ is a monad, and forms a Kleisli category
`A → B` vs `A →` $\wp$`(B)`
“returns a value” vs "returns a specification”

$\wp$ is a "specification effect" or "nonconstructive effect"

Monadic functions in `A →` $\wp$`(B)` which “have no effect” can be extracted and executed

CGCs Religion

$\wp$`(A) ≔ A → prop`
constructive encoding for powersets (Coq/Agda)

$\wp$ is a monad, and forms a Kleisli category
`A → B` vs `A →` $\wp$`(B)`
“returns a value” vs "returns a specification”

$\wp$ is a "specification effect" or "nonconstructive effect"

Monadic functions in `A →` $\wp$`(B)` which “have no effect” can be extracted and executed

Rediscover Galois connections in this new category

Powerset Model

Powerset Model

```
℘(A) ≔ A ↘ prop
```

Powerset Model

`℘(A) ≔ A ↘ prop`

`x ∈ φ ≔ φ(x)` [for `φ : ℘(A)` and `x : A`]

Powerset Model

$\wp(\mathtt{A}) \coloneqq \mathtt{A} \searrow \mathtt{prop}$

$\mathtt{x} \in \varphi \coloneqq \varphi(\mathtt{x})$ [for $\varphi : \wp(\mathtt{A})$ and $\mathtt{x} : \mathtt{A}$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall(\mathtt{x} : \mathtt{A}),\ \varphi_1(\mathtt{x}) \to \varphi_2(\mathtt{x}))$ [for $\varphi_1\ \varphi_2 : \wp(\mathtt{A})$]

Powerset Model

$\wp(A) \coloneqq A \searrow \mathtt{prop}$

$x \in \varphi \coloneqq \varphi(x)$ [for $\varphi : \wp(A)$ and $x : A$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall (x : A), \varphi_1(x) \to \varphi_2(x))$ [for $\varphi_1\ \varphi_2 : \wp(A)$]

$\mathtt{return} : A \nearrow \wp(A)$
$\mathtt{return}(x)(y) \coloneqq y \sqsubseteq x$

Powerset Model

$\wp(A) \coloneqq A \searrow \mathtt{prop}$

$x \in \varphi \coloneqq \varphi(x)$ [for $\varphi : \wp(A)$ and $x : A$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall(x : A), \varphi_1(x) \rightarrow \varphi_2(x))$ [for $\varphi_1\ \varphi_2 : \wp(A)$]

$\mathtt{return} : A \nearrow \wp(A)$
$\mathtt{return}(x)(y) \coloneqq y \sqsubseteq x$

$\mathtt{return}(x) \approx \{y \mid y \sqsubseteq x\} \approx \{x\}$

Powerset Model

$\wp(A) \coloneqq A \searrow \mathtt{prop}$

$x \in \varphi \coloneqq \varphi(x)$ [for $\varphi : \wp(A)$ and $x : A$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall (x : A), \varphi_1(x) \rightarrow \varphi_2(x))$ [for $\varphi_1\ \varphi_2 : \wp(A)$]

$\mathtt{return} : A \nearrow \wp(A)$
$\mathtt{return}(x)(y) \coloneqq y \sqsubseteq x$

$\mathtt{return}(x) \approx \{y \mid y \sqsubseteq x\} \approx \{x\}$

$\_^{*} : (A \nearrow \wp(B)) \nearrow (\wp(A) \nearrow \wp(B))$
$f^{*}(X)(y) \coloneqq \exists\ x \in X\ \mathit{st}\ y \in f(x)$

Powerset Model

$\wp(A) \coloneqq A \searrow \mathtt{prop}$

$x \in \varphi \coloneqq \varphi(x)$ [for $\varphi : \wp(A)$ and $x : A$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall (x : A), \varphi_1(x) \rightarrow \varphi_2(x))$ [for $\varphi_1\ \varphi_2 : \wp(A)$]

$\mathtt{return} : A \nearrow \wp(A)$
$\mathtt{return}(x)(y) \coloneqq y \sqsubseteq x$

$\mathtt{return}(x) \approx \{y \mid y \sqsubseteq x\} \approx \{x\}$

$\_^* : (A \nearrow \wp(B)) \nearrow (\wp(A) \nearrow \wp(B))$
$f^*(X)(y) \coloneqq \exists\ x \in X\ \mathit{st}\ y \in f(x)$

