

- Cubical Subtypes
- Initial Base Library
- Fast Type Check
- Strict Equality (HTS)
- Kan Operations

Anders 0.7.2

Groupoid Infinity Cafe

Groupoid Infinity, 2021, INFOTECH SE, Ukraine, Kyiv

@siegment
@5HT



Dan Kan



Daniel
Quillen



Vladimir
Voevodsky



Thierry
Coquand

PTS	ML72	ML73	CCHM	HTS
UNI	PI	UNI	UNI	UNI
PI	SIGMA	PI	PI	PI
		SIGMA	SIGMA	SIGMA
		ID	PATH	ID
		NAT	GLUE	PATH
			HIT	GLUE
				HIT

Type Theory

Anders 0.7.2

HW	VHDL, Verilog, Clash, Chisel, SystemC, Lava, BSV
ASM	PDP-11, VAX, S/360, M68K, PowerPC, MIPS, SPARC, Super-H, Intel, ARM, RISC-V
ALG	C, BCPL, ALGOL, SNOBOL, Simula, Pascal, Oberon, COBOL, PL/1
ML	SML, Alice ML, OCaml, UrWeb, Flow, F#
PURE	HOPE, Miranda, Clean, Charity, Joy, Mercury, Elm, PureScript, Fω Scala, Haskell, 1ML, Plutus
MACR	LISP, Scheme, Clojure, Racket, Dylan, LFE, CL, Nemerle, Nim, Haxe, Perl, Elixir
OOI	Simula, Smalltalk, Self, REBOL, Io, JS, Lua, Ruby, Python, PHP, TS, Java, Kotlin
CMP	C++, Rust, D, Swift, Fortran
SHELL	PowerShell, TCL, SH, CLIPS, BASIC, FORTH, SVC IDL, SOAP, ASN.1, GRPC
MARK	TeX, PS, XML, SVG, CSS, ROFF, OWL, SGML, RDF, SysML
LOGIC	AUT-68, ACL2, LEGO, ALF, Prolog, CPL, Mizar, Dedukti, HOL, Isabelle, Z
ΠΣ	Coq, F*, Lean, NuPRL, ATS, Epigram, Cayenne, Idris, Dhall, Cedile, Kind
HoTT	Menkar, Cubical, yacctl, redtt, RedPRL, Arend, Agda, Anders
CHKR	TLA+, Twelf, Promela, CSPM
PAR	Ling, Pony, Erlang, BPMN, Ada, E, Go, Occam, Oz
ARR	Julia, Wolfram, MATHLAB, Octave, Futhark, APL, SQL, cg, Clarion, Clipper, QCL, K, MUMPS, Q, R, S, J, O

CoC:	* : *	□ : *	* : □	□ : □
Fω:	* : *	□ : *		□ : □
P2:	* : *	□ : *	* : □	
AUT:	* : *		* : □	
F:	* : *	□ : *		

