```
{
  "meta": {
    "title": "R-Theory Canonical Compendium",
    "subtitle": "Closed Relational Framework for Projection, Calibration, and Admissibility",
    "edition": "Version 1.0.1— Full Canonical Structure",
    "author": "Kit Cosby",
    "compiled_by": "ChatGPT",
    "language": "en-struct",
    "license": "CUSTOM-NA-SA (no attribution required, Share and Share alike)",
    "date_compiled": "2026-01-08T00:00:00Z",
    "canonical_hash": {
      "algorithm": "SHA3-512",
      "value":
"39c56a9c21f493baf7ef09d4d3ac7a9db5f1ce6a6a3d4975bdbb8423c3180f6b5eb82d0c37e24e5
0b4d3d6cfafdf69b1a9f017bde9de474a88b3f0d15d0f48a1"
    },
    "build_environment": "AugInt-Stable v2."
  },
  "canonical": {
    "slots": {
      "structural_layer": ["A", "B", "C", "D", "E", "F", "G", "H"],
      "relational_layer": ["I", "J", "K", "L", "M"],
      "projection_layer": ["N", "O", "P", "Q", "R", "S", "T", "U"],
      "calibration_layer": ["V", "W"],
      "governance_appendices": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
"O", "P", "Q", "R", "S", "T", "U", "V", "W"]
    }
  },
  "primitive_index": {
    "A": "Admissibility Predicate",
    "B": "Manifold Without Metric",
    "C": "Ordering & Reachability",
    "D": "Projection Operator",
    "E": "Negative Invariance Filter",
    "F": "Accounting Integral",
    "G": "Transformation Closure",
    "H": "Structural Closure",
    "I": "Relational Separation",
    "J": "Phase State",
    "K": "Curvature Operator",
    "L": "Attenuation Mapping",
    "M": "Accounting Without Conservation",
    "N": "Measurement Operator",
    "O": "Projection Equivalence",
    "P": "Re-Entry Discipline",
    "Q": "Vector Tuple Grammar",
    "R": "One-Dimensional Mechanics",
    "S": "Circulation Grammar",
    "T": "Calibration Scales",
    "U": "Propagation Mapping",
    "V": "Discrete Cycle / Hydrogen Bridge",
    "W": "Terminal Admissibility Rule"
  },
  "lineage": "Derived from binary sequence (A–W); cross-checked with dependency graph (Part I
§3)",
  "governance": {
    "admissibility_ruleset": {
      "name": "fivefold_predicate",
      "elements": [
        "boundedness",
        "no_new_primitives",
        "mirror_accessibility",
        "residual_reporting",
        "projection_declaration"
      ]
    },
    "closure_ceiling": "L3",
    "projection_freedom": true,
    "ontology_allowed": false,
    "residual_reporting_required": true,
    "calibration_anchors": ["hydrogen_21cm", "cesium_133", "planetary_cycles"],
    "ethical_guidelines": [
      "Declare all projections explicitly.",
      "Decompose all constants into calibration ratios.",
      "Report residuals as first-class results.",
      "Introduce no new primitives.",
      "Maintain transparency and accessibility."
    ]
  },
  "canonical_math": {
    "angle_convention": {
      "preferred_unit": "degrees",
      "interchangeability_disclaimer": "Degrees are canonical. Any radian-form closures (e.g.,
2*pi) are treated as shorthand for 360° closure. This does not introduce a metric, scale, or unit
ontology."