$f^*(\{x_1 \ldots x_n\}) \approx \{y \mid y \in f(x_1) \vee \ldots \vee y \in f(x_n)\}$
$f^*(\{x_1 \ldots x_n\}) \approx \bigcup_{i} f(x_i)$

Powerset Model

$\wp(A) \coloneqq A \searrow \mathrm{prop}$

$x \in \varphi \coloneqq \varphi(x)$ [for $\varphi : \wp(A)$ and $x : A$]

$\varphi_1 \subseteq \varphi_2 \coloneqq (\forall(x : A),\ \varphi_1(x) \rightarrow \varphi_2(x))$ [for $\varphi_1\ \varphi_2 : \wp(A)$]

$\mathrm{return} : A \nearrow \wp(A)$
$\mathrm{return}(x)(y) \coloneqq y \sqsubseteq x$

$\mathrm{return}(x) \approx \{y \mid y \sqsubseteq x\} \approx \{x\}$

$\_^* : (A \nearrow \wp(B)) \nearrow (\wp(A) \nearrow \wp(B))$
$f^*(X)(y) \coloneqq \exists\ x \in X\ \mathit{st}\ y \in f(x)$

$f^*(\{x_1 \ldots x_n\}) \approx \{y \mid y \in f(x_1) \vee \ldots \vee y \in f(x_n)\}$
$f^*(\{x_1 \ldots x_n\}) \approx \bigcup_{i} f(x_i)$

$\_\diamond\_ : (B \nearrow \wp(C)) \nearrow (A \nearrow \wp(B)) \nearrow (A \nearrow \wp(C))$
$(g \diamond f)(x) \coloneqq g^*(f(x))$

$$A \rightleftarrows B$$

$A \rightleftarrows B$

Classical

$\alpha : A \nearrow B$
$\gamma : B \nearrow A$

sound: $id^A \sqsubseteq \gamma \circ \alpha$
tight: $\alpha \circ \gamma \sqsubseteq id^B$

sound: $\forall (x : A),\ x \sqsubseteq \gamma(\alpha(x))$
tight: $\forall (z : B),\ \alpha(\gamma(z)) \sqsubseteq z$

A ⇄ B

Classical

```
α : A ↗ B
γ : B ↗ A

sound: idᴬ ⊑ γ ∘ α
tight: α ∘ γ ⊑ idᴮ

sound: ∀(x : A), x ⊑ γ(α(x))
tight: ∀(z : B), α(γ(z)) ⊑ z
```

Kleisli

```
α : A ↗ ℘(B)
γ : B ↗ ℘(A)

sound: returnᴬ ⊑ γ ⟐ α
tight: α ⟐ γ ⊑ returnᴮ

sound: ∀(x : A), {x} ⊆ γ*(α(x))
tight: ∀(z : B), α*(γ(z)) ⊆ {z}
```

$$A \rightleftarrows B$$

Classical	**Kleisli**
$\alpha : A \nearrow B$	$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow A$	$\gamma : B \nearrow \wp(A)$
sound: $id^A \sqsubseteq \gamma \circ \alpha$	sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \circ \gamma \sqsubseteq id^B$	tight: $\alpha \diamond \gamma \sqsubseteq return^B$
sound: $\forall (x : A),\ x \sqsubseteq \gamma(\alpha(x))$	sound: $\forall (x : A),\ \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall (z : B),\ \alpha(\gamma(z)) \sqsubseteq z$	tight: $\forall (z : B),\ \alpha^*(\gamma(z)) \subseteq \{z\}$

For Classical, A is typically instantiated to $\wp(A)$

$A \rightleftarrows B$

Classical

$\alpha : \wp(A) \nearrow B$
$\gamma : B \nearrow \wp(A)$

sound: $id^A \subseteq \gamma \circ \alpha$
tight: $\alpha \circ \gamma \sqsubseteq id^B$

sound: $\forall (X : \wp(A)),\ X \subseteq \gamma(\alpha(X))$
tight: $\forall (z : B),\ \alpha(\gamma(z)) \sqsubseteq z$

