Process Geometry for CT/HoTT: Triangles-in-Squares, Tetrahedra-in-Cubes, and dHoTT Movies

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

This paper develops a visual calculus for Category Theory and Homotopy Type Theory by mapping algebraic structure into small, drawable solids. We start from a triangle-in-square: objects S, T, Time; functors F:S->T, C:T->Time, G:S->Time; and a 2-cell alpha : C∘F ==> G. The square S -G-> Time <-C- T with diagonal F yields a process object P (strict: pullback S x_Time T; lax: comma (G ↓ C)), letting the diagonal factor through P via a certificate. Lifting to 3D, a tetrahedron-in-cube adds Resources R; faces encode feasibility/cost laws, while the interior holds a 3-cell coherence theta between rival 2-paths. In dynamic HoTT (dHoTT), we index these shapes by an interval I, turning processes into movies u |-> Process(u) with phase changes witnessed by higher homotopies. We supply a shape dictionary (monoids, monoidal functors, adjunctions, limits/colimits, distributive laws), a desk-ready “process kit” (cards, cube nets, flipbook strips), and two case studies (capacity-bounded scheduling; module action with cost monotonicity). The result is a compact, compositional vernacular: draw a face to state a law, fill a volume to certify coherence, and run a movie to track change.

Keywords: category theory, homotopy type theory, dHoTT, pullback, comma object, higher homotopy, cubical type theory, process geometry, triangle-in-square, tetrahedron-in-cube, coherence, associativity pentagon, adjunction, limits and colimits, distributive laws, scheduling, module action, visualization, pedagogy, Sigma type, Id type, witnesses. 56 pages. A collaboration with GPT-5.

I. Prelude: Why a geometric vernacular for abstract algebra

Abstract algebra in CT/HoTT is natively higher-dimensional: equalities between maps, witnesses between equalities, and coherences between witnesses. Yet most of our day-to-day reasoning fits on paper. This paper proposes a compact visual vernacular—"process geometry"—that lets you draw those higher facts as small shapes whose faces state laws and whose volumes certify coherences. The guiding idea is simple: a face is a law; a filler is a proof; a movie is a higher homotopy.

We start from a minimal pattern, the triangle-in-square. Data: objects S, T, Time; functors F:S->T, C:T->Time, G:S->Time; and a 2-cell alpha : C \circ F ==> G. The outer square S -G-> Time <-C- T plus the diagonal F yields a process object P with a universal property: strict P = S x_{Time} T (pullback), lax P = (G \downarrow C) (comma). In HoTT terms, the same shape is a dependent pair of choices and a witness:
Process_2D := Sigma s:S . Sigma t:T . Id_Time(C t , G s).
The diagonal factors through P via the certificate alpha, making the picture a working calculus for feasibility, bounds, and certification.

Lifting one dimension, a tetrahedron inside a cube adds a fourth vertex R (e.g., resources). The four triangular faces carry the 2D laws simultaneously; the interior encodes a 3-cell coherence theta relating the rival 2-paths that the faces generate. In dynamic HoTT (dHoTT), we index such solids by an interval I, making a path u |-> Process(u): a flipbook where frame-to-frame changes are governed by higher homotopies rather than ad hoc edits.

Why this helps: (1) Compositionality. Small shapes paste to make large arguments; universal properties guarantee canonical fillers. (2) Auditability. Each face/edge has a name, every filler corresponds to a concrete witness (Id term, naturality square, or inequality certificate). (3) Dual use. The same diagram can be read categorically (pullbacks, comma objects, Beck–Chevalley) or type-theoretically (Sigma/Id, transports, paths). (4) Pedagogy and practice. Paper cards, cube nets, and short "movies" let one demo coherence theorems at a desk while staying faithful to formal structure.

Scope and limits. We stay in copy-safe ASCII math; we separate strict vs lax readings (equality vs comparison morphisms) and note when Time is merely posetal. We do not attempt a full formalization here; instead we give a schema that is easy to encode in HoTT/cubical sketches and easy to animate. The remainder of the paper turns this schema into a small toolkit (2D, 3D, dHoTT), a dictionary of shapes for standard algebra, and two worked case studies.

II. Conventions and core schema

Goal. Fix a minimal, copy-safe ASCII language for drawing and reading "process geometry" so each picture has an unambiguous CT/HoTT meaning.

A) Ambient settings (two parallel readings)

1. Categorical (Cat/2-Cat):

- Objects: S (structure), T (tasks), Time (a category carrying cost/ordering).
- Functors: F:S->T, C:T->Time, G:S->Time.
- 2-cell (certificate): alpha : C \circ F => G (natural transformation).
- Strict vs lax:
    - Strict: C \circ F = G (on the nose).
    - Lax/inequality: in a posetal Time, take alpha_s : C(F(s)) <= G(s) or >=, with a chosen orientation (we use C(F(s)) <= G(s) unless stated otherwise).

2. Type-theoretic (HoTT/cubical sketch):

- Types: S, T, Time.
- Functions: F:S->T, C:T->Time, G:S->Time.
- Witness (2-path): alpha_s : Id_Time(C(F(s)), G(s)).
- Posetal variant: replace Id_Time(x,y) by a proposition Comp(x,y) such as x <= y.

B) Picture grammar (face = law, filler = proof, movie = homotopy)

- Nodes (vertices) name objects/types.
- Edges name functors/functions.
- Faces (2D regions) state equations/inequalities (laws).
- Filled faces carry their witness (equalities, naturality 2-cells, or comparisons).
- Volumes (3D cells) carry coherences between face-witnesses (3-cells).
- A movie (row of frames) is a path in the type of processes; inter-frame coherence is a higher path.

C) The base triangle-in-square schema

Data:

- S,T,Time; F:S->T; C:T->Time; G:S->Time; alpha : C \circ F => G.

Square view:

- Outer square: S -G-> Time <-C- T.
- Diagonal: F:S->T.
- Fourth corner (the “process object”) P with projections p1:P->S, p2:P->T.

Definitions:

- Strict case (pullback):
  P := S x_{Time} T, with p1,p2 s.t. G \circ p1 = C \circ p2.
- Lax case (comma):
  P := (G \downarrow C), objects are triples (s,t,w) with w: G(s) -> C(t) in Time.

Factorization/lift:

- There is a canonical phi:S->P given by
  phi(s) := (s, F(s), alpha_s).
- Projections satisfy p1 \circ phi = id_S and p2 \circ phi = F.

HoTT encoding (strict/equality reading):

- Process_2D := \Sigma s:S.\ \Sigma t:T.\ Id_Time(C(t), G(s)).
- Lift: phi(s) := (s, F(s), alpha_s).

D) Orientation and bounds (posetal Time)

- Default orientation: C(F(s)) <= G(s). Read as “computed cost is bounded by structural capacity.”
- Reversed orientation (if desired): G(s) <= C(F(s)) (“structural lower bound”).
- Either way, alpha_s is the witness inhabiting the hom-set of Time (a proposition in the posetal case).

E) Minimal existence assumptions

- For strict pullback P to exist in Cat: assume Time has the needed limits and the square is a pullback in your ambient 2-category (or work objectwise).
- For lax/comma P: always exists in Cat.
- In HoTT: Id types and \Sigma types are available; homotopy pullbacks give the strict reading automatically:
  hPullback(C,G) := \Sigma s:S.\ \Sigma t:T.\ Id_Time(C(t), G(s)).

F) Naturality and pasting

- If alpha is natural, then for any f:s->s' in S, the square
  C(F(s)) --C(F(f))--> C(F(s'))
  | alpha_s | alpha_s'
  v v
  G(s) ---------G(f)--------> G(s')
  commutes (strictly or laxly).
- Pasting rule: two triangle-in-square tiles that share an edge compose to a rectangle; their process objects compose via the universal property (pullback pasting or comma composition).

G) 3D extension placeholder (used in Sec. IV)

- Add a fourth object R (Resources). New functors alloc_from_s:S->R, demand_from_t:T->R.
- Faces: time agreement (C vs G), resource agreement (alloc vs demand), and two mixed faces.
- Interior 3-cell theta witnesses that the two composite 2-paths around the shell coincide up to a 3-path.

HoTT sketch for later reference:

- Process_3D :=
  \Sigma s:S.\ \Sigma t:T.\ \Sigma r:R.
  (w_time : Id_Time(C(t), G(s)))
  x (w_res : Id_R(alloc_from_s(s), demand_from_t(t)))
  x (theta : coherence(w_time, w_res)).

H) dHoTT (movie) convention (used in Sec. V)

- Parameter u in I := [0,1].
- A “movie” is u |-> (S_u,T_u,Time_u; F_u,C_u,G_u; alpha_u), packed as a path in the process type.
- Continuous deformation: alpha_u varies by a higher path; discontinuous “phase jump” frames carry an explicit higher witness that glues the two regimes.

I) ASCII notational rules

- Composition: (C \circ F)(s) = C(F(s)).
- Equality/witness: Id_X(x,y) for equalities in type X; use "<=" for posetal comparisons.
- Products and sums: \Pi for dependent products (rare here), \Sigma for dependent sums.

- No Unicode symbols; straight quotes only; function types as A->B; natural transformations with "=>".

J) Reading guide for desk use

- When you color a face, you have provided (or assumed) its witness.
- When you fill a volume, you have provided a coherence between face-witnesses.
- The diagonal edge always denotes a chosen “implementation” functor (e.g., F:S->T) that factors through the process object.

This fixed schema makes each later section mechanically checkable: Sec. III instantiates 2D P and phi; Sec. IV adds R and the interior 3-cell; Sec. V turns static solids into dHoTT movies; Sec. VI maps standard algebra to these shapes; Sec. VII packages the visuals as cards/nets/flipbooks; the case studies apply the same grammar verbatim.

III. 2D calculus: triangle-in-square (process as pullback/comma)

3.1 Data and picture
Given objects/types S, T, Time and maps

- F:S->T, C:T->Time, G:S->Time,
- witness alpha : C∘F => G (strict “=” or lax “<=”),
  draw the outer square S -G-> Time <-C- T and the diagonal F:S->T. The fourth corner P is “the process object.”

3.2 Strict reading (Cat/HoTT equality): pullback
Define

- P := S x_{Time} T, with projections p1:P->S, p2:P->T, s.t. G∘p1 = C∘p2.
  Universal property: for any Q with q1:Q->S, q2:Q->T and equality beta : C∘q2 = G∘q1, there exists a unique u:Q->P with p1∘u=q1, p2∘u=q2 and whose pullback equality is beta.