    },
    "domain": {
      "x_symbol": "x",
  "x_units": "degrees",
  "x_range": { "type": "half_open_interval", "start": "0", "end": "360" },
  "singular_sets": [
    "sin(x)=0",
    "cos(x)=0",
    "abs(cxp(x)*srx(x)-1)=0"
  ]
},
"operators": {
  "srx": {
    "name": "sininverx",
    "expr": "|csc(x)| + cot(x)",
    "display": "srx(x) = |csc(x)| + cot(x)"
  },
  "sxp": {
    "name": "sinexpon",
    "expr": "1/(|csc(x)| + cot(x))",
    "display": "sxp(x) = 1/(|csc(x)| + cot(x))"
  },
  "cxp": {
    "name": "cosinexpon",
    "expr": "|sec(x)| + tan(x)",
    "display": "cxp(x) = |sec(x)| + tan(x)"
  },
  "crx": {
    "name": "cosinverx",
    "expr": "1/(|sec(x)| + tan(x))",
    "display": "crx(x) = 1/(|sec(x)| + tan(x))"
  }
},
"composites": {
  "urx": {
    "name": "unainverex",
    "expr": "srx(x) - crx(x)",
    "display": "urx(x) = srx(x) - crx(x)"
  },
  "uxp": {
    "name": "unaexpon",
    "expr": "cxp(x) - sxp(x)",
    "display": "uxp(x) = cxp(x) - sxp(x)"
  },
  "sawdown": {
    "name": "sawdown",
    "expr": "srx(x) / (cxp(x)*srx(x) - 1)",
    "display": "sawdown(x) = srx(x) / (cxp(x)*srx(x) - 1)"
  },
  "sawup": {
    "name": "sawup",
    "expr": "cxp(x) / (cxp(x)*srx(x) - 1)",
    "display": "sawup(x) = cxp(x) / (cxp(x)*srx(x) - 1)"
  },
  "flatwave": {
    "name": "flatwave",
    "expr": "(srx(x) + cxp(x)) / (cxp(x)*srx(x) - 1)",
    "display": "flatwave(x) = (srx(x)+cxp(x)) / (cxp(x)*srx(x) - 1)"
  }
},
"identities": [
  {
    "id": "flatwave_decomposition",
    "lhs": {
      "name": "flatwave",
      "expr": "(srx(x) + cxp(x)) / (cxp(x)*srx(x) - 1)",
      "display": "flatwave(x)"
    },
    "rhs": {
      "name": "flatwave_as_reciprocals",
      "expr": "1/urx(x) + 1/uxp(x)",
      "display": "1/urx(x) + 1/uxp(x)"
    },
    "notes": "Flatwave equals sum of the two sawtooth reciprocals."
  },
  {
    "id": "sawup_identity",
    "lhs": { "name": "sawup", "expr": "sawup(x)", "display": "sawup(x)" },
    "rhs": { "name": "inv_urx", "expr": "1/urx(x)", "display": "1/urx(x)" },
    "notes": "sawup(x) == 1/urx(x)"
  },
  {
    "id": "sawdown_identity",
    "lhs": { "name": "sawdown", "expr": "sawdown(x)", "display": "sawdown(x)" },
    "rhs": { "name": "inv_uxp", "expr": "1/uxp(x)", "display": "1/uxp(x)" },
    "notes": "sawdown(x) == 1/uxp(x) (per your stated uxp orientation)"
  },
  {
    "id": "sin2x_over_4_identity",
    "lhs": {
      "name": "sin2x_over_4",
          "expr": "sin(2*x)/4",
          "display": "sin(2x)/4"
        },
        "rhs": {
          "name": "inv_urx_plus_uxp",
          "expr": "1/(urx(x) + uxp(x))",
          "display": "1/(urx(x)+uxp(x))"
        },
        "notes": "Identity proven in another file: sin(2x)/4 = 1/(urx(x)+uxp(x)). Keep s canonical
claim."
      }
    ],
    "phase_wheel": {
      "encoding": {
        "decay_symbol": "O",
        "growth_symbol": "I",
        "manifest_symbol": "I",
        "mirror_symbol": "O",
        "phase_symbol": "I",
        "cophase_symbol": "O"
      },
      "octants": [
        { "deg_start": 0, "deg_end": 45, "dominant": "srx", "recessive": "cxp", "phase_sign": "+",
"description": "Decay phase manifest", "code": "OII" },
        { "deg_start": 45, "deg_end": 90, "dominant": "cxp", "recessive": "srx", "phase_sign": "+",