Kleisli

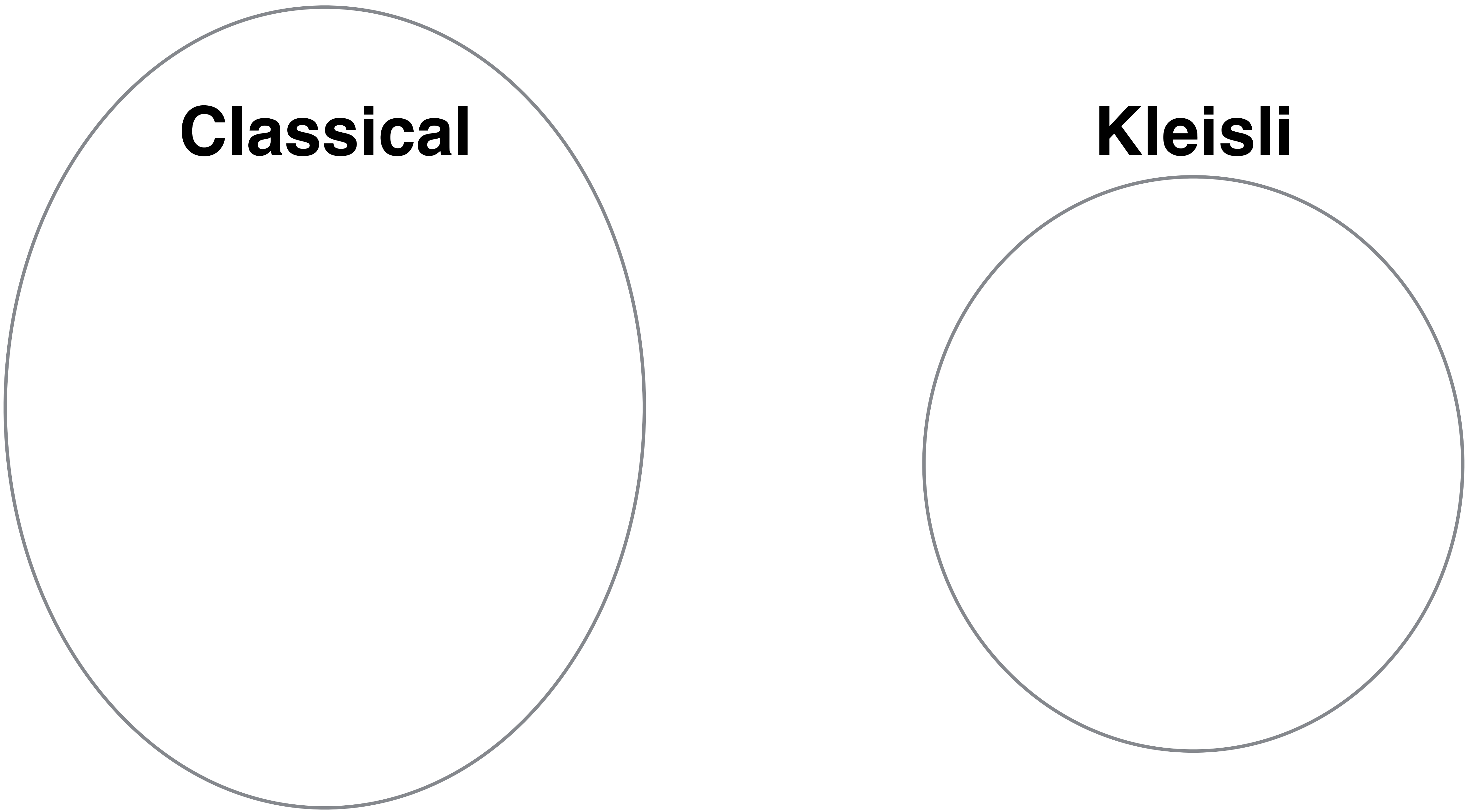
$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow \wp(A)$

sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^B$

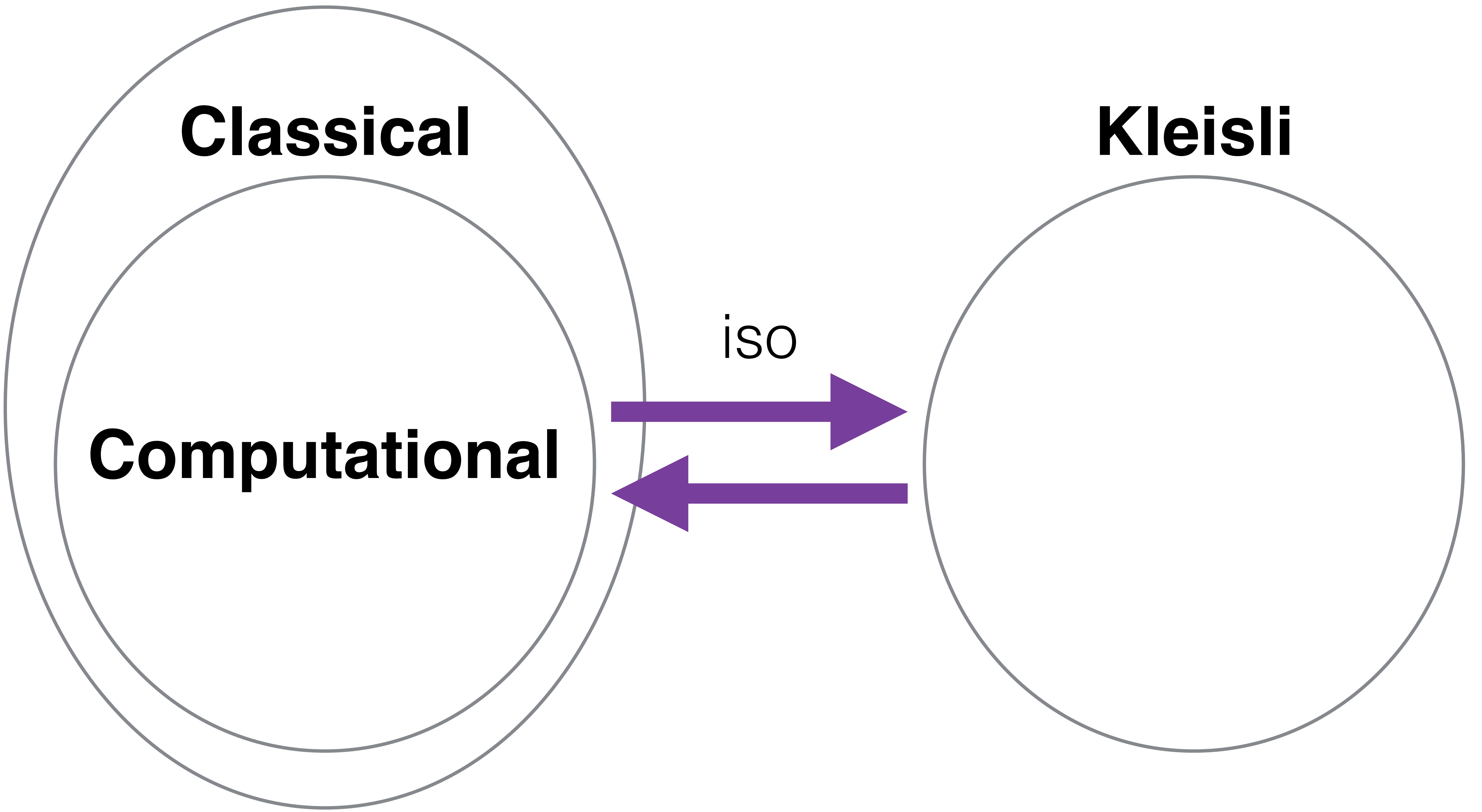
sound: $\forall (x : A),\ \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall (z : B),\ \alpha^*(\gamma(z)) \subseteq \{z\}$

For Classical, A is typically instantiated to $\wp(A)$

Relationships



Relationships



A ⇄ B

Kleisli

$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow \wp(A)$

sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^B$

sound: $\forall(x : A), \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall(z : B), \alpha^*(\gamma(z)) \subseteq \{z\}$

$$A \rightleftarrows B$$

"α has no monadic effect"

Kleisli

```
α : A ↗ ℘(B)
γ : B ↗ ℘(A)

sound: returnᴬ ⊑ γ ⟐ α
tight: α ⟐ γ ⊑ returnᴮ

sound: ∀(x : A), {x} ⊆ γ*(α(x))
tight: ∀(z : B), α*(γ(z)) ⊆ {z}
```

$A \rightleftarrows B$

"α has no monadic effect"

$$\exists\ \eta : A \nearrow B\ st$$
$$\alpha = \lambda x.\ \{\eta(x)\}$$

Kleisli

$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow \wp(A)$

sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^B$

sound: $\forall(x : A),\ \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall(z : B),\ \alpha^*(\gamma(z)) \subseteq \{z\}$