Diagonal factorization: phi:S->P by phi(s)=(s, F(s)) with the pullback equality instantiated by alpha_s.

3.3 HoTT equality encoding: homotopy pullback

- Process_2D := Sigma s:S. Sigma t:T. Id_Time(C(t), G(s)).
- p1(s,t,w)=s, p2(s,t,w)=t.
- phi(s) := (s, F(s), alpha_s).
  Equivalence: Process_2D is the standard homotopy pullback of C and G. The universal property above becomes the eliminator for Sigma with paths in Time.

3.4 Lax reading (posetal Time or 2-categorical): comma object
When Time is a poset/category of costs, treat alpha_s as a comparison rather than equality.

- P := (G ↓ C) with objects (s,t,w) where w: G(s) -> C(t) in Time (read “G(s) <= C(t)” if posetal).
- Projections p1,p2 are as before; phi(s)=(s, F(s), alpha_s : G(s)->C(F(s))).
  All comma objects exist in Cat; you avoid existence worries of strict pullbacks while retaining the same desk picture.

3.5 Reading guides

- Feasibility: a point of P is exactly a feasible pair (s,t) with a time-certificate w.
- Certification: alpha packages the certificate that the chosen diagonal F lifts into P.
- Bounds: choose orientation (default C(F(s)) <= G(s)). Flip if you prefer structural lower bounds.

3.6 Naturality and transport

If alpha is natural in s, then for f:s->s' the square

C(F(s)) --C(F(f))--> C(F(s'))

| alpha_s | alpha_s'

v v

G(s) ---------G(f)------> G(s')

commutes (strictly/laxly). In HoTT, naturality gives a transport law on Id_Time along G(f) and C(F(f)), ensuring phi is functorial wrt S-maps (when T,Time have the needed structure).

3.7 Small worked examples (desk-ready)

(A) Scheduling upper bounds

- S = machine topology; T = admissible schedules for a fixed job; Time = (R>=0, +, <=).
- C = makespan:T->Time; G = capacity-bound:S->Time (e.g., critical resource inverse throughput).
- Point of P: (s,t,w) with w : G(s) <= C(t) (or reversed).
- phi(s) = (s, F(s), alpha_s) says: “the schedule F(s) meets the structural bound.”

(B) Measurement with calibration

- S = calibrated instrument state; T = measurement protocols; Time = error budget poset.
- C = protocol-induced error; G = structural error floor.
- P collects all (instrument, protocol) pairs with a witnessed relation of floors vs observed error.

(C) Rewriting/confluence (diamond)

- T = terms; S = contexts; Time = rewrite-length poset.
- C = length-to-normal-form; G = contextual bound.

- P contains pairs (context, term) with a confluence-friendly certificate.

3.8 Pasting and Beck–Chevalley (why squares matter)
Two triangle-in-square tiles that share an edge paste to a rectangle; on the strict side this states a pullback-pasting lemma; on the lax side this is a comma pasting. A “Beck–Chevalley” condition appears when you swap “pull forward then cost” with “cost then pull back,” and a filled rectangle asserts the needed isomorphism/inequality. Practically: if you refactor either F or C, the commutation witness lets phi persist through the refactor without re-proving feasibility from scratch.

3.9 Computation rules (how to use P)

- Projection: (s,t,w) in P projects to s (structure) and t (task).
- Refinement: if w':C(t)->C(t') and monotone, postcompose to get (s,t', w'∘w) in P.
- Aggregation: if Time is monoidal and C is lax monoidal, witnesses add: w_total := w1 ⨁ w2.
- Pullback elimination (strict): to define h:P->X, give h(s,t,w) that respects p1,p2 equalities; in practice, define h on S and T and make it independent of the choice of w by functoriality.

3.10 Failure modes and what the picture tells you

- No lift (missing phi): your alpha is absent; diagonal does not land in P (implementation not certified).
- Inconsistent bounds: cannot produce w for given (s,t); the square is drawn but the face is unfilled.
- Non-cartesian pitfall: if your ambient category lacks pullbacks, move to the comma reading.
- Witness fragility: if alpha is not natural, lifts may fail to compose; mark the affected edge and treat it as local (do not paste globally).

3.11 Desk checklist (2D card)

- Label edges: F, C, G.
- Mark the face law you intend (equality or <=).
- Write P at the fourth corner; annotate “pullback/comma.”
- Draw phi and tag its witness alpha.
- If you paste two cards, explicitly copy the shared edge labels; only claim global coherence once both faces are filled.

Summary. The triangle-in-square turns “pick tasks; compute cost; compare to structure” into a universal object P that *is* the space of certified implementations, with a canonical factorization of your chosen diagonal. This is the atomic tile we will lift to 3D (Sec. IV) and then run as a movie in dHoTT (Sec. V).

IV. 3D calculus: tetrahedron-in-cube (add Resources R)

4.1 Data and picture
Add a fourth object R = Resources (e.g., memory/bandwidth/energy/inventory). Extend the 2D tile to a cube with four triangular faces and one interior tetrahedron.

Objects/types: S, T, Time, R
Maps:

- F : S->T
- C : T->Time, G : S->Time (time side)
- A : S->R, D : T->R (resource side)

Faces (2D laws):

- Time-face: alpha : C∘F => G (strict “=” or lax “<=”)
- Resource-face: rho : D∘F => A (strict “=” or lax “<=” in R)

- Mixed faces: naturality/monotonicity squares that relate (C,G) to (A,D) via a capacity functor cap : R->Time (monotone or functorial).

Interior (3D coherence): a 3-cell theta that says “the time certificate agrees with the capacity-of-resources certificate,” made precise below.

4.2 Strict reading (equalities): a limit-and-coherence object
First, package the two faces as pullbacks/comma objects:

- P_time := S x_{Time} T with witness w_time : Id_Time(C(t), G(s))
- P_res := S x_{R} T with witness w_res : Id_R(D(t), A(s))

Assume a functor cap : R->Time and its action on Id yields cap_*(w_res) : Id_Time(cap(D(t)), cap(A(s))).

Define the **process cube object** Q as the pullback over S×T of these two face-objects together with an interior coherence:

- Carrier (HoTT sketch):
- Process_3D :=
- Sigma s:S.
- Sigma t:T.
- w_time : Id_Time(C(t), G(s)) -- time-face witness
- x w_res : Id_R(D(t), A(s)) -- resource-face witness
- x theta : Id_{Id_Time}(
- w_time ,
- bridge(s,t,w_res)
-)

where bridge(s,t,w_res) is the 2-path in Time obtained by transporting along cap_*(w_res) and any fixed identification of G(s) with cap(A(s)) you adopt (e.g., a chosen isomorphism gamma_s : Id_Time(G(s), cap(A(s)))). Concretely,

bridge := (Id_Time C(t) = cap(D(t))) via cap_*(w_res)

;; (Id_Time cap(D(t)) = cap(A(s)))

;; (Id_Time cap(A(s)) = G(s)) via gamma_s^{-1}

and theta asserts the equality of 2-paths (a 3-path).

Universal property (informal): giving any cube that supplies both faces and a choice of gamma coherently determines (up to unique equality) a map into Process_3D.

4.3 Lax reading (posets/2-cats): comma + inequality coherence
If Time and R are posetal, replace equalities by comparisons:

- P_time := (G ↓ C) with w_time : C(t) <= G(s) (or oriented the other way).
- P_res := (A ↓ D) with w_res : A(s) <= D(t) (or oriented the other way).
- Monotone cap : R->Time yields cap_*(w_res) : cap(A(s)) <= cap(D(t)).

Interior theta becomes a **2-cell of inequalities** that compares the two routes in Time:

theta : C(t) <= G(s) and cap(A(s)) <= cap(D(t))

-> reconcile(w_time , cap_*(w_res) , gamma)

where gamma fixes a structural relationship between G(s) and cap(A(s)) (e.g., G(s) = cap(A(s)) or G(s) >= cap(A(s))). Orientation choices yield upper/lower bound variants; pick one convention and stick to it.

4.4 Two concrete patterns

(A) Capacity-bounded scheduling (GPU cluster)

- S = hardware topology; T = admissible schedules; R = resource vectors (cores, mem, bw); Time = (R>=0,+,<=).
- A(s) = available resources of s; D(t) = demanded resources of t.
- cap : R->Time maps resources to throughput-bound makespan.

- alpha : C∘F <= G is the “makespan <= structural bound” certificate.
- rho : D∘F <= A is the “demand <= available” certificate.
- theta witnesses that alpha matches cap_*(rho) after identifying G(s) with cap(A(s)).

(B) Module action with cost monotonicity

- S = module M with structure parameter s (e.g., decomposition); T = endomorphisms End(M); R = ring R (or resource proxy like op-count vectors); Time = complexity poset.
- A(s) encodes structural “budget” (e.g., sparsity pattern capacity); D(t) encodes operation demand; cap translates budgets to complexity bounds.
- The cube ensures that algebraic composition obeys both the time-face and resource-face and that these two faces cohere via theta.

4.5 Composition and pasting (3D)

- Face pasting: gluing two cubes along a shared face pastes their 3-cells; theta’s compose by whiskering and vertical composition in Id_{Id_Time}.
- Edge refactors: replacing C by C' with a natural isomorphism induces a transported theta' without re-proving resource-face facts.
- Projection/use: projecting Process_3D to P_time forgets resources; projecting to P_res forgets time. The interior theta lets you recover one certificate from the other via cap.

4.6 Computation rules (how to manipulate witnesses)

- Refinement in T: if t -> t' with monotone C and D, then w_time' := (C t <= C t') ;; w_time, w_res' := w_res ;; (D t <= D t').
- Refinement in S: if s -> s' with monotone G and A, transport both witnesses along G and A and update gamma accordingly.
- Aggregation: for sequential composition t1 * t2, if C and cap are monoidal then

w_time_total := w_time1 ⊕ w_time2, cap_*(w_res_total) := cap_*(w_res1) ⊕ cap_*(w_res2), and theta_total := theta1 ⊕ theta2.