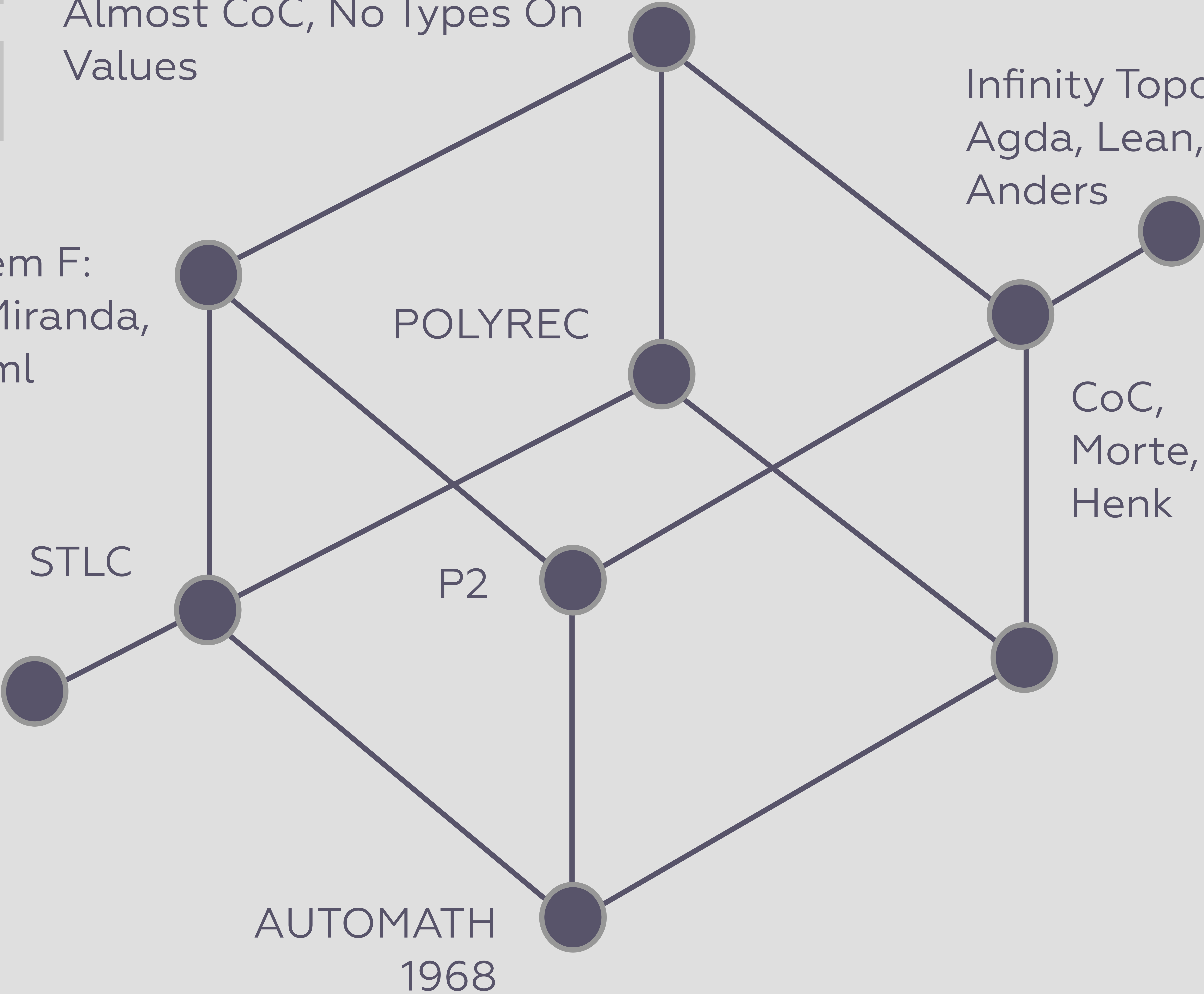
System Fω: Haskell, Scala, 1ML
Almost CoC, No Types On Values

System F:
ML, Miranda, OCaml

Infinity Topoi,
Agda, Lean,
Anders

CoC,
Morte,
Henk

Untyped SLC:
Erlang, LISP,
JavaScript



Anders 0.7.2

OCaml Internal AST

```
type exp =  
  | EPre of int | EKan of int  
  | EVar of name | EHole  
  | EPi of exp * (name * exp) | ELam of exp * (name * exp) | EApp of exp * exp  
  | ESig of exp * (name * exp) | EPair of exp * exp | EFst of exp | ESnd of exp  
  | Eld of exp | ERef of exp | EJ of exp  
  | EPathP of exp | EPLam of exp | EAppFormula of exp * exp  
  | EI | EDir of dir | EAnd of exp * exp | EOr of exp * exp | ENeg of exp  
  | ETransp of exp * exp | EPartial of exp | ESystem of system  
  | ESub of exp * exp * exp | EInc of exp | EOuc of exp
```


cosmos := U nat | V nat
 var := var name | hole
 pi := Π name E E | λ name E E | E E
 sigma := Σ name E E | (E,E) | E.1 | E.2
 id := Id E | ref E | idJ E
 path := Path E | Ei | E @ E
 I := I | 0 | 1 | E \vee E | E \wedge E | \neg E
 part := Partial E E | [(E=I) \rightarrow E, ...]
 sub := inc E | ouc E | E [I \rightarrow E]
 kan := transp E E | hcomp E
 glue := Glue E | glue E | unglue E E

Informal BNF

E := cosmos | var | MLTT | CCHM | HIT

HIT := inductive E E | ctor name E | match E E

CCHM := path | I | part | sub | kan | glue

MLTT := pi | sigma | id

Cosmos

Anders 0.7.2

```
inductive cosmos : U
  | fibrant: nat → cosmos
  | pretypes: nat → cosmos
```

Agda 2.6.2

```
inductive cosmos : U
  | prop: nat → cosmos
  | fibrant: nat → cosmos
  | pretypes: nat → cosmos
  | strict: nat → cosmos
  | omega: cosmos
  | lock: cosmos
```