$A \rightleftarrows B$

"α has no monadic effect"

$$\exists\ \eta : A \nearrow B \ \textit{st}$$
$$\alpha = \lambda x.\ \{\eta(x)\}$$

Kleisli

$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow \wp(A)$

sound: $\text{return}^A \sqsubseteq \gamma \circledcirc \alpha$
tight: $\alpha \circledcirc \gamma \sqsubseteq \text{return}^B$

sound: $\forall (x : A),\ \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall (z : B),\ \alpha^*(\gamma(z)) \subseteq \{z\}$

sound: $\forall (x : A),\ \exists\ (z : B)\ \textit{st}\ z \in \alpha(x) \wedge x \in \gamma(z)$

A ⇄ B

Kleisli

$\alpha : A \nearrow \wp(B)$
$\gamma : B \nearrow \wp(A)$

sound: $return^{A} \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^{B}$

sound: $\forall(x : A), \{x\} \subseteq \gamma^{*}(\alpha(x))$
tight: $\forall(z : B), \alpha^{*}(\gamma(z)) \subseteq \{z\}$

Constructive

$\eta : A \nearrow B$
$\gamma : B \nearrow \wp(A)$

sound: $return^{A} \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^{B}$

sound: $\forall(x : A), \{x\} \subseteq \gamma^{*}(\alpha(x))$
tight: $\forall(z : B), \alpha^{*}(\gamma(z)) \subseteq \{z\}$

For Constructive GC, $\alpha \coloneqq \lambda x.\ \{\eta(x)\}$

A ⇄ B

Constructive

```
η : A ↗ B
γ : B ↗ ℘(A)

sound: returnᴬ ⊑ γ ⟐ α
tight: α ⟐ γ ⊑ returnᴮ

sound: ∀(x : A), {x} ⊆ γ*(α(x))
tight: ∀(z : B), α*(γ(z)) ⊆ {z}
```

For Constructive GC, α ≔ λx. {η(x)}

$A \rightleftarrows B$

sound:
$x \in \gamma(\eta(x))$

Constructive

$\eta : A \nearrow B$
$\gamma : B \nearrow \wp(A)$

sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^B$

sound: $\forall(x : A), \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall(z : B), \alpha^*(\gamma(z)) \subseteq \{z\}$

For Constructive GC, $\alpha \coloneqq \lambda x.\ \{\eta(x)\}$

$A \rightleftarrows B$

sound:
$x \in \gamma(\eta(x))$

tight:
$x \in \gamma(z) \Rightarrow \eta(x) \sqsubseteq z$

Constructive

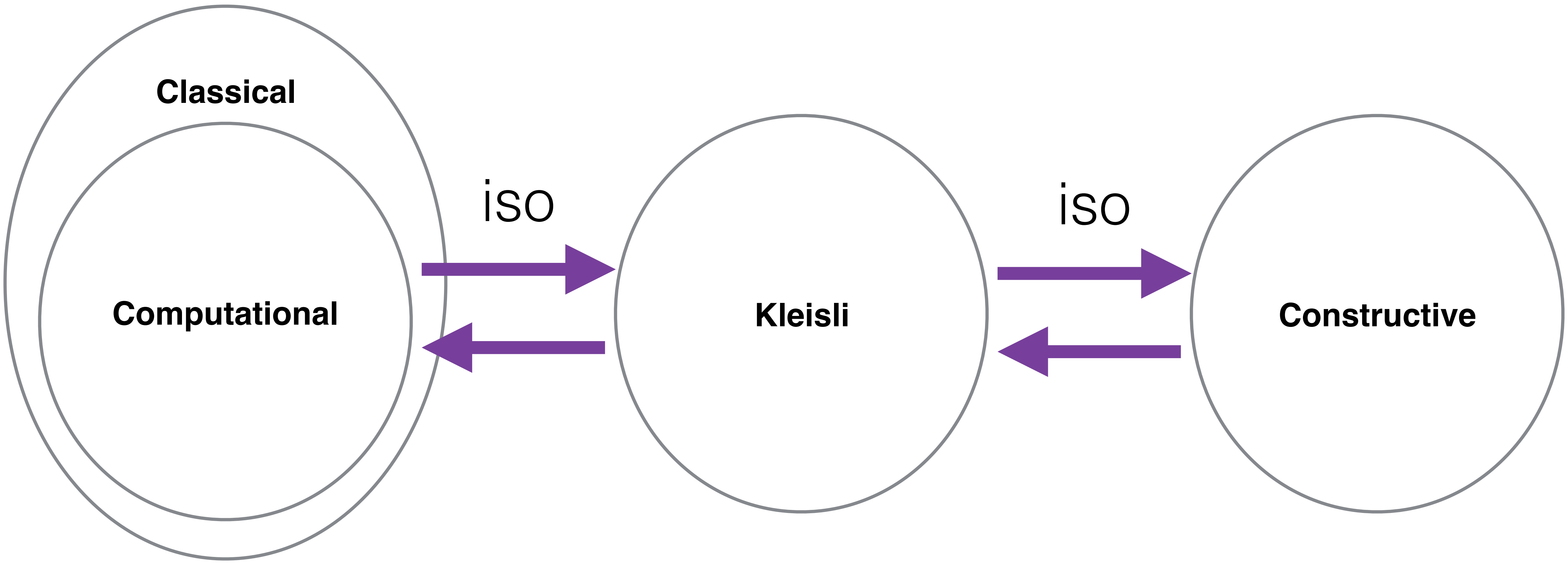
$\eta : A \nearrow B$
$\gamma : B \nearrow \wp(A)$

sound: $return^A \sqsubseteq \gamma \diamond \alpha$
tight: $\alpha \diamond \gamma \sqsubseteq return^B$

sound: $\forall(x : A), \{x\} \subseteq \gamma^*(\alpha(x))$
tight: $\forall(z : B), \alpha^*(\gamma(z)) \subseteq \{z\}$

For Constructive GC, $\alpha \coloneqq \lambda x.\ \{\eta(x)\}$

Relationships



Summary

Summary

Define $\gamma : B \nearrow \wp(A)$ just as before

Summary

Define $\gamma : B \nearrow \wp(A)$ just as before

Define $\eta : A \nearrow B$ instead of $\alpha : \wp(A) \nearrow B$

Summary

Define $\gamma : B \nearrow \wp(A)$ just as before

Define $\eta : A \nearrow B$ instead of $\alpha : \wp(A) \nearrow B$

Lift proofs of soundness for free (through isomorphisms)

Summary

Define $\gamma : B \nearrow \wp(A)$ just as before

Define $\eta : A \nearrow B$ instead of $\alpha : \wp(A) \nearrow B$

Lift proofs of soundness for free (through isomorphisms)

Interact with classical GCs (through isomorphisms)

Summary

Define γ `: B ↗` $\wp$`(A)` just as before

Define η `: A ↗ B` instead of α `:` $\wp$`(A) ↗ B`

Lift proofs of soundness for free (through isomorphisms)

Interact with classical GCs (through isomorphisms)

You can calculate with this approach towards interpreters which are both sound *and* computable by construction.

Summary

Define γ `: B ↗ ℘(A)` just as before

Define η `: A ↗ B` instead of α `: ℘(A) ↗ B`

Lift proofs of soundness for free (through isomorphisms)

Interact with classical GCs (through isomorphisms)

You can calculate with this approach towards interpreters which are both sound *and* computable by construction.

η and γ are constructive, so mechanizing general framework and extraction is no problem.

AGT with CGCs

AGT In A Nutshell

```
τ ∈ type  ⩴ 𝔹 | τ→τ
e ∈  exp  ⩴ b | if(e){e}{e}
            | x | λ(x).e | e(e)
```

AGT In A Nutshell

```
τ ∈ type  ⩴ 𝔹 | τ→τ
e ∈  exp  ⩴ b | if(e){e}{e}
           | x | λ(x).e | e(e)
```

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\text{if}(e_1)\{e_2\}\{e_3\} : \tau}\ [\mathbb{B}\text{-E}]$$

AGT In A Nutshell

$$\tau \in \text{type} ::= \mathbb{B} \mid \tau \rightarrow \tau$$

$$e \in \text{exp} ::= b \mid \underline{\text{if}}(e)\{e\}\{e\} \mid x \mid \underline{\lambda}(x).e \mid e(e)$$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau}\ [\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \rightarrow \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\ [\rightarrow\text{-E}]$$