4.7 Failure modes (what the cube tells you)

- Feasible-but-incoherent: both faces have witnesses, but no theta aligns them; time and resource stories disagree (investigate cap or gamma).
- Resource infeasible: cannot produce rho; the resource-face remains unfilled.
- Capacity model mismatch: gamma_s is wrong (G(s) not comparable to cap(A(s))); revise structural model or orientation.
- Non-functorial cap: without cap_*(w_res) you cannot bridge faces; either enrich R/Time or keep only a 2D account.

4.8 Desk model (how to draw/build)

- Draw a cube in perspective. Label front face the time-square (S -G-> Time <-C- T with diagonal F). Label a side face the resource-square (S -A-> R <-D- T with the same diagonal F).
- Inside, sketch a faint tetrahedron connecting the three "operational" vertices (S,T,R/Time) with the fourth being the **Process_3D** corner.
- On the time-face, write alpha; on the resource-face, write rho; inside the tetrahedron, write theta.
- If you use nets: print left/right cube nets; put identical edge labels on glued faces; keep the diagonal F consistent across both faces.

Summary. The 3D tetrahedron-in-cube records **two certified faces** (time and resources) and a **coherence** that binds them. Algebraically, it is a fibered combination of a time-constraint comma/pullback with a resource-constraint comma/pullback, plus a 3-path theta that equates the two induced time routes. Practically, theta is your guarantee that "the schedule's time certificate is exactly the capacity-of-resources certificate," so you can compute or verify either side and transport it across the cube.

V. 4D dynamics (dHoTT): movies of processes

5.1 Idea and setup

Static tiles (2D) and cubes (3D) become **movies** when indexed by an interval I = [0,1]. A movie is a path of processes:

- Data vary with u in I: S_u, T_u, Time_u, R_u; maps F_u, C_u, G_u, A_u, D_u; witnesses alpha_u, rho_u; coherence theta_u.
- dHoTT view: a movie is a dependent path
- Movie := Path(Process, u |-> Process(u))

where, e.g., for the 2D case

Process_2D(u) := Sigma s:S_u. Sigma t:T_u. Id_{Time_u}(C_u(t), G_u(s)).

For 3D:

Process_3D(u) :=

Sigma s:S_u. Sigma t:T_u.

w_time : Id_{Time_u}(C_u(t), G_u(s))

x w_res : Id_{R_u}(D_u(t), A_u(s))

x theta : coherence_u(w_time,w_res).

5.2 Cylinder and homotopy language

Categorically, use a **cylinder** functor along I:

D : Shape x I -> Cat

that assigns to each u a diagram and to each morphism in I (u->v) a natural transformation between diagrams. A movie is then a homotopy of diagrams; endpoints D(-,0), D(-,1) are the first and last frames.

5.3 Transport along time

Given a path p:u~>v in I, dHoTT provides transport maps

tr_S(p) : S_u -> S_v, tr_T(p) : T_u -> T_v, tr_Time(p) : Time_u -> Time_v, ...

and similarly for witnesses:

tr_Id(p) : Id_{Time_u}(x,y) -> Id_{Time_v}(tr_Time(p)(x), tr_Time(p)(y)).

Hence any point (s,t,w,...) in Process(u) transports canonically to a point in Process(v). This is the formal content of “frames glue coherently.”

5.4 Continuity vs phase changes
We distinguish two regimes:

- **Continuous drift:** all objects/maps/witnesses vary by paths; alpha_u and rho_u change by higher paths; theta_u changes by a 4D witness.
- **Phase jump (controlled discontinuity):** at u=u*, some map is replaced by an equivalent map via a specified iso/natural equivalence; the jump is legalized by inserting an explicit higher cell across a small collar [u*−epsilon,u*+epsilon]. Practically, one adds a short sub-movie that carries the equivalence data.

5.5 Operations on movies

- **Concatenation:** if Movie_1 ends where Movie_2 begins, reparameterize and paste: Movie := Movie_1 * Movie_2.
- **Reversal:** Movie^{-1} via u |-> 1-u; witnesses transport contravariantly.
- **Whiskering:** postcompose a movie with a natural iso of C or G (or cap) to obtain a new movie without re-proving face fillers; theta updates by whiskering in Id_{Id_Time}.
- **Projection:** forgetting resources yields a 2D movie; forgetting time yields a resource movie; coherence guarantees you can reconstruct one certificate from the other when cap is fixed.

5.6 Stability and invariants
Useful diagnostics over u:

- **Face coverage:** proportion of frames whose time-face and resource-face are filled.

- **Slack(u):** in posetal Time, define slack := minimal d with C_u(t) + d >= G_u(s). Nonincreasing slack indicates tightening bounds.
- **Certificate energy:** integrate a norm of witness change, e.g., the path-length of alpha_u in a chosen metric on Time; low energy suggests robust schedules or algebraic normal forms.

5.7 Example A (GPU scheduling under hardware drift)

- u parametrizes a rolling upgrade of memory bandwidth.
- R_u increases via A_u(s); cap_u:R_u->Time_u tightens the structural makespan bound.
- Movie supplies alpha_u : C_u(F_u(s)) <= G_u(s) and rho_u : D_u(F_u(s)) <= A_u(s).
- theta_u proves that the improved alpha_u coincides with cap_{u*}(rho_u) as bandwidth changes.
- A controlled phase jump swaps F_u to a new scheduler at u=u*; a short collar movie carries the natural iso between old/new C_u on admissible schedules.

5.8 Example B (module action as decomposition evolves)

- u refines a decomposition of M (sparsity pattern improves).
- A_u(s) reflects the refined budget; D_u(t) is unchanged; cap_u translates budgets to cost.
- alpha_u tightens from equality to inequality (or vice versa) depending on orientation; theta_u witnesses consistency between algebraic and resource narratives throughout the refinement.

5.9 Failure modes and remedies

- **Non-liftable jump:** diagonal F_u cannot be lifted through P_u at some frame (alpha_u missing). Remedy: localize the jump with a collar supplying a comparison morphism or relax to comma form.

- **Cap mismatch over time:** cap_u not functorial in u; theta_u cannot be updated. Remedy: replace cap_u by a functor into a fixed Time and add an endpoint isomorphism.
- **Witness drift without control:** alpha_u varies but not naturally; pasting fails. Remedy: record naturality-in-u explicitly as a higher naturality square.

5.10 Storyboard protocol (practical dHoTT on paper)

- **Keyframes:** draw 5–10 frames at u=0,0.1,...,1 with the same edge labels; color faces that are filled.
- **Witness log:** beneath each frame, record alpha_u, rho_u, and a short tag for theta_u.
- **Event markers:** mark phase jumps with a small collar strip showing the inserted higher cell.
- **Audit strip:** right margin lists invariants (face coverage, slack trend, certificate energy).

Summary. A dHoTT movie is a path in the type of processes that preserves the tile/cube grammar frame-by-frame. Transport along I turns handwritten flipbooks into bona fide higher homotopies, while collars make discrete redesigns lawful. This provides a disciplined way to track redesigns, upgrades, and proofs that evolve over time without losing coherence.

VI. Shape dictionary for algebra (from 2D to 4D)

6.1 Overview

We index common algebraic/categorical notions by small shapes. Edges are functors/maps, faces are laws (equalities or comparisons), filled faces carry witnesses, and volumes carry coherences (higher witnesses). HoTT encodings are given as Sigma/Id sketches; posetal/lax readings replace Id_X(x,y) with relations like x <= y.

6.2 Monoids and associativity (triangles, squares, pentagons)
Data: a monoid (A, *, e).

- Unit triangles (left/right): faces for e * x = x and x * e = x.
  HoTT: w_L(x): Id_A(*(e,x), x), w_R(x): Id_A(*(x,e), x).
- Associativity square/pentagon:
  - Square (local): *(*(x,y), z) = *(x, *(y,z)).
  - Pentagon (global coherence): the 5 associativity paths between fourfold products agree.
    Picture: draw two unit triangles sharing an edge for each unitor; draw a pentagon whose 5 edges are associativity rebracketings; its 2D filler is the Mac Lane coherence.
    HoTT: a 2-path assoc(x,y,z) : Id_A(*(*(x,y), z), *(x, *(y,z))); a 3-path filling enforces pentagon coherence among composites of assoc.

6.3 Monoidal categories (interchange squares; braiding hexagons)
Data: (C, ⊗, I) monoidal; optionally braided/symmetric.

- Interchange square: naturality of ⊗ on morphisms; face asserts (f1 ⊗ g1) ; (f2 ⊗ g2) = (f1 ; f2) ⊗ (g1 ; g2).
- Unit triangles: left/right unitors λ, ρ as filled faces.
- Associator pentagon: as in 6.2, but functorial.
- Braiding hexagon (braided case): two hexagonal faces witnessing compatibility of associator with braiding β.
  HoTT: witnesses are dependent Ids over object tuples; the hexagon interior is a 3-path equating the two composites of associator and braiding Ids.

6.4 Adjunctions (unit/counit triangles; snake identities as squares)
Data: F ⊣ U with unit η: id_C => U F and counit ε: F U => id_D.

- Two triangles: F --η--> U F and F U --ε--> id commute to give snake identities.
  Picture: draw two triangle-in-square tiles sharing the FU/UF edge; the

square fillers are the snake identities.
HoTT: for c:C and d:D, provide Ids
(1) U(ε_d) ; η_{U d} = id_{U d},
(2) ε_{F c} ; F(η_c) = id_{F c}.
These squares' fillers are the adjunction coherence.

6.5 Limits and colimits (pullback/pushout squares; pasting)

- Pullback square: face law is universal property; process object P = A x_C B.
- Pushout square: dual picture with comma/pushout object.
- Beck–Chevalley/pasting: two adjacent pullbacks/pushouts paste to a larger rectangle whose filler asserts preservation or exchange.
  HoTT: pullback as Sigma with Id, as in Sec. III; pasting corresponds to composing Sigma/Id eliminations.

6.6 Distributive laws (prisms with an interior 3-cell)
Data: two monoidal structures (+, 0) and (*, 1) with a distributive law.
Picture: a triangular prism; one axis for +, one for *, third for functoriality.

- Faces: left and right distributivity squares, unit compatibility squares.
- Interior: a 3-cell ensuring the two routes for distributing over triples agree.
  HoTT: Ids for each distributive face; a 3-path equates composite 2-paths.

6.7 Fibrations/opfibrations (cartesian/opfib squares; transport)
Data: p:E->B a fibration.

- Cartesian square: lift a base arrow f:b->b' to E with universal property. Face carries the cartesian-lift witness.
- Cleavage coherence: pasting two cartesian squares equals lifting along composite; interior 3-cell is the associativity of transport.
  HoTT: transport along paths in B; cartesian-ness is a universal property of lifts; 3-path witnesses functoriality of transport.

