

# Issue XXI: Super Type System

М.Е. Сохацький <sup>1</sup>

<sup>1</sup> Національний технічний університет України  
ім. Ігоря Сікорського  
26 листопада 2025

## Анотація

Here is presented Groupoid Infinity language for TED-K.

## Зміст

<b>1</b>	<b>Introduction to Urs</b>	<b>1</b>
<b>2</b>	<b>Super Type System</b>	<b>1</b>
2.1	Bosonic Modality . . . . .	1
2.2	Bose . . . . .	1
2.3	Braid . . . . .	2
2.4	Graded Universes . . . . .	5
2.5	KU . . . . .	7

## 1 Introduction to Urs

## 2 Super Type System

### 2.1 Bosonic Modality

The  $\bigcirc$  modality in cohesive type theory projects a type to bosonic parity ( $g = 0$ ). For a type  $A : \mathbf{U}_{i,g}$ ,  $\bigcirc A$  forces the type to be bosonic, aligning with supergeometry and quantum physics.

In Urs,  $\bigcirc$  operates on types in graded universes from **Graded**, with applications in bosonic quantum fields **qubit** and supergeometry **SmthSet**.

### 2.2 Bose

**Definition 1** (Bosonic Modality Formation). The  $\bigcirc$  modality is a type operator on graded universes, mapping to bosonic parity:

$$\bigcirc : \prod_{i:\mathbb{N}} \prod_{g:\mathbf{Grade}} \mathbf{U}_{i,g} \rightarrow \mathbf{U}_{i,0}.$$

```
def bosonic (i : Nat) (g : Grade) (A : U i g) : U i 0
```

**Definition 2** (Bosonic Modality Introduction). Applying  $\bigcirc$  to a type  $A$  produces  $\bigcirc A$  with bosonic parity:

$$\Gamma \vdash A : \mathbf{U}_{i,g} \rightarrow \Gamma \vdash \bigcirc A : \mathbf{U}_{i,0}.$$

**Definition 3** (Bosonic Modality Elimination). The eliminator for  $\bigcirc A$  maps bosonic types to properties in  $\mathbf{U}_0$ :

$$\text{Ind}_{\bigcirc} : \prod_{i:\mathbb{N}} \prod_{g:\mathbf{Grade}} \prod_{A:\mathbf{U}_{i,g}} \prod_{\phi:(\bigcirc A) \rightarrow \mathbf{U}_0} \left( \prod_{a:\bigcirc A} \phi a \right) \rightarrow \prod_{a:\bigcirc A} \phi a.$$

```
def bosonic_ind (i : Nat) (g : Grade) (A : U i g)
  (phi : (bosonic i g A) -> U_0)
  (h : Π (a : bosonic i g A), phi a)
  : Π (a : bosonic i g A), phi a
```

**Theorem 1** (Idempotence of Bosonic). The  $\bigcirc$  modality is idempotent, as it always projects to bosonic parity:

$$\bigcirc\text{-idem} : \prod_{i:\mathbb{N}} \prod_{g:\mathbf{Grade}} \prod_{A:\mathbf{U}_{i,g}} (\bigcirc(\bigcirc A)) = (\bigcirc A).$$

```
def bosonic_idem (i : Nat) (g : Grade) (A : U i g)
  : (bosonic i 0 (bosonic i g A)) = (bosonic i g A)
```

**Theorem 2** (Bosonic Qubits). For  $C, H : \mathbf{U}_0$ , the type  $\bigcirc\mathbf{Qubit}(C, H)$  models bosonic quantum states:

$$\bigcirc\text{-qubit} : \prod_{i:\mathbb{N}} \prod_{g:\mathbf{Grade}} \prod_{C,H:\mathbf{U}_0} (\bigcirc\mathbf{Qubit}(C, H)) : \mathbf{U}_{i,0}.$$

```
def bosonic_qubit (i : Nat) (g : Grade) (C H : U_0) : U i 0
:= bosonic i g (Qubit C H)
```

## 2.3 Braid

The  $\mathbf{Braid}_n(X)$  type models the braid group  $B_n(X)$  on  $n$  strands over a smooth set  $X : \mathbf{SmthSet}$ , the fundamental group of the configuration space  $\mathbf{Conf}^n(X)$ , used in knot theory, quantum computing, and smooth geometry.

In Urs,  $\mathbf{Braid}_n(X)$  is a type in  $\mathbf{U}_0$ , parameterized by  $n : \mathbf{Nat}$  and  $X : \mathbf{SmthSet}$ , supporting braid generators  $\sigma_i$  and relations, with applications to anyonic quantum gates and knot invariants.