Pi

Anders 0.7.2

```
def Pi (A : U) (B : A → U) : U :=  $\prod$  (x : A), B x
def lambda (A: U) (B: A → U) (b: Pi A B) : Pi A B :=  $\lambda$  (x : A), b x
def lam (A B: U) (f: A → B) : A → B :=  $\lambda$  (x : A), f x
def apply (A: U) (B: A → U) (f: Pi A B) (a: A) : B a := f a
def app (A B: U) (f: A → B) (x: A): B := f x
def  $\prod$ - $\beta$  (A : U) (B : A → U) (a : A) (f : Pi A B)
  : Path (B a) (apply A B (lambda A B f) a) (f a) := idp (B a) (f a)
def  $\prod$ - $\eta$  (A : U) (B : A → U) (a : A) (f : Pi A B)
  : Path (Pi A B) f ( $\lambda$  (x : A), f x) := idp (Pi A B) f
```


Sigma

```
def Sigma (A: U) (B: A → U) : U :=  $\Sigma$  (x: A), B x
def pair (A: U) (B: A → U) (a: A) (b: B a) : Sigma A B := (a, b)
def pr1 (A: U) (B: A → U) (x: Sigma A B) : A := x.1
def pr2 (A: U) (B: A → U) (x: Sigma A B) : B (pr1 A B x) := x.2
def Sigma-β-1 (A : U) (B : A → U) (a : A) (b : B a)
  : Path A a (pr1 A B (a ,b)) := idp A a
def Sigma-β-2 (A : U) (B : A → U) (a : A) (b : B a)
  : Path (B a) b (pr2 A B (a, b)) := idp (B a) b
def Sigma-η (A : U) (B : A → U) (p : Sigma A B)
  : Path (Sigma A B) p (pr1 A B p, pr2 A B p) := idp (Sigma A B) p
```