6.8 Sheaves/descent (Cech squares; gluing faces; higher coherence)
Data: cover {U_i -> X}, presheaf F.

- Overlap squares: F(U_i) and F(U_j) glued via F(U_ij); filled faces are matching/equalizer witnesses.
- Triple overlaps: prism/cube encoding cocycle condition; interior 3-cell enforces associativity of gluing.
  HoTT: matching family as Sigma over i with Ids on overlaps; 3-path fills the Cech 2-cocycle constraint.

6.9 Rewriting and 2-categories (critical pair diamonds; confluence)

- Diamond: two one-step rewrites from t converge to a common term; filled diamond is local confluence.
- Newmann's lemma (if terminating): paste diamonds along a strip to get global confluence rectangle.
  HoTT: Id terms for rewrites; 2-path filler for local diamond; higher path for strip composition.

6.10 Enriched/weighted structures (Time-enrichment)

- If hom-sets are weighted by Time (e.g., costs), every face becomes a comparison and every interior a compatibility inequality.
- Shapes remain the same; witnesses live in Hom_Time; pasting uses monoidal structure of Time (e.g., +).
  HoTT: replace Id with a proposition Comp(x,y); monoidal addition gives composition of witnesses.

6.11 Probabilistic/stochastic variants (Markov faces)

- Time becomes a stochastic order or [0,1] with >=; faces carry likelihood bounds or expectations.
- Interior 3-cell is a coupling/coherence between two probabilistic certifications.

HoTT: witnesses are evidence terms for inequalities on expectations; higher paths encode coupling equalities.

6.12 Higher operads/associahedra (polytope cells as coherences)

- Associahedron K_n: vertices = parenthesizations; edges = rotations; 2-faces = pentagons; higher cells = higher associativity.
- Draw K_4 (pentagon) as in 6.2; K_5 produces a 3D polytope; its volume is the 3-cell enforcing all pentagon compatibilities at once.
  HoTT: nested Ids along the polytope 1-skeleton; volume filler is a higher Id collapsing the entire polytope to a point in the space of proofs.

6.13 Adjacent calculus: profunctors and Kan extensions (mate squares)

- Mate correspondence: a square whose filler is the unit/counit “mate” transformation.
- Kan extension: comma square exhibits Lan_F or Ran_F as a universal process object; Beck–Chevalley square witnesses base change.
  HoTT: Sigma/Id encodings of universal properties; higher path for mate round-trips.

6.14 Reading guide (how to use the dictionary)

- Pick a shape that matches your law (triangle, square, prism, cube, pentagon/hexagon).
- Label edges with the functors/maps you actually use.
- Choose strict vs lax reading; write the witness type (Id_X or relation).
- Fill each face only when you have an explicit witness; do not imply interiors without listing which composite 2-paths they compare.
- For dynamics, replicate the same shape across frames and track which faces/volumes stay filled; add collar movies at refactors.

Summary. This dictionary compresses standard coherence theorems and constructions into a small repertoire of drawable shapes. Each entry has (i) a 2D

face law, (ii) an interior coherence when needed, and (iii) a HoTT sketch that turns pictures into Sigma/Id data. This keeps algebraic arguments legible on paper while staying faithful to CT/HoTT semantics and scalable to dHoTT movies.

VII. The Process Kit: desk-ready visual tools

7.1 What's in the kit

- Face cards (2D): printable tiles for triangles and squares with edge placeholders for F, C, G, A, D and a witness slot (alpha, rho, …).
- Cube nets (3D): left-handed and right-handed nets with a consistent spot for the diagonal F on the two key faces (Time-face and Resource-face).
- Flipbook strips (4D): a row of small frames to storyboard u ↦ Process(u) with "collar" mini-frames for phase jumps.
- Legend overlay: a minimal symbol key for equality vs inequality vs higher-cell tags.
- Audit sheets: checklists for pasting, sharing edges, and recording universal properties.

7.2 Face cards (2D)

- Size: 3×5 in (or A6) works well; wide margins for annotations.
- Square card template: corners labeled S, T, Time; edge labels G:S->Time, C:T->Time; diagonal slot F:S->T; fourth corner P.
- Witness box: "alpha : C∘F => G" (strict "=" or lax "<="). Fill the face only after you supply the witness.
- Pullback vs comma: tick one box [P = S x_Time T] or [P = (G ↓ C)].
- Pasting marks: draw small triangles on shared edges to enforce alignment when two cards are taped into a rectangle.

7.3 Cube nets (3D)

- Provide both handed nets; print on cardstock.
- Two designated faces:
  Face-T (time): S -G-> Time <-C- T with diagonal F.
  Face-R (resource): S -A-> R <-D- T with the same diagonal F.
- Interior tag: “theta : coherence(time-face, resource-face via cap:R->Time)”.
- Assembly rule: the edge labeled F must meet the corresponding F on the adjacent face; write alpha on Face-T, rho on Face-R, theta inside after assembly.
- B/W friendly marking:
  equality Id_X(x,y) = solid fill;
  inequality x <= y = diagonal hatch;
  higher cell (theta) = dotted fill.

7.4 Flipbook strips (dHoTT movies)

- Ten frames per strip (0.0,…,1.0). Same edge labels every frame.
- Under each frame: alpha_u, rho_u, theta_u short tags (e.g., “alpha_u := <= tight”, “theta_u := bridged via cap”).
- Phase jump collar: insert 2–3 micro-frames around u* with the equivalence/nat-iso you use to legalize the jump.

7.5 Legend (copy-safe ASCII)

- Equality witness: Id_X(x,y).
- Comparison witness (posetal): x <= y (arrow lives in Time or R).
- 2-cell (natural transformation): tau : F=>G.
- 3-cell (coherence of 2-cells): Theta : Id_{Id_X}(p, q).
- Process objects: P_2D = Sigma s:S. Sigma t:T. Id_Time(C t, G s).
  Process_3D = Sigma s,t. w_time × w_res × theta.

- Default orientation: C(F(s)) <= G(s); flip only if explicitly stated.

7.6 Validation checklists

A) 2D card ready?

- Edges labeled F, C, G.
- Orientation chosen (<= or =).
- P marked as pullback or comma.
- phi(s) = (s, F(s), alpha_s) noted.
- If pasting, shared edge text matches exactly.

B) 3D cube ready?

- Both faces filled (alpha on time-face, rho on resource-face).
- cap:R->Time specified.
- gamma_s (linking G(s) and cap(A(s))) chosen if needed.
- theta stated (what two 2-paths it equates).
- Handedness recorded (L or R) and diagonal F consistent across faces.

C) Movie ready?

- Frames numbered, same labels.
- Transport law recorded (what changes with u).
- Collars added at jumps.
- Invariants logged (face coverage, slack trend, certificate energy).

7.7 Workflow (desk and classroom)

- Build once: print a stack of blank cards/nets; keep a small legend card at hand.
- Work small to large: prove a face locally on a card, then paste; only after two faces are certified, assemble the cube and write theta.

- Record universals, not instances: for pullbacks/comma, write the universal property (factorization/lift) you actually use, not just the diagram.
- Film the proof: use the flipbook to capture refactors (changing C, switching F, updating cap) with collars, so later readers can replay the homotopy.

7.8 Extending the kit

- New shapes: pentagon (associator), hexagon (braiding), triangular prism (distributivity).
- Enriched Time: replace Id with Comp in a chosen enrichment (metrics, probabilities); keep the hatch pattern but annotate the enrichment (e.g., "[E[·]] <=" for expectation).
- Typed corners: allow S[T] annotations (e.g., S(M) = module at decomposition M) to clarify parameterized settings.

7.9 Safety rails (consistency rules)

- One meaning per edge: the same printed edge label must denote the same functor/function everywhere it appears in a pasted figure.
- Face first, volume second: never claim a 3-cell theta without writing exactly which two 2-paths it compares.
- Naturality explicitly: if you rely on naturality in s or t, draw the naturality square as a separate card and check it before pasting.
- Orientation discipline: do not mix <= and >= on the same face unless you display the comparison morphisms that convert one to the other.

Summary. The kit turns the paper grammar into a tangible method: face cards to state and certify laws, cube nets to bind dual certifications with a coherence, and flipbook strips to run lawful redesigns as movies. It is minimal, compositional, B/W friendly, and faithful to the CT/HoTT semantics fixed in earlier sections.

VIII. Case Study A: capacity-bounded scheduling on a GPU cluster

8.1 Problem setup (objects, maps, witnesses)

- Objects/types
  S := cluster topology/state (GPUs, memory, interconnect)
  T := admissible schedules for a fixed job DAG J
  R := resource vectors (cores, mem, bw, shared cache, …)
  Time := nonnegative reals with + and <= (read as makespan or cost)
- Functors/functions
  F : S->T (choose a concrete schedule/placement from structure)
  C : T->Time (analyzer: compute makespan of a schedule)
  G : S->Time (predictor: structural upper bound on makespan)
  A : S->R (available resources for a structure)
  D : T->R (demanded resources for a schedule)
  cap : R->Time (capacity model mapping resources to a time bound)
- Witnesses (faces)
  alpha_s : C(F(s)) <= G(s) (time face)
  rho_s : D(F(s)) <= A(s) (resource face)
- Interior coherence (volume)
  theta_s : Id_{Comp_Time}(alpha_s , bridge_s) where
  bridge_s := [C(F(s)) <= cap(D(F(s))) <= cap(A(s))] plus a chosen gamma_s : Id_Time(G(s) , cap(A(s))) or comparison G(s) >= cap(A(s)).
  Intuition: “the schedule’s time certificate equals (or refines to) the capacity-of-resources certificate.”

8.2 Building the 2D process object and lifting the diagonal

- 2D time tile: P_time := (G ↓ C) = Sigma s:S. Sigma t:T. (C(t) <= G(s)).
- Canonical lift of the diagonal F: phi_time(s) := (s, F(s), alpha_s).
- 2D resource tile: P_res := (A ↓ D) = Sigma s:S. Sigma t:T. (D(t) <= A(s)).
- Canonical lift: phi_res(s) := (s, F(s), rho_s).

8.3 The 3D process cube (strict sketch)
Process_3D :=
Sigma s:S. Sigma t:T.
w_time : Id_Time(C(t) , G(s)) (or comparison C(t) <= G(s))
x w_res : Id_R(D(t) , A(s)) (or comparison D(t) <= A(s))
x theta : coherence(w_time , cap_*(w_res) , gamma_s)

In the posetal/inequality reading (recommended here), replace Id_* by "<=" witnesses and theta equates two composite inequalities.

8.4 Concrete capacity model (cap)
Choose a simple conservative model; examples:

- max-rate bound: cap(cores, bw, mem) := max{ work/cores_rate , bytes/bw , footprint/mem_rate }
- critical-resource sum: cap(r) := sum_i work_i / rate_i(r)
- calibrated affine form: cap(r) := a0 + a1/r1 + a2/r2 + … with a_i >= 0

cap must be monotone in R and compatible with composition (for sequential tasks, cap(r1)+cap(r2) is admissible).

8.5 Schedulers (diagonal F)
F(s) returns a legal schedule: placement π of nodes of J onto devices/streams, plus an order that respects J's edges and device constraints. Typical policy families:

- list scheduling by longest path (LPT)
- heterogeneous earliest finish time (HEFT)
- bin-packing by memory footprint, then by compute
  All produce a T-object with demands D(F(s)) computed from per-node footprints and link transfers.

8.6 Certificates (how to get alpha and rho)

- rho_s : verify per-device and per-link resource constraints. For each device/link/channel, accumulate D(F(s)) and compare to A(s). rho_s is the conjunction of these comparisons.

- alpha_s : compute C(F(s)) as the critical-path length on the scheduled DAG with device/link latencies; compare to G(s). How to get G(s): use cap(A(s)) (and set gamma_s : G(s) = cap(A(s))) or keep G independent and relate them via a comparison gamma_s : G(s) >= cap(A(s)).

8.7 The interior coherence theta (how to build it)
Construct bridge_s as the composite inequality:
C(F(s)) <= cap(D(F(s))) <= cap(A(s))
The first step holds if cap upper-bounds the analyzer C on any schedule demand; the second is monotonicity of cap and rho_s. Choose gamma_s :

- strict identification: G(s) = cap(A(s)) (model equals predictor)
- or comparison: G(s) >= cap(A(s)) (predictor is looser than cap)
  Then define theta_s to assert that alpha_s matches bridge_s (up to gamma_s). In practice: store both numeric gaps and a reason code (e.g., “alpha via critical path”; “bridge via cap∘D and rho”).

8.8 Minimal example (desk numbers)
Let J have three kernels:

- k1: 4e12 flops, 8 GiB read; k2: 2e12 flops, 4 GiB read; k3: 2e12 flops, 4 GiB read; with edges k1->k2, k1->k3 (fan-out), k2 and k3 join via host copy (serialize at the end).
  Structure s: two GPUs on NVLink; each GPU: 10e12 flops/s, 16 GiB mem; NVLink: 300 GiB/s; host copy: 20 GiB/s.
  Choose F(s): place k1 on GPU0; k2 on GPU1; k3 on GPU0; pipeline edges over NVLink; final host copy after both complete.

Demands (simplified):

- compute: k1 0.4 s, k2 0.2 s, k3 0.2 s;
- NVLink: two copies of 4 GiB at 300 GiB/s ≈ 0.013 s each (overlappable);
- host copy: 8 GiB at 20 GiB/s = 0.4 s.
  Analyzer C(F(s)): critical path ≈ max{ k1 0.4 -> parallel k2/k3 0.2, host copy 0.4 } = 0.8 s (since host copy gates completion).

Resources A(s): per-GPU mem OK (<=16 GiB), flops OK (utilization <=1), NVLink OK (two 0.013 s transfers non-saturating). Thus rho_s holds.

Capacity model cap(r): max{ flops_work/total_flops , bytes_total/host_bw }
cap(A(s)) = max{ (4+2+2)e12 / (2*10e12) , 8 GiB / 20 GiB/s } = max{ 0.4 s , 0.4 s } = 0.4 s.
But analyzer gave 0.8 s because of serialization at the end (host copy cannot overlap fully). Update cap to include serialization factor:
cap'(r) := max{ flops/total_flops , host_bytes/host_bw } + overlap_penalty.
Take overlap_penalty = max(0, host_bytes/host_bw − parallelizable_window).
With a conservative window 0.4 s, we get cap'(A(s)) = 0.8 s.
Set G(s) := cap'(A(s)) = 0.8 s, and define
alpha_s : C(F(s)) = 0.8 <= 0.8 = G(s)
rho_s : D(F(s)) <= A(s) (checked per channel)
bridge_s : C(F(s)) <= cap'(D(F(s))) <= cap'(A(s)) = 0.8
theta_s : equates alpha_s with bridge_s (both 0.8).
Takeaway: theta_s forces the capacity model to encode the real non-overlap; once corrected, the faces and volume cohere.

8.9 Metrics and invariants (for audits)

- face_coverage := 1 if alpha_s and rho_s exist, else 0.
- slack_time := G(s) − C(F(s)) (>= 0 desirable).
- res_util := D(F(s)) / A(s) per channel (<= 1 desirable).
- theta_gap := | (numeric alpha side) − (numeric bridge side) | (aim for 0).
  Log these per s and per job; store reason codes for any nonzero theta_gap.

8.10 Failure modes and what the cube says

- alpha missing: analyzer finds C(F(s)) > G(s) with no model justification; either G is too tight or F is poor.
- rho missing: over-commit on some resource; schedule is infeasible.

- theta mismatch: faces hold separately, but cap does not reflect analyzer semantics (e.g., missed serialization, contention, NUMA). Fix cap or add gamma_s to reconcile.

8.11 How to use the cube operationally

- Design-time: propose F, compute D and C, propose cap, obtain alpha, rho, then tune cap until theta ≈ 0 (model learning).
- Run-time: monitor A(s) drift (e.g., throttling); use phi(s) = (s, F(s), alpha_s, rho_s, theta_s) to certify the current schedule; if any witness fails, trigger a collar movie to switch F or update cap.
- Post-mortem: paste cubes from adjacent runs; use pasting to justify that a refactor (new F or new C) preserved certification.

8.12 Desk checklist (apply Sec. VII kit)

- On Face-T: write C, G, F, alpha.
- On Face-R: write D, A, F, rho.
- Inside: write cap and theta; record gamma_s if G != cap(A).
- If you change F or C, add a flipbook collar with the natural iso or comparison you use; transport alpha, rho, theta across the collar.

Summary. The GPU scheduling cube encodes, in one glance, feasibility (rho), bounded makespan (alpha), and their coherence (theta via cap). The universal lift phi:S->Process_3D packages a certified implementation, and the invariants let you audit, tune, and lawfully evolve the schedule (Sec. V movies) while keeping algebraic and resource stories in lock-step.

IX. Case Study B: module action with cost monotonicity

9.1 Problem setup (objects, maps, witnesses)

- Objects/types
  S := structural presentation of a module M (e.g., chosen

basis/decomposition)
T := End(M) (admissible endomorphisms or finite words in them)
R := resource proxy (op-count vectors, nnz patterns, memory traffic)
Time := complexity/cost poset (e.g., N with + and <=, or R_{>=0} with + and <=)

- Functors/functions
  F : S->T (choose an implementation in End(M) from a structure)
  C : T->Time (cost of an endomorphism or word, e.g., op-count -> time)
  G : S->Time (structural capacity/upper bound from S)
  A : S->R (available algebraic/structural budget: bandwidth windows, cache tiling, sparsity)
  D : T->R (demand: ops, nnz touched, bytes moved)
  cap : R->Time (model mapping demands to cost; monotone and typically subadditive)

- Witnesses
  alpha_s : C(F(s)) <= G(s) (time face)
  rho_s : D(F(s)) <= A(s) (resource face)
  theta_s : coherence(alpha_s , cap_*(rho_s) , gamma_s) (interior 3-cell; gamma_s links G(s) with cap(A(s)))

9.2 Algebraic laws encoded by faces

- Monotonicity/subadditivity of cost: C(t1 * t2) <= C(t1) + C(t2).
- Action respects structure: if S refines (better basis/decomposition), then G(s') <= G(s) and A(s') >= A(s).
- Demand homomorphism: D(t1 * t2) <= D(t1) \oplus D(t2) in a suitable monoid on R.
  Each law is drawn as a square face; its witness is a 2-cell (equality or comparison).

9.3 Typical choices for the model

- M as an R-module with chosen basis; T words in a generating set {E_i} (endomorphisms).
- R = Z_{>=0}^k for k resource channels (multiplies, adds, loads, stores, cache lines, nnz); \oplus = componentwise +.
- cap(r) = a_0 + sum_i a_i * r_i with a_i >= 0 (affine upper bound), or a max-of-affines to capture bottlenecks.
- G(s) = cap(A(s)) (set gamma_s to an equality), or G(s) >= cap(A(s)) (gamma_s as comparison if G is a safety margin).

9.4 Worked example: sparse linear action
Let M = R^n with a sparsity pattern determined by s in S (block structure and ordering).

- T contains t = "apply B, then permute P, then triangular solve T_s" (word in End(M)).
- D(t) = (nnz(B), nnz(P), nnz(T_s), bytes_read, bytes_write).
- A(s) = channel budgets induced by cache/tiling of s: (L2_lines_per_ms, mem_bw, block reuse window, nnz per block).
- C(t) = measured/estimated time from op-counts (Roofline-style).
  Pick F(s) := "use block-order from s, apply B in blocked SpMV, P as in-place permutation, T_s by blocked forward-substitution."

Compute witnesses:

- rho_s : check each channel c that D_c(F(s)) <= A_c(s) (e.g., loads <= L2 window, bytes <= mem_bw * slot, nnz/block within reuse limits).
- alpha_s : compute C(F(s)) and compare to G(s) (if G(s)=cap(A(s)), alpha is immediate once rho and cap's monotonicity hold).

- theta_s : compare alpha_s to bridge_s := C(F(s)) <= cap(D(F(s))) <= cap(A(s)); set theta_s to identify alpha_s with bridge_s (or certify alpha_s as a refinement if G(s) > cap(A(s))).

9.5 Composition and reassociation (algebra ⇔ geometry)

- Algebraic composition t1 * t2 corresponds to pasting two time-faces horizontally; subadditivity witness yields the rectangular filler C(t1*t2) <= C(t1)+C(t2).
- Changing basis (s -> s') is a vertical face: transport A,G along the refinement; a collar movie (Sec. V) carries the natural isomorphism between old/new decompositions and updates alpha,rho,theta.

9.6 Diagnostics and invariants

- face_coverage := 1 if both alpha_s and rho_s exist.
- slack_time := G(s) − C(F(s)) (>= 0 desired).
- res_util_c := D_c(F(s)) / A_c(s) per channel (<= 1 desired).
- multiplicativity_gap := C(t1*t2) − (C(t1)+C(t2)) (should be <= 0 under subadditivity model).
- theta_gap := numeric difference between alpha side and cap-bridge side (aim for 0; nonzero implies cap mismatch).

9.7 Failure modes and remedies

- rho failure (resource infeasible): D(F(s)) exceeds A(s) on some channel. Remedy: change F(s) (reorder blocks), or refine s to improve A(s).
- alpha failure (time overrun): C(F(s)) > G(s). Remedy: either improve schedule (F), loosen G(s), or fix cap so that G(s)=cap(A(s)) reflects true bottlenecks.
- theta mismatch: cap underestimates some cost (e.g., permutation thrash). Remedy: enrich R with a new channel (e.g., TLB misses), update cap and A,D.

9.8 Minimal numeric sketch

Suppose A(s) = (mul=3e9/s, add=3e9/s, load=30e9 B/s, store=20e9 B/s).
For F(s): D(F(s)) = (mul=1.2e9, add=1.2e9, load=18e9 B, store=12e9 B).
cap(r) = max{ mul/3e9, add/3e9, load/30e9, store/20e9 }.
Then cap(D(F(s))) = max{0.4, 0.4, 0.6, 0.6} = 0.6 s; cap(A(s)) is a bound function of capacities -> choose G(s)=0.6 s.
Analyzer C(F(s)) from a microbenchmark predicts 0.62 s.

- rho_s holds (all demands within budgets).
- alpha_s fails if strict (0.62 <= 0.6 is false); choose lax: G(s) >= cap(A(s)) and set G(s)=0.65 s (safety margin). Now alpha_s : 0.62 <= 0.65.
- bridge_s : 0.62 <= cap(D(F(s)))=0.6 <= cap(A(s))<=0.65, so alpha_s refines bridge_s; theta_s notes refinement chain, theta_gap := 0.03.

9.9 Using the cube operationally

- Design: pick s to maximize slack_time while keeping res_util_c <= 1; F(s) should minimize C while respecting D<=A.
- Verification: fill Face-R first (rho), then Face-T (alpha), then interior theta via cap.
- Evolution: as s is refined (new basis), run a collar movie transporting witnesses; as t is factored (t = t2 * t1), paste faces and use subadditivity to maintain alpha.

9.10 Pedagogical read

- Algebra students see: unit/associativity laws as pasted faces; distributivity as a prism.
- Systems students see: R as counters, cap as a monotone functor, theta as “model-to-measurement” coherence.
- Both read the same cube: the geometry is a common language.

Summary. The module-action cube ties three views—algebra (End(M)), systems (resources), and analysis (cost)—into one object: a certified implementation with

coherent witnesses. Faces encode homomorphism/monotonicity; the interior enforces that time-as-measured equals time-as-modeled from resources, up to an explicit higher witness.

X. Implementation sketch (lightweight formalization)

10.1 Minimal HoTT core (strict/equality reading)
Types S, T, Time; functions F:S->T, C:T->Time, G:S->Time; witness alpha_s : \Id_{Time}(C(F(s)), G(s)).
Define the 2D process type:
Process_2D := \Sigma s:S.\ \Sigma t:T.\ \Id_{Time}(C(t), G(s)).
Projections: p1(s,t,w)=s, p2(s,t,w)=t. Canonical lift: \phi(s) := (s, F(s), alpha_s).
3D extension (with R, A:S->R, D:T->R, cap:R->Time, gamma_s relating G(s) and cap(A(s))):
Process_3D := \Sigma s:S.\ \Sigma t:T.\ (w_time:\Id_{Time}(C(t),G(s))) \times (w_res:\Id_{R}(D(t),A(s))) \times (theta:\Id_{\Id_{Time}}(w_time, bridge(s,t,w_res,gamma_s))).
Here bridge is the 2-path in Time induced by cap on w_res and gamma_s. Transport and pasting are by ordinary path operations.

10.2 Lax/posetal variant (inequalities instead of equalities)
Assume Time and R are posets/categories. Replace \Id_{X}(x,y) by Comp_X(x,y), e.g., x <= y.
Process_2D^{lax} := \Sigma s:S.\ \Sigma t:T.\ (C(t) <= G(s)).
Process_3D^{lax} := \Sigma s,t.\ (C(t) <= G(s)) \times (D(t) <= A(s)) \times theta, where theta certifies that the time-face certificate equals/refines the cap-composed resource certificate, given an orientation and gamma_s.

10.3 Naturality-in-s and functoriality
If alpha is natural, record for f:s->s' the square
C(F(s)) --C(F(f))--> C(F(s'))
| alpha_s | alpha_{s'}
v v
G(s) ---------G(f)--------> G(s')
as its own face card; in HoTT this is a dependent path over s with transport along f. Similar records exist for A, D, cap, and gamma.

10.4 Shape schema (serialization you can validate or render)
All-ASCII, no Unicode. One file per shape.

Header
type: "triangle_in_square" | "tetrahedron_in_cube" | "prism" | "pentagon" | ...
strictness: "strict" | "lax"
orientation: "C(F(s)) <= G(s)" (or "G(s) <= C(F(s))")

Objects
S: {name:"Structure"}
T: {name:"Tasks"}
Time: {name:"Time"}
R: {name:"Resources", optional:true}

Maps
F: {src:"S", dst:"T"}
C: {src:"T", dst:"Time"}
G: {src:"S", dst:"Time"}
A: {src:"S", dst:"R", optional:true}
D: {src:"T", dst:"R", optional:true}
cap: {src:"R", dst:"Time", optional:true}

Witnesses
alpha: {expr:"C \circ F => G", mode:"=", "<=", ">=", data:{...}}
rho: {expr:"D \circ F => A", mode:"=", "<=", ">=", data:{...}, optional:true}
gamma: {expr:"G ~ cap(A)", mode:"=", "<=", ">=", data:{...}, optional:true}

Interior
theta: {compares:["alpha","cap(rho)+gamma"], mode:"=", "refines", note:"..."}

Frame (for movies; see 10.7)
u: 0.0..1.0
deltas: {which maps/witnesses changed and how}
collar: true|false (marks a phase-jump micro-frame)

10.5 Validation rules (deterministic, B/W friendly)
V1 Edge consistency: every repeated label denotes the identical map type (same src/dst).
V2 Face completeness: a square is “filled” only if its witness is present and its mode matches strictness (strict requires "=").
V3 Lift check: in a triangle-in-square, accept a diagonal F only if alpha exists; then record \phi(s) as the canonical lift.
V4 3D coherence: in a tetrahedron-in-cube, accept theta only if alpha, rho, cap, and gamma are present and orientations allow composing to the bridge term.
V5 Monotonicity: when strictness="lax", require declared monotonicity of cap and of any composing map used to build bridges.
V6 Pasting integrity: when two shapes share an edge, the shared edge objects and map labels must match exactly; otherwise reject the paste.
V7 Movie continuity: between adjacent frames, either all changes are given as paths (continuous) or a collar is flagged with an explicit comparison/iso.

10.6 Computation hooks (optional semantics)
H1 Evaluators: plug-in functions to compute numeric C(t), G(s), D(t), A(s), and cap(r) for concrete case studies.
H2 Inequality checker: confirms C(F(s)) <= G(s) numerically (within tolerance) and constructs alpha with a reason code.
H3 Bridge builder: constructs cap(D(F(s))) <= cap(A(s)) from rho and cap monotonicity; logs each step.
H4 Theta comparator: compares the alpha route and the bridge route; yields theta with gap and provenance.

H5 Metrics: face_coverage, slack_time, res_util per channel, theta_gap; stored per frame.

10.7 Movie generator pipeline (dHoTT storyboard to frames)
Step 1 Parse schema for frame u=0; validate V1–V4; compute metrics (H5).
Step 2 For each planned edit (map change, orientation flip, cap update), decide: continuous drift (add a path note) or phase jump (insert collar micro-frames).
Step 3 For continuous drift, record deltas and re-run H1–H4; for collars, add explicit comparison/iso witnesses enabling transport across the jump.
Step 4 Emit N frames (SVG or PDF is fine) with identical edge labels; color/hatch faces that are filled; annotate alpha, rho, theta succinctly.
Step 5 Export a per-frame audit log (CSV): u, face_coverage, slack_time, res_util_*, theta_gap, reason_codes.
Step 6 Optional: render a flipbook GIF by stacking frames in order; embed the audit strip as a bottom band.

10.8 Minimal “strict vs lax” toggle (one code path, two readings)

- If strictness="strict": witnesses must use "="; P is read as homotopy pullback; theta as equality of 2-paths.
- If strictness="lax": witnesses use "<=" (or your chosen relation); P is read as a comma object; theta as a refinement certificate.
  The renderer changes only face fill pattern (solid for "=", hatched for "<=") and interior fill (dotted). The validator switches V2/V4 accordingly; H2/H4 compute inequalities rather than equalities.

10.9 Reproducibility and versioning

- Every schema file carries a version, a UUID, and a deterministic hash over (Objects, Maps, Witnesses, Interior, strictness, orientation).
- Movies include a manifest listing frames, their hashes, and the list of collars.
- A “paste ledger” records which shapes were glued along which edges, enabling third parties to reconstruct large arguments from small, checkable tiles.

Summary. This lightweight formalization keeps the pictures honest without demanding a full proof assistant: strict and lax readings are two modes on the same schema; validation is syntactic plus optional numeric hooks; movies are sequences of validated frames with collars for lawful jumps. It is enough to print cards, assemble cubes, and publish flipbook proofs while retaining clear semantics for CT/HoTT readers.

XI. Evaluation and pedagogy

11.1 What counts as “good” process geometry
We evaluate along four axes that mirror the paper’s goals:

- Correctness: faces and volumes correspond to valid CT/HoTT witnesses (strict “=” vs lax “<=” respected).
- Compositionality: small tiles paste into larger arguments without rework (universal properties carry through).
- Readability: a trained reader can recover the intended Sigma/Id (or comparison) content from the picture alone.
- Portability to code: a shape-schema instance validates (Sec. 10) and can drive numeric hooks in the case studies.

11.2 Instrumentation and metrics
Given a project (cards, cubes, movie), compute per-frame and aggregate metrics (Sec. 8–9):

- face_coverage in [0,1]: fraction of required faces with witnesses.
- slack_time := G(s) - C(F(s)) (strict reading expects 0; lax expects >= 0).
- res_util_* per channel: D(F(s))/A(s) (<= 1 is feasible).
- theta_gap: numeric difference between the “alpha route” and the “cap-bridge route.”
- paste_integrity: proportion of shared edges that match exactly (V1, V6).

- movie_continuity: proportion of changes explained by continuous drift vs collars (V7).
  These serve both as QA (engineering) and as rubrics (pedagogy).

11.3 Bench protocols (engineering)
E1 Strict vs lax toggle: render the same argument twice; verify that only witness modes and fills change, not structure or conclusions.
E2 Pasting stress test: glue 10–20 2D tiles into a lattice; run validator (V1–V6); record paste_integrity and number of re-label fixes needed.
E3 Refactor stability: apply a collar movie that swaps C for C' via a natural iso; confirm invariants (face_coverage, theta_gap) remain within tolerance.
E4 Model auditing (case studies): deliberately perturb cap; measure change in theta_gap; confirm that restoring a missing channel or constraint collapses theta_gap.

11.4 Learning protocols (pedagogy)
L1 Two-hour studio: students receive blank cards and a legend. Task: (i) draw a triangle-in-square for a chosen scenario, (ii) instantiate a comma P, (iii) factor the diagonal. Graded by face_coverage and a short prose extraction of Process_2D.
L2 Cube lab: teams assemble left/right cube nets; fill time- and resource-faces for a toy example; propose a cap and write the interior coherence. Rubric: theta explained as “which two 2-paths it compares.”
L3 Flipbook mini-project: storyboard a refactor (change F or C) using a collar; grade by movie_continuity and a one-paragraph account of transports.
L4 Oral “edge audit”: one teammate points to any repeated edge in a pasted diagram and explains its meaning; credit only if the meaning is identical across tiles (tests V1).

11.5 Reader extraction tasks (does the picture communicate?)
For each artifact, ask a fresh reader to:

- Write the Process_* Sigma/Id (or comparison) type underlying the figure.
- List which faces are strict and which are lax.

- Identify the universal property claimed (pullback/comma) and the factorization map (phi) if present.
- For cubes: name the two 2-paths the interior coherence compares. Score exactness (binary) and time-to-extraction (seconds); target: high exactness, sub-minute extraction.

11.6 Threats to validity and mitigations

- Ambiguous labels (edge name reused with a different meaning). Mitigation: V1/V6 checks; template edges pre-printed.
- Hidden assumptions (monotone cap, naturality of alpha) not shown on cards. Mitigation: require separate “assumption cards” pasted adjacent to faces they support.
- Over-fitting cap to measurements (theta_gap ~ 0 by construction). Mitigation: hold out test instances; report theta_gap on unseen workloads.
- Cognitive overload in large pastes. Mitigation: hierarchical pasting; collapse verified sub-diagrams into single “macro-cards.”

11.7 Accessibility and B/W constraints

- No reliance on color; use fill styles: solid “=”, hatched “<=”, dotted interior for coherences.
- Font-agnostic ASCII only (per Sec. 2 and 10); avoid Unicode drift.
- Provide left/right cube nets to avoid handedness confusion; print margins wide for annotations.

11.8 Classroom sequence (3×50 min)
Day 1: L1 studio (triangle-in-square) + quick shares.
Day 2: L2 cube lab (resources) + short quiz extracting Process_3D.
Day 3: L3 flipbook mini-project + plenary “edge audit.”
Assessment: combine rubric scores (face_coverage, paste_integrity, movie_continuity) with a brief written extraction of Sigma/Id content.

11.9 What success looks like

- Engineering: reproducible theta_gap ~ 0 on calibrated models; stable invariants across refactors via collars; high paste_integrity in large arguments.
- Pedagogy: near-100% face_coverage on student cards; median sub-minute extraction times; correct identification of phi and universal properties without prompting.

11.10 Limitations and future evaluation

- Non-cartesian settings (lack of pullbacks) push everything into comma form; visuals stay, but proofs may lengthen.
- Very high-dimensional coherences (beyond 3-cells) need polytopes (associahedra, permutohedra); we give only a sketch (Sec. 6).
- Empirical user studies (learning gains, retention) are proposed but not yet run; Section 12 positions related methods, and Section 13 outlines a plan to integrate proof assistants for automated checking.

Summary. The evaluation plan treats the visuals as checkable artifacts with clear semantics and measurable invariants. In the classroom, the same metrics become rubrics; in engineering, they become dashboards. Success is when small, honest tiles paste into large, honest arguments that a new reader can extract to Sigma/Id (or comparison) data quickly and correctly.

XII. Related work (pointer map)

12.1 String diagrams and monoidal coherence

- Mac Lane coherence (“pentagon,” “triangle”) underlies our face/volume grammar.
- Joyal–Street, Kelly–Laplaza: graphical calculi for monoidal/compact closed categories; our face fillers are the 2-cells those calculi certify.

- Selinger (“A survey of graphical languages for monoidal categories”): systematic treatment of rewriting on string diagrams; our “pasting + collars” mirrors diagram rewriting with explicit witnesses.
- Coecke–Kissinger (“Picturing Quantum Processes”): large-scale pedagogy via pictures; we adopt the same philosophy but broaden from symmetric monoidal structure to pullbacks/comma objects and higher coherences.

12.2 Higher categories and coherence polytopes

- Stasheff associahedra and Mac Lane’s pentagon supply canonical 2D/3D shapes for associativity; our dictionary (Sec. 6) treats these polytopes as ready-made volumes to fill.
- Leinster’s higher category texts and Baez–Stay on “physics, topology, logic and computation” motivate using small shapes as compositional program logic.
- Operads/PROPs (Loday–Vallette): multi-input coherence managed by polyhedral cells; we recast select operadic axioms as prisms/cubes with interior 3-cells.

12.3 Limits, comma, and Beck–Chevalley

- Pullbacks/pushouts as universal faces are standard; our triangle-in-square is the smallest “live” pullback/comma gadget amenable to desk use.
- Beck–Chevalley and base-change laws: our pasting rectangles with fillers are the picture-level manifestation.

12.4 Sheaves, descent, and gluing

- Mac Lane–Moerdijk (“Sheaves in Geometry and Logic”), Johnstone (“Sketches of an Elephant”): descent data as matching families; our Cech squares and triple-overlap prisms are the visual proxy for those equalizer/coequalizer axioms.

12.5 HoTT and cubical type theory

- HoTT Book (UF): Id-types as paths; higher paths as coherences; our Sigma/Id sketches are directly HoTT-readable.
- Cubical TT (Bezem–Coquand–Huber; Cohen–Coquand–Hüber–Mörtberg): intervals and composition/transport; our flipbook “movies” and collars correspond to cubical paths and homotopies with explicit composition structure.
- Higher inductive types: our “volumes as proofs” matches HIT constructors for glued spaces.

12.6 Applied category theory and pedagogy

- Spivak (“Category Theory for Scientists”), Fong–Spivak (“Seven Sketches”), Milewski’s notes: visual teaching of CT; our kit (cards, cube nets, flipbooks) is a minimal, hardware-store version aimed at coherence-heavy workflows.
- Diagrammatic rewriting (Lack, Kissinger, Dixon, et al.): confluence/termination methods; our “diamond” faces and movie collars give a lightweight, hand-checkable analog.

12.7 Scheduling, resources, and quantitative enrichment

- Enriched categories and Lawvere metrics motivate “Time as an object” with comparisons as morphisms; our lax reading (<=) is precisely the enriched view.
- Weighted limit/colimit ideas justify treating cap:R->Time as a functor and theta as compatibility of two weighted routes.

12.8 Proof assistants and lightweight formality

- Agda/Coq/Lean HoTT libraries, Cubical Agda: formal back-ends that can host our Sigma/Id encodings.
- Rewriting frameworks (Quantomatic, Globular, homotopy.io): rich precedents for interactive higher-dimensional diagram checking; our JSON schema (Sec. 10) is a lean front-end that could export to these tools.

12.9 Positioning

- We do not replace string-diagram calculi; we supply a *process-geometry overlay* emphasizing universal properties (pullback/comma) and higher coherences (3-cells) with dHoTT movies for evolution over time. The emphasis is on *hand-held auditability*: face = law, volume = coherence, movie = lawful change.

XIII. Discussion and outlook

13.1 What this buys you (and why now)
Process geometry compresses higher algebra into small, drawable solids with explicit witnesses. It sits between fully formal proof assistants and informal whiteboard sketches: faces are laws with named evidence; volumes are coherences; movies are lawful redesigns. This "middle layer" is timely for CT/HoTT pedagogy, systems design, and iterative research notes.

13.2 Risks and scope limits

- Non-cartesian settings: when strict pullbacks fail, default to comma; proofs lengthen but pictures remain stable.
- Overfitting cap: theta can be gamed by tuning models; require held-out audits and reason codes.
- Proof debt: pictures without witnesses are placeholders; the kit enforces "face first, volume second," but discipline is still human-driven.

13.3 Open problems (technical)

- Automated fillers: given edge labels and partial faces, synthesize witnesses (alpha, rho) or certify non-existence.
- Minimal witnesses: compute "shortest" theta (e.g., fewest comparison steps) and canonical representatives up to higher paths.
- Pasting normal forms: confluence/termination for large tiled arguments; diamond criteria for our square calculus.

- Enriched Time: extend validation to metrics/probabilities (Lawvere metric spaces; expectation order), with composition laws for witnesses.
- Dynamic collars: canonical construction of micro-movies for common refactors (map replacement by a natural iso; orientation flips).

13.4 Integration with formal tools

- Export: Sigma/Id sketches and comma data to HoTT/Cubical Agda/Lean; import reduced witnesses back into the schema.
- Diagrammatic assistants: bridge to Globular/Quantomatic; treat our shapes as front-end "tiles," back-end as proof search/check.
- Typed rendering: render validated tiles to SVG/PDF with machine-readable witness IDs (hashes) for immutable citation.

13.5 Roadmap (library + benchmarks)

- v0.1: reference schema + validator (Sec. 10), printables (Sec. 14), two worked bundles (Secs. 8–9).
- v0.2: movie collars and invariant dashboards (slack_time, theta_gap).
- v0.3: enriched Time modules (metric/probabilistic), distributivity prisms, braiding hexagons.
- Bench set: (i) GPU scheduling cubes; (ii) module-action cubes; (iii) adjunction tiles; (iv) Cech gluing prisms. Release both pictures and witness logs.

13.6 Pedagogy research plan

- Pre/post tests on extraction tasks (write the Sigma/Id type from a picture).
- Retention after 1–2 weeks; transfer to new shapes (e.g., from pullbacks to adjunction triangles).
- Studio ethnography: how teams enforce edge consistency (V1/V6) and narrate theta.

13.7 Outlook

We expect “process geometry” to become a small but durable vernacular: a set of audited figures that can be pasted, filmed, and formalized. The long-term bet is two-way traffic: sketches that are easy to formalize, and formal proofs that render back into one-page, witness-rich pictures for humans.

XIV. Appendices

A. Printable card set and cube nets (left/right)

A.1 Square face card (triangle-in-square)

- Corners: top-left S, top-right Time, bottom-right T, bottom-left P.
- Edges: S->Time (G), T->Time (C).
- Diagonal: S->T (F).
- Witness box: “alpha : C∘F => G” with mode “=” or “<=”.
- Tick: [P = S x_Time T] or [P = (G ↓ C)].
- Margin lines for “phi(s) := (s, F(s), alpha_s)”.

A.2 Cube net (two faces pre-labeled)

- Face-T (time): S-G->Time, T-C->Time, diagonal F:S->T.
- Face-R (resource): S-A->R, T-D->R, diagonal F:S->T.
- Interior tag (after assembly): “theta : coherence(alpha, cap∘rho, gamma)”.
- Provide both handed nets; print on cardstock; hatch pattern for “<=”, solid fill for “=”, dotted interior for theta.

A.3 Flipbook strip

- Ten small squares in a row (u=0.0,…,1.0). Same labels each frame.
- Under each frame: “alpha_u, rho_u, theta_u” short notes.
- Insert 2–3 micro-frames (collar) around any phase jump.

B. Legend and style guide (copy-safe ASCII)

B.1 Symbols

- Equality witness: Id_X(x,y).
- Comparison (poset): x <= y (lives in Hom_Time or Hom_R).
- Natural transformation / 2-cell: tau : F=>G.
- 3-cell (coherence of 2-cells): Theta : Id_{Id_X}(p, q).
- Products/sums: Sigma, Pi (write as \Sigma, \Pi in text).
- Composition: C∘F written “C(F(s))” in running text.

B.2 Fill styles (B/W)

- “=” solid face; “<=” hatched face; theta dotted interior.
- Edge labels must match identically wherever pasted; one meaning per edge.

C. Worked proofs for Secs. III–IV (expanded sketches)

C.1 Universal property (pullback reading)
Given C:T->Time, G:S->Time. Define
Process_2D := \Sigma s:S.\ \Sigma t:T.\ \Id_{Time}(C(t), G(s)).
Projections p1(s,t,w)=s, p2(s,t,w)=t satisfy G∘p1 = C∘p2 by w.
Universal mapping: for any Q with q1:Q->S, q2:Q->T and beta: \Id_{Time}(C∘q2 , G∘q1), define
u(q) := (q1(q), q2(q), beta(q)). Then p1∘u=q1, p2∘u=q2, and u is unique by \Sigma/Id elimination. Hence Process_2D is the homotopy pullback.

C.2 Comma reading (lax)
Process_2D^{lax} := \Sigma s,t.\ (C(t) <= G(s)).
The same universal construction holds with beta a comparison; composition stability uses monotonicity of Time.

C.3 Bridge and theta (3D)
With A:S->R, D:T->R, cap:R->Time, and gamma_s relating G(s) and cap(A(s)), define bridge(s,t,w_res) as the composite in Time obtained by applying cap to w_res and

composing with gamma_s.
Process_3D := \Sigma s,t.\ w_time \times w_res \times theta,
where theta : \Id_{Id_{Time}}(w_time , bridge(s,t,w_res)).
In the lax case, theta certifies refinement/equality of two comparison chains.

D. JSON schema examples (ASCII, minimal)

D.1 2D strict (pullback)


```
{
"type": "triangle_in_square",
"strictness": "strict",
"orientation": "C(F(s)) = G(s)",
"objects": {"S":{}, "T":{}, "Time":{}},
"maps": {"F":{"src":"S","dst":"T"},
"C":{"src":"T","dst":"Time"},
"G":{"src":"S","dst":"Time"}},
"witnesses": {
"alpha": {"expr":"C \circ F => G", "mode":"="}
},
"interior": {},
"notes": {"phi":"phi(s)=(s,F(s),alpha_s)"}
}
```


D.2 3D lax (comma + coherence)


```
{
"type": "tetrahedron_in_cube",
"strictness": "lax",
"orientation": "C(F(s)) <= G(s)",
"objects": {"S":{}, "T":{}, "Time":{}, "R":{}},
"maps": {"F":{"src":"S","dst":"T"},
"C":{"src":"T","dst":"Time"},
"G":{"src":"S","dst":"Time"},
"A":{"src":"S","dst":"R"},
```

"D":{"src":"T","dst":"R"},
"cap":{"src":"R","dst":"Time"}},
"witnesses": {
"alpha":{"expr":"C \circ F => G","mode":"<="},
"rho":{"expr":"D \circ F => A","mode":"<="},
"gamma":{"expr":"G ~ cap(A)","mode":">="}
},
"interior": {
"theta":{"compares":["alpha","cap(rho)+gamma"],"mode":"refines","note":"bridg
e via cap monotone"}
}
}

References

[1] Mac Lane, S. *Categories for the Working Mathematician* (2nd ed.). Springer, 1998.

[2] Mac Lane, S. “Natural associativity and commutativity.” *Rice Univ. Studies* 49 (1963), 28–46.

[3] Stasheff, J. “Homotopy associativity of H-spaces I, II.” *Trans. Amer. Math. Soc.* 108 (1963), 275–312.

[4] Joyal, A.; Street, R. “The geometry of tensor calculus I.” *Adv. Math.* 88 (1991), 55–112.

[5] Kelly, G. M.; Laplaza, M. L. “Coherence for compact closed categories.” *J. Pure Appl. Algebra* 19 (1980), 193–213.

[6] Selinger, P. “A survey of graphical languages for monoidal categories.” In *New Structures for Physics* (Lecture Notes in Physics 813), Springer, 2011, 289–355.

[7] Coecke, B.; Kissinger, A. *Picturing Quantum Processes.* Cambridge Univ. Press, 2017.

[8] Leinster, T. *Higher Operads, Higher Categories.* Cambridge Univ. Press, 2004.

[9] Baez, J.; Stay, M. “Physics, topology, logic and computation: a Rosetta Stone.” In *New Structures for Physics* (LNP 813), Springer, 2011, 95–172.

[10] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* Institute for Advanced Study, 2013.

[11] Bezem, M.; Coquand, T.; Huber, S. “A model of type theory in cubical sets.” *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, 2014.

[12] Cohen, C.; Coquand, T.; Huber, S.; Mörtberg, A. “Cubical Type Theory: A constructive interpretation of the univalence axiom.” *Logical Methods in Computer Science* 14:4 (2018).

[13] Lurie, J. *Higher Topos Theory.* Princeton Univ. Press, 2009.

[14] Mac Lane, S.; Moerdijk, I. *Sheaves in Geometry and Logic.* Springer, 1992.

[15] Johnstone, P. T. *Sketches of an Elephant: A Topos Theory Compendium.* Oxford Univ. Press, 2002.

[16] Lawvere, F. W. "Metric spaces, generalized logic, and closed categories." *Rend. Sem. Mat. Fis. Milano* 43 (1973), 135–166 (1974).

[17] Spivak, D. I. *Category Theory for the Sciences.* MIT Press, 2014.

[18] Fong, B.; Spivak, D. I. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory.* Cambridge Univ. Press, 2019.

[19] Loday, J.-L.; Vallette, B. *Algebraic Operads.* Springer, 2012.

[20] Lack, S. "A 2-categories companion." In *Towards Higher Categories*, Springer, 2010, 105–191. (Background on bicategorical pasting and coherence.)

[21] Street, R. "Fibrations in bicategories." *Cahiers de Topologie et Géométrie Différentielle* 21 (1980), 111–160. (Beck–Chevalley/base change perspective.)

[22] Dixon, L.; Kissinger, A.; Merry, R. "Quantomatic: A proof assistant for diagrammatic reasoning." *ITP 2010 Workshop on User Interfaces for Theorem Provers*, 2010. (Tool reference.)

[23] Bar, K.; Heunen, C.; Kissinger, A.; Vicary, J. "Globular: an online proof assistant for higher-dimensional rewriting." *arXiv:1601.08105*, 2016. (Tool reference.)

[24] Homotopy.io team. "homotopy.io: interactive proofs in higher categories." Project note, 2016–. (Tool reference.)

[25] Youvan, D. C. "Going Higher Than Voevodsky in Category Theory (CT) and Homotopy Type Theory (HoTT) — 'Hotter than HoTT'." ResearchGate, Sept 2025. DOI: 10.13140/RG.2.2.22756.64644.

[26] Youvan, D. C. "Twin Cubes — Executable Storyboards and Physical Manipulatives for CT/HoTT (including source code)." ResearchGate, Aug 2025. DOI: 10.13140/RG.2.2.28641.77920.

[27] Selinger, P. “Functoriality of graphical calculi and coherence.” *ENTCS* 143 (2006), 99–122. (Further background on diagram rewriting.)

[28] Carboni, A.; Kelly, G. M.; Walters, R. F. C.; Wood, R. J. “A 2-categorical approach to change of base and geometric morphisms.” *Cahiers de Topologie et Géométrie Différentielle Catégoriques* 32 (1991), 47–95. (Beck–Chevalley and base change.)

(Items [20]–[24], [27]–[28] serve as anchors for pasting laws, bicategorical base change, and tool ecosystems referenced in the text.)

Process Geometry for CT/HoTT