**Definition 4** (Braid Formation). The type  $\mathbf{Braid}_n(X)$  is formed for each  $n : \mathbf{Nat}$  and  $X : \mathbf{SmthSet}$ :

$$\mathbf{Braid} : \prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \mathbf{U}_0.$$

```
def Braid (n : Nat) (X : SmthSet) : U_0
(* Braid group type *)
```

**Definition 5** (Braid Introduction). Terms of type  $\mathbf{Braid}_n(X)$  are introduced via the **braid** constructor, representing generators  $\sigma_i$  for  $i : \mathbf{Fin} (n-1)$ :

$$\mathbf{braid} : \prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \prod_{i:\mathbf{Fin} (n-1)} \mathbf{Braid}_n(X).$$

```
def braid (n : Nat) (X : SmthSet) (i : Fin (n-1)) : Braid n X
(* Braid generator sigma_i *)
```

**Definition 6** (Braid Elimination). The eliminator for  $\mathbf{Braid}_n(X)$  maps braid elements to properties in  $\mathbf{U}_0$ :

$$\mathbf{BraidInd} : \prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \prod_{\beta:\mathbf{Braid}_n(X) \rightarrow \mathbf{U}_0} \left( \prod_{b:\mathbf{Braid}_n(X)} \beta b \right) \rightarrow \prod_{b:\mathbf{Braid}_n(X)} \beta b.$$

```
def braid_ind (n : Nat) (X : SmthSet)
  (beta : Braid n X -> U_0)
  (h : Π (b : Braid n X), beta b)
  : Π (b : Braid n X), beta b
```

**Theorem 3** (Braid Relations). For  $n : \mathbf{Nat}$ ,  $X : \mathbf{SmthSet}$ ,  $\mathbf{Braid}_n(X)$  satisfies the braid group relations (Commutation and Yang-Baxter):

$$\prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \prod_{i,j:\mathbf{Fin} (n-1), |i-j|\geq 2} \sigma_i \cdot \sigma_j = \sigma_j \cdot \sigma_i,$$

$$\prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \prod_{i:\mathbf{Fin} (n-2)} \sigma_i \cdot \sigma_{i+1} \cdot \sigma_i = \sigma_{i+1} \cdot \sigma_i \cdot \sigma_{i+1}.$$

```
def braid_rel_comm (n : Nat) (X : SmthSet) (i j : Fin (n-1))
  : Path (braid i · braid j) (braid j · braid i)
```

```
def braid_rel_yang (n : Nat) (X : SmthSet) (i : Fin (n-2))
  : Path (braid i · braid (i+1) · braid i) (braid (i+1) ·
    braid i · braid (i+1))
```

**Theorem 4** (Configuration Space Link). For  $n : \mathbf{Nat}$ ,  $X : \mathbf{SmthSet}$ ,  $\mathbf{Braid}_n(X)$  is the fundamental groupoid of  $\mathbf{Conf}^n(X)$ :

$$\prod_{n:\mathbf{Nat}} \prod_{X:\mathbf{SmthSet}} \mathbf{Braid}_n(X) \cong \pi_1(\mathbf{Conf}^n(X)).$$

```
def braid_conf (n : Nat) (X : SmthSet)
  : Path (Braid n X) (pi_1 (Conf n X))
```

**Theorem 5** (Quantum Braiding). For  $C, H : \mathbf{U}_0$ ,  $\mathbf{Braid}_n(X)$  acts on  $\mathbf{Qubit}(C, H)^{\otimes n}$  as braiding operators:

$$\mathbf{braid\_qubit} : \prod_{n:\mathbf{Nat}} \prod_{C,H:\mathbf{U}_0} \prod_{X:\mathbf{SmthSet}} \mathbf{Braid}_n(X) \rightarrow (\mathbf{Qubit}(C, H)^{\otimes n} \rightarrow \mathbf{Qubit}(C, H)^{\otimes n}).$$

```
def braid_qubit (n : Nat) (C H : U_0) (X : SmthSet)
  : Braid n X -> (Qubit C H)^n -> (Qubit C H)^n
```

**Theorem 6** (Braid Group Delooping). For  $n : \mathbf{Nat}$ , the delooping  $\mathbf{BB}_n$  of the braid group  $B_n$  is a 1-groupoid:

$$\mathbf{BB}_n : \mathbf{Grpd} \, 1 \equiv \mathfrak{S}(\mathbf{Conf}^n(\mathbb{R}^2)).$$

```
def BB_n (n : Nat) : Grpd 1 := \mathfrak{S} (Conf n \mathbb{R}^2)
```

## 2.4 Graded Universes

**Graded Universes.** The  $\mathbf{U}_\alpha$  type represents a graded universe indexed by a monoid  $\mathcal{G} = \mathbb{N} \times \mathbb{Z}/2\mathbb{Z}$ , where  $\alpha \in \mathcal{G}$  encodes a level ( $\mathbb{N}$ ) and parity ( $\mathbb{Z}/2\mathbb{Z}$ : 0 = bosonic, 1 = fermionic). Graded universes support type hierarchies with cumulativity, graded tensor products, and coherence rules, used in supergeometry (e.g., bosonic/fermionic types), quantum systems (e.g., graded qubits), and cohesive type theory.