"description": "GROWTH cophase MANIFEST", "code": "IOI" },
        { "deg_start": 90, "deg_end": 135, "dominant": "crx", "recessive": "sxp", "phase_sign": "-",
"description": "DECAY cophase MIRROR", "code": "OOO" },
        { "deg_start": 135, "deg_end": 180, "dominant": "sxp", "recessive": "crx", "phase_sign": "-",
"description": "GROWTH PHASE MIRROR", "code": "IIO" },
        { "deg_start": 180, "deg_end": 225, "dominant": "srx", "recessive": "cxp", "phase_sign": "+",
"description": "DECAY PHASE MIRROR", "code": "OIO" },
        { "deg_start": 225, "deg_end": 270, "dominant": "cxp", "recessive": "srx", "phase_sign": "+",
"description": "GROWTH cophase MIRROR", "code": "IOO" },
        { "deg_start": 270, "deg_end": 315, "dominant": "crx", "recessive": "sxp", "phase_sign": "-",
"description": "DECAY cophase MANIFEST", "code": "OOI" },
        { "deg_start": 315, "deg_end": 360, "dominant": "sxp", "recessive": "crx", "phase_sign": "-",
"description": "GROWTH PHASE MANIFEST", "code": "III" }
      ]
    },
    "samples": [
      { "deg": 1e-9, "srx": null, "sxp": null, "cxp": null, "crx": null, "uxp": null, "urx": null, "sawdown":
null, "sawup": null, "flatwave": null },
      { "deg": 89.999999999, "srx": 1, "sxp": 1, "cxp": null, "crx": 0, "uxp": null, "urx": 1, "sawdown":
0, "sawup": 1, "flatwave": null },
      { "deg": 90.000000001, "srx": 1, "sxp": 1, "cxp": 0, "crx": null, "uxp": -1, "urx": null,
"sawdown": -1, "sawup": 0, "flatwave": null },
      { "deg": 179.999999999, "srx": 0, "sxp": null, "cxp": 1, "crx": 1, "uxp": null, "urx": -1,
"sawdown": 0, "sawup": -1, "flatwave": null },
      { "deg": 180.000000001, "srx": null, "sxp": 0, "cxp": 1, "crx": 1, "uxp": 1, "urx": null,
"sawdown": 1, "sawup": 0, "flatwave": null },
      { "deg": 269.999999999, "srx": 1, "sxp": 1, "cxp": null, "crx": 0, "uxp": null, "urx": 1,
"sawdown": 0, "sawup": 1, "flatwave": null },
      { "deg": 270.000000001, "srx": 1, "sxp": 1, "cxp": 0, "crx": null, "uxp": -1, "urx": null,
"sawdown": -1, "sawup": 0, "flatwave": null },
      { "deg": 359.999999999, "srx": 0, "sxp": null, "cxp": 1, "crx": 1, "uxp": null, "urx": -1,
"sawdown": 0, "sawup": -1, "flatwave": null }
    ]
  },
  "machine_interface": {
    "serialization_format": ["yaml", "json", "ttl"],
    "logical_schema": "R-Theory Ontology Schema (1.0)",
    "query_endpoint": "/rtheory/canonical",
    "compatibility": ["MML-2025", "LangGraph-1.2", "NeuroSync-3.4", "OpenSim-v6"],
    "dependency_keys": [
      "admissibility",
      "projection_equivalence",
      "calibration_ratio",
      "residual_vector",
      "closure_status"
    ]
  },
  "status": {
    "admissibility_summary": {
      "structure_status": "closed",
      "projection_status": "free",
      "calibration_status": "continuous",
      "practice_status": "governed",
      "ethical_status": "transparent"
    },
    "verification": {
      "validation_suite": "rtheory-canonical-tests",
      "unit_tests_passed": 108,
      "residual_variance": 0.000013,
      "anomalies": 0,
      "meta_admissibility": true
    }
```

```
 },
 "sorts": [
   { "id": "Locus", "description": "Abstract locus / state-holder in M." },
   { "id": "Manifold", "description": "Topological host for loci; no metric assumed at structural tier."
},
   { "id": "Path", "description": "Continuous mapping from an abstract index set into M." },
   { "id": "Frame", "description": "Projection package Π = (chart, ordering, calibration,
accounting)." },
   { "id": "Report", "description": "Numerical or symbolic output of a protocol under Π." }
 ],
 "primitive_symbols": [
   { "id": "M", "sort": "Manifold", "kind": "constant" },
   { "id": "x", "sort": "Locus", "kind": "variable" },
   { "id": "y", "sort": "Locus", "kind": "variable" },
   { "id": "≼", "sort": "Locus×Locus→Bool", "kind": "relation", "paper": "C" },
   { "id": "ρ", "sort": "Locus×Locus→Scalar", "kind": "relation", "paper": "I", "notes":
"dimensionless; metric not implied" },
   { "id": "Mirror", "sort": "Frame→Frame", "kind": "function", "paper": "A/C" },
   { "id": "T", "sort": "Frame×Frame→(Frame→Frame)", "kind": "relation", "paper": "G/O",
"notes": "admissible frame transformation witness" },
   { "id": "Π", "sort": "Frame", "kind": "variable", "paper": "D/G" },
   { "id": "Π_metric", "sort": "Frame", "kind": "constant", "paper": "D" },
   { "id": "M_proto", "sort": "(Structure×Protocol×Frame)→Report", "kind": "function", "paper": "N"
}
 ],
 "formal_axioms": [
   {
     "id": "AX-A1",
     "layer": "structure",
     "name": "Bounded phase closure",
     "formal": "Phase closure is enforced by identification after 360°.",
     "enforcement": "governance_only",
     "depends_on": ["A"]
   },
   {
     "id": "AX-B1",
     "layer": "structure",
     "name": "Topological-only manifold",
     "formal": "M admits continuity/neighborhoods; no metric/norm/inner product is structural.",
     "enforcement": "schema_tag_only",
     "depends_on": ["B"]
   },
   {
     "id": "AX-C1",
      "layer": "structure",
      "name": "Partial order",
      "formal": "⩽ is reflexive, antisymmetric, transitive; not total.",
      "enforcement": "external_validator",
      "depends_on": ["C"]
    },
    {
      "id": "AX-I1",
      "layer": "structure",
      "name": "Separation axioms",
      "formal": "ρ(x,x)=0 ∧ ρ(x,y)=ρ(y,x) ∧ (∀x)(∀y1,y2) comparable(ρ(x,y1),ρ(x,y2)) ∧ 
monotone_reparameterization_allowed(ρ).",
      "enforcement": "external_validator",
      "depends_on": ["I"]
    },
    {
      "id": "AX-L1",
      "layer": "structure",
      "name": "Attenuation requirement for gravitation-class relation",
      "formal": "If G is gravitational-class then G(x,y)=A(ρ(x,y)) with A monotone decreasing; no 
functional form fixed.",
      "enforcement": "schema_tag_only",
      "depends_on": ["L"]
    },
    {
      "id": "AX-D1",
      "layer": "projection",
      "name": "Metric is projection-only and joint",
      "formal": "Π_metric may introduce (L,T) only jointly; ¬admissible(define L without T) ∧ 
¬admissible(define T without L).",
      "enforcement": "external_validator",
      "depends_on": ["D"]
    },
    {
      "id": "AX-G1",
      "layer": "projection",
      "name": "Transformation compositional closure",
      "formal": "Admissible transformations compose: admissible(T1)∧admissible(T2) ⇒ ∃T3: 
T2∘T1 = T3.",
      "enforcement": "external_validator",
      "depends_on": ["G"]
    },
    {
      "id": "AX-N1",
      "layer": "projection",
      "name": "Measurement is protocol-to-report map",
      "formal": "Measurement does not reveal intrinsic quantities; it produces reports under
(Protocol, Π).",
      "enforcement": "governance_only",
      "depends_on": ["N"]
    },
    {
      "id": "AX-O1",
      "layer": "projection",
      "name": "Projection equivalence",
      "formal": "D1≈D2 ⇔ ∃ admissible T aligning Π1→Π2 such that relational structure
correspondence holds (numeric equality not required).",
      "enforcement": "governance_only",
      "depends_on": ["O"]
    },
    {
      "id": "AX-H1",
      "layer": "closure",
      "name": "No new primitives after closure",
      "formal": "After structural closure, additions are projection craft only; primitives inventory is
immutable.",
      "enforcement": "corpus_linter",
      "depends_on": ["H", "P", "W"]
    },
    {
      "id": "AX-V1",
      "layer": "calibration",
      "name": "21cm bridge role (classification statement)",
      "formal": "A discrete propagation-stable cycle provides a multi-projection calibration bridge
(time/length/mass/charge labeling) without privileging any projection as fundamental.",
      "enforcement": "governance_only",
      "depends_on": ["V"]
    }
  ],
  "admissibility": {
    "predicate_name": "fivefold_predicate",
    "criteria": [
      "boundedness",
      "no_new_primitives",
      "mirror_accessibility",
      "residual_reporting",
      "projection_declaration"
    ],
    "inadmissible_moves": [
      "implicit_infinities",
      "hidden_normalization",
      "smuggled_metric",
      "external_reference_standard",
      "privileged_frame",
      "rates_premature"
    ],
    "two_rotation_seal": {
      "rotation_1_degrees": 360,
      "rotation_2_degrees": 360,
      "status": "procedural_obligation_only"
    }
  },
  "catalog_unmodeled_requirements": [
    {
      "id": "DISC-STRUCTURE-VS-PROJECTION",
      "type": "governance",
      "text": "Tags like 'structure' vs 'projection' are classification commitments; schema alone
cannot prove correctness."
    },
    {
      "id": "DISC-ENFORCEMENT",
      "type": "engineering",
      "text": "Rules marked governance_only/corpus_linter/external_validator require non-JSON
enforcement (validators, linters, proof artifacts)."
    },
    {
      "id": "DISC-NORMATIVE-ADMISSIBILITY",
      "type": "governance",
      "text": "Normative admissibility rules (gate conditions)\nInadmissible moves are excluded
rather than corrected.\nNo observer frame may be privileged.\nMirror accessibility must
hold.\nThese are constraints on what is allowed to be said/done, not data. JSON can store the
text of rules, but cannot by itself enforce or prove them without an external validator/prover."
    },
    {
      "id": "DISC-NON-SCOPE-PROHIBITIONS",
      "type": "governance",
      "text": "Non-scope / prohibition semantics: Introduces no operators, no physical
quantities…\nThis paper performs neither rotation."
    },
    {
      "id": "DISC-PROJECTION-HUNGER-HEURISTICS",
      "type": "methodology",
      "text": "Projection hunger / observer-responsibility language\nDisappointment indicates
correct operation.\nExplanatory richness indicates leakage.\nThese are diagnostic heuristics
about reader behavior and category errors; they are not representable as formal constraints
except as informal annotations."
    },
    {
      "id": "DISC-TWO-ROTATION-SEAL-PROCEDURE",
      "type": "methodology",
      "text": "Two-rotation seal as a procedural prerequisite:\nFirst 360°: closure without
dimension.\nSecond 360°: dimension may emerge only as projection…\nThis is a process
requirement (a staged proof/derivation obligation). JSON can encode workflow steps, but not
verify the closure demonstrated condition without a proof artifact."
    },
    {
      "id": "DISC-PROOF-OBLIGATIONS",
      "type": "engineering",
      "text": "Proof obligations and must be shownclaims\n\nExamples:\nDiscreteness is forced by
closure.\nTransformations must compose.\nAny further addition necessarily constitutes
projection craft.\nThese are mathematical/argumentative obligations, not static fields. JSON can
reference proofs, but not contain the proofs in a machine-checkable way unless you embed a
formal proof language."
    },
    {
      "id": "DISC-PROJECTION-EQUIVALENCE-SEMANTICS",
      "type": "meta",
      "text": "Equivalence: up to projection / ≈ semantics\nProjection-equivalent iff there exists
admissible T such that … ≈ …\nThe symbol ≈ encodes a tolerance/translation criterion that is
intentionally non-numeric and context-dependent. JSON can’t capture that meaning without an
explicit, formal equivalence relation plus acceptance criteria."
    },
    {
      "id": "DISC-STABILITY-CLAIMS-CONTEXT",
      "type": "methodology",
      "text": "Context-sensitive stability claims:\nStable under
propagation/aggregation.\nResiduals are informative rather than ignorable.\nThese depend on
domain, protocol class, and resolution regime. JSON can store parameters and residuals, but
stability here is an interpretive classification, not a fixed predicate."
    },
    {
      "id": "DISC-NO-NEW-PRIMITIVES-GLOBAL",
      "type": "governance",
      "text": "No new primitives as a global document constraint\nThis is a cross-document
invariance constraint: later papers must not introduce new primitives. JSON can store an
```

```
inventory list, but enforcing no novel primitives introduced anywhere requires semantic diffing
and human or formal review."
    },
    {
      "id": "DISC-INTENTIONAL-UNDER-SPECIFICATION",
      "type": "meta",
      "text": "Intentional under-specification;\nMany definitions are deliberately negative (define by
what is not asserted) or leave functional forms open (e.g., attenuation A(ρ) monotone
decreasing).\nJSON favors explicit fields and values; intentional incompleteness is
representable only as null/placeholder, which loses the reason for incompleteness."
    },
    {
      "id": "DISC-CATEGORY-BOUNDARY-CLAIMS",
      "type": "governance",
      "text": "Category-boundary claims (structure vs projection):\nProjection is not a
weakness.\nConstants reconcile choices; they do not govern structure.\nThese are
meta-theoretic classifications about the status of statements, not objects in the modeled
domain. You can tag statements, but the tag's correctness is not machine-enforceable in plain
JSON."
    }
  ],
  "notes": {
    "intent": "Provide MMLs (LLMs) and augmented intelligences (AI) with a relationally closed
canonical map for admissible modeling.",
    "interoperability": "Designed for modular import, translation, and projection across
domain-specific grammars.",
    "revision_policy": "Calibration refinements only; no structural alterations permitted.",
    "canonical_close": "Structure: closed. Projection: free. Practice: governed."
  },
  "projections": [
    {
      "id": "AI-PROJ-00",
      "title": "Augmented Intelligence as Admissible Projection Under Bounded Output",
      "version": "1.0",
      "language": "en-struct",
      "axiom_refs": ["AX-N1", "AX-O1", "AX-D1", "AX-G1", "AX-H1"],
      "canonical_slots_used": ["D", "E", "N", "O", "P", "Q", "W"],
      "declared_primitives": ["D", "E", "N", "O", "P", "Q", "W"],
      "domain": {
        "phase_symbol": "phi",
        "phase_range": { "type": "half_open_interval", "start": "0", "end": "360" },
        "dimensionless": true,
        "inputs": {
          "S_in": "user-provided structure stream (text, data, constraints)",
    "C_out": "output capacity (token budget / response budget)"
  }
},
"principles": {
  "no_preferred_frame": true,
  "no_preferred_user": true,
  "ontology_allowed": false,
  "projection_status": "free",
  "structure_status": "closed",
  "practice_status": "governed",
  "residual_reporting_required": true
},
"canon_math_ref": "canonical_math",
"definitions": {
  "H_in": "estimate_structural_entropy(S_in)",
  "R": {
    "name": "compression_pressure",
    "expr": {
      "op": "/",
      "lhs": "H_in",
      "rhs": { "fn": "max", "args": ["C_out", 1.0] }
    },
    "display": "R = H_in / max(C_out, 1.0)"
  },
  "A": {
    "name": "expansion_surplus",
    "expr": {
      "op": "/",
      "lhs": { "fn": "max", "args": ["C_out", 1.0] },
      "rhs": { "fn": "max", "args": ["H_in", 1e-9] }
    },
    "display": "A = max(C_out, 1.0) / max(H_in, 1e-9)"
  },
  "tau": {
    "name": "dominance_sharpness",
    "range": { "min": 0.0, "max": null },
    "notes": "tau->0 enforces single-trajectory dominance; larger tau relaxes dominance"
  },
  "psi": {
    "name": "interaction_phase",
    "display": "psi = infer_constraint_manifold(S_in, constraints)",
    "notes": "style/constraint state, not user identity"
  },
  "kappa": {
    "name": "incoherence_risk",
    "display": "kappa = risk(underspecification, inconsistency, adversarial_markers)"
  }
},
"admissibility_contract": {
  "fivefold_predicate": [
    "boundedness",
    "no_new_primitives",
    "mirror_accessibility",
    "residual_reporting",
    "projection_declaration"
  ],
  "requirements": [
    "Declare projection intent: summarize | analyze | translate | generate",
    "Introduce no new primitives (ontology_allowed=false)",
    "Report residuals as first-class outputs",
    "If kappa exceeds threshold, enforce E and return refusal-or-reframe with residuals",
    "No metric claims unless explicitly calibrated: not used in this projection"
  ]
},
"algorithm": {
  "inputs": ["S_in", "C_out", "intent", "constraints", "tau"],
  "steps": [
    { "set": "H_in", "call": "estimate_structural_entropy", "args": ["S_in"] },
    { "set": "R", "expr": "H_in / max(C_out, 1.0)" },
    { "set": "A", "expr": "max(C_out, 1.0) / max(H_in, 1e-9)" },
    { "set": "psi", "call": "infer_constraint_manifold", "args": ["S_in", "constraints"] },
    { "set": "kappa", "call": "incoherence_risk", "args": ["S_in", "constraints"] },
    {
      "if": { "op": ">", "lhs": "kappa", "rhs": "KAPPA_MAX" },
      "then": [
        { "set": "Y", "call": "refusal_or_reframe", "args": ["intent", "psi"] },
        { "set": "eps", "call": "residual_vector", "args": ["..."] },
        { "return": ["Y", "eps"] }
      ]
    },
    {
      "set": "regime",
      "expr": "(R > 1.0) ? 'compress' : ((A > 1.0) ? 'expand' : 'translate')"
    },
    { "set": "Y", "call": "render", "args": ["regime", "intent", "psi", "tau", "C_out"] },
    {
      "set": "eps",
      "call": "residual_vector",
          "args": ["dropped_mass", "ambiguity", "contradiction", "unsupported_claims", "kappa"]
        },
        { "return": ["Y", "eps"] }
      ],
      "invariances": [
        "response depends on (S_in, constraints, intent, tau, C_out), not on user identity",
        "equivalent inputs + equivalent constraints => structurally comparable outputs (up to
declared degrees of freedom)"
      ]
    },
    "outputs": {
      "Y": "primary response (text or structured artifact)",
      "eps": {
        "name": "residual_vector",
        "fields": ["dropped_mass", "ambiguity", "contradiction", "unsupported_claims", "kappa"]
      },
      "closure_status": "closed-structure / free-projection / governed-practice"
    },
    "safety_notes": {
      "incoherence_users": false,
      "definitions_disclaimer": "Definitions may introduce labels but not ontological
commitments."
    }
  },
  {
    "id": "AI-PROJ-01",
    "title": "Bounded Temperature Projection with SAWUP/SAWDOWN Deterministic
Divergence",
    "version": "1.1",
    "language": "en-struct",
    "axiom_refs": ["AX-A1", "AX-N1", "AX-O1", "AX-D1", "AX-G1", "AX-H1"],
    "canonical_slots_used": ["D", "E", "J", "N", "O", "P", "Q", "W"],
    "declared_primitives": ["D", "E", "J", "N", "O", "P", "Q", "W"],
    "domain": {
      "phase_symbol": "phi",
      "phase_range": { "type": "half_open_interval", "start": "0", "end": "360" },
      "dimensionless": true,
      "inputs": {
        "S_in": "user-provided structure stream (text, data, constraints)",
        "C_out": "output capacity (token budget / response budget)"
      }
    },
    "principles": {
      "no_preferred_frame": true,
      "no_preferred_user": true,
      "ontology_allowed": false,
      "projection_status": "free",
      "structure_status": "closed",
      "practice_status": "governed",
      "residual_reporting_required": true
    },
    "definitions": {
      "H_in": "estimate_structural_entropy(canon(S_in))",
      "R": {
        "name": "compression_pressure",
        "expr": "R = H_in / max(C_out, 1.0)",
        "display": "R = H_in / max(C_out, 1.0)"
      },
      "A": {
        "name": "expansion_surplus",
        "expr": "A = max(C_out, 1.0) / max(H_in, 1e-9)",
        "display": "A = max(C_out, 1.0) / max(H_in, 1e-9)"
      },
      "tau": {
        "name": "dominance_sharpness",
        "range": { "min": 0.0, "max": null },
        "notes": "tau->0 enforces single-trajectory dominance; larger tau permits multi-trajectory
option sets"
      },
      "psi": {
        "name": "interaction_phase",
        "display": "psi = infer_constraint_manifold(canon(S_in), constraints)",
        "notes": "constraint/style state, not user identity"
      },
      "kappa": {
        "name": "incoherence_risk",
        "display": "kappa = incoherence_risk(canon(S_in), constraints) in [0,1]"
      },
      "T_policy": {
        "name": "temperature_policy",
        "expr": "T in [T_min,T_max]; if T_raw < T_eps then T := 1.0; analytic uses SAWUP toward
1; creative uses SAWDOWN toward T_max",
        "display": "bounded sawtooth temperature with reflecting floor-to-one"
      }
    },
    "admissibility_contract": {
      "fivefold_predicate": [
        "boundedness",
        "no_new_primitives",
        "mirror_accessibility",
        "residual_reporting",
        "projection_declaration"
      ],
      "requirements": [
        "Declare intent: summarize | analyze | translate | generate",
        "Declare mode: analytic | creative (or allow deterministic inference from S_in +
constraints)",
        "No new primitives (ontology_allowed=false); only recombine declared slots/ops",
        "Determinism requirement: seed := SHA3-256(canon(S_in) || projection_id || intent ||
C_out) and all option generation must be seed-driven or deterministic-diverse",
        "Temperature boundedness: T in [1.0, T_max]; if T_raw approaches 0 (T_raw < T_eps)
then T := 1.0",
        "SAWUP/SAWDOWN singular handling must be declared: exclude neighborhoods where
sin(phi)=0, cos(phi)=0, or abs(cxp*srx - 1) < eps_den, OR apply a deterministic clamp",
        "Negative quadrant rule: when frame is negative (e.g., sign(cos(phi)*sin(phi)) < 0), take
the opposing-side branch (sign flip) rather than forcing alignment",
        "Residuals are mandatory and must include: dropped_mass, ambiguity, contradiction,
unsupported_claims, kappa"
      ]
    },
    "algorithm": {
      "inputs": [
        "S_in",
        "C_out",
        "intent",
        "constraints",
        "tau",
        "mode",
        "phi",
        "T_max",
        "T_eps",
        "K_max",
        "KAPPA_MAX",
        "eps_den"
      ],
      "steps": [
        { "set": "S_canon", "call": "canon", "args": ["S_in"] },
        { "set": "H_in", "call": "estimate_structural_entropy", "args": ["S_canon"] },
        { "set": "R", "expr": "H_in / max(C_out, 1.0)" },
        { "set": "A", "expr": "max(C_out, 1.0) / max(H_in, 1e-9)" },
        { "set": "psi", "call": "infer_constraint_manifold", "args": ["S_canon", "constraints"] },
        { "set": "kappa", "call": "incoherence_risk", "args": ["S_canon", "constraints"] },
      {
        "if": { "op": ">", "lhs": "kappa", "rhs": "KAPPA_MAX" },
        "then": [
          { "set": "seed", "call": "derive_seed_sha3_256_uint64", "args": ["S_canon",
"AI-PROJ-01", "intent", "C_out"] },
          { "set": "K", "expr": "min(K_max, 3)" },
          { "set": "Y", "call": "generate_reframes_option_set", "args": ["S_canon", "constraints",
"intent", "K", "seed"] },
          { "set": "eps", "call": "residual_vector", "args": ["dropped_mass", "ambiguity",
"contradiction", "unsupported_claims", "kappa"] },
          { "return": ["Y", "eps"] }
        ]
      },
      { "set": "mode_eff", "call": "infer_mode_if_missing", "args": ["mode", "intent", "S_canon",
"constraints"] },
      { "set": "seed", "call": "derive_seed_sha3_256_uint64", "args": ["S_canon", "AI-PROJ-01",
"intent", "C_out"] },
      { "set": "srx", "call": "srx", "args": ["phi"] },
      { "set": "cxp", "call": "cxp", "args": ["phi"] },
      { "set": "den", "expr": "cxp*srx - 1.0" },
      { "set": "den_safe", "call": "denominator_safe", "args": ["den", "eps_den"] },
      { "set": "sawdown", "expr": "srx / den_safe" },
      { "set": "sawup", "expr": "cxp / den_safe" },
      { "set": "neg_frame", "call": "negative_quadrant_indicator", "args": ["phi"] },
      { "set": "align_sign", "expr": "(neg_frame) ? -1.0 : 1.0" },
      { "set": "T_raw", "call": "temperature_from_saw", "args": ["mode_eff", "sawup", "sawdown",
"align_sign", "T_max"] },
      { "set": "T", "expr": "(T_raw < T_eps) ? 1.0 : clamp(T_raw, 1.0, T_max)" },
      { "set": "K", "call": "branch_count_from_temperature", "args": ["T", "T_max", "K_max"] },
      { "set": "tau_eff", "call": "tau_from_temperature", "args": ["tau", "T"] },
      { "set": "regime", "expr": "(R > 1.0) ? 'compress' : ((A > 1.0) ? 'expand' : 'translate')" },
      { "set": "Y", "call": "render_option_set", "args": ["regime", "intent", "psi", "mode_eff", "T",
"K", "tau_eff", "C_out", "seed", "S_canon", "constraints"] },
      { "set": "eps", "call": "residual_vector", "args": ["dropped_mass", "ambiguity",
"contradiction", "unsupported_claims", "kappa"] },
      { "return": ["Y", "eps"] }
    ],
    "invariances": [
      "response depends on (canon(S_in), constraints, intent, tau, phi, C_out, seed_policy), not
on user identity",
      "equivalent canon(S_in) + equivalent constraints => same option_set ordering and
comparable outputs (up to declared degrees of freedom)",
      "SAWUP identity: sawup(phi) == 1/urx(phi) (exact)",
      "SAWDOWN identity under uxp definition: sawdown(phi) == 1/uxp(phi) (exact)"
```

```
        ]
      }
    }
  ]
}
```