Conceptual Topology Language (CTL): A Categorical Framework for Semantic Computation

A minimal semantic calculus based on category-theoretic structure, modeling logical operations (AND, OR, XOR, NOT, implication) over contextual frames (Z-frames), and enabling semantic evaluation via conceptual morphisms. This α-version demonstrates the construction of exponential objects, semantic XOR, and Z-dependent entailment classification (χ).

Logical Operations in Conceptual Topology

Following CTL Examples are simplified version. You need to clarify the Z frame as follows.

```
girl ⇔ woman | Human ∧ Femenity
```

Logical Symbol	CTL Example	Categorical Structure
$\top$ (true)	The concept of truth	Terminal object 1
$\bot$ (false)	A null concept	Initial object 0 or Zero morphism
$\neg p$ (not p)	σ(Z).Not(f)	Zero morphism (rupture)
$\neg\neg p$	Equalizer(σ∘σ, id)	Equalizer (closure of negation)
$p \wedge q$	girl ∧ dog ≅ mammal	Pullback (common intersection)
$p \vee q$	dog ∨ cat ≅ pet	Coproduct (union of categories)
$p \veebar q$	man ∨ woman ≅ genderness	$p \veebar q := (p \vee q) \ominus (p \wedge q)$
$p \Rightarrow q$	dog ⇒ mammal	Exponential object (Hom(p, q))
$p \Leftrightarrow q$	girl ⇔ woman	Isomorphism (bidirectional morphism)
$p = q$	girl = girl	Identiy Morphism (conceptual identity)
$p \cong q$	girl = woman	Isomorphism (conceptual identity)
$\exists x.\ P(x)$	∃ dog. dog ∈ pet	Co-limit (existential quantification)
$\forall x.\ P(x)$	∀ dog. dog ∈ mammal	Limit (universal quantification)

XOR

Example1:

```
using z-space Female
p ⊻ q := (p ∨ q) ⊖ (p ∧ q)
girl ⊻ woman
= (girl ∨ woman)  ⊖  (girl ∧ woman)
= age-attribute
```

Diagram:

```
             Female
            /     \
        girl       woman
          \         /
           \       /
         girl ∧ woman   ← Pullback
              |
              v
         girl ⊻ woman   ← XOR
              ^
              |
         girl ∨ woman   ← Coproduct
```

Example 2:

```
using z-space Human

prince = (king ∧ man) ∧ young
princess = (queen ∧ woman) ∧ young
prince ⊻ princess  = genderness

More precisely
Let:
  A = prince
  B = princess
  Z = Human
  f = A ∨ B | Z    // Coproduct
  g = A ∧ B | Z    // Pullback

Then:
  h = f ⊻ g | Z    // Difference morphism: Genderness

Therefore:
  Genderness = prince  ⊻ princess | Z
```

Diagram:

```
               Human                   Human
               /   \                   /   \
        prince       princess    prince    princess
           \          /             \          /
            \        /               \        /
          prince ∨ princess        prince ∧ princess
                 |                          |
                 |────────────⊖──────────|
                              |
                              v
                      prince ⊻ princess
                              |
                              v
                         Genderness
```

AND in Diagram

```
prince = (king ∧ man) ∧ young

Let:
 Z = Human
 A = king
 B = man
 C = young

Then:
 P₁ = A ∧ B | Z        (Pullback: adult male royalty)
 P₂ = P₁ ∧ C | Z       (Pullback: adds age attribute)

Therefore:
 prince := P₂ | Z


      Human
     /    \
    /      \
   /        \
king         man
   \          /       Human
    \        /         /
     ↘     ↙         ↙
    king × man    young
         \          /
          \        /
           ↘     ↙
            prince
```

king - man + woman = queen

```
queen = (king ⊻ man) ∧ woman | Human
```

Diagram:

```
        Human                    Human
        /   \                    /   \
    king      man            king    man
      \         /              \       /
       \      /                 \    /
     king ∨ man               king ∧ man
          |                        |
          |─────────────⊖──────────|
                        |
                        ∨
                  king ⊻ man
                        |           Human
                        ∨               |
                     Royalty        woman
                        |               |
                        |───────────────|
                               |
                             queen
```

Application

```
func exponential(a: Concept, b: Concept, z: Frame) -> ConceptualMorphism:
  """
  Construct exponential object B^A under Z-frame.
  Represents: 'if a then b' interpreted within context z.
  """
  return σ(z).>(a, b)  // represents B^A

func eval(f: Conceptual Morphism, a: Concept, z: Frame) → Concept:
  return f × a | z

//if press then open → it opens
val1 = eval("press", "open", "door") → door → opened door | Door

//if press then open + you press → it opens
f = exponential("press", "open", "door")
val2 = eval(f, "you press") → open | Door // open → open | Dorr

// if means truth = χ(val1, val2)
// χ: entailment classifier → Ω
if val1 ⇒ val2:
    print("This is about door being opened")
```

simbols

```
Z : Intermediating variable (conceptual anchor; Z-frame)
| : Frame separator (indicates morphism is mediated by Z-frame)
→: Morphic Flow
↛: Ruptured morphism
F : Cross-category morphism (used in cross-category flow under shared Z-frame)
//: Used to narrate meaning flow of morphic chains.
¬: Absence

M|Z : Monoid of Conceptual Flow under Z-frame
R|Z := { rupture(f) | rupture(f, σ(f) | Z) ≠ ∅ }
e|Z : Identity element of M|Z
D(Aₙ₋₁ | Z) : Morphic chain under Z frame

σ : Conceptual Shifting Morphism
>> : Generalization relation (A >> X ≡ A ⊑ X)
<< : Specialization relation (X >> A ≡ X ⊑ A)
rupture(f, σ(f) | Z) ≠ ∅ : Indicates conceptual rupture
η : Quasi-Natural Transformation: Contextual alignment between morphic chains.

⊕:  Conceptual morphism set addition in σ or morphic merger such as:
    (k₂ ∘ k₁) ⊕ (q₂ ∘ q₁) = human → royalty | Z'
⊖:  Conceptual morphism set subtraction
    Removes specified morphisms from a morphic chain or set.
```

Notations

```
Concept / Word (lexeme):
    - Lower case (e.g., puppy, dog, girl, she)

Z Frame (conceptual anchor):
    - Upper case (e.g., Mammal, Human, Agency, Domesticated, Royalty)

Type variables (A, B, X, Y, Z in formal definitions):
    - Follow standard formal notation (uppercase)

Example:
puppy → dog | Mammal
A → B | Z

Morphism: f, g, h
Functor: F
```

Simplified Form of Identity Morphism:

1. f: X → X | X (Category-theoretic identity)
   In simplified form: X
   or more explicitly: id_X

2. f: X → X | Z (Mediated identity with conceptual flow)
   In simplified form: X | Z

σ Operator

```
σ(X). Not(x){ A ↛/B | Z}     →     Rupture under Z frame
σ(X). so_much(x){A → B | Z}  →     Preservation & amplification under Z frame
σ(X). >>(x,y)                →     Conceptual Shifting x to y (Generalization) as
function form
σ(X). <<(x,y)                →     Downward Shifting x to y (Specialization) as
function form
σ(X). >(x,y)                 →     Conceptual Shifting
```

Conceptual Morphism Set Operators

```
Addition (⊕):
σ(X). ⊕(f, Aₙ₋₁ | Z): D(Aₙ₋₁ | Z) → D(Bₙ₋₁ | Z) | Z
σ(X). ⊕(f₁, f₂) : Aₙ₋₁ :={f₁, f₂}

Subtraction (⊖):
⊖: Aₙ₋₁ ⊖ {f_i}
σ(X). ⊖(f, Aₙ₋₁ | Z): D(Aₙ₋₁ | Z) → D(Bₙ₋₁ | Z) | Z

- ⊕ operator is σ_safe if Z alignment is preserved.
- ⊖ operator is potentially σ_unsafe but can be σ_safe if resulting chain
preserves the underlying morphic continuity Z.
```

σ Typing Hierarchy

σ_safe: $D(A_{n-1} | Z) \to D(B_{n-1} | Z) | Z$ (Preserves global coherence)
σ_unsafe: $D(A_{n-1} | Z) \to \{ \text{rupture}(f_1), \ldots, \text{rupture}(f_n) | \neg Z \}$ (Global coherence lost)

Note: σ_safe behaves as Quasi-Natural Transformation.
σ_unsafe induces rupture, and cannot be captured globally.

This is version α 1.1

Conceptual Topos Named as 概念位相論 / Conceptual Topology

This theory, named 概念位相論 or Conceptual Topoloy, was proposed by **No Name Yet Exist**.

Meaning no longer escapes.
It circulates within the morphic fibration.
We, once again, govern the topology of meaning.

GitHub: https://github.com/No-Name-Yet-Exist/Conceptual-Topology
Note: https://note.com/xoreaxeax/n/n3711c1318d0b
Zenodo: https://zenodo.org/records/15455079