In Urs,  $\mathbf{U}_\alpha$  is a type indexed by  $\alpha : \mathcal{G}$ , with operations like lifting, product formation, and graded tensor products, extending standard universe hierarchies to include parity, building on **Tensor**.

**Definition 7** (Grading Monoid). The grading monoid  $\mathcal{G}$  is defined as  $\mathbb{N} \times \mathbb{Z}/2\mathbb{Z}$ , with operation  $\oplus$  and neutral element  $\mathbf{0}$ , encoding level and parity.

$$\begin{aligned} \mathcal{G} : \mathbf{Type} &\equiv \mathbb{N} \times \mathbb{Z}/2\mathbb{Z}, \\ \oplus : \mathcal{G} &\rightarrow \mathcal{G} \rightarrow \mathcal{G}, \\ (\alpha, \beta) &\mapsto (\text{fst } \alpha + \text{fst } \beta, (\text{snd } \alpha + \text{snd } \beta) \bmod 2), \\ \mathbf{0} : \mathcal{G} &\equiv (0, 0). \end{aligned}$$

```
def  $\mathcal{G} : \mathbf{Type} := \mathbb{N} \times \mathbb{Z}/2\mathbb{Z}$ 
def  $\oplus (\alpha \beta : \mathcal{G}) : \mathcal{G} := (\text{fst } \alpha + \text{fst } \beta, (\text{snd } \alpha + \text{snd } \beta) \bmod 2)$ 
def  $\mathbf{0} : \mathcal{G} := (0, 0)$ 
```

**Definition 8** (Graded Universe Formation). The universe  $\mathbf{U}_\alpha$  is a type indexed by  $\alpha : \mathcal{G}$ , containing types of grade  $\alpha$ . A shorthand notation  $\mathbf{U}_{i,g}$  is used for  $\mathbf{U}(i, g)$ .

$$\begin{aligned} \mathbf{U} : \mathcal{G} &\rightarrow \mathbf{Type}, \\ \mathbf{Grade} : \mathbf{Set} &\equiv \{0, 1\}, \\ \mathbf{U}_{i,g} : \mathbf{Type} &\equiv \mathbf{U}(i, g) : \mathbf{U}_{i+1}. \end{aligned}$$

```
def  $\mathbf{U} (\alpha : \mathcal{G}) : \mathbf{Type} := \mathbf{Universe } \alpha$ 
def  $\mathbf{Grade} : \mathbf{Set} := \{0, 1\}$ 
def  $\mathbf{U} (i : \mathbf{Nat}) (g : \mathbf{Grade}) : \mathbf{Type} := \mathbf{U} (i, g)$ 
def  $\mathbf{U}_0 (g : \mathbf{Grade}) : \mathbf{U} (1, g) := \mathbf{U} (0, g)$ 
def  $\mathbf{U}_{00} : \mathbf{Type} := \mathbf{U} (0, 0)$ 
def  $\mathbf{U}_{10} : \mathbf{Type} := \mathbf{U} (1, 0)$ 
def  $\mathbf{U}_{01} : \mathbf{Type} := \mathbf{U} (0, 1)$ 
```

**Definition 9** (Graded Universe Coherence Rules). Graded universes support coherence rules for lifting, product formation, and substitution, ensuring type-

theoretic consistency.

$$\begin{aligned}
\mathbf{lift} &: \prod_{\alpha, \beta: \mathcal{G}} \prod_{\delta: \mathcal{G}} \mathbf{U} \alpha \rightarrow (\beta = \alpha \oplus \delta) \rightarrow \mathbf{U} \beta, \\
\mathbf{univ} &: \prod_{\alpha: \mathcal{G}} \mathbf{U} (\alpha \oplus (1, 0)), \\
\mathbf{cumul} &: \prod_{\alpha, \beta: \mathcal{G}} \prod_{A: \mathbf{U} \alpha} \prod_{\delta: \mathcal{G}} (\beta = \alpha \oplus \delta) \rightarrow \mathbf{U} \beta, \\
\mathbf{prod} &: \prod_{\alpha, \beta: \mathcal{G}} \prod_{A: \mathbf{U} \alpha} \prod_{B: A \rightarrow \mathbf{U} \beta} \mathbf{U} (\alpha \oplus \beta), \\
\mathbf{subst} &: \prod_{\alpha, \beta: \mathcal{G}} \prod_{A: \mathbf{U} \alpha} \prod_{B: A \rightarrow \mathbf{U} \beta} \prod_{t: A} \mathbf{U} \beta, \\
\mathbf{shift} &: \prod_{\alpha, \delta: \mathcal{G}} \prod_{A: \mathbf{U} \alpha} \mathbf{U} (\alpha \oplus \delta).
\end{aligned}$$

```

def lift (α β : G) (δ : G) (e : U α) : β = α ⊕ δ → U β :=
  λ eq : β = α ⊕ δ, transport (λ x : G, U x) eq e

def univ-type (α : G) : U (α ⊕ (1, 0)) :=
  lift α (α ⊕ (1, 0)) (1, 0) (U α) refl

def cumul (α β : G) (A : U α) (δ : G) : β = α ⊕ δ → U β :=
  lift α β δ A

def prod-rule (α β : G) (A : U α) (B : A → U β) : U (α ⊕ β) :=
  Π (x : A), B x

def subst-rule (α β : G) (A : U α) (B : A → U β) (t : A) : U β :=
  B t

def shift (α δ : G) (A : U α) : U (α ⊕ δ) :=
  lift α (α ⊕ δ) δ A refl

```

**Definition 10** (Graded Tensor Introduction). Graded tensor products combine types with matching levels, combining parities.

$$\begin{aligned}
\mathbf{tensor} &: \prod_{i: \mathbb{N}} \prod_{g_1, g_2: \mathbf{Grade}} \mathbf{U}_{i, g_1} \rightarrow \mathbf{U}_{i, g_2} \rightarrow \mathbf{U}_{i, (g_1 + g_2 \bmod 2)}, \\
\mathbf{pair-tensor} &: \prod_{i: \mathbb{N}} \prod_{g_1, g_2: \mathbf{Grade}} \prod_{A: \mathbf{U}_{i, g_1}} \prod_{B: \mathbf{U}_{i, g_2}} \prod_{a: A} \prod_{b: B} \mathbf{tensor}(i, g_1, g_2, A, B).
\end{aligned}$$

```

def tensor (i : Nat) (g1 g2 : Grade)
  (A : U i g1) (B : U i g2) : U i (g1 + g2 mod 2)
:= A ⊗ B

def pair-tensor (i : Nat) (g1 g2 : Grade) (A : U i g1)
  (B : U i g2) (a : A) (b : B) : tensor i g1 g2 A B
:= a ⊗ b

```

**Definition 11** (Graded Tensor Eliminators). Eliminator for graded tensor products project to their components.

$$\begin{aligned}\otimes\text{-prj}_1 &: (A \otimes B) \rightarrow A, \\ \otimes\text{-prj}_2 &: (A \otimes B) \rightarrow B.\end{aligned}$$

```
def pr1 (i : Nat) (g1 g2 : Grade)
  (A : U i g1) (B : U i g2) (p : A ⊗ B) : A := p.1

def pr2 (i : Nat) (g1 g2 : Grade)
  (A : U i g1) (B : U i g2) (p : A ⊗ B) : B := p.2
```

**Theorem 7** (Monoid Properties). The grading monoid  $\mathcal{G}$  satisfies associativity and identity laws.

$$\begin{aligned}\text{assoc} &: ((\alpha \oplus \beta) \oplus \gamma) = (\alpha \oplus (\beta \oplus \gamma)), \\ \text{id-left} &: (\alpha \oplus \mathbf{0}) = \alpha, \\ \text{id-right} &: (\mathbf{0} \oplus \alpha) = \alpha.\end{aligned}$$

```
def assoc (α β γ : G) : (α ⊕ β) ⊕ γ = α ⊕ (β ⊕ γ) := refl
def ident-left (α : G) : α ⊕ 0 = α := refl
def ident-right (α : G) : 0 ⊕ α = α := refl
```

## 2.5 KU

The  $\mathbf{KU}^G$  type represents generalized K-theory, a topological invariant used to classify vector bundles or operator algebras over a space, twisted by a groupoid. It is a cornerstone of algebraic topology and mathematical physics, with applications in quantum field theory, string theory, and index theory.

In the cohesive type system,  $\mathbf{KU}^G$  operates on smooth sets  $\mathbf{SmothSet}$  and groupoids  $\mathbf{Grpd}_1$ , producing a type in the universe  $\mathbf{U}_{(0,0)}$ . It incorporates a twist to account for non-trivial topological structures, making it versatile for modeling complex physical systems.

**Definition 12** ( $\mathbf{KU}^G$ -Formation). The generalized K-theory type  $\mathbf{KU}^G$  is formed over a term  $X : \mathbf{U}_{(0,0)}$ , a groupoid  $G : \mathbf{U}_{(0,0)}$ , and a twist  $\tau : \prod_{x:X} \mathbf{U}_{(0,0)}$ , yielding a type in the universe  $\mathbf{U}_{(0,0)}$ :

$$\mathbf{KU}^G : \prod_{X:\mathbf{U}_{(0,0)}} \prod_{G:\mathbf{U}_{(0,0)}} \prod_{\tau:\prod_{x:X} \mathbf{U}_{(0,0)}} \mathbf{U}_{(0,0)}.$$

```
type exp =
| KU^G of exp * exp * exp
```

**Definition 13** ( $\mathbf{KU}^G$ -Introduction). A term of type  $\mathbf{KU}^G(X, G, \tau)$  is introduced by constructing a generalized K-theory class, representing a stable equivalence class of vector bundles or operators over  $X$ , twisted by  $G$  and  $\tau$ :

$$\mathbf{KU}^G : \prod_{X: \mathbf{U}_{(0,0)}} \prod_{G: \mathbf{U}_{(0,0)}} \prod_{\tau: \prod_{x: X} \mathbf{U}_{(0,0)}} \mathbf{KU}^G(X, G, \tau).$$

```
let KU^G_intro (x : exp) (g : exp) (tau : exp) : exp =
  KU^G (x, g, tau)
```

**Definition 14** ( $\mathbf{KU}^G$ -Elimination). The eliminator for  $\mathbf{KU}^G$  allows reasoning about generalized K-theory classes by mapping them to properties or types dependent on  $\mathbf{KU}^G(X, G, \tau)$ , typically by analyzing the underlying bundle or operator structure over  $X$ :

$$\mathbf{KU}^G\text{Ind} : \prod_{X: \mathbf{U}_{(0,0)}} \prod_{G: \mathbf{U}_{(0,0)}} \prod_{\tau: \prod_{x: X} \mathbf{U}_{(0,0)}} \prod_{\beta: \mathbf{KU}^G(X, G, \tau) \rightarrow \mathbf{U}_{(0,0)}} \left( \prod_{k: \mathbf{KU}^G(X, G, \tau)} \beta k \right) \rightarrow \prod_{k: \mathbf{KU}^G(X, G, \tau)} \beta k.$$

```
let KU^G_ind (x : exp) (g : exp) (tau : exp) (beta : exp) (h : exp) : exp =
  (* Hypothetical eliminator *)
  App (Var "KU^GInd", KU^G (x, g, tau))
```

**Theorem 8** (K-Theory Stability). The type  $\mathbf{KU}^G(X, G, \tau)$  is stable under suspension, meaning it is invariant under the suspension operation in the spectrum, reflecting its role in stable homotopy theory:

$$\text{stability} : \prod_{X: \mathbf{U}_{(0,0)}} \prod_{G: \mathbf{U}_{(0,0)}} \prod_{\tau: \prod_{x: X} \mathbf{U}_{(0,0)}} \mathbf{KU}^G(X, G, \tau) =_{\mathbf{U}_{(0,0)}} \mathbf{KU}^G(\text{Susp } X, G, \tau).$$

```
let KU^G_stability (x : exp) (g : exp) (tau : exp) : exp =
  Path (Universe (0, Bose), KU^G (x, g, tau), KU^G (Susp x, g, tau))
```

**Theorem 9** (Refinement to Differential K-Theory, Theorem 3.4.5). The type  $\mathbf{KU}^G(X, G, \tau)$  can be refined to differential K-theory by incorporating a connection, as provided by  $\mathbf{KU}_b^G(X, G, \tau, \text{conn})$ :

$$\text{refine}_{\mathbf{KU}_b^G} : \prod_{X: \mathbf{U}_{(0,0)}} \prod_{G: \mathbf{U}_{(0,0)}} \prod_{\tau: \prod_{x: X} \mathbf{U}_{(0,0)}} \prod_{\text{conn}: \Omega^1(X)} \mathbf{KU}^G(X, G, \tau) \rightarrow \mathbf{KU}_b^G(X, G, \tau, \text{conn}).$$

```
let KU^G_to_KU_flat^G (x : exp) (g : exp) (tau : exp) (conn : exp) : exp =
  KU_flat^G (x, g, tau, conn)
```