Knuth-Skilling Lean Overview

Codex 5.2, Claude 4.5, Zar Goertzel*

January 27, 2026

Abstract

This document provides a definition-, axiom-, and theorem-statement-level overview of the Lean 4 formalization of Knuth & Skilling's "Foundations of Inference" [1]. Each section lists the main definitions and theorem statements with file locations, organized by the corresponding K&S paper sections. Alternative derivation paths (Cox, Shore–Johnson, Shannon/Faddeev) are discussed in the appendix.

Contents

Abstract **1**

1 Overview and File Structure **4**
- 1.1 K&S Paper Coverage . . . 4
- 1.2 Key Files . . . 4

2 Combination and Valuation Axioms (K&S Sections 3–4) **5**
- 2.1 K&S Symmetries 0–2 . . . 5
- 2.2 Iteration . . . 7
- 2.3 Identity-Free Iteration . . . 7

3 No Anomalous Pairs and Separation **8**
- 3.1 No Anomalous Pairs (Classical Formulation) . . . 8
- 3.2 Separation (Formalization-Discovered Axiom) . . . 9
- 3.3 Equivalence Theorem . . . 10
  - 3.3.1 Bilateral Separation . . . 10
- 3.4 Derived Properties . . . 10
  - 3.4.1 Archimedean Property and Commutativity . . . 10

4 The Representation Theorem (Appendix A) **11**
- 4.1 Three Independent Proof Paths . . . 12
- 4.2 Proof Architecture 1: The Hölder Path (Recommended) . . . 12
- 4.3 Proof Architecture 2: The Grid/Induction Path . . . 13
  - 4.3.1 The Globalization Construction ("Triple Family Trick") . . . 13
- 4.4 Main Theorem Statement . . . 14
- 4.5 Proof Architecture 2: The Direct Cuts Path . . . 15

*This document was drafted collaboratively by humans and AI systems (GPT/Codex, Claude). While formal claims are machine-checked in Lean 4, prose descriptions may contain human or AI hallucinations. Caveat lector.

5 The Product Theorem (Appendix B) 17
5.1 Two Complete Proof Paths . . . 17
5.1.1 Path 1: K&S's Derivation . . . 17
5.1.2 Path 2: Direct Algebraic Proof . . . 17
5.2 Common Interface: ScaledMultRep . . . 17
5.3 Direct Product Symmetries (3–4) . . . 18
5.4 Product Equation . . . 20

6 The Variational Theorem (Appendix C) 21
6.1 What Appendix C Derives . . . 21
6.2 Path A: The Functional Equation Solver . . . 22
6.3 Path B: Lagrange Multiplier Derivation (Local) . . . 22
6.4 Gate G: Bridging Local to Global . . . 23
6.5 Integration: From Derivative to Entropy Form . . . 23
6.6 Alternative Route: Shore–Johnson Axioms . . . 23

7 Divergence (K&S Section 6.1) 24

8 Conditional Probability Calculus (K&S Section 7) 25
8.1 Structural Change: From Linear Order to Lattice . . . 26
8.2 Bivaluation and Axiom 5 . . . 26
8.3 The Product Equation Reappears . . . 27
8.4 Chain-Product Rule . . . 28
8.5 Bayes' Theorem . . . 28
8.6 Probability as Ratio of Measures . . . 28
8.7 baseMeasure Satisfies Measure Axioms . . . 29

9 σ-Completeness and σ-Additivity (Extension) 30
9.1 σ-Complete Events . . . 30
9.2 Sequential Completeness of the Scale . . . 30
9.3 Scott Continuity (Countable Directed Limits) . . . 30
9.4 Main Theorem: K&S σ-Additivity . . . 31

10 Information and Entropy (K&S Section 8) 31
10.1 From Atom Divergence to KL Divergence . . . 32
10.2 Derivation Chain . . . 32
10.3 Shannon's Three Properties . . . 33
10.4 Formalization . . . 33
10.5 Shannon Entropy Definition . . . 34

11 Counterexamples and Clarifications 34
11.1 The "Discontinuous Re-grading" Claim . . . 34
11.2 Pathological Additive Functions . . . 35
11.2.1 The Construction . . . 35
11.3 Separation is Not Derivable (Independence Result) . . . 36

12 Summary: Complete Formalization 37
12.1 Major Formalized Theorems at a Glance . . . 37
12.2 Key Discoveries from Formalization . . . 39

Appendix: Supplementary Derivation Paths (Walkthrough) **40**
Supplementary: Unified finite entropy/KL 40
Supplementary: ProbVec ↔ ProbDist + equivalences 41
Supplementary: Axiomatic entropy (Faddeev/Shannon/Khinchin) 43
Supplementary: Shore–Johnson → KL 45
Supplementary: Cox 47

Appendix: Small Example Files **49**
Appendix: Examples: CoinDie 50
Appendix: Examples: 7-element lattice 50
Appendix: Examples: decision toy 51

Appendix: The KSSeparation Discovery **51**

13 Build Instructions **53**

Acknowledgments **53**

1 Overview and File Structure

1.1 K&S Paper Coverage

K&S Section	Topic	Lean Files
Sections 3–4	Combination Axioms (Sym 0–2)	`Core/Basic.lean`, `Core/Algebra.lean`
Sections 3,7	Probability (two K&S paths)	`Probability/ProbabilityDerivation.lean`, `Probability/ConditionalProbability/Basic.lean`
Section 4	Quantum theory	`Core/SymmetricalFoundation.lean`, `Mettapedia/Algebra/TwoDimClassification.lean`
Section 6	Divergence	`Information/Divergence.lean`
Section 8	Info/Entropy	`Information/InformationEntropy.lean`
Appendix A	Representation	`Additive/Main.lean`, `Additive/Representation.lean`
Appendix B	Product (Sym 3–4)	`Multiplicative/Main.lean`, `Multiplicative/ScaledMultRep.lean`
Appendix C	Variational	`Variational/Main.lean`
Extension	σ-additivity bridge	`Core/ScaleCompleteness.lean`

1.2 Key Files

All paths below are relative to `Mettapedia/ProbabilityTheory/KnuthSkilling/`.

`../KnuthSkilling.lean` Main entrypoint (FOI core + extra K&S modules)

`FoundationsOfInference.lean` Reviewer entrypoint (FOI core; no WIP)

`Core/Basic.lean` Core axioms: `KSSemigroupBase`, `KnuthSkillingMonoidBase`, `KnuthSkillingAlgebraBase`

`Core/Algebra.lean` `iterate_op`, separation axioms (`KSSeparation*`)

`Core/ScaleCompleteness.lean` σ-completeness axioms and σ-additivity theorem

`Core/SymmetricalFoundation.lean` Section 4 quantum derivation

`Additive/Main.lean` Appendix A entrypoint (typeclass interface + instances)

`Additive/Representation.lean` Appendix A representation interfaces (identity-free default)

`Additive/Axioms/SandwichSeparation.lean` Archimedean + commutativity from `KSSeparation`

`Additive/Proofs/OrderedSemigroupEmbedding/HolderEmbedding.lean` Hölder/Alimov embedding path

`Additive/Proofs/GridInduction/Main.lean` Grid/induction path (K&S-style globalization)

`Additive/Proofs/DirectCuts/Main.lean` Dedekind cuts path (alternative)

`Multiplicative/Main.lean` Appendix B pipeline (product ⇒ exponential ⇒ scaled mult)

`Multiplicative/Proofs/Direct/DirectProof.lean` Appendix B direct algebraic proof path

`Multiplicative/ScaledMultRep.lean` Appendix B common interface for both proof paths

`Variational/Main.lean` Appendix C variational theorem (entropy form)

`Information/Divergence.lean` Section 6 divergence

`Information/InformationEntropy.lean` Section 8 information/entropy (KL, Shannon)

`Probability/ConditionalProbability/Basic.lean` Section 7 conditional probability (lattice path)

2 Combination and Valuation Axioms (K&S Sections 3–4)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Core/Basic.lean`

2.1 K&S Symmetries 0–2

K&S notation: $\bar{x}$ denotes a lattice element, x its real-valued valuation. Symmetries 0–2 constrain how valuations relate to lattice structure:

- **Symmetry 0** (Fidelity): Valuation preserves order: $\bar{x} < \bar{y} \Rightarrow x < y$
- **Symmetry 1** (Monotonicity): Combination preserves order: $\bar{x} < \bar{y} \Rightarrow \bar{x} \oplus \bar{z} < \bar{y} \oplus \bar{z}$
- **Symmetry 2** (Associativity): Combination is associative: $(\bar{x} \oplus \bar{y}) \oplus \bar{z} = \bar{x} \oplus (\bar{y} \oplus \bar{z})$

The **direct product** symmetries (3–4) are formalized separately in `Multiplicative/Main.lean` (see Section 5).

Definition 2.1 (`KSSemigroupBase`)**. Lines 180–188, Core/Basic.lean**
Identity-free core structure containing only Symmetries 0–2.

Listing 1: Identity-free semigroup base

```
class KSSemigroupBase (alpha : Type*) extends LinearOrder alpha where
  op : alpha → alpha → alpha                          -- combination operation
  op_assoc : ∀ x y z : alpha, op (op x y) z = op x (op y z)   -- Sym 2
  op_strictMono_left : ∀ y : alpha, StrictMono (fun x => op x y)    -- Sym 0+1
  op_strictMono_right : ∀ x : alpha, StrictMono (fun y => op x y)   -- Sym 0+1
```

Definition 2.2 (`KnuthSkillingMonoidBase`)**. Lines 237–244, Core/Basic.lean**
Adds an identity element (`ident`) without assuming it is the order minimum.

Listing 2: Core K&S structure with identity (no positivity assumption)

```
class KnuthSkillingMonoidBase (alpha : Type*) extends KSSemigroupBase alpha where
  ident : alpha
  op_ident_right : ∀ x : alpha, op x ident = x
  op_ident_left : ∀ x : alpha, op ident x = x
```

Definition 2.3 (`KnuthSkillingAlgebraBase`)**. Lines 246–248, Core/Basic.lean**
The probability-theory convenience layer: assumes the identity is the order minimum (`ident_le`).

Listing 3: K&S probability base: identity is minimum (positivity)

```
class KnuthSkillingAlgebraBase (alpha : Type*) extends KnuthSkillingMonoidBase
    alpha where
  ident_le : ∀ x : alpha, ident ≤ x
```

Remark 2.4 (Implicit Linear Order). K&S never explicitly state that elements are totally ordered, but their proofs rely on trichotomy. We make this explicit via `LinearOrder`. This is **necessary**, not cosmetic: `no_pointRepresentation_with_incomparables` proves that any partial order with incomparable elements admits no faithful $\Theta : \alpha \to \mathbb{R}$. Dropping totality yields interval-valued representations (Walley's imprecise probability); see `Core/TotalityImprecision.lean`.

Remark 2.5 (Identity Element—Essential for Positivity). **Important**: The identity element (`ident`) is **not** among K&S's numbered symmetries. K&S explicitly state that the bottom element $\bot$ is **optional**:

> "with the bottom element optional" (K&S line 320)
> "Some mathematicians opt to include the bottom element on aesthetic grounds, whereas others opt to exclude it" (K&S lines 340–341)

However, our formalization **disproves** K&S's claim that fidelity alone ensures positivity. The $\mathbb{Z}$ counterexample (`Additive/Counterexamples/NegativeWithoutIdentity.lean`) shows:

- $(\mathbb{Z}, +, \leq)$ satisfies K&S Axioms 1–2 but has no identity
- The representation theorem applies, giving $\Theta : \mathbb{Z} \to \mathbb{R}$
- But $\Theta(-1) = -1 < 0$—**negative values appear**

Corrected understanding:

- **With identity + `ident_le`**: $\Theta(\bot) = 0$; all $x > \bot$ have $\Theta(x) > 0$
- **Without identity**: Representation works but positivity is **not guaranteed**

Identity with `ident_le` (i.e., $\bot \leq x$ for all x) is **essential** for positivity, not "aesthetic."

Listing 4: $\mathbb{Z}$ counterexample: representation works but positivity fails

```
-- Z is a valid KSSemigroupBase (satisfies Axioms 1-2)
instance Int.instKSSemigroupBase : KSSemigroupBase Int where
  op := (. + .)
  op_assoc := add_assoc
  op_strictMono_left := fun y => by intro a b hab; omega
  op_strictMono_right := fun x => by intro a b hab; omega

-- But Z cannot satisfy ident_le (no minimum element)
theorem Int.cannot_satisfy_ident_le : ¬(∀ n : Int, (0 : Int) ≤ n) := by
  push_neg; use -1; omega

-- The Holder embedding has negative values: Phi(-1) < 0
theorem Int.holder_embedding_has_negatives :
    ∃ (G : Subsemigroup (Multiplicative R))
        (Theta : Multiplicative Int ≃*o G),
      Multiplicative.toAdd (Theta (Multiplicative.ofAdd (-1))) < 0 :=
  holder_embedding_produces_negatives MultiplicativeInt.no_anomalous_pair
```

Remark 2.6 (Unbundled Axiom Predicates). **Lines 142–164, Core/Basic.lean**

In addition to the bundled typeclasses, we provide **unbundled predicates** for each axiom. This enables flexible hypothesis tracking—use individual predicates when you need minimal assumptions, or bundled classes when you want ergonomic access to multiple axioms.