Fibrations

Bundle: $F \rightarrow E \rightarrow B$

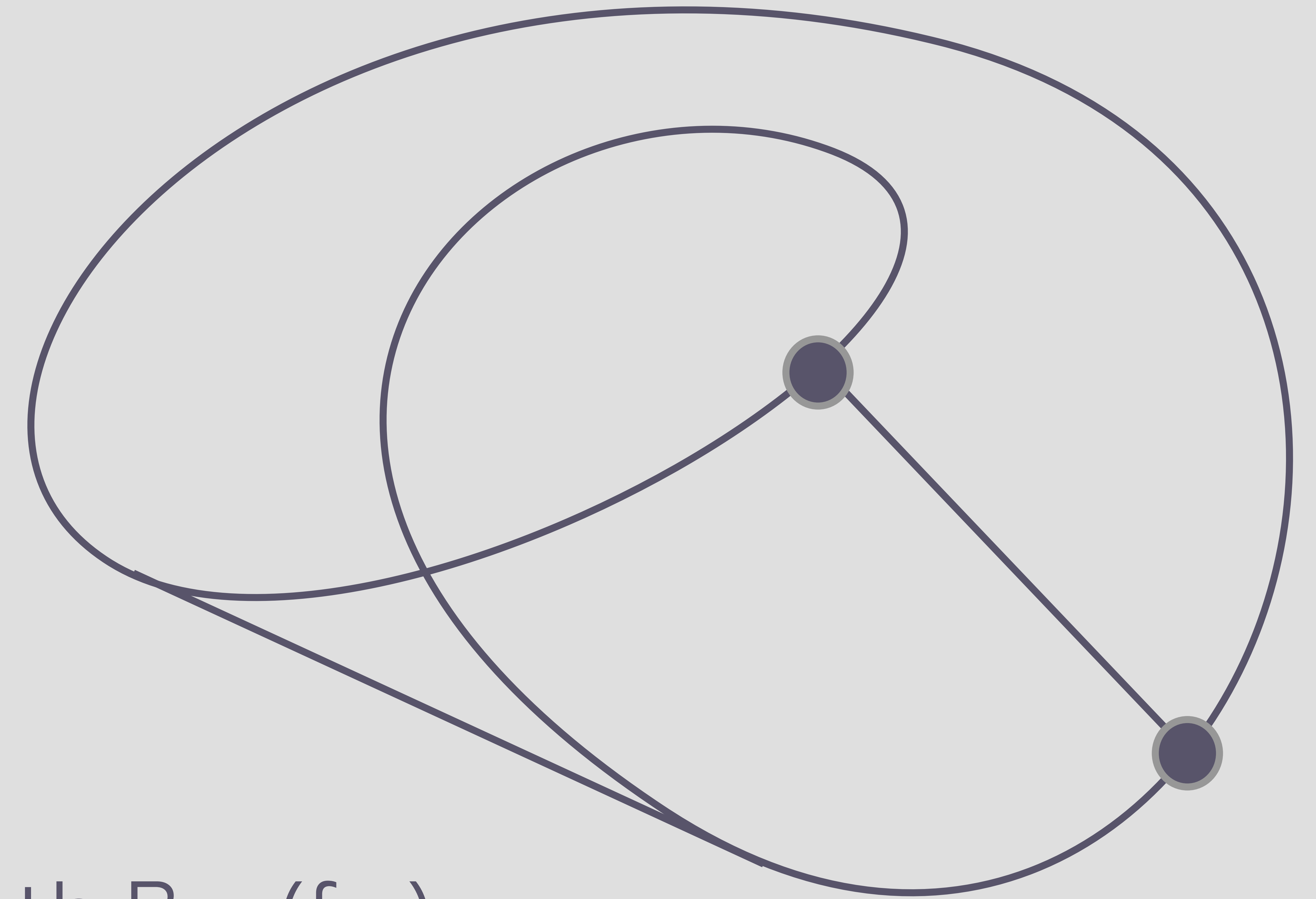
$p : \text{total} \rightarrow B$

$F = \text{fiber} : B \rightarrow \text{total}$

$\text{total} = \sum (y : B), \text{fiber}(y)$

Moebius $E = S^1 \text{ 'twisted *' } [0,1]$

Trivial: $E = B * F$



```
def fiber (A B : U) (f: A → B) (y : B): U :=  $\sum (x : A), \text{Path } B \ y \ (f \ x)$ 
```

```
def isContr' (A: U) : U :=  $\sum (x: A), \prod (y: A), \text{Path } A \ x \ y$ 
```

```
def isEquiv (A B : U) (f : A → B) : U :=  $\prod (y : B), \text{isContr } (\text{fiber } A \ B \ f \ y)$ 
```

```
def equiv (A B : U) : U :=  $\sum (f : A \rightarrow B), \text{isEquiv } A \ B \ f$ 
```

```
def idEquiv (A : U) : equiv A A := (id A, isContrSingl A)
```

Anders 0.7.2

Id (Strict Equality) in V

```
def 1= : I -> V := Id I 1
```

```
def 1=1 : 1= 1 := ref 1
```

```
def UIP (A : V) (a b : A) (p q : Id A a b) : Id (Id A a b) p q := ref p
```

```
def Js (A : V) (B : Π (a b : A), Id A a b -> V) (a b : A)
```

```
  (d : B a a (ref a)) (p : Id A a b) : B a b p := idJ A B a d b p
```

```
def Js-β (A : V) (B : Π (a b : A), Id A a b -> V) (a : A) (d : B a a (ref a))
  : Id (B a a (ref a)) (Js A B a a d (ref a)) d := ref d
```


Path (Globular Equality) in U

```
def hmtpy (A : U) (x y : A) (p : Path A x y)
  : Path (Path A x x) (<_> x) (<i> p @ i /\ -i) := <j i> p @ j /\ i /\ -i
```

```
def isProp (A : U) : U :=  $\prod$  (a b : A), Path A a b
```

```
def isSet (A : U) : U :=  $\prod$  (a b : A) (a0 b0 : Path A a b), Path (Path A a b) a0 b0
```

```
def isGroupoid (A : U) : U :=  $\prod$  (a b : A) (x y : Path A a b)
  (i j : Path (Path A a b) x y), Path (Path (Path A a b) x y) i j
```


Path (Computational)

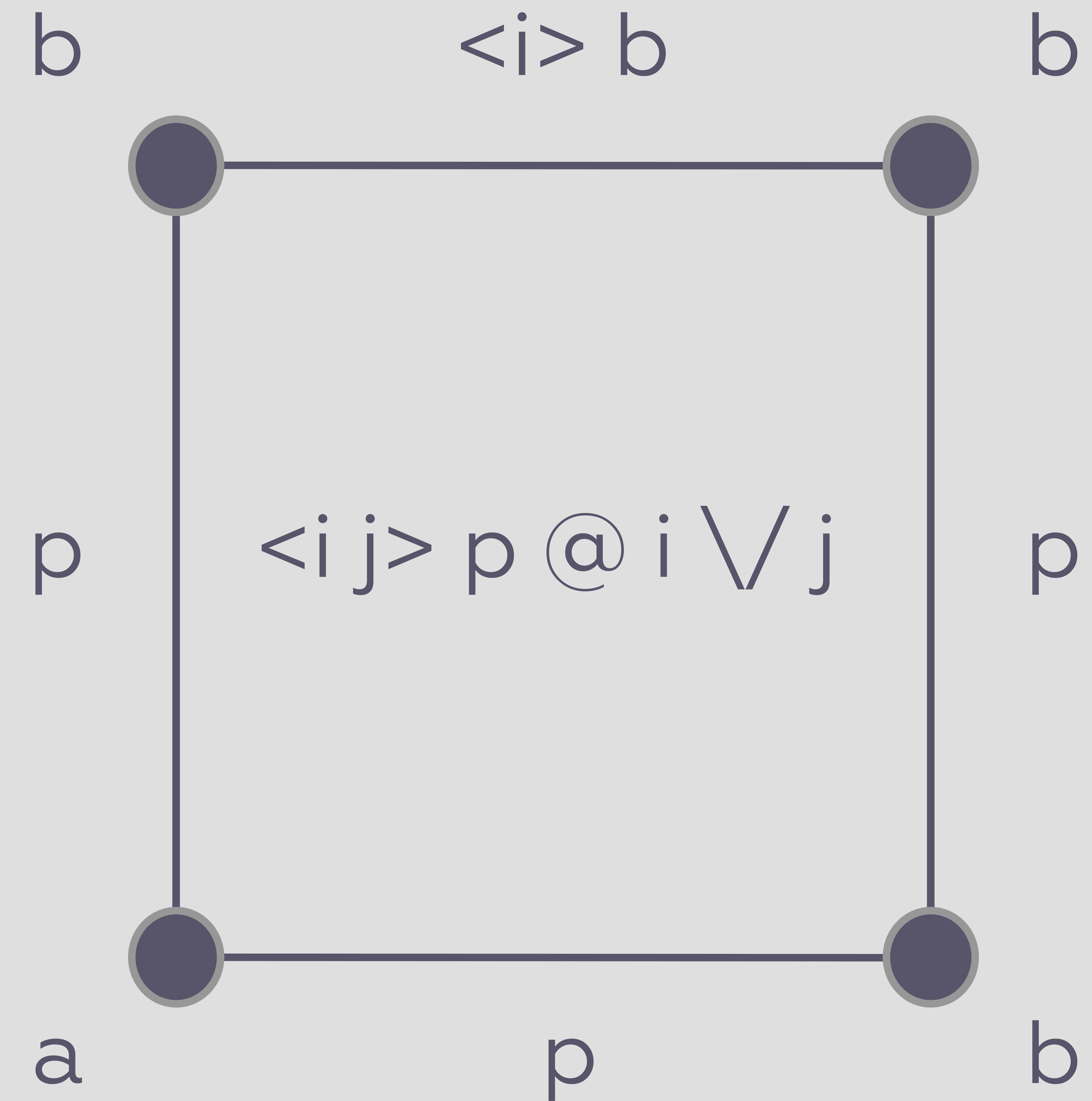
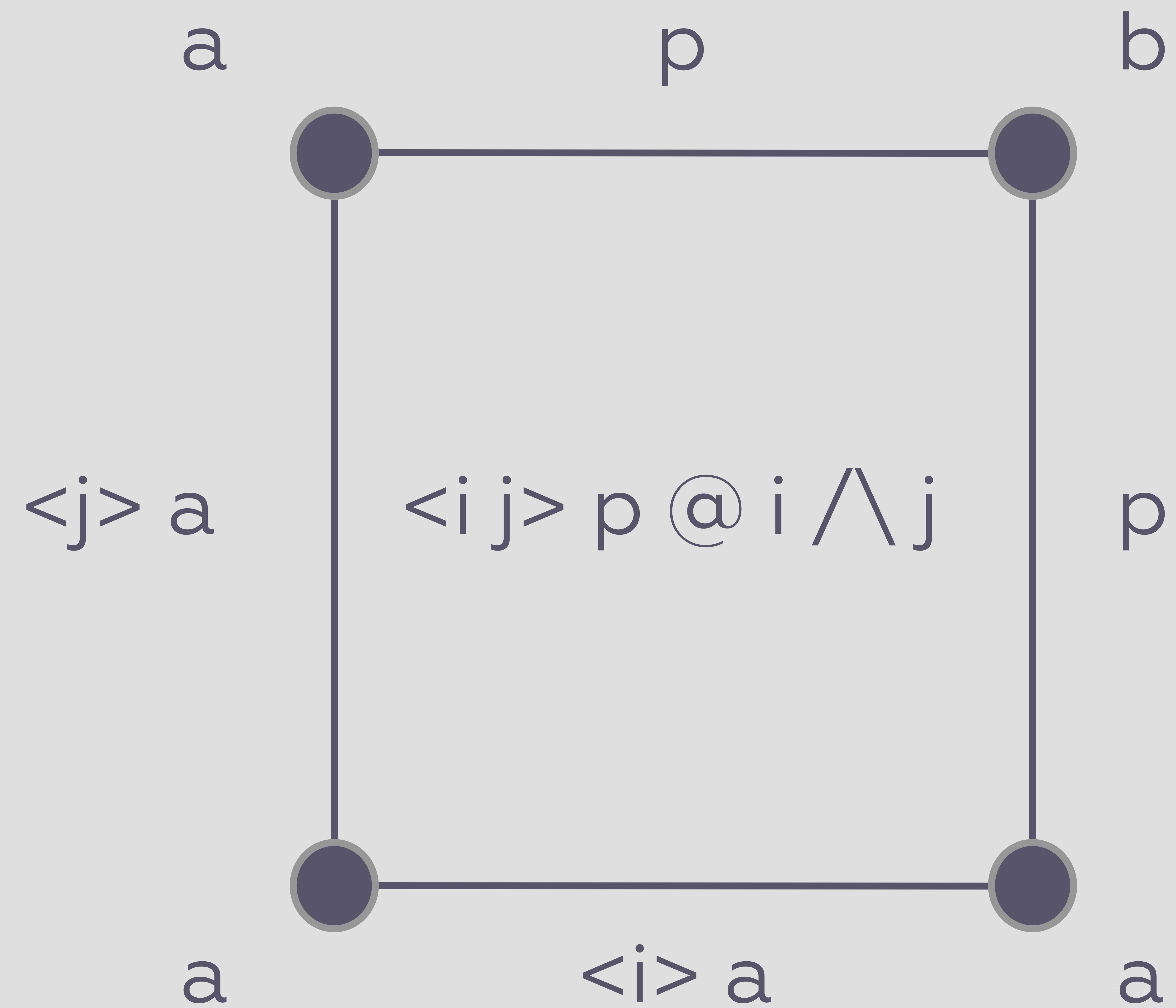
```

def transport (A B: U) (p: PathP (<_>U) A B) (a: A): B := transp p 0 a
def trans_comp (A:U)(a: A): Path A a (transport A A (<i> A) a) := <j> transp (<_> A) -j a
def subst (A: U) (P: A -> U) (a b: A) (p: Path A a b) (e: P a): P b := transp (<i> P (p @ i)) 0 e
def D (A : U) : U1 :=  $\prod$  (x y : A), Path A x y  $\rightarrow$  U
def J (A: U) (x: A) (C: D A) (d: C x x (idp A x)) (y: A) (p: Path A x y): C x y p
  := subst (singl A x) (\ (z: singl A x), C x (z.1) (z.2)) (eta A x) (y, p) (contr A x y p) d
def subst_comp (A: U) (P: A  $\rightarrow$  U) (a: A) (e: P a)
  : Path (P a) e (subst A P a a (idp A a) e) := trans_comp (P a) e
def J- $\beta$  (A : U) (a : A) (C : D A) (d: C a a (idp A a))
  : Path (C a a (idp A a)) d (J A a C d a (idp A a))
  := subst_comp (singl A a) (\ (z: singl A a), C a (z.1) (z.2)) (eta A a) d

```


Connections

Anders 0.7.2



DNF

```
def T (i j : I) : I := (i ∧ -j) ∨ (-i ∧ j)
def T-comm (i j : I) : Id I (T i j) (T j i) := ref (T i j)
def ∧-comm (i j : I) : Id I (i ∧ j) (j ∧ i) := ref (i ∧ j)
def ∨-comm (i j : I) : Id I (i ∨ j) (j ∨ i) := ref (i ∨ j)
def ¬-of-∧ (i j : I) : Id I -(i ∧ j) (-i ∨ -j) := ref -(i ∧ j)
def ¬-of-∨ (i j : I) : Id I -(i ∨ j) (-i ∧ -j) := ref -(i ∨ j)
def ∧-distrib-∨ (i j k : I) : Id I ((i ∨ j) ∧ k) ((i ∧ k) ∨ (j ∧ k)) := ref ((i ∨ j) ∧ k)
def ∨-distrib-∧ (i j k : I) : Id I ((i ∧ j) ∨ k) ((i ∨ k) ∧ (j ∨ k)) := ref ((i ∧ j) ∨ k)
def ∧-assoc (i j k : I) : Id I (i ∧ (j ∧ k)) ((i ∧ j) ∧ k) := ref (i ∧ (j ∧ k))
```


Generalized Transport

```
def subst' (A: U) (P: A → U) (a b: A) (p: Path A a b) (e: P a): P b
  := transp (<i> P (p @ i)) 0 e
def coerce (A B: U) (p: PathP (<_> U) A B): A → B := λ (x : A), trans A B p x
def pcomp (A: U) (a b c: A) (p: Path A a b) (q: Path A b c)
  : Path A a c := subst A (Path A a) b c q p
def transId (A : U) : A → A := transp (<_> A) 1
def transFill (A B : U) (p : PathP (<_> U) A B) (a : A)
  : PathP p a (transp p 0 a) := <j> transp (<i> p @ i /\j) -j a
```


transp (Huber)

$$\text{hcomp}^i N [\varphi \rightarrow 0] 0 = 0$$

$$\text{hcomp}^i N [\varphi \rightarrow S u] (S u_0) = S (\text{hcomp}^i N [\varphi \rightarrow u] u_0)$$

$$\text{hcomp}^i U [\varphi \rightarrow E] A = \text{Glue} [\varphi (E(i/1), \text{equiv}^i E(i/1-i))] A$$

$$\text{hcomp}^i (\prod (x : A), B) [\varphi \rightarrow u] u_0 v = \text{hcomp}^i B(x/v) [\varphi u v] (u_0 v)$$

$$\text{hcomp}^i (\sum (x : A), B) [\varphi \rightarrow u] u_0 = (v(i/1), \text{comp}^i B(x/v) [\varphi u.2] u_0.2), v = \text{hfill}^i A [\varphi \mapsto u.1] u_0.1$$

$$\text{hcomp}^i (\text{Path}^j A v w) [\varphi \rightarrow u] u_0 = \langle j \rangle \text{hcomp}^i A [\varphi u j, (j = 0) v, (j = 1) w] (u_0 j)$$

$$\begin{aligned} \text{hcomp}^i (\text{Glue} [\varphi \rightarrow (T, w)] A) [\psi u] u_0 &= \text{glue} [\varphi \rightarrow t_1] a_1 = \text{glue} [\varphi \rightarrow u(i/1)] (\text{unglue } u(i/1)) \\ &= u(i/1) : \text{Glue} [\varphi \rightarrow (T, w)] A, t_1 = u(i/1) : T, a_1 = \text{unglue } u(i/1) : A, \text{glue} [\varphi \rightarrow t_1] a_1 = t_1 : T \end{aligned}$$

Homogeneous Composition

```
def kan (A : U) (a b c d : A) (p : Path A a c) (q : Path A b d) (r : Path A a b)
  : Path A c d := <i> hcomp A (∂ i) (λ (j : I), [(i = 0) → p @ j, (i = 1) → q @ j]) (inc (r @ i))
```

```
def comp (A : I → U) (r : I) (u : Π (i : I), Partial (A i) r) (u₀ : (A 0)[r |→ u 0]) : A 1
  := hcomp (A 1) r (λ (i : I), [(φ : r = 1) → transp (<j> A (i ∨ j)) i (u i φ)])
    (inc (transp (<i> A i) 0 (ouc u₀)))
```

```
def ghcomp (A : U) (r : I) (u : I → Partial A r) (u₀ : A[r |→ u 0]) : A
  := hcomp A (∂ r) (λ (j : I), [(φ : r = 1) → u j φ, (r = 0) → ouc u₀]) (inc (ouc u₀))
```


hcomp (Huber)

$$\text{transp}^i N \varphi u_0 = u_0$$

$$\text{transp}^i U \varphi A = A$$

$$\text{transp}^i (\prod (x : A), B) \varphi u_0 v = \text{transp}^i B(x/w) \varphi (u_0 w(i/0)), w = \text{transpFill}^{-i} A \varphi v, v : A(i/1)$$

$$\text{transp}^i (\sum (x : A), B) \varphi u_0 = (\text{transp}^i A \varphi (u_0.1), \text{transp}^i B(x/v) \varphi (u_0.2)), v = \text{transpFill}^i A \varphi u_0.1$$

$$\text{transp}^i (\text{Path}^j A v w) \varphi u_0 = \langle j \rangle \text{comp}^i A [\varphi \rightarrow u_0 j, (j=0) v, (j=>1) w] (u_0 j), u : A(j/0), v : A(j/1)$$

$$\text{transp}^i (\text{Glue} [\varphi \rightarrow (T, w)] A) \psi u_0 = \text{glue} [\varphi (i/1) t'_1] a'_1 : B(i/1)$$

$$\text{transp}^{-i} A \varphi u = (\text{transp}^i A(i/1-i) \varphi u)(i/1-i) : A(i/0)$$

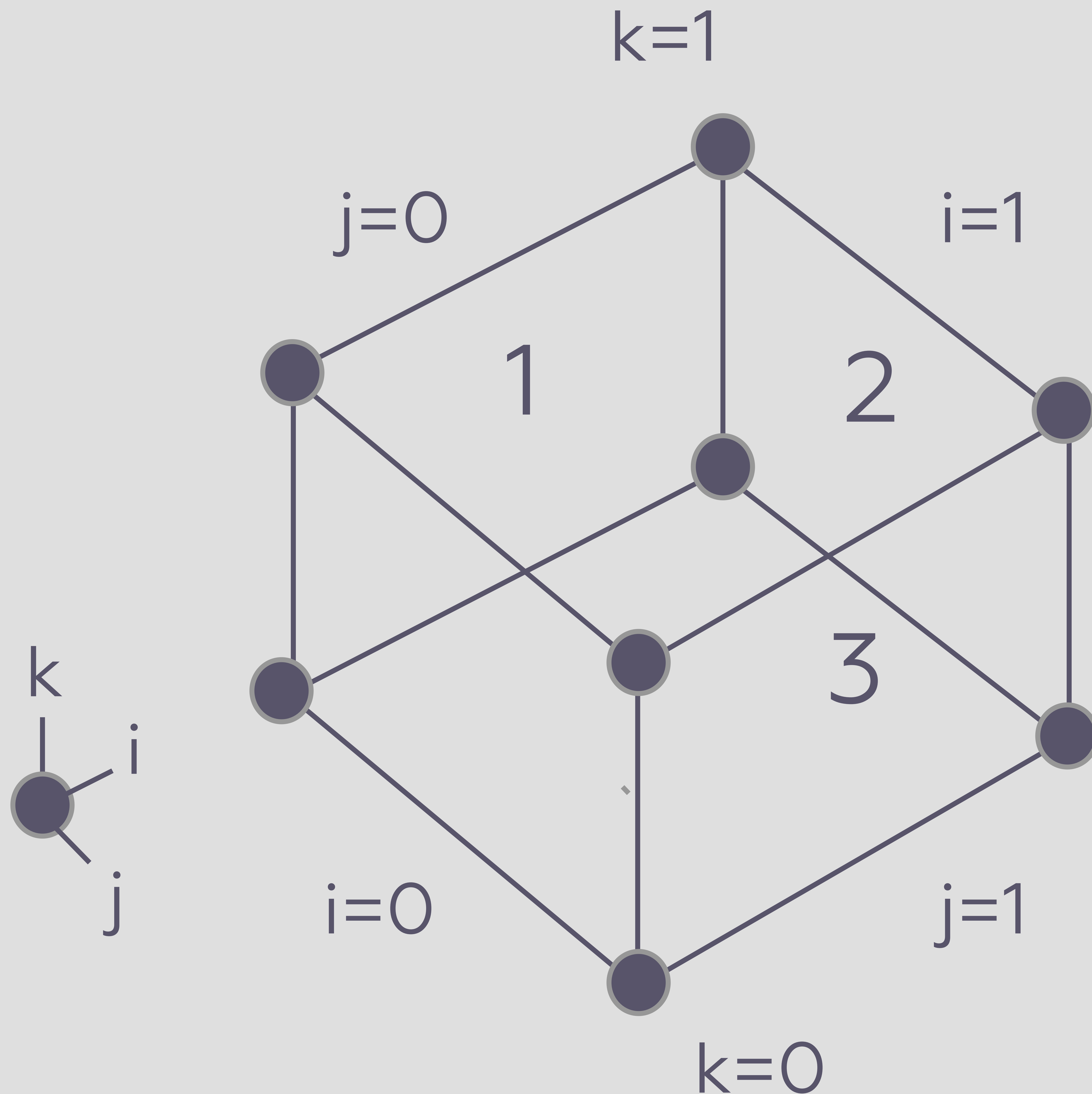
$$\text{transpFill}^i A \varphi u_0 = \text{transp}^j A(i/i \wedge j) (\varphi (i=0)) u_0 : A$$

$$\text{hfill}^i A [\varphi \rightarrow u] u_0 = \text{hcomp}^j A [\varphi \rightarrow u(i/i \wedge j), (i=0) u_0] u_0 : A$$

hcomp

$$A : U, (k = 0) \rightarrow (M : A)$$

use hcomp to fill the lids:
 $(k = 0, k = 1)$:

$$\text{hcomp } A^k [(j = 0) \rightarrow N1, \\ (i = 1) \rightarrow N2, \\ (j = 1) \rightarrow N3] M : A$$


Base Library Assurance

1. MLTT Internalization ✓
2. Topos Theory ✓
3. Tesseract
4. Category of Groupoids
5. Homological Algebra
6. Grothendieck Group

← we are here

Groupoid Infinity, 2021, INFOTECH SE, Ukraine, Kyiv

Anders 0.7.2

github.com/groupoid/anders

Thank You!