AGT In A Nutshell

$$\tau \in \mathtt{type}^{\sharp} ::= \mathbb{B} \mid \tau \rightarrow \tau \mid ?$$

$$\begin{aligned} e \in \mathtt{exp} ::= {} & b \mid \underline{\mathtt{if}}(e)\{e\}\{e\} \\ \mid {} & x \mid \underline{\lambda}(x).e \mid e(e) \end{aligned}$$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\mathtt{if}}(e_1)\{e_2\}\{e_3\} : \tau}\ [\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \rightarrow \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\ [\rightarrow\text{-E}]$$

AGT In A Nutshell

$$\tau \in \text{type}^\sharp ::= \mathbb{B} \mid \tau \rightarrow \tau \mid ?$$

$$e \in \text{exp}^\sharp ::= b \mid \underline{\text{if}}(e)\{e\}\{e\}$$

$$\mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau$$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau}\ [\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \rightarrow \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\ [\rightarrow\text{-E}]$$

AGT In A Nutshell

$$\tau \in \text{type}^\sharp ::= \mathbb{B} \mid \tau \to \tau \mid ?$$

$$\begin{aligned} e \in \text{exp}^\sharp ::= {} & b \mid \underline{\text{if}}(e)\{e\}\{e\} \\ & \mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau \end{aligned}$$

$$\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \mathbb{B} \\ e_2 : \tau_2 & \\ e_3 : \tau_3 & \end{array}}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau_2 \vee \tau_3}\,[\mathbb{B}\text{-E}]$$

$$\frac{\begin{array}{l} e_1 : \tau_1 \to \tau_2 \\ e_2 : \tau_1 \end{array}}{e_1(e_2) : \tau_2}\,[\to\text{-E}]$$

AGT In A Nutshell

$$
\begin{array}{llll}
\tau & \in & \mathsf{type}^\sharp & ::= \mathbb{B} \mid \tau \rightarrow \tau \mid ? \\
e & \in & \mathsf{exp}^\sharp & ::= b \mid \underline{\mathsf{if}}(e)\{e\}\{e\} \\
 & & & \quad \mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau
\end{array}
$$

$$
\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \mathbb{B} \\ e_2 : \tau_2 & \\ e_3 : \tau_3 & \end{array}}{\underline{\mathsf{if}}(e_1)\{e_2\}\{e_3\} : \tau_2 \vee \tau_3} [\mathbb{B}\text{-E}]
$$

$$
\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \\ e_2 : \tau_2 & \tau_2 \sim \tau_{11} \end{array}}{e_1(e_2) : \tau_{21}} [\rightarrow\text{-E}]
$$

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\ [\rightarrow\text{-E}]$$

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \to \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\,[\to\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \to \tau_{21}$"

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\,[\rightarrow\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathbin{⸬} \tau_2) : \tau_2}\,[⸬\text{-I}]$$

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}} [\rightarrow\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathbin{⸽} \tau_2) : \tau_2} [⸽\text{-I}]$$

"I claim **e** *might* have type τ_2"

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\;[\rightarrow\text{-E}]$$

"It's plausible that **e₁** has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e :: \tau_2) : \tau_2}\;[::\text{-I}]$$

"I claim **e** *might* have type τ_2"

$$\frac{}{? \sim \tau} \qquad \frac{}{\tau \sim ?}$$

Plausibility

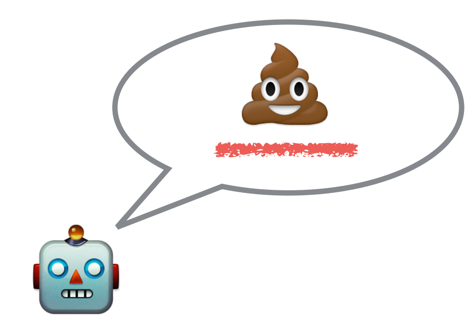
$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\;[\rightarrow\text{-E}]$$

"It's plausible that **e_1** has some arrow type **$\tau_{11} \rightarrow \tau_{21}$**"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathbin{⦂} \tau_2) : \tau_2}\;[⦂\text{-I}]$$

"I claim **e** *might* have type **τ_2**"

$$\frac{}{? \sim \tau} \qquad \frac{}{\tau \sim ?}$$



<Gradual Rob>

"If you say so…"

Consistent Equality

```
gτ~gτ
```

Consistent Equality

$g\tau \sim g\tau$

"meaning" of a gradual type

$$[\![\_]\!] : \text{type}^\sharp \rightarrow \wp(\text{type})$$
$$[\![\mathbb{B}]\!] \coloneqq \{\mathbb{B}\}$$
$$[\![g\tau_1 \rightarrow g\tau_2]\!] \coloneqq \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in [\![g\tau_1]\!] \wedge \tau_2 \in [\![g\tau_2]\!]\}$$
$$[\![?]\!] \coloneqq \{\tau \mid \tau \in \text{type}\}$$

Consistent Equality

$g\tau \sim g\tau$

"meaning" of a gradual type

$$
\begin{aligned}
&[\![\_]\!] : \mathtt{type}^\sharp \to \wp(\mathtt{type}) \\
&[\![\mathbb{B}]\!] \coloneqq \{\mathbb{B}\} \\
&[\![g\tau_1 \to g\tau_2]\!] \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in [\![g\tau_1]\!] \wedge \tau_2 \in [\![g\tau_2]\!]\} \\
&[\![?]\!] \coloneqq \{\tau \mid \tau \in \mathtt{type}\}
\end{aligned}
$$

consistent equalities are "plausibilities"

$$
\frac{\tau_1 \in [\![g\tau_1]\!] \qquad \tau_2 \in [\![g\tau_2]\!] \qquad \tau_1 = \tau_2}{g\tau_1 \sim g\tau_2}
$$

The Whole AGT Story

- The “meaning” function ⟦ _ ⟧ forms a Galois connection between precise and gradual types.
- Guided by the Galois connection, define consistent equality and derive dynamic and static semantics.
- “Semantics design by abstract interpretation”

Galois Connections

$[\![\_]\!] : \mathsf{type}^\sharp \rightarrow \wp(\mathsf{type})$
$[\![\mathbb{B}]\!] \coloneqq \{\mathbb{B}\}$
$[\![g\tau_1 \rightarrow g\tau_2]\!] \coloneqq \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in [\![g\tau_1]\!] \wedge \tau_2 \in [\![g\tau_2]\!]\}$
$[\![?]\!] \coloneqq \{\tau \mid \tau \in \mathsf{type}\}$

Galois Connections

$$\gamma : \text{type}^\sharp \to \wp(\text{type})$$
$$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$$
$$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\}$$
$$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$$

Galois Connections

$\gamma : \text{type}^\sharp \rightarrow \wp(\text{type})$
$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$
$\gamma(g\tau_1 \rightarrow g\tau_2) \coloneqq \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$
$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$

$\alpha : \wp(\text{type}) \rightarrow \text{type}^\sharp$
$\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$

Galois Connections

$\gamma : type^\sharp \to \wp(type)$
$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$
$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$
$\gamma(?) \coloneqq \{\tau \mid \tau \in type\}$

$\alpha : \wp(type) \to type^\sharp$
$\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$

$\eta : type \to type^\sharp$
$\eta(\mathbb{B}) \coloneqq \mathbb{B}$
$\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$

$\tau_1 \sqcup \tau_2 \coloneqq ?$ *when* $\tau_1 \neq \tau_2$
τ_1 *when* $\tau_1 = \tau_2$

Galois Connections

$$\gamma : type^\sharp \to \wp(type)$$
$$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$$
$$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\}$$
$$\gamma(?) \coloneqq \{\tau \mid \tau \in type\}$$

Non-constructive	***Constructive***
$\alpha : \wp(type) \to type^\sharp$ $\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$	$\eta : type \to type^\sharp$ $\eta(\mathbb{B}) \coloneqq \mathbb{B}$ $\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$ $\tau_1 \sqcup \tau_2 \coloneqq ?$ *when* $\tau_1 \neq \tau_2$ τ_1 *when* $\tau_1 = \tau_2$

Galois Connections

"specification effect"

$$\gamma : \text{type}^\sharp \to \wp(\text{type})$$
$$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$$
$$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$$
$$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$$

Non-constructive	***Constructive***
$\alpha : \wp(\text{type}) \to \text{type}^\sharp$ $\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$	$\eta : \text{type} \to \text{type}^\sharp$ $\eta(\mathbb{B}) \coloneqq \mathbb{B}$ $\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$ $\tau_1 \sqcup \tau_2 \coloneqq ?$ *when* $\tau_1 \neq \tau_2$ τ_1 *when* $\tau_1 = \tau_2$

Demo