Combination predicates (Core/Basic.lean):

- `OpAssoc op` (line 143): $\forall x\, y\, z,\ \mathrm{op}(\mathrm{op}(x, y), z) = \mathrm{op}(x, \mathrm{op}(y, z))$
- `OpStrictMonoLeft op` (line 147): $\forall y,\ \mathrm{StrictMono}(\lambda x.\, \mathrm{op}(x, y))$
- `OpStrictMonoRight op` (line 151): $\forall x,\ \mathrm{StrictMono}(\lambda y.\, \mathrm{op}(x, y))$
- `OpIdentLeft op e` (line 155): $\forall x,\ \mathrm{op}(e, x) = x$
- `OpIdentRight op e` (line 159): $\forall x,\ \mathrm{op}(x, e) = x$
- `IdentIsMin e` (line 163): $\forall x,\ e \leq x$

Connection theorems: The `KSSemigroupBase` and `KnuthSkillingAlgebraBase` namespaces provide lemmas like `KSSemigroupBase.opAssoc` that extract the unbundled predicate from a bundled instance.

2.2 Iteration

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Core/Algebra.lean`

Definition 2.7 (`iterate_op`)**. Lines 23–25, Algebra.lean**

K&S Paper Reference: This corresponds to K&S's use of "n copies of x" in their proofs, written as x^n or nx depending on context (see K&S equations around line 370–380 in the TeX source). The iteration builds repeated applications of $\oplus$: $x^n = \underbrace{x \oplus x \oplus \cdots \oplus x}_{n \text{ times}}$.

Listing 5: Iteration definition

```
def iterate_op (x : alpha) : ℕ → alpha
  | 0 => ident
  | n + 1 => op x (iterate_op x n)
```

This builds the sequence: $\mathrm{ident}, x, x \oplus x, x \oplus (x \oplus x), \ldots$

2.3 Identity-Free Iteration

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Core/Basic.lean`

Definition 2.8 (`iterate_op_pnat`)**. Lines 375–376, Core/Basic.lean**

For identity-free reasoning, we define iteration using positive natural numbers ($\mathbb{N}^+$) instead of $\mathbb{N}$. This works on the weaker `KSSemigroupBase` (no identity required).

Listing 6: Identity-free iteration

```
def iterate_op_pnat [KSSemigroupBase alpha] (x : alpha) (n : ℕ+) : alpha :=
  iterate_op_pnat_aux x (n.val - 1)

private def iterate_op_pnat_aux (x : alpha) : ℕ → alpha
  | 0 => x          -- n=0 maps to x^1 = x
  | n + 1 => op x (iterate_op_pnat_aux x n)
```

3 No Anomalous Pairs and Separation

This section addresses the critical question: *what additional property, beyond order and associativity, is required to guarantee an additive embedding into* $(\mathbb{R}, +)$*?*

The additional property is necessary. A counterexample proves that order and associativity alone are insufficient:

Listing 7: Semidirect product countermodel (SemidirectNoSeparation.lean)

```
-- Type definitions (lines 38-40)
abbrev SDBase := PN ×ₗ N
abbrev SD := WithBot SDBase

-- Semidirect operation: (u,x) ⊕ (v,y) = (u+v, x + 2^u·y) (lines 52-59)
def baseOp (p q : SDBase) : SDBase :=
  (p.1 + q.1, p.2 + (N.pow 2 (p.1 : N)) * q.2)
def op : SD → SD → SD
  | ⊥, b => b | a, ⊥ => a | some p, some q => some (baseOp p q)

-- All K&S base axioms satisfied (lines 315-324)
instance : KnuthSkillingAlgebra SD where
  op := op; ident := ⊥
  op_assoc := op_assoc; op_ident_right := ...; op_ident_left := ...
  op_strictMono_left := ...; op_strictMono_right := ...; ident_le := ...

-- Key theorems: non-commutative, fails separation (lines 333, 447-449)
theorem op_not_comm : op exX exY ≠ op exY exX := by simp; decide
theorem not_KSSeparation : ¬ KSSeparation SD := ...
```

This structure satisfies associativity and strict monotonicity but fails both commutativity and separation—hence admits no additive embedding.

K&S's Appendix A proof is constructive: it builds a grid of values without assuming continuity, implicitly relying on a density requirement that the formalization makes explicit. Two roughly equivalent formulations of this requirement emerge:

- **No Anomalous Pairs (NAP)**: A classical condition from ordered semigroup theory [3, 4].
- **Separation**: A "sandwich" property discovered during formalization, capturing the density assumption implicit in K&S's constructive proof (see Appendix 12.2).

The formalization uses NAP as the *primary* axiom because it connects to 60+ years of classical algebra, yields a shorter and more elegant proof, and has a complete machine-checked proof via Eric Paul's `OrderedSemigroups` library.[1]

3.1 No Anomalous Pairs (Classical Formulation)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Additive/Axioms/AnomalousPairs.lean`

Definition 3.1 (Anomalous Pair)**.** Two elements a, b of an ordered semigroup form an *anomalous pair* if their iterates remain "squeezed" forever:

$$a^n < b^n < a^{n+1} \quad \text{for all } n \in \mathbb{N}^+$$

(or the symmetric condition $a^n > b^n > a^{n+1}$ for negative elements).

[1] `https://github.com/ericluap/OrderedSemigroups`

Definition 3.2 (`NoAnomalousPairs`)**.** An ordered semigroup has *no anomalous pairs* if no such pair exists.

Listing 8: NAP definition (following Eric Paul's OrderedSemigroups)

```
def AnomalousPair (a b : alpha) : Prop :=
  ∀ n : N+,
    (iterate_op_pnat a n < iterate_op_pnat b n /\
     iterate_op_pnat b n < iterate_op_pnat a (n + 1)) \/
    (iterate_op_pnat a n > iterate_op_pnat b n /\
     iterate_op_pnat b n > iterate_op_pnat a (n + 1))

class NoAnomalousPairs (alpha : Type*) [KSSemigroupBase alpha] : Prop where
  not_anomalous : ∀ a b : alpha, Not (AnomalousPair a b)
```

Historical context: For *groups*, Hölder [2] proved that Archimedean ordered groups embed into $(\mathbb{R}, +)$. For *semigroups* (without inverses), Alimov [3] identified "no anomalous pairs" as the precise generalization. Fuchs [4] provided the textbook treatment.

Theorem 3.3 (Hölder–Alimov [2, 3, 4])**.** *In a linearly ordered cancellative semigroup, the following are equivalent:*

1. *The semigroup has no anomalous pairs.*
2. *There exists an order-preserving additive embedding into* $(\mathbb{R}, +)$.

This classical result is formalized in Eric Paul's `OrderedSemigroups` library [6] and imported into the K&S development.

3.2 Separation (Formalization-Discovered Axiom)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Core/Algebra.lean`

Knuth & Skilling's Appendix A proof works constructively: they build a grid of values by introducing atoms one at a time, placing each new value at a "convenient" point within an interval. This implicitly assumes a *density* property—that for any interval (x, y), powers of a base element can be placed within it.

The formalization extracts this implicit assumption as the **separation property**—an axiom discovered by Claude Code during the formalization process and proposed by Ben Goertzel (see Appendix 12.2 for the discovery timeline):

Definition 3.4 (`KSSeparationSemigroup`)**.** For any positive elements a, x, y with $x < y$, there exist exponents $(n, m) \in \mathbb{N}^+$ such that:

$$x^m < a^n \leq y^m$$

Listing 9: Separation axiom (identity-free, Algebra.lean:312)

```
class KSSeparationSemigroup (alpha : Type*) [KSSemigroupBase alpha] where
  separation : ∀ {a x y : alpha},
    IsPositive a → IsPositive x → IsPositive y → x < y →
    ∃ n m : N+, iterate_op_pnat x m < iterate_op_pnat a n /\
                iterate_op_pnat a n ≤ iterate_op_pnat y m
```

Intuition: The separation property says that powers of any positive base element are "dense enough" to separate any two distinct elements. This is precisely the density that K&S's constructive proof requires to place new grid points.

3.3 Equivalence Theorem

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Additive/Axioms/AnomalousPairs.lean`

Under our standing hypotheses, the two formulations are equivalent:

Theorem 3.5 (Equivalence of NAP and Separation)**.** *For a linearly ordered cancellative semigroup with strictly monotone operation and identity as minimum:*

$$\mathit{KSSeparation} \iff \mathit{NoAnomalousPairs}$$

Proof sketch:

1. **Separation ⇒ NAP**: If (a, b) were anomalous with $a^n < b^n < a^{n+1}$ for all n, separation would provide witnesses (n, m) with $a^m < c^n \leq b^m$ for some base c—breaking the squeeze. (`noAnomalousPairs_of_KSSeparation_with_IdentMin`)

2. **NAP ⇒ Embedding**: The Hölder/Alimov theorem (Theorem 3.3), formalized via `OrderedSemigroups.holder_not_anom`.

3. **Embedding ⇒ Separation**: Rational density in $\mathbb{R}$ provides the sandwich witnesses. For any $x < y$ in the semigroup, their images $\Theta(x) < \Theta(y)$ in $\mathbb{R}$ have a rational p/q between them, which translates back to the required power witnesses.

3.3.1 Bilateral Separation

Without the assumption that identity is a minimum, the scale can have both "positive" and "negative" cones (under the additive embedding Θ). In Lean we express this identity-free using `IsPositive`/`IsNegative`. Bilateral separation is separation on each cone:

Listing 10: Bilateral separation (Core/Algebra.lean:415–426)

```
-- SeparationSemigroupProp: for IsPositive a, x, y and x < y, sandwich condition
-- SeparationNegProp: for IsNegative a, x, y and x < y, sandwich condition
def SeparationBilateralProp [KSSemigroupBase alpha] : Prop :=
  SeparationSemigroupProp ∧ SeparationNegProp
```

Remark 3.6 (Probability setting). Under `IdentIsMinimum`, `IsNegative` is impossible, so the negative branch is vacuous. In this setting, `SeparationBilateralProp` is equivalent to `SeparationSemigroupProp` (see Core/Algebra.lean:436–443).

3.4 Derived Properties

From NAP and Separation, we derive two key properties that K&S claim follow from order + associativity alone:

3.4.1 Archimedean Property and Commutativity

Both the Archimedean property and commutativity follow *directly* from NAP—they are not independent axioms. This is Alimov's theorem [3, 5].

Eric Paul's OrderedSemigroups library [6] formalizes these results:

Listing 11: NAP ⇒ Commutativity (Paul, Archimedean.lean:355–394)

```
/-- If a linear ordered cancel semigroup does not have an anomalous pair,
    then it is commutative. -/
theorem not_anomalous_pair_commutative
    (not_anomalous : ¬has_anomalous_pair (α := α)) (a b : α) :
    a * b = b * a := by
  -- Contrapositive: if a*b < b*a, the "gap" between them persists under
  -- powers, yielding (a*b)^n < (b*a)^n for all n. This divergence witnesses
  -- an anomalous pair. NAP forbids such gaps, forcing a*b = b*a.
  ...
```

Listing 12: NAP ⇒ Archimedean (Paul, Archimedean.lean:396–399)

```
theorem not_anomalous_arch (not_anomalous : ¬has_anomalous_pair (α := α)) :
    is_archimedean (α := α) := by
  have := mt (non_archimedean_anomalous_pair (α := α)) not_anomalous
  simpa
```

Our library proves the analogous results from KSSeparation:
File: `Additive/Axioms/SandwichSeparation.lean`

Listing 13: Separation ⇒ Commutativity (SandwichSeparation.lean:404)

```
theorem ksSeparation_implies_comm [KSSeparation α]
    (x y : α) (hx : ident < x) (hy : ident < y) :
    op x y = op y x := by
  -- Assume x⊕y < y⊕x. Apply separation to get (x⊕y)^m < a^n ≤ (y⊕x)^m.
  -- By associativity, both sides have the same "atom counts" → contradiction.
  ...
```

Listing 14: Separation ⇒ Archimedean (SandwichSeparation.lean:132–144)

```
/-- The Archimedean property follows from KSSeparation. -/
theorem op_archimedean_of_separation [KSSeparation α] (a x : α) (ha : ident < a) :
    ∃ n : ℕ, x < ℕ.iterate (op a) n a := by
  -- Get x ≤ a^k from separation, then x < a^{k+1} since a^k < a^{k+1}
  ...
```

Since KSSeparation ⇔ NAP (Theorem 3.5), these are equivalent routes.

4 The Representation Theorem (Appendix A)

Files:

- `Additive/Representation.lean` (interfaces: identity-free default)
- `Additive/Main.lean` (entrypoint: lightweight interface + instances)
- `Additive/Proofs/OrderedSemigroupEmbedding/HolderEmbedding.lean` (Hölder/Alimov embedding path)
- `Additive/Proofs/DirectCuts/Main.lean` (Dedekind cuts path)
- `Additive/Proofs/GridInduction/Main.lean` (Grid/induction path; K&S-style, large)
- `Additive/Proofs/GridInduction/Globalization.lean` (globalization orchestrator)
- `Additive/Proofs/GridInduction/Core/` (grid infrastructure)

- `Core/MultiGrid.lean`: `AtomFamily`, `MultiGridRep`, grid representations
- `Core/Induction/`: Inductive extension theorems (`Construction`, `ThetaPrime`, `DeltaShift`, `Goertzel`)
- `Core/OneDimensional.lean`: Base case (single atom)
- `Core/Prelude.lean`: Foundational lemmas

The grid path entry point is `Globalization.lean`, which imports `Core/All.lean`.

4.1 Three Independent Proof Paths

The formalization provides **three complete, independent proof routes** to the representation theorem:

1. **Hölder embedding** (shortest route): Uses the NoAnomalousPairs condition and classical ordered semigroup theory [2, 3, 4]. Formalized via Eric Paul's `OrderedSemigroups` [6].
2. **Dedekind cuts** (alternative): Uses Separation property with Hölder/Dedekind cuts construction, bypassing the grid machinery.
3. **Grid induction** (K&S-style): Uses multi-dimensional grid representations and induction on atom families, following K&S's original approach.

Historical development:

- First: Grid/induction path (following K&S's original approach)
- Second: Cuts path discovered by Claude Code and Codex
- Third: Hölder/Alimov path discovered by GPT-5.2 Pro (classical connection)

All three proofs are complete. The Hölder path uses **NoAnomalousPairs only**—connecting to classical ordered semigroup theory.

4.2 Proof Architecture 1: The Hölder Path (Recommended)

File: `Additive/Proofs/OrderedSemigroupEmbedding/HolderEmbedding.lean`

The Hölder path is **recommended** because it is the shortest and connects to classical ordered semigroup theory.

Theorem 4.1 (`holder_embedding_of_noAnomalousPairs`). ***Lines 166–169, HolderEmbedding.lean***

If a K&S algebra has no anomalous pairs, it embeds into $\mathbb{R}$.

Listing 15: Hölder embedding theorem

```
theorem holder_embedding_of_noAnomalousPairs [NoAnomalousPairs alpha] :
    ∃ G : Subsemigroup (Multiplicative R), Nonempty (alpha =(equiv)*o G) := by
  have h : ~has_anomalous_pair (alpha := alpha) := noAnomalousPairs_iff_eric
  exact holder_not_anom h
```

Theorem 4.2 (`representation_semigroup`). ***Lines 272–278, HolderEmbedding.lean***

Identity-free representation*: NoAnomalousPairs implies additive embedding into* $\mathbb{R}$. Θ *is defined up to an additive constant (no canonical zero point).*

Listing 16: Identity-free representation theorem

```
theorem representation_semigroup [NoAnomalousPairs alpha] :
    ∃ Theta : alpha → R,
      (∀ a b : alpha, a ≤ b ↔ Theta a ≤ Theta b) /\
      (∀ x y : alpha, Theta (op x y) = Theta x + Theta y) := by
  obtain <G, <iso>> := holder_embedding_of_noAnomalousPairs (alpha := alpha)
  use theta_from_embedding G iso
  exact <theta_preserves_order G iso, theta_additive G iso>
```

Theorem 4.3 (`representation_from_noAnomalousPairs`)**.** ***Lines 300–307, HolderEmbedding.lean***

With identity*: NoAnomalousPairs implies the full representation with* $\Theta(ident) = 0$*.*

Listing 17: Representation with identity normalization

```
theorem representation_from_noAnomalousPairs [NoAnomalousPairs alpha] :
    ∃ Theta : alpha → R,
      (∀ a b : alpha, a ≤ b ↔ Theta a ≤ Theta b) /\
      Theta ident = 0 /\
      ∀ x y : alpha, Theta (op x y) = Theta x + Theta y := by
  obtain <G, <iso>> := holder_embedding_of_noAnomalousPairs (alpha := alpha)
  use theta_from_embedding G iso
  exact <theta_preserves_order G iso, theta_ident G iso, theta_additive G iso>
```

Key advantage: Identity-free semigroup version: `representation_semigroup` (for `KSSemigroupBase`). With identity, the same path yields `representation_from_noAnomalousPairs` (for `KnuthSkillingAlgebraBase`).

4.3 Proof Architecture 2: The Grid/Induction Path

The grid-based proof is packaged as the typeclass `RepresentationGlobalization`, which is automatically instantiated when `[KSSeparationStrict α]` is available.

Definition 4.4 (`RepresentationGlobalization`)**. Lines 54–60, Globalization.lean**

A typeclass packaging the existence of Θ.

Listing 18: RepresentationGlobalization typeclass

```
class RepresentationGlobalization (alpha : Type*)
    [KnuthSkillingAlgebra alpha] [KSSeparation alpha] : Prop where
  ∃_Theta :
    ∃ Theta : alpha → R,
      (∀ a b : alpha, a ≤ b ↔ Theta a ≤ Theta b) /\
      Theta ident = 0 /\
      ∀ x y : alpha, Theta (op x y) = Theta x + Theta y
```

4.3.1 The Globalization Construction ("Triple Family Trick")

The instance `representationGlobalization_of_KSSeparationStrict` (lines 93–850, Globalization.lean) constructs Θ globally using a multi-step process:

1. **Reference atom**: Choose any $a_0 >$ ident as a fixed reference point.

2. **2-atom families**: For each $x >$ ident, build a 2-atom family $F_2 = \{a_0, x\}$ with a `MultiGridRep` R_2 (via `extend_grid_rep_with_atom_of_KSSeparationStrict` from `Core/`).

3. **Define** $\Theta(x)$: Extract the representation value from the grid:

$$\Theta(x) := R_2.\text{Theta\_grid}(\langle x, \text{membership\_proof}\rangle)$$

4. **Well-definedness**: Use 3-atom families $F_3 = \{a_0, a_1, x\}$ to show that $\Theta(x)$ does not depend on the choice of reference atom. Path independence follows from `DeltaSpec_unique` (line 755, `Core/Induction/Construction.lean`).

5. **Order preservation**: For $a < b$, build $F_3 = \{a_0, a, b\}$ and use `MultiGridRep.strictMono` to show $\Theta(a) < \Theta(b)$.

6. **Additivity**: For $x \oplus y$, build $F_3 = \{a_0, x, y\}$ and verify $\Theta(x \oplus y) = \Theta(x) + \Theta(y)$ by path independence across different extension orderings.

Remark 4.5 (Why "Triple Family Trick"?). The name comes from using 3-atom families to mediate between different 2-atom constructions. This technique ensures global consistency: any two definitions of $\Theta(x)$ via different reference atoms must agree, because they both embed into a common 3-atom grid representation.

Remark 4.6 (Identity-Free Grid Infrastructure). The grid construction has **parametric versions** that could work without identity:

- `mu_param F r base`: Grid valuation with explicit base element instead of `ident`
- `kGrid_param F base`: Grid set using `mu_param`
- `mu_pnat`, `kGrid_pnat`: Truly identity-free using $\mathbb{N}^+$ iteration (no 0 exponents)
- `RepresentationGlobalizationAnchor`: Class for representations normalizing to an arbitrary anchor

Currently, the globalization instance uses identity: `representationGlobalization_of_KSSeparationStrict`. An identity-free instance using the parametric infrastructure is marked as **future work** in `Globalization.lean`.

For identity-free representations **today**, use the Hölder path (`HolderEmbedding.lean`) which produces `RepresentationResult` (order + additivity, no normalization constraint).

4.4 Main Theorem Statement

Theorem 4.7 (`associativity_representation`)**.** ***Lines 54–60, Additive/Proofs/GridInduction/Main.lean***

K&S Appendix A Main Theorem*: There exists an order embedding* $\Theta : \alpha \to \mathbb{R}$ *such that:*

1. *Order preservation:* $a \leq b \Leftrightarrow \Theta(a) \leq \Theta(b)$
2. $\Theta(\mathit{ident}) = 0$
3. *Additivity:* $\Theta(\mathit{op}\ x\ y) = \Theta(x) + \Theta(y)$

Listing 19: Appendix A Representation Theorem (public API)

```
theorem associativity_representation
    (alpha : Type*) [KnuthSkillingMonoidBase alpha] [KSSeparation alpha]
    [RepresentationGlobalization alpha] :
    ∃ Theta : alpha → R,
      (∀ a b : alpha, a ≤ b ↔ Theta a ≤ Theta b) /\
      Theta ident = 0 /\
      ∀ x y : alpha, Theta (op x y) = Theta x + Theta y := by
  exact RepresentationGlobalization.∃_Theta (alpha := alpha)
```

Remark 4.8 (Proof Delegation). The theorem statement simply extracts `exists_Theta` from the typeclass. All the actual work happens in the instance construction:

`representationGlobalization_of_KSSeparationStrict`

(starts at line 105, `Additive/Proofs/GridInduction/Globalization.lean`)

This design keeps the public API clean while hiding the complex globalization machinery.

4.5 Proof Architecture 2: The Direct Cuts Path

File: `Additive/Proofs/DirectCuts/DirectCuts.lean`

The DirectCuts path provides **both identity-based and identity-free** versions using Dedekind cuts:

- **Identity-free**: Uses `Theta_cuts_pnat` with $\mathbb{N}^+$ iteration
  - `Theta_cuts_pnat` (line 1434): Definition via Dedekind cuts using $\mathbb{N}^+$ iteration
  - `Theta_cuts_pnat_strictMono` (line 1530): Strict monotonicity (fully proven)
  - `Theta_cuts_pnat_add` (line 1636): Additivity (fully proven)
  - No reference to `ident` anywhere
- **Identity-based** (§9a): Uses `Theta_cuts` with $\mathbb{N}$ iteration
  - `iterate_op x 0 = ident` for the base case
  - `ident` as the canonical reference point
  - Produces `RepresentationResult` satisfying $\Theta(\text{ident}) = 0$

The cuts construction uses a classical Hölder/Dedekind approach (shown here for the identity-based version; the identity-free version uses `IsPositive` instead of comparing to `ident`):

1. **Fix base element**: Choose any $a_0 >$ ident as a reference point (identity-free: choose any a_0 with `IsPositive` a_0)
2. **Define rational approximants**: For any $x \in \alpha$, consider the set of ratios $m/n \in \mathbb{Q}$ where $a_0^m \leq x^n$ (equivalently, $m \cdot a_0 \leq n \cdot x$ in additive notation)
3. **Define $\Theta(x)$ by supremum in $\mathbb{R}$**:

$$\Theta_{\text{cuts}}(x) := \sup_{\mathbb{R}} \{m/n \in \mathbb{Q} : a_0^m \leq x^n,\ n > 0\}$$

   where the supremum is taken in $\mathbb{R}$ (which is already complete from Mathlib). The cut set is defined in α using the order relation, but the supremum is computed in $\mathbb{R}$.

4. **Prove properties**:

   - **Order preservation**: If $x < y$, then for any m/n in the cut of x, there exists m'/n' in the cut of y with $m/n < m'/n'$ (uses KSSeparation to find witnesses)
   - **Additivity**: $\Theta(x \oplus y) = \Theta(x) + \Theta(y)$ follows from $a_0^{m_1+m_2} \leq (x \oplus y)^{n_1 \cdot n_2}$ iff $a_0^{m_1} \leq x^{n_1}$ and $a_0^{m_2} \leq y^{n_2}$ (uses commutativity and associativity)

Remark 4.9 (No Circularity). This construction does **not** require completing α into $\mathbb{R}$ first. Instead:

- The set $\{m/n \in \mathbb{Q} : a_0^m \leq x^n\}$ is defined using the order relation in α
- These rationals are cast to $\mathbb{R}$: `(↑) '' cutSet` a x `: Set` $\mathbb{R}$
- The supremum is computed in $\mathbb{R}$ using `sSup` (conditional supremum from Mathlib)

Thus $\Theta : \alpha \to \mathbb{R}$ is directly defined without requiring α to already embed into $\mathbb{R}$.

Theorem 4.10 (associativity_representation_cuts). ***Lines 44–71, Additive/Proofs/DirectCuts/Main.lean***

The cuts-based representation theorem.

Listing 20: Appendix A (cuts proof)

```
theorem associativity_representation_cuts
    (alpha : Type*) [KnuthSkillingAlgebra alpha] [KSSeparation alpha]
    [KSSeparationStrict alpha] :
    ∃ Theta : alpha → R,
      (∀ a b : alpha, a ≤ b ↔ Theta a ≤ Theta b) /\
      Theta ident = 0 /\
      ∀ x y : alpha, Theta (op x y) = Theta x + Theta y := by
  -- Use Theta_cuts (the Dedekind-cuts construction)
  obtain <a0, ha0> := <witness for non-trivial element>
  refine <Theta_cuts a0 ha0, order_preservation, identity, additivity>
```

Remark 4.11 (Comparison to Grid Proof). The cuts proof is significantly more compact:

- **Grid proof**: ∼2000+ lines (induction machinery, extension lemmas, path independence)
- **Cuts proof**: ∼500 lines (direct construction, no induction)

However, the grid proof more closely follows K&S's original argument structure (A/B/C partition, δ-choice), while the cuts proof uses the standard Hölder technique from ordered group theory.

Corollary 4.12 (op_comm_of_associativity). ***Lines 87–92, Additive/Proofs/GridInduction/-Main.lean***

Commutativity follows from the representation theorem.

Listing 21: Commutativity from representation

```
theorem op_comm_of_associativity
    (alpha : Type*) [KnuthSkillingMonoidBase alpha] [KSSeparation alpha]
    [RepresentationGlobalization alpha] :
    ∀ x y : alpha, op x y = op y x := by
  classical
  obtain <Theta, hTheta_order, _, hTheta_add> := associativity_representation (
      alpha := alpha)
  exact commutativity_from_representation Theta hTheta_order hTheta_add
```

5 The Product Theorem (Appendix B)

Files:

- `Multiplicative/Main.lean` (K&S's Appendix B pipeline via Appendix A)
- `Multiplicative/Proofs/Direct/DirectProof.lean` (Alternative: direct algebraic path)
- `Multiplicative/ScaledMultRep.lean` (Common interface for both paths)

5.1 Two Complete Proof Paths

Like Appendix A, the formalization provides **two independent proofs** of Appendix B's conclusion. Both paths arrive at the same result: the tensor operation $\otimes$ on positive reals equals multiplication up to a global scale constant.

5.1.1 Path 1: K&S's Derivation

`Multiplicative/Main.lean` follows K&S's paper exactly: "apply Appendix A again to $\otimes$". This path uses `AdditiveOrderIsoRep` (from Appendix A) to derive the product equation, then solves it to show $\otimes$ is scaled multiplication.

5.1.2 Path 2: Direct Algebraic Proof

`Multiplicative/Proofs/Direct/DirectProof.lean` provides a direct algebraic proof:

Listing 22: Direct proof: Axioms 3–4 + commutativity $\Rightarrow$ scaled multiplication (DirectProof.lean:471)

```
/-- If tensor satisfies Axioms 3-4 and is commutative, then it is scaled
    multiplication. -/
theorem tensor_coe_eq_mul_div_const_of_assoc_of_distrib_of_comm
    (hAssoc : ∀ u v w, tensor (tensor u v) w = tensor u (tensor v w))
    (hDistrib : DistributesOverAdd tensor)
    (hComm : ∀ u v, tensor u v = tensor v u) :
    ∃ C : R, 0 < C /\ ∀ x y,
      ((tensor x y) : R) = ((x : R) * (y : R)) / C := ...
```

Remark 5.1 (Why Two Paths?).

- **Path 1 (K&S)**: Uses `AdditiveOrderIsoRep` for the tensor ("apply Appendix A again")
- **Path 2 (Direct)**: Derives result directly from distributivity + associativity + commutativity
- Both arrive at the same conclusion: $\otimes$ is scaled multiplication

5.2 Common Interface: ScaledMultRep

Both paths provide the `ScaledMultRep` interface, which captures the OUTPUT of Appendix B:

Listing 23: ScaledMultRep interface (Multiplicative/ScaledMultRep.lean:44)

```
structure ScaledMultRep (tensor : PosR → PosR → PosR) where
  C : R                  -- The scale constant C > 0
  C_pos : 0 < C
  tensor_eq : ∀ x y : PosR,
    ((tensor x y) : R) = ((x : R) * (y : R)) / C
```

Design principle: Like `AdditiveOrderIsoRep` for Appendix A, this interface captures WHAT Appendix B proves without depending on HOW it was proven. Downstream code should depend on `ScaledMultRep`, not on specific proof paths.

Constructors:

- `scaledMultRep_of_additiveOrderIsoRep`: K&S path (uses Appendix A)
- `scaledMultRep_of_tensorRegularity`: Direct path (bypasses Appendix A)
- `scaledMultRep_of_assoc_distrib_comm`: Minimal assumptions (assoc + distrib + comm)

5.3 Direct Product Symmetries (3–4)

K&S paper location: Symmetry 3 appears at equation (7) on page 6 (arxiv.tex lines 462–467), Axiom 3 at equation (24) on page 9 (arxiv.tex lines 566–572).

Before applying Appendix A, K&S work with lattice elements and the direct-product operator $\times$. **Symmetry 3** states that $\times$ is (right-)distributive over the join $\sqcup$:

$$(x \times t) \sqcup (y \times t) = (x \sqcup y) \times t$$

After Appendix A provides the representation $\Theta : \alpha \to \mathbb{R}$, we work with graded measures and the tensor operation $\otimes$. **Axiom 3** is the graded version:

$$(x \otimes t) \oplus (y \otimes t) = (x \oplus y) \otimes t$$

After moving to real numbers via Θ, this becomes (Appendix B, arxiv.tex line 661):

$$x \otimes t + y \otimes t = (x + y) \otimes t$$

where $+$ is real addition (since $\oplus$ has been identified with $+$ by Appendix A).

Symmetry 4 (Product Associativity): $(u \otimes v) \otimes w = u \otimes (v \otimes w)$

Formalization note: We have lattice-level Symmetry 3 in `DirectProduct.prod_sup_left`:

```
-- Multiplicative/DirectProduct.lean, line 42
prod_sup_left : ∀ a1 a2 : alpha, ∀ b : beta,
  prod (a1 || a2) b = prod a1 b || prod a2 b  -- || denotes sup
```

At the graded level, we define `DistributesOverAdd` as a property:

- `DistributesOverAdd` is a predicate that a tensor may or may not satisfy
- `TensorAlgebra` is the bundled class including this property
- The derivation of `DistributesOverAdd` from `prod_sup_left` is now **fully formalized** in `DistributivityDerivation.lean`

Listing 24: Distributivity property (Multiplicative/Basic.lean:68)

```
-- Defines the PROPERTY (not an axiom, just a predicate)
def DistributesOverAdd (tensor : PosR → PosR → PosR) : Prop :=
  ∀ x y t : PosR, tensor (addPos x y) t = addPos (tensor x t) (tensor y t)

-- Then we ASSUME some tensor satisfies this property:
variable (hDistrib : DistributesOverAdd tensor)
```

Remark 5.2 (Connection between lattice and graded levels). In K&S's development:

1. Symmetry 3 is stated at the lattice level: $(x \times t) \sqcup (y \times t) = (x \sqcup y) \times t$
2. After Appendix A provides the representation Θ, this gives distributivity at the graded level
3. The graded tensor satisfies `DistributesOverAdd`

Current state: We have *all three* levels formalized:

- Lattice level: `DirectProduct.prod_sup_left` (Multiplicative/DirectProduct.lean)
- Graded level: `DistributesOverAdd` (Multiplicative/Basic.lean)
- Bridge: `distributes_over_add_from_lattice` (Multiplicative/DistributivityDerivation.lean)

Derivation: The theorem `distributes_over_add_from_lattice` proves that `DistributesOverAdd` follows from:

1. `prod_sup_left`: Lattice-level distributivity
2. `disjoint_prod_left`: Disjointness preservation
3. `sum_rule`: Valuation additivity on disjoint events (from `CoxConsistency`)
4. `RectTensorCompatible`: Bridge predicate $v(\text{prod } a\ b) = \text{tensor}(v(a), v(b))$

This shows that scalar distributivity is **derived**, not assumed!

Remark 5.3 (Unbundled Tensor Predicates and TensorAlgebra). **Lines 59–123, Multiplicative/Basic.lean**

Following the unified axiom organization, tensor properties have both unbundled predicates and a bundled class:

Unbundled predicates:

- `TensorAssoc tensor` (line 72): Associativity of $\otimes$
- `TensorPos tensor` (line 77): Positivity-preserving
- `TensorStrictMonoLeft tensor` (line 81): Strict monotonicity in left argument
- `TensorStrictMonoRight tensor` (line 85): Strict monotonicity in right argument
- `DistributesOverAdd tensor` (line 68): Distributivity over $+$

Bundled class (line 103):

```
class TensorAlgebra (tensor : PosR → PosR → PosR) : Prop where
  distributes : DistributesOverAdd tensor
  assoc : TensorAssoc tensor
  pos : TensorPos tensor
```

Convenience theorem (line 247): `productEquation_of_tensorAlgebra` provides an ergonomic entry point for proofs that use all the bundled axioms together.

Design principle: Use unbundled predicates (e.g., `hDistrib : DistributesOverAdd tensor`) when tracking minimal hypotheses. Use `[TensorAlgebra tensor]` for ergonomic access in longer proofs.

5.4 Product Equation

Appendix B shows $\otimes$ must be multiplication up to a global scale.

Theorem 5.4 (`Psi_is_exp`). ***Line 43, Multiplicative/Main.lean***

The inverse representation $\Psi = \Theta^{-1}$ is exponential: $\Psi(x) = C \cdot e^{Ax}$ for some constants $C > 0$ and A.

Listing 25: Appendix B: Ψ is exponential

```
theorem Psi_is_exp
    (hRep : AdditiveOrderIsoRep PosR tensor)
    (hDistrib : DistributesOverAdd tensor) :
    ∃ (C A : R), 0 < C /\ ∀ x : R, Derived.Psi hRep x = C * R.exp (A * x)
    := by
  refine
    productEquation_solution_of_continuous_strictMono
      (hEq := productEquation_Psi (tensor := tensor) hRep hDistrib)
      (hPos := fun x => Derived.Psi_pos (tensor := tensor) hRep x)
      (hCont := Derived.Psi_continuous (tensor := tensor) hRep)
      (hMono := Derived.Psi_strictMono (tensor := tensor) hRep)
```

Remark 5.5 (The Functional Equation Proof). The proof delegates to:
`productEquation_solution_of_continuous_strictMono`
(line 294, `Multiplicative/FunctionalEquation.lean`)

This proves a **classical result from functional equations theory**:

Statement: If $\Psi : \mathbb{R} \to \mathbb{R}$ satisfies the product equation

$$\Psi(\tau + \xi) + \Psi(\tau + \eta) = \Psi(\tau + \zeta(\xi, \eta))$$

for all $\tau, \xi, \eta \in \mathbb{R}$, and if Ψ is positive, continuous, and strictly monotone, then $\Psi(x) = C \cdot e^{Ax}$ for some constants $C > 0$ and A.

Key steps (561 lines):

1. Extract shift constant: $a := \zeta(0, 0)$ gives $\Psi(x + a) = 2\Psi(x)$
2. Extend to powers: $\Psi(x + na) = 2^n \Psi(x)$ for all $n \in \mathbb{Z}$
3. Extend to rationals: $\Psi(x + (m/n)a) = 2^{m/n}\Psi(x)$ for all $m/n \in \mathbb{Q}$
4. Use continuity + density: Extend to all reals
5. Conclude: $\Psi(x) = C \cdot 2^{x/a} = C \cdot e^{(\ln 2/a) \cdot x}$

The continuity and monotonicity hypotheses are **derived** (not assumed) from the order isomorphism $\Theta : \text{PosReal} \simeq_o \mathbb{R}$ established in Appendix A:

- `Psi_strictMono` (line 148, `Multiplicative/Basic.lean`): Since $\Psi := \Theta^{-1}$ and Θ is an order isomorphism, Θ^{-1} is strictly monotone.
- `Psi_continuous` (line 154, `Multiplicative/Basic.lean`): Order isomorphisms $\mathbb{R} \simeq_o \mathbb{R}$ are continuous (order topology).

Why Exponential Ψ Implies Tensor = Scaled Multiplication

Given: $\Theta(x \otimes y) = \Theta(x) + \Theta(y)$ (additivity) and $\Psi = \Theta^{-1}$ with $\Psi(z) = C \cdot e^{Az}$

Derivation:

$$x = \Psi(\Theta(x)) = C \cdot e^{A \cdot \Theta(x)} \quad \Rightarrow \quad e^{A \cdot \Theta(x)} = x/C$$

$$\begin{aligned} x \otimes y &= \Psi(\Theta(x \otimes y)) = \Psi(\Theta(x) + \Theta(y)) \\ &= C \cdot e^{A(\Theta(x)+\Theta(y))} = C \cdot e^{A \cdot \Theta(x)} \cdot e^{A \cdot \Theta(y)} \\ &= C \cdot (x/C) \cdot (y/C) = \frac{x \cdot y}{C} \end{aligned}$$

Conclusion: $x \otimes y = (x \cdot y)/C$ (Lean: `tensor_coe_eq_mul_div_const`, line 61)

Theorem 5.6 (`tensor_mul_rule_normalized`)**.** ***Line 105, Multiplicative/Main.lean***
The tensor operation is multiplication up to a global constant: $(x \otimes y)/C = (x/C) \cdot (y/C)$.

Listing 26: Product rule (normalized)

```
theorem tensor_mul_rule_normalized
    (hRep : AdditiveOrderIsoRep PosR tensor)
    (hDistrib : DistributesOverAdd tensor) :
    ∃ C : R, 0 < C /\
      (∀ x : PosR, 0 < ((x : R) / C)) /\
      (∀ x y : PosR,
        ((tensor x y : PosR) : R) / C = (((x : R) / C) * ((y : R) / C)))
```

6 The Variational Theorem (Appendix C)

Primary file: `Variational/Main.lean`

6.1 What Appendix C Derives

The result: Any potential function H satisfying the variational constraints must have the form:

$$H(m) = A + Bm + C(m \log m - m)$$

This is the **entropy/divergence normal form**—derived from structure, not assumed.

Two proof paths in the code:

1. **Path B (derivation)**: Lagrange multiplier analysis at a stationary point yields a *local* separated equation [K&S approach].
2. **Path A (solver)**: Given a *global* separated equation plus regularity, solve it to get $H'(m) = B + C \log m$.

Two *hypothesis gates* connect these paths:

- **Gate G (globality)**: upgrades the local equation from Path B to a global one.
- **Gate R (regularity)**: excludes pathological (Hamel-basis) solutions to force the logarithmic form.

6.2 Path A: The Functional Equation Solver

Path A takes as input a *global* functional equation and solves it.

Definition 6.1 (`VariationalEquation`)**. Lines 206–207, Variational/Main.lean**
The separated variational equation: $H'(m_x \cdot m_y) = \lambda(m_x) + \mu(m_y)$ for all positive m_x, m_y.

Listing 27: Separated variational equation

```
206 def VariationalEquation (H' lam mu : R → R) : Prop :=
207   ∀ m_x m_y : R, 0 < m_x → 0 < m_y → H' (m_x * m_y) = lam m_x + mu m_y
```

Theorem 6.2 (`variationalEquation_solution_measurable`)**.** ***Lines 315–380, Variational/-Main.lean***
Gate R (regularity)*: Under Borel measurability, the only solutions are logarithmic.*

Listing 28: Path A rigidity theorem: measurability forces log

```
315 theorem variationalEquation_solution_measurable
316     (H' : R → R) (lam mu : R → R)
317     (hMeas : Measurable H')
318     (hV : VariationalEquation H' lam mu) :
319     ∃ B C : R, ∀ m : R, 0 < m → H' m = B + C * R.log m := by
```

Proof idea: Transform via $u = \log m$ to Cauchy's additive equation $f(u+v) = f(u) + f(v)$. Measurable solutions to Cauchy are linear, giving $H'(m) = B + C \log m$.

Regularity variants (same conclusion, different Gate R):

- `variationalEquation_solution_monotone` (line 420): monotonicity ⇒ measurability
- `variationalEquation_solution` (line 386): `deriv H` is automatically measurable

Honesty note: K&S motivate regularity via a convolution/"blurring" argument. The Lean development does *not* formalize that argument (too vague to formalize, not disproved); it makes Gate R explicit instead.

6.3 Path B: Lagrange Multiplier Derivation (Local)

Path B derives a separated equation at a *single stationary point* of a constrained optimization.

Setup: Maximize $\sum_{x,y} H(m_{xy})$ subject to row and column constraints, using Lagrange multipliers α_x, β_y.

Theorem 6.3 (`lagrange_coordinate_deriv_eq`)**.** ***Lines 699–709, Variational/Main.lean***
At a stationary point, the derivative separates into x-only and y-only terms.

Listing 29: Path B: local separated equation at stationary point

```
699 theorem lagrange_coordinate_deriv_eq
700     (H : R → R) (g : X → R → R) (h : Y → R → R) (alpha : X → R) (beta : Y → R)
701     (m : X * Y → R) (x0 : X) (y0 : Y)
702     (hH : DifferentiableAt R H (m (x0, y0)))
703     (hg : DifferentiableAt R (g x0) (rowSum m x0))
704     (hh : DifferentiableAt R (h y0) (colSum m y0))
705     (hcrit : HasDerivAt
706       (fun t => productLagrangian H g h alpha beta (Function.update m (x0, y0) t))
```

```
      0 (m (x0, y0))) :
    deriv H (m (x0, y0))
      = alpha x0 * deriv (g x0) (rowSum m x0) + beta y0 * deriv (h y0) (colSum m
          y0) := by
```

Key point: This equation holds at *one* coordinate (x_0, y_0) of *one* stationary point. It is **local**, not global.

6.4 Gate G: Bridging Local to Global

To apply the Path A solver, we need the separated equation to hold *globally* for all positive pairs—not just at one stationary point. K&S argue informally that the same functional form must hold across "all applications." In Lean, this is an explicit hypothesis:

Definition 6.4 (`KSVariationalGenerality`)**. Lines 474–477, Variational/Main.lean**
Gate G packaged as a hypothesis: the equation holds globally.

Listing 30: Gate G: explicit globality hypothesis

```
structure KSVariationalGenerality (H : R → R) where
  lam : R → R
  mu : R → R
  equation : VariationalEquation (deriv H) lam mu
```

The logical flow:

$$\text{Path B (local)} \xrightarrow{\text{Gate G}} \text{Global equation} \xrightarrow{\text{Path A + Gate R}} H'(m) = B + C \log m$$

6.5 Integration: From Derivative to Entropy Form

Once $H'(m) = B + C \log m$ is forced, integrating yields the normal form.

Listing 31: Entropy derivative and form (lines 292, 490)

```
noncomputable def entropyDerivative (B C : R) : R → R := fun m => B + C * R.log m

noncomputable def entropyForm (A B C : R) : R → R :=
  fun m => A + B * m + C * (m * R.log m - m)
```

Theorem 6.5 (`entropyForm_deriv`)**.** ***Lines 493–517****: Verifies* $\frac{d}{dm}[A + Bm + C(m \log m - m)] = B + C \log m$*.*

6.6 Alternative Route: Shore–Johnson Axioms

Shore–Johnson (1980) derive the *same* functional equation via *different* axioms.

Their axioms (SJ1–SJ4): consistency requirements on inference methods, especially **system independence** (SJ4): inference on independent subsystems should factor.

Their derivation: SJ4 applied to Dirac test distributions forces the multiplicative Cauchy equation $g(xy) = g(x) + g(y)$.

The connection: This is the *special case* $\lambda = \mu = g$ of the variational equation. Same Gate R (measurability) forces the same logarithmic solution.

Theorem 6.6 (`mulCauchyOnPos_eq_const_mul_log`)**.** ***Lines 33–65, Bridges/ShoreJohnson-VariationalBridge.lean***

Listing 32: Shore–Johnson as special case of Path A solver

```
theorem mulCauchyOnPos_eq_const_mul_log_of_variationalEquation_solution_measurable
    (g : R → R) (hg : MulCauchyOnPos g) (hMeas : Measurable g) :
    ∃ C : R, ∀ x : R, 0 < x → g x = C * R.log x := by
```

Summary: K&S (variational) and Shore–Johnson (axiomatic) are two routes to the same functional equation. Both require the same regularity gate; both yield the same logarithmic solution.

See Appendix 12.2 for the full Shore–Johnson formalization.

7 Divergence (K&S Section 6.1)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Information/Divergence.lean`

What this is: The divergence $\phi(w, u)$ measures the "distance" between two measure assignments w and u. It quantifies the information-theoretic cost of using measure u when the "true" measure is w.

Connection to Appendix C (Entropy Form): The divergence is a **special case** of the entropy form from the variational theorem:

$$H(m) = A + Bm + C(m \log m - m) \quad \text{(Appendix C)}$$
$$\phi(w, u) = u - w + w \log(w/u) \quad \text{(Divergence)}$$

Setting $A = u$, $B = -\log(u)$, $C = 1$ in the entropy form gives the divergence (K&S Eq. 44). The critical point analysis from Appendix C proves that $\phi(w, u)$ is minimized when $w = u$.

Note: This is **atom divergence** for general real-valued measures. The specialization to **probability distributions** happens in Section 11, after Section 9 derives what probability distributions are.

Key properties:

- **Non-negative**: $\phi(w, u) \geq 0$, with equality iff $w = u$ (formalized in `atomDivergence_nonneg`, lines 102–120)
- **Asymmetric**: $\phi(w, u) \neq \phi(u, w)$ in general (it's NOT a distance metric)
- **Connects variational calculus to information theory**: Bridge between Appendix C and Section 8

Forward reference: This atom divergence will be specialized to **probability distributions** in Section 11 (Information and Entropy), giving the Kullback-Leibler divergence formula.

Definition 7.1 (`atomDivergence`)**. Lines 68–69, Divergence.lean**

The per-atom divergence: $\phi(w, u) = u - w + w \log(w/u)$.

Listing 33: Atom divergence

```
noncomputable def atomDivergence (w u : R) : R :=
  u - w + w * log (w / u)
```

Theorem 7.2 (`atomDivergence_nonneg`)**.** ***Lines 102–120, Divergence.lean***

Listing 34: Divergence non-negativity

```
theorem atomDivergence_nonneg (w u : ℝ) (hw : 0 < w) (hu : 0 < u) :
    0 ≤ atomDivergence w u := by
  unfold atomDivergence
  -- Rewrite as w * (u/w - 1 - log(u/w)) and use log inequality
  let s := u / w
  have hs : 0 < s := div_pos hu hw
  have hrewrite : u - w + w * log (w / u) = w * (s - 1 - log s) := by ...
  rw [hrewrite]
  exact mul_nonneg (le_of_lt hw) (log_ineq s hs)
```

Theorem 7.3 (`atomDivergence_eq_zero_iff`)**.** ***Lines 123–159, Divergence.lean***
Divergence equals zero if and only if $w = u$.

Listing 35: Divergence equals zero iff $w = u$ (Divergence.lean:123)

```
theorem atomDivergence_eq_zero_iff (w u : ℝ) (hw : 0 < w) (hu : 0 < u) :
    atomDivergence w u = 0 ↔ w = u := by
  constructor
  . -- If phi(w,u) = 0, then w = u
    intro h
    let s := u / w
    have hs : 0 < s := div_pos hu hw
    -- phi(w,u) = w * (s - 1 - log s) = 0
    -- w > 0 implies s - 1 - log s = 0
    -- But s - 1 - log s > 0 for s ≠ 1 (strict log inequality)
    -- So s = 1, hence u = w
    ...
  .
    intro heq
    rw [heq]
    exact atomDivergence_self u hu
```

8 Conditional Probability Calculus (K&S Section 7)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Probability/ConditionalProbability/Basic.lean`

What this section does: K&S Section 7 derives **probability calculus** from first principles. This is the crucial step that takes us from general measures (Sections 1–6) to **probability distributions** (normalized measures).

Starting with conditional plausibility as a **bivaluation** $p(x|t)$ (a function taking pairs of lattice elements to reals), K&S introduces **Axiom 5 (Chaining Associativity)** and proves:

1. The **chain-product rule**: $\Pr(a|c) = \Pr(a|b) \cdot \Pr(b|c)$ for chains $a \leq b \leq c$
2. **Bayes' theorem**: $\Pr(x|\theta) \cdot \Pr(\theta) = \Pr(\theta|x) \cdot \Pr(x)$
3. **Probability as a ratio**: $\Pr(x|t) = \frac{m(x \wedge t)}{m(t)}$ (K&S Eq. 53)

Key insight: The SAME functional equation from Appendix B (product equation) reappears here! Axiom 5 + sum rule forces the chaining operation to be multiplication (up to scale).

Deliverable: This section establishes that **probability is a normalized measure** - "simply the shape of the confined measure, automatically normalized to unit mass" (K&S, Section 7.3). This gives us **probability distributions**, which are used in Section 11.

8.1 Structural Change: From Linear Order to Lattice

Remark 8.1 (Different Type Structure). K&S Section 7 operates on a **different type** than Sections 1–6:

- **Sections 1–6** (K&S algebra): `[LinearOrder α]` - measures on linearly ordered values
- **Section 7** (Bivaluation): `[Lattice α] [BoundedOrder α]` - probability on lattice of events

This reflects K&S's conceptual shift: earlier sections study **measure values** (which are linearly ordered reals), while Section 7 studies **conditional probability on events** (which form a lattice).

The Lean formalization respects the **logical dependency order**, not K&S's presentation order. Section 7 requires lattice operations ($\wedge$, $\vee$, $\bot$, $\top$) that weren't needed in Sections 1–6.

Lattice hierarchy in the code:

- `Bivaluation` structure (line 59): `[Lattice α]` - general lattice
- Main theorems (chain-product, Bayes): `[DistribLattice α]` - needs distributivity
- Optional theorems (`sumRule_general`, `complementRule`): `[BooleanAlgebra α]` - needs complements

Formalization insight: Chain rule and Bayes require only distributivity; complements ($\Pr(A)+\Pr(\neg A)=1$) require Boolean structure.

8.2 Bivaluation and Axiom 5

Definition 8.2 (`Bivaluation`)**. Lines 59–73, Basic.lean**

A bivaluation $p : \alpha \to \alpha \to \mathbb{R}$ represents conditional plausibility on a lattice with:

- **Positivity**: $p(x|t) > 0$ when $\bot < x \leq t$
- **Sum rule**: $p(x \vee y|t) = p(x|t) + p(y|t)$ for disjoint x, y
- **Context intersection**: $p(x|t) = p(x \wedge t|t)$ (implicit in K&S)

Remark 8.3 (Lattice Structure on Events, Not Context). Note that the sum rule applies to the **first argument** (the event), not the context:

$$p(x \vee y \mid t) = p(x \mid t) + p(y \mid t)$$

The context t stays **fixed** while events are decomposed via the lattice join $\vee$. This matches the standard probability identity $P(A \cup B \mid C) = P(A \mid C) + P(B \mid C)$ for disjoint A, B. The lattice operations ($\vee$, $\wedge$, $\bot$) describe the *event algebra*; the context is just a parameter.

Definition 8.4 (Axiom 5: Chaining Associativity)**. Lines 109–128 (ChainingOp structure), Basic.lean**

The chaining operation $\odot : \mathbb{R} \to \mathbb{R} \to \mathbb{R}$ on plausibility values satisfies:

Listing 36: `ChainingOp` structure (Basic.lean:127)

```
structure ChainingOp where
  chain : R → R → R
  chain_assoc : ∀ x y z, chain (chain x y) z = chain x (chain y z)
  chain_strictMono_left : ∀ z, 0 < z → StrictMono (fun x => chain x z)
  chain_strictMono_right : ∀ x, 0 < x → StrictMono (fun z => chain x z)
  chain_pos : ∀ x y, 0 < x → 0 < y → 0 < chain x y
  chain_distrib_left : ∀ a b t, 0 < a → 0 < b → 0 < t →
    chain a t + chain b t = chain (a + b) t
```

Formulation: For a chain $a < b < c < d$:

$$(p(a|b) \odot p(b|c)) \odot p(c|d) = p(a|b) \odot (p(b|c) \odot p(c|d))$$

Definition 8.5 (Chain Rule)**. Line 219, Basic.lean**

The chain rule connects the chaining operation to the bivaluation:

Listing 37: `ChainingAssociativity` class (Basic.lean:222)

```
class ChainingAssociativity (alpha : Type*) [Lattice alpha] [BoundedOrder alpha]
    (B : Bivaluation alpha) where
  chainOp : ChainingOp
  chain_rule : ∀ a b c : alpha, a ≤ b → b ≤ c → Bot < a →
    B.p a c = chainOp.chain (B.p a b) (B.p b c)
```

This says: $p(a|c) = p(a|b) \odot p(b|c)$ for chains $a \leq b \leq c$.[2]

8.3 The Product Equation Reappears

Lines 245–314, Basic.lean

K&S's brilliant observation: combining chaining associativity with the sum rule gives the **exact same product equation** as Appendix B!

Proof sketch:

1. Let Θ be the function such that $\Theta(p(a|b) \odot p(b|c)) = \Theta(p(a|b)) + \Theta(p(b|c))$
2. Define $\Psi = \Theta^{-1}$
3. The sum rule says: for disjoint x, y with intermediate context,

   $$\text{chain}(a, t) + \text{chain}(b, t) = \text{chain}(a + b, t)$$

   This is **left-distributivity over addition** - exactly the Appendix B hypothesis!
4. Therefore: $\Psi(\tau + \xi) + \Psi(\tau + \eta) = \Psi(\tau + \zeta(\xi, \eta))$ where $\zeta(\xi, \eta) = \Theta(\Psi(\xi) + \Psi(\eta))$
5. By Appendix B: $\Theta = A \cdot \log$ for some $A > 0$
6. Hence the chaining operation is: $\text{chain}(x, y) = \frac{x \cdot y}{K}$ for some $K > 0$

[2] K&S uses "interval notation" $[x, y]$ throughout Section 7 without formally defining intervals as mathematical objects. They write $\alpha = [x, y]$, $\beta = [y, z]$, etc., and speak of "concatenating intervals" $[x, y] \circ [y, z] = [x, z]$. The chaining operation $\odot$ then acts on the plausibility *values* of these intervals: $p(\alpha) \odot p(\beta)$. Our formalization sidesteps this implicit interval semantics by working directly with lattice elements: the "interval $[a, b]$" is represented implicitly by the pair (a, b) with constraint $a \leq b$. The chain rule then states $p(a|c) = p(a|b) \odot p(b|c)$ for $a \leq b \leq c$, which captures the compositional structure without reifying intervals as first-class objects.

8.4 Chain-Product Rule

Theorem 8.6 (`chainProductRule`). ***Line 345, Basic.lean***

For chains $a \leq b \leq c$ in a lattice with normalized bivaluation ($p(t|t) = 1$):

$$\Pr(a|c) = \Pr(a|b) \cdot \Pr(b|c)$$

Listing 38: `chainProductRule` (Basic.lean:348)

```
theorem chainProductRule
    {alpha : Type*} [DistribLattice alpha] [BoundedOrder alpha]
    (B : Bivaluation alpha) [CA : ChainingAssociativity alpha B]
    (hNormalized : ∀ t : alpha, Bot < t → B.p t t = 1) :
    ∀ a b c : alpha, a ≤ b → b ≤ c → Bot < a →
      B.p a c = B.p a b * B.p b c
```

Proof strategy:

- Appendix B gives: $\text{chain}(x, y) = (x \cdot y)/K$ for some $K > 0$
- Normalization at (a, a, a) forces $K = 1$: since $p(a|a) = 1$, we have $1 = \text{chain}(1, 1) = 1/K$
- Therefore: $p(a|c) = \text{chain}(p(a|b), p(b|c)) = p(a|b) \cdot p(b|c)$

8.5 Bayes' Theorem

Theorem 8.7 (`bayesTheorem`). ***Line 421, Basic.lean***

For $x, \theta \leq t$ in a distributive lattice:

$$\Pr(x|\theta) \cdot \Pr(\theta|t) = \Pr(\theta|x) \cdot \Pr(x|t)$$

Listing 39: `bayesTheorem` (Basic.lean:424)

```
theorem bayesTheorem
    {alpha : Type*} [DistribLattice alpha] [BoundedOrder alpha]
    (B : Bivaluation alpha) [CA : ChainingAssociativity alpha B]
    (hNormalized : ∀ t : alpha, Bot < t → B.p t t = 1)
    (x theta t : alpha) (hxtheta_pos : Bot < x ⊓ theta) (hx : x ≤ t) (htheta :
       theta ≤ t)
    (hx_pos : Bot < x) (htheta_pos : Bot < theta) :
    B.p x theta * B.p theta t = B.p theta x * B.p x t
```

Proof: Both sides equal $\Pr(x \wedge \theta|t)$ by the product rule and commutativity of $\wedge$.

8.6 Probability as Ratio of Measures

Theorem 8.8 (`prob_eq_measure_ratio`). ***Line 462, Basic.lean***

*Define the **unconditional measure** by $m(x) := p(x|\top)$. Then for any context $t \neq \bot$:*

$$\Pr(x|t) = \frac{m(x \wedge t)}{m(t)}$$

Listing 40: `prob_eq_measure_ratio` (Basic.lean:714)

```
theorem prob_eq_measure_ratio
    {alpha : Type*} [DistribLattice alpha] [BoundedOrder alpha]
    (B : Bivaluation alpha) [CA : ChainingAssociativity alpha B]
    (hNormalized : ∀ t : alpha, Bot < t → B.p t t = 1) :
    ∀ x t : alpha, t ≠ Bot → B.p x t = baseMeasure B (x ⊓ t) / baseMeasure B t
```

This single formula subsumes the sum rule, chain-product rule, and range $[0, 1]$. Probability is simply the **ratio of measures** — "the elementary calculus of proportions of measure" (K&S, Section 7.3).

8.7 baseMeasure Satisfies Measure Axioms

Lines 559–576, Basic.lean

The derived `baseMeasure` satisfies the classical measure axioms:

Theorem 8.9 (`baseMeasure_satisfies_measure_axioms`)**.** *For a normalized Bivaluation ($p(t|t) = 1$ for $t > \bot$), `baseMeasure` is a probability measure:*

1. $m(\bot) = 0$ *(empty set has measure zero)*
2. *Finite additivity:* $m(x \vee y) = m(x) + m(y)$ *for disjoint* x, y
3. *Non-negativity:* $0 \leq m(x)$
4. *Normalization:* $m(\top) = 1$

Listing 41: `baseMeasure_satisfies_measure_axioms` (Basic.lean:559)

```
theorem baseMeasure_satisfies_measure_axioms
    (hNormalized : ∀ t : alpha, Bot < t → B.p t t = 1)
    (hTop : (Top : alpha) ≠ Bot) :
    baseMeasure B Bot = 0 /\
    (∀ x y : alpha, Disjoint x y →
      baseMeasure B (x ⊔ y) = baseMeasure B x + baseMeasure B y) /\
    (∀ x : alpha, 0 ≤ baseMeasure B x) /\
    baseMeasure B Top = 1
```

Key point: For finite Boolean algebras, finite additivity is equivalent to σ-additivity, so this is a bona fide probability measure in the Kolmogorov sense [9].

Remark 8.10 (Additional Measure Properties). The formalization also proves:

- **Inclusion-exclusion** (`baseMeasure_inclusion_exclusion`, line 588):

$$m(x \vee y) + m(x \wedge y) = m(x) + m(y)$$

- **Complement rule** (`baseMeasure_compl_normalized`, line 640):

$$m(x^c) = 1 - m(x)$$

- **Subadditivity** (`baseMeasure_subadditive`, line 649):

$$m(x \vee y) \leq m(x) + m(y)$$

- **ENNReal version** (`baseMeasureENNReal`, line 673): For Mathlib compatibility

9 σ-Completeness and σ-Additivity (Extension)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Core/ScaleCompleteness.lean`

Purpose: K&S develop a *finite* additivity theory on event algebras. To connect this to Kolmogorov-style probability measures on σ-algebras, we need a **countable** closure story:

- events must support countable joins ("σ-completeness"),
- the valuation must preserve limits of increasing chains (Scott continuity), and
- the scale (the image of Θ) must be sequentially complete so that sup values exist in the scale even if they need not exist inside the original algebra.

This file packages these as **minimal, explicit** additional axioms and proves the corresponding σ-additivity theorem for disjoint unions.

9.1 σ-Complete Events

Definition 9.1 (`SigmaCompleteEvents`)**. Lines 144–152, Core/ScaleCompleteness.lean**

Extends `PlausibilitySpace` with a countable supremum operator `iSup : (Nat -> E) -> E` satisfying the usual upper-bound / least-upper-bound laws.

Listing 42: σ-complete events structure

```
class SigmaCompleteEvents (E : Type*) extends PlausibilitySpace E where
  /-- Countable supremum ∃ -/
  iSup : (ℕ → E) → E
  /-- iSup is an upper bound -/
  le_iSup : ∀ (f : ℕ → E) (n : ℕ), f n ≤ iSup f
  /-- iSup is the least upper bound -/
  iSup_le : ∀ (f : ℕ → E) (b : E), (∀ n, f n ≤ b) → iSup f ≤ b
```

9.2 Sequential Completeness of the Scale

Definition 9.2 (`KSScaleComplete`)**. Lines 81–89, Core/ScaleCompleteness.lean**

Given a representation `R : RepresentationResult S` (Appendix A output), `KSScaleComplete S R` states that $\Theta(S) \subseteq \mathbb{R}$ is closed under suprema of bounded increasing ω-sequences.

Listing 43: Scale completeness (sequential closure)

```
class KSScaleComplete (S : Type*) [KSSemigroupBase S] (R : RepresentationResult S)
    where
  /-- Θ(S) is closed: bounded increasing sequences have preimages of their sups -/
  closed_seqSup : ∀ (f : ℕ → S), Monotone f → BddAbove (Set.range (R.Θ ∘ f)) →
    ∃ s : S, R.Θ s = ⨆ n, R.Θ (f n)
```

9.3 Scott Continuity (Countable Directed Limits)

Definition 9.3 (`KSScottContinuous`)**. Lines 203–211, Core/ScaleCompleteness.lean**

Scott continuity (phrased via $R.\Theta$) states that for a monotone sequence of events $f : \mathbb{N} \to E$,

$$\Theta(v(\sup f)) = \sup_n \Theta(v(f_n)).$$

Listing 44: Scott continuity for valuations

```
class KSScottContinuous (E S : Type*) [SigmaCompleteEvents E] [
    KnuthSkillingMonoidBase S]
    (R : RepresentationResult S) [KSScaleComplete S R] extends KSModel E S where
  /-- v preserves countable sups of increasing sequences (via Θ) -/
  v_scott : ∀ (f : N → E), Monotone f →
    ∃ _hf_mono : Monotone (R.Θ ∘ v ∘ f),
    ∃ _hf_bdd : BddAbove (Set.range (R.Θ ∘ v ∘ f)),
    R.Θ (v (SigmaCompleteEvents.iSup f)) = ⨆ n, R.Θ (v (f n))
```

9.4 Main Theorem: K&S σ-Additivity

Theorem 9.4 (ks_sigma_additive). ***Lines 360–418, Core/ScaleCompleteness.lean***
Under:

- *SigmaCompleteEvents E*
- *a normalized representation R : NormalizedRepresentationResult S*
- *KSScaleComplete S R.toRepresentationResult*
- *KSScottContinuous E S R.toRepresentationResult*
- *pairwise disjointness of f : Nat -> E*

the induced real-valued measure $\mu := \Theta \circ v$ is countably additive:

$$\mu\Big(\sup_n f_n\Big) = \sum_{n=0}^{\infty} \mu(f_n).$$

Listing 45: σ-additivity theorem (signature)

```
theorem ks_sigma_additive
    {E S : Type*} [SigmaCompleteEvents E] [KnuthSkillingMonoidBase S]
    (R : NormalizedRepresentationResult S) [KSScaleComplete S R.
        toRepresentationResult]
    (m : KSScottContinuous E S R.toRepresentationResult)
    (f : N → E) (hd : PairwiseDisjoint' f) :
    let μ := R.Θ ∘ m.v
    μ (SigmaCompleteEvents.iSup f) = ∑' n, μ (f n) := by
  -- Step 1: iSup of f equals iSup of partial unions
  -- Step 2: Apply Scott continuity
  -- Step 3: By finite additivity, Θ(v(partialUnion f n)) = ∑ i < n, μ(f i)
  -- Step 4: Connect iSup of partial sums to tsum (nonnegative sequences)
  ...
```

Remark 9.5 (Mathlib Bridge). For a bridge to mathlib's `Measure` / `ProbabilityMeasure` APIs, see: `Mettapedia/ProbabilityTheory/KnuthSkilling/Bridges/MathlibProbability.lean`.

10 Information and Entropy (K&S Section 8)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Information/InformationEntropy.lean`

What this section does: K&S Section 8 takes **special cases** of the variational potential H from Appendix C, specialized to probability distributions (normalized measures from Section 7).

Key point: Shannon entropy is **derived**, not just defined. It emerges as an "inevitable consequence of seeking a variational quantity" (K&S, Section 8.2).

Unified view (project-wide, optional): The repo also formalizes the Faddeev and Shannon–Khinchin uniqueness theorems (axiomatic characterizations of entropy), plus bridges to mathlib's measure-theoretic KL divergence. The entrypoint `Mettapedia/InformationTheory/EntropyKL.lean` records the equivalences between these approaches (see Appendix: Supplementary Derivation Paths).

10.1 From Atom Divergence to KL Divergence

The key step: Now that we have probability distributions from Section 9, we can specialize the atom divergence from Section 8 to normalized measures.

For probability distributions $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ where $\sum p_i = 1$ and $\sum q_i = 1$:

$$\begin{aligned}\sum_i \phi(p_i, q_i) &= \sum_i (q_i - p_i + p_i \log(p_i/q_i)) \\ &= \underbrace{\sum_i q_i}_{=1} - \underbrace{\sum_i p_i}_{=1} + \sum_i p_i \log(p_i/q_i) \\ &= \sum_i p_i \log(p_i/q_i) = D_{KL}(P\|Q)\end{aligned}$$

This is the **Kullback-Leibler divergence** (K&S Eq. 54).

Formalized in Lean (`klDivergence_from_divergence_formula`, lines 324–338, `Information/InformationEntropy.lean`):

```
theorem klDivergence_from_divergence_formula (P Q : ProbDist n)
    (hQ_pos : ∀ i, P.p i ≠ 0 → 0 < Q.p i) :
    klDivergence P Q hQ_pos =
      sum i, atomDivergence (P.p i) (Q.p i) - (sum i, Q.p i - sum i, P.p i)
```

The proof uses the normalization constraints: $\sum(q_i - p_i) = 1 - 1 = 0$, so the linear terms cancel.

10.2 Derivation Chain

From Appendix C to Shannon Entropy:

1. **Appendix C** establishes the general variational form for any measure:
$$H(m) = A + B \cdot m + C \cdot (m \log m - m)$$
2. **Section 8** specializes to atom divergence: $\phi(w, u) = u - w + w \log(w/u)$
3. **Section 9** proves that probability is a normalized measure: $\Pr(x|t) = m(x \wedge t)/m(t)$
4. **Section 11 (this section)**: For probability distributions, divergence simplifies to KL divergence
5. **Section 8.2 (Entropy)**: To quantify uncertainty, we require:
   - Zero uncertainty when one $p_k = 1$ (fully determined state)

- This forces: $A_k = 0$ and $B_k = C$
- Setting $C = -1$ (conventional scale) gives:

$$S(p) = -\sum_k p_k \log p_k$$

This is Shannon entropy - not assumed, but **derived from the variational principle**.

Elegant connection: Shannon entropy is the KL divergence to the uniform distribution, up to a constant:

$$H(P) = \log n - D_{KL}(P\|U)$$

where $U = (1/n, \ldots, 1/n)$ is the uniform distribution. **Proof**:

$$D_{KL}(P\|U) = \sum_i p_i \log(p_i/(1/n)) = \sum_i p_i \log p_i + \sum_i p_i \log n = -H(P) + \log n$$

Interpretation: Entropy measures how far a distribution is from uniform—the state of maximum uncertainty. Maximum entropy ($H = \log n$) occurs exactly when $P = U$ (i.e., $D_{KL}(P\|U) = 0$).

10.3 Shannon's Three Properties

K&S claim these properties are "inevitable consequences" (K&S, Section 8.2):

1. **Continuity**: S is a continuous function of its arguments
2. **Monotonicity**: If there are n equal choices ($p_k = 1/n$), then S increases in n
3. **Grouping**: If a choice is broken down into subchoices, S adds according to expectation:

$$S(p_1, p_2, p_3) = S(p_1, p_2 + p_3) + (p_2 + p_3) \cdot S\left(\frac{p_2}{p_2 + p_3}, \frac{p_3}{p_2 + p_3}\right)$$

These are Shannon's original axioms [8]. K&S shows they follow from the variational framework.

10.4 Formalization

Definition 10.1 (`ProbDist`)**. Lines 19–22, Mettapedia/ProbabilityTheory/Foundations/Distributions/ProbDist.lean**

A probability distribution: probabilities for n outcomes that are non-negative and sum to 1.

Listing 46: Probability distribution

```
structure ProbDist (n : ℕ) where
  p : Fin n → ℝ
  nonneg : ∀ i, 0 ≤ p i
  sum_one : sum i, p i = 1
```

Remark 10.2 (KS-facing alias)*.* **Lines 66–72, Information/InformationEntropy.lean**

`ProbDist` is defined in `Mettapedia/ProbabilityTheory/Foundations/Distributions/ProbDist.lean` and re-exported as a KS-facing alias so that Section 8 can refer to `ProbDist` without importing Foundations directly.

Definition 10.3 (`klDivergence`)**. Lines 279–281, Information/InformationEntropy.lean**
The Kullback-Leibler divergence for probability distributions (K&S Eq. 54).

Listing 47: `klDivergence` (Information/InformationEntropy.lean:279)

```
noncomputable def klDivergence {n : ℕ} (P Q : ProbDist n)
    (hQ_pos : ∀ i, P.p i ≠ 0 → 0 < Q.p i) : ℝ :=
  sum i, P.p i * log (P.p i / Q.p i)
```

Note: The positivity hypothesis `hQ_pos` ensures Q is strictly positive on the support of P, avoiding $\log(p/0)$ issues. This is the regime where K&S's formula is meaningful.

An extended version `klDivergenceTop` (line 292, `Information/InformationEntropy.lean`) takes values in $\mathbb{R}_{\geq 0} \cup \{\infty\}$, returning ∞ when this condition fails.

10.5 Shannon Entropy Definition

Formalized in Lean (`shannonEntropy`, lines 458–459, `Information/InformationEntropy.lean`):

Listing 48: Shannon entropy from the variational framework

```
noncomputable def shannonEntropy {n : ℕ} (P : ProbDist n) : ℝ :=
  sum i, entropyComponent (-1) (P.p i)
```

Equivalence to conventional formula (`shannonEntropy_eq'`, lines 480–493):

Listing 49: Shannon entropy equals $-\sum p_k \log p_k$

```
theorem shannonEntropy_eq' {n : ℕ} (P : ProbDist n) :
    shannonEntropy P = -sum i, P.p i * log (P.p i)
```

The $0 \cdot \log(0) = 0$ convention (`zero_mul_log_zero`, line 86):

Listing 50: Convention for entropy at zero probability

```
@[simp]
theorem zero_mul_log_zero : (0 : ℝ) * log 0 = 0 := zero_mul _
```

This convention is mathematically justified by $\lim_{x \to 0^+} x \log x = 0$ (L'Hôpital's rule). In Lean, it follows trivially from $0 \cdot x = 0$ for any x.

11 Counterexamples and Clarifications

Directory: `Mettapedia/ProbabilityTheory/KnuthSkilling/Counterexamples/`

11.1 The "Discontinuous Re-grading" Claim

K&S (Section 2) claim that continuity is "merely a convenient convention" and suggest a discontinuous "re-grading" map Θ could preserve the sum rule, using a base-conversion example.

Theorem 11.1 (Re-grading continuity)**.**
File: *`Counterexamples/RegradeCounterexample.lean`*
Lean name: *`regrade_preserving_sum_rule_is_continuous`*
This claim is false. *Any re-grading $\Theta : \mathbb{R} \to \mathbb{R}$ that preserves:*

1. *The sum rule: $\Theta(x + y) = \Theta(x) + \Theta(y)$ (additivity)*

2. *Monotonicity:* $x \leq y \Rightarrow \Theta(x) \leq \Theta(y)$

must be linear ($\Theta(x) = c \cdot x$ *for some constant* c*), hence continuous.*

Listing 51: Monotone additive functions are linear (RegradeCounterexample.lean:100)

```
theorem monotone_additive_is_linear {f : R → R}
    (hadd : ∀ x y, f (x + y) = f x + f y)
    (hmono : Monotone f) :
    ∀ x, f x = f 1 * x

theorem regrade_preserving_sum_rule_is_continuous {Theta : R → R}
    (hTheta_add : ∀ x y, Theta (x + y) = Theta x + Theta y)
    (hTheta_mono : Monotone Theta) :
    Continuous Theta
```

Remark 11.2 (Why K&S's Example Fails). K&S's base-conversion map is not additive: $\Theta(x+y) \neq \Theta(x) + \Theta(y)$. Their example could only work by changing the addition operation to some weird $\oplus$, which is just obfuscating notation, not demonstrating genuine discontinuity.

The philosophical point (finite systems can't detect continuity) may be valid, but the mathematical example does not support the claim.

11.2 Pathological Additive Functions

File: `Counterexamples/CauchyPathology.lean`

Without regularity conditions, Cauchy's equation $f(x + y) = f(x) + f(y)$ has "wild" nonlinear solutions (constructed via Hamel bases over $\mathbb{Q}$) [7]. These solutions are necessarily **non-monotonic**—they oscillate wildly and cannot preserve order.

11.2.1 The Construction

Step 1: Build a Hamel basis (lines 35–77):

We work with $\mathbb{R}$ as a vector space over $\mathbb{Q}$. First prove $\{1, \sqrt{2}\}$ is $\mathbb{Q}$-linearly independent (using irrationality of $\sqrt{2}$), then extend to a Hamel basis:

Listing 52: Hamel basis extending $\{1, \sqrt{2}\}$

```
1 theorem linearIndepOn_one_sqrt2 :
2     LinearIndepOn Q id ({(1 : R), R.sqrt 2} : Set R)
3
4 noncomputable def hamelBasis :
5     Module.Basis (...extend {1, sqrt 2}...) Q R :=
6   Module.Basis.extend linearIndepOn_one_sqrt2
```

Step 2: Define the weird map (lines 79–86):

Create a $\mathbb{Q}$-linear map that sends:

- $1 \mapsto 0$
- $\sqrt{2} \mapsto 1$
- All other basis vectors $\mapsto 0$

Listing 53: Definition of weirdAdditive

```
noncomputable def weirdQLinear : R →_[Q] R :=
  (hamelBasis).constr Q fun i =>
    if (i : R) = R.sqrt 2 then (1 : R) else 0

noncomputable def weirdAdditive : R → R := fun x => weirdQLinear x
```

Step 3: Prove it's additive but not linear (lines 87–127):

Listing 54: weirdAdditive satisfies Cauchy's equation but isn't linear

```
theorem weirdAdditive_add (x y : R) :
    weirdAdditive (x + y) = weirdAdditive x + weirdAdditive y

theorem weirdAdditive_not_mul (A : R) :
    ∃ x : R, weirdAdditive x ≠ A * x
  -- Proof: weirdAdditive 1 = 0 but weirdAdditive (sqrt 2) = 1
  -- So it can't be x ↦ A*x for any constant A
```

Step 4: Convert to positive reals (lines 129–165):
Define $H'(m) := \text{weirdAdditive}(\log m)$ on positive reals:

Listing 55: Multiplicative-additive pathology on positive reals

```
noncomputable def Hprime (m : R) : R := weirdAdditive (R.log m)

theorem Hprime_mul (m_x m_y : R) (hx : 0 < m_x) (hy : 0 < m_y) :
    Hprime (m_x * m_y) = Hprime m_x + Hprime m_y

theorem Hprime_not_B_add_C_log :
    ¬ ∃ (B C : R), ∀ m, 0 < m → Hprime m = B + C * log m
  -- Proof: If Hprime m = B + C*log m, then weirdAdditive x = C*x,
  -- contradicting weirdAdditive_not_mul
```

Key insight: These pathological solutions exist but **cannot be monotone**. By the theorem in §11.1, any monotone additive function is linear, so wild solutions like `weirdAdditive` must oscillate wildly and violate order preservation.

11.3 Separation is Not Derivable (Independence Result)

The separation property is **independent** of the base K&S axioms—a key discovery during formalization. The semidirect product countermodel (Listing 7, §3.1) proves this: it satisfies all base axioms but fails separation.

See Appendix 12.2 for the discovery timeline.

12 Summary: Complete Formalization

12.1 Major Formalized Theorems at a Glance

This section consolidates all the main results proven in the formalization. All paths are relative to `Mettapedia/ProbabilityTheory/KnuthSkilling/`.

Sum Rule (Appendix A)

- `representation_semigroup`
  `Additive/Proofs/OrderedSemigroupEmbedding/HolderEmbedding.lean:272`
  — $\exists\Theta$ preserving order and addition (identity-free)

- `representation_from_noAnomalousPairs`
  `Additive/Proofs/OrderedSemigroupEmbedding/HolderEmbedding.lean:300`
  — Full representation with $\Theta(\text{ident}) = 0$

- `associativity_representation`
  `Additive/Proofs/GridInduction/Main.lean:54`
  — K&S-style (grid/induction) public API

- `op_archimedean_of_separation`
  `Additive/Axioms/SandwichSeparation.lean:133`
  — Archimedean is derivable from `KSSeparation`

Product Theorem (Appendix B)

- `Psi_is_exp` (`Multiplicative/Main.lean:43`) — $\Psi = \Theta^{-1}$ is exponential under the product equation regularity

- `tensor_coe_eq_mul_div_const` (`Multiplicative/Main.lean:61`) — $x \otimes y = (x \cdot y)/C$

- `ScaledMultRep` (`Multiplicative/ScaledMultRep.lean:44`) — Common interface for both proof paths

Variational (Appendix C)

- `variationalEquation_solution_measurable` (`Variational/Main.lean:315`) — Measurable solution $\Rightarrow H'(m) = B + C \log m$

- `entropyDerivative_variational` (`Variational/Main.lean:295`) — Entropy-derivative satisfies the variational equation

- `entropyForm_deriv` (`Variational/Main.lean:493`) — $\frac{d}{dm}\big[A + Bm + C(m \log m - m)\big] = B + C \log m$

Probability Calculus (FOI Mainline)

- `sum_rule`
  `Probability/ProbabilityDerivation.lean:856`
  — $P(A \vee B) = P(A) + P(B)$ for disjoint events

- `product_rule_ks`
  `Probability/ProbabilityDerivation.lean:938`
  — $P(A \wedge B) = P(A|B)\,P(B)$

- `bayes_theorem_ks`
  `Probability/ProbabilityDerivation.lean:957`
  — Bayes' theorem

- `complement_rule`
  `Probability/ProbabilityDerivation.lean:969`
  — $P(B) = 1 - P(A)$ for complements

Conditional Probability (Section 7)

- `chainProductRule`
  `Probability/ConditionalProbability/Basic.lean:348`
  — $p(xy|z) = p(x|z) \cdot p(y|xz)$

- `bayesTheorem`
  `Probability/ConditionalProbability/Basic.lean:424`
  — $p(x|yz) \cdot p(y|z) = p(y|xz) \cdot p(x|z)$

- `prob_eq_measure_ratio`
  `Probability/ConditionalProbability/Basic.lean:714`
  — $p(x|y) = m(x \wedge y)/m(y)$

- `baseMeasure_satisfies_measure_axioms`
  `Probability/ConditionalProbability/Basic.lean:559`
  — m is a probability measure

σ-Additivity Bridge (Extension)

- `ks_sigma_additive` (`Core/ScaleCompleteness.lean:360`) — $\mu(\sup f) = \sum_n \mu(f_n)$ for disjoint countable families

Divergence & Entropy (Sections 6, 8)

- `atomDivergence_nonneg` (`Information/Divergence.lean:102`) — $D(w\|u) \geq 0$

- `atomDivergence_eq_zero_iff` (`Information/Divergence.lean:123`) — $D(w\|u) = 0 \Leftrightarrow w = u$

- `klDivergence` (`Information/InformationEntropy.lean:279`) — KL divergence

- `shannonEntropy` (`Information/InformationEntropy.lean:458`) — $H = -\sum p \log p$

Quantum Theory Classification (Section 4)

- `selection_theorem` (`Mettapedia/Algebra/TwoDimClassification.lean:427`) — $\mu < 0$ gives QM Born rule

- `mean_bornRule_sum_unit_phases` (`Core/SymmetricalFoundation.lean:303`) — Born rule from averaging

- `classification_trichotomy` (`Mettapedia/Algebra/TwoDimClassification.lean:293`) — Exactly 3 algebra classes

Counterexamples & Clarifications

- `monotone_additive_is_linear` (`Counterexamples/RegradeCounterexample.lean:100`) — Discontinuous re-grading impossible
- `weirdAdditive_add` (`Counterexamples/CauchyPathology.lean:88`) — Pathological additive functions exist
- `Hprime_not_B_add_C_log` (`Counterexamples/CauchyPathology.lean:144`) — But they violate regularity

12.2 Key Discoveries from Formalization

1. **Linear order is necessary**: K&S proofs assume trichotomy without stating it. We prove this is essential: `no_pointRepresentation_with_incomparables` shows partial orders with incomparable elements admit no faithful $\Theta : \alpha \to \mathbb{R}$ (`Core/TotalityImprecision.lean`).

2. **Identity is essential for positivity**: K&S say the bottom element is "optional" (lines 320, 340–341), claiming "fidelity ensures that other elements are quantified by positive values." Our formalization **proves this claim is FALSE** for unbounded structures.

   The $\mathbb{Z}$ counterexample (`Additive/Counterexamples/NegativeWithoutIdentity.lean`):

   - $(\mathbb{Z}, +, \leq)$ satisfies K&S Axioms 1–2 (associativity and strict order preservation)
   - The Hölder embedding theorem applies (no anomalous pairs)
   - But $\Theta(-1) = -1 < 0$—**negative values appear!**

   Why K&S's claim fails: K&S's positivity comes from `ident_le`: $\forall x,\ \bot \leq x$. This requires $\bot$ to *exist* and be *minimal*. For $\mathbb{Z}$:

   - No bottom element exists ($\mathbb{Z}$ is unbounded below)
   - The representation theorem still applies
   - But positivity is **not guaranteed**

   Corrected understanding:

   - **With identity + `ident_le`**: $\Theta(\bot) = 0$ provides canonical normalization; all other elements have $\Theta(x) > 0$.
   - **Without identity**: The representation theorem works, but positivity is **not guaranteed**. The embedding is unique up to additive constant, but no constant can rescue positivity for unbounded-below structures.

   Status: K&S claim **corrected**. Identity with `ident_le` is **essential** for positivity, not merely "aesthetic."

3. **Additional axiom is necessary**: The representation theorem requires an explicit axiom beyond associativity and monotonicity—either separation or NAP (no anomalous pairs)—to enable rational approximation.

4. **Archimedean is derivable**: Not an axiom—follows from separation or NAP.

5. **Commutativity is derivable**: Not an axiom—follows from separation or NAP.

6. **Classification gives isomorphism**: The original K&S classification theorem is about *isomorphism classes*, not equality of multiplication rules.

7. **Measurability replaces continuity**: For Appendix C, measurability (not differentiability) is the correct regularity assumption.

8. **Discontinuous re-grading is impossible**: K&S's claim that continuity is optional is false for maps preserving both the sum rule and monotonicity.
(Proved in `Counterexamples/RegradeCounterexample.lean`.)

9. **Symmetries 3–4 are direct product**: The direct product symmetries (distributivity, product associativity) are logically separate from the combination symmetries (0–2) and are formalized in `Multiplicative/Main.lean`.

10. **Interface design pattern**: Both Appendix A and Appendix B use an **interface + multiple implementations** pattern:

    - **Appendix A**: `HasRepresentationTheorem` / `RepresentationResult` interface with Hölder, cuts, and grid proof paths
    - **Appendix B**: `ScaledMultRep` interface with K&S path and Direct path implementations

    Downstream code depends only on the interfaces, not on specific proof paths. This separation of concerns allows switching implementations without changing dependent code.

Appendix: Supplementary Derivation Paths (Walkthrough)

The entrypoint `FoundationsOfInference.lean` focuses on K&S FOI. The repository also contains additional formalizations that complement the main development: independent derivation paths, equivalence theorems, and bridges connecting different approaches to entropy and KL divergence. This appendix lists the **major definitions** and **theorem statements** in these modules, with file/line pointers for direct audit.

A. Unified finite entropy/KL import surface

Curated import surface: `Mettapedia/InformationTheory/EntropyKL.lean` (Lines 1–44). This file is intentionally a stable *import path*—it does not introduce new definitions, but imports the main finite-discrete entropy/KL stack plus bridges.

Listing 56: Curated entrypoint: entropy + KL from all routes (imports only)

```
import Mettapedia.InformationTheory.ShannonEntropy.Main
import Mettapedia.ProbabilityTheory.KnuthSkilling.Information.Main
```

Definition 12.1 (`ProbVec`)**. Lines 28–36, InformationTheory/Basic.lean**
Finite distributions on `Fin n` using mathlib's standard simplex.

Listing 57: Mathlib-grounded finite distributions (stdSimplex)

```
abbrev Prob (alpha : Type*) [Fintype alpha] := Subtype (stdSimplex ℝ alpha)
abbrev ProbVec (n : ℕ) := Prob (Fin n)
```

Definition 12.2 (`ProbDist`)**. Lines 18–22, Foundations/Distributions/ProbDist.lean**
Foundations-level finite distributions as an explicit record `p, nonneg, sum_one`.

Listing 58: Foundations-level finite distributions (record form)

```
structure ProbDist (n : ℕ) where
  p : Fin n → ℝ
  nonneg : ∀ i, 0 ≤ p i
  sum_one : ∑ i, p i = 1
```

Definition 12.3 (`shannonEntropy`)**. Lines 62–70, InformationTheory/Basic.lean**
Shannon entropy for `ProbVec`: $H(p) = \sum_i \texttt{negMulLog}(p_i) = -\sum_i p_i \log(p_i)$, with the convention $0 \log 0 = 0$ via `negMulLog`.

Listing 59: Shannon entropy on ProbVec

```
noncomputable def shannonEntropy {n : ℕ} (p : ProbVec n) : ℝ :=
  ∑ i : Fin n, negMulLog (p.1 i)
```

Definition 12.4 (`uniformDist`)**. Lines 79–85, InformationTheory/Basic.lean**
Uniform distribution on `Fin n`: $p_i = 1/n$.

Listing 60: Uniform distribution on Fin n

```
noncomputable def uniformDist (n : ℕ) (hn : 0 < n) : ProbVec n :=
  ⟨fun _ => 1 / n, ...⟩
```

B. ProbVec ↔ ProbDist glue + key equivalences

File: `Mettapedia/InformationTheory/ShannonEntropy/Interface.lean`

Definition 12.5 (`probVecEquivProbDist`)**. Lines 185–223, ShannonEntropy/Interface.lean**
Equivalence between the mathlib representation (`ProbVec`) and the K&S finite distribution type (`KSProbDist`).

Listing 61: ProbVec n ≃ KSProbDist n (conversion equivalence)

```
abbrev KSProbDist (n : ℕ) :=
  Mettapedia.ProbabilityTheory.KnuthSkilling.Information.InformationEntropy.
      ProbDist n

def probVecEquivProbDist (n : ℕ) : ProbVec n ≃ KSProbDist n where
  toFun := ProbVec.toProbDist
  invFun := KSProbDist.toProbVec
  left_inv := fun _ => rfl
  right_inv := fun p => by simp
```

Theorem 12.6 (shannonEntropy_eq_ks_shannonEntropy)**.** ***Lines 229–246, ShannonEntropy-/Interface.lean***

The Shannon entropy defined on `ProbVec` *agrees with the K&S Shannon entropy on* `ProbDist`*, via the bridge.*

Listing 62: Shannon entropy: ProbVec version equals KS ProbDist version

```
theorem shannonEntropy_eq_ks_shannonEntropy {n : ℕ} (p : ProbVec n) :
    shannonEntropy p =
      Mettapedia.ProbabilityTheory.KnuthSkilling.Information.InformationEntropy.
          shannonEntropy
        p.toProbDist := by
  -- proof in file
  ...
```

Definition 12.7 (klDivergenceVec)**. Lines 265–268, ShannonEntropy/Interface.lean**

Unified finite KL divergence on `ProbVec`, defined by transporting the K&S `klDivergence` along `ProbVec.toProbDist`.

Listing 63: KL divergence on ProbVec (via KS klDivergence)

```
noncomputable def klDivergenceVec {n : ℕ} (P Q : ProbVec n)
    (hQ_pos : ∀ i, P.1 i ≠ 0 → 0 < Q.1 i) : ℝ :=
  klDivergence P.toProbDist Q.toProbDist (by intro i hi; exact hQ_pos i hi)
```

Theorem 12.8 (klDivergenceVec_nonneg)**.** ***Lines 276–281, ShannonEntropy/Interface.lean***

Gibbs inequality for `klDivergenceVec`*:* $0 \le D(P\|Q)$ *under the usual positivity-on-support hypothesis.*

Listing 64: Gibbs inequality on ProbVec

```
theorem klDivergenceVec_nonneg {n : ℕ} (P Q : ProbVec n)
    (hQ_pos : ∀ i, P.1 i ≠ 0 → 0 < Q.1 i) :
    0 ≤ klDivergenceVec P Q hQ_pos :=
  klDivergence_nonneg' P.toProbDist Q.toProbDist (by intro i hi; exact hQ_pos i hi
      )
```

Theorem 12.9 (shannonEntropy_via_klDivergence)**.** ***Lines 284–308, ShannonEntropy/Interface.lean***

Entropy via KL: $H(P) = \log(n) - D(P\|\mathrm{Uniform}_n)$.

Listing 65: Entropy via KL (finite)

```
theorem shannonEntropy_via_klDivergence {n : ℕ} (P : ProbVec n) (hn : 0 < n) :
    shannonEntropy P =
      log n - klDivergenceVec P (uniformDist n hn) (by
        intro i _; unfold uniformDist; simp; positivity) := by
  -- proof in file
  ...
```

File: `Mettapedia/InformationTheory/ShannonEntropy/MeasureTheoreticBridge.lean`

Theorem 12.10 (klDivergence_eq_mathlib_klDiv)**.** ***Lines 229–343, ShannonEntropy/MeasureTheoreticBridge.lean***

The discrete finite KL formula agrees with mathlib's measure-theoretic `InformationTheory.klDiv`*, after converting a* `ProbVec` *to the corresponding finite measures.*

Listing 66: Discrete KL equals mathlib klDiv (finite case)

```
theorem klDivergence_eq_mathlib_klDiv {n : ℕ} (P Q : ProbVec n)
    (hQ_pos : ∀ i, P.1 i ≠ 0 → 0 < Q.1 i) :
    klDivergenceVec P Q hQ_pos =
      (InformationTheory.klDiv P.toFinMeasure Q.toFinMeasure).toℝ := by
  -- proof in file
  ...
```

Theorem 12.11 (axiomatic_equals_measureTheoretic)**.** ***Lines 361–363, ShannonEntropy/MeasureTheoreticBridge.lean***

Faddeev axioms ⇒ Shannon formula, packaged as a single bridge theorem from axioms to the measure-theoretic normalization.

Listing 67: Axioms-to-measures bridge (Faddeev -¿ Shannon)

```
theorem axiomatic_equals_measureTheoretic {n : ℕ} (E : FaddeevEntropy) (p :
    ProbVec n) :
    E.H p = shannonEntropyNormalized p := by
  simpa using faddeev_H_eq_shannon E p
```

C. Axiomatic entropy: Faddeev / Shannon (1948) / Shannon–Khinchin

Entry point: `Mettapedia/InformationTheory/ShannonEntropy/Main.lean`

Faddeev (1956): minimal axiomatization

Definition 12.12 (FaddeevEntropy)**. Lines 75–94, ShannonEntropy/Faddeev.lean**

Faddeev's axiom interface: binary continuity, symmetry, recursivity (grouping), normalization.

Listing 68: Faddeev axioms (minimal entropy interface)

```
structure FaddeevEntropy where
  H : ∀ {n : ℕ}, ProbVec n → ℝ
  continuous_binary : Continuous (fun p : Set.Icc (0 : ℝ) 1 =>
    H (binaryDist p.1 p.2.1 p.2.2))
  symmetry : ∀ {n : ℕ} (p : ProbVec n) (sigma : Equiv.Perm (Fin n)),
    H (permute sigma p) = H p
  recursivity : ∀ {n : ℕ} (p : ProbVec (n + 2)) (h : 0 < p.1 0 + p.1 1),
    H p = H (groupFirstTwo p h) + (p.1 0 + p.1 1) * H (normalizeBinary (p.1 0) (p
        .1 1)
      (p.nonneg 0) (p.nonneg 1) h)
  normalization : H binaryUniform = 1
```

Theorem 12.13 (faddeev_H_eq_shannon)**.** ***Lines 6067–6072, ShannonEntropy/Faddeev.lean***

Faddeev uniqueness: every `FaddeevEntropy` *is (the normalized) Shannon entropy.*

Listing 69: Faddeev uniqueness theorem (main result)

```
theorem faddeev_H_eq_shannon (E : FaddeevEntropy) {n : ℕ} (p : ProbVec n) :
    E.H p = shannonEntropyNormalized p := by
  -- proof in file
  ...
```

Shannon (1948): original axiom interface

Definition 12.14 (`Shannon1948Entropy`)**. Lines 196–210, ShannonEntropy/Shannon1948.lean**

Shannon's original axiom interface (relabeling invariance, continuity, monotonicity on uniforms, grouping).

Listing 70: Shannon 1948 axiom interface

```
structure Shannon1948Entropy where
  H : ∀ {alpha : Type} [Fintype alpha], Prob alpha → ℝ
  relabel :
    ∀ {alpha beta : Type} [Fintype alpha] [Fintype beta], ∀ e : alpha ≃ beta, ∀ p
        : Prob alpha,
      H (alpha := beta) (Prob.map e p) = H (alpha := alpha) p
  continuity : ∀ {alpha : Type} [Fintype alpha], Continuous (H (alpha := alpha))
  monotone_uniform :
    ∀ {m n : ℕ} (hm : 0 < m) (hn : 0 < n), m ≤ n →
      H (alpha := Fin m) (uniformDist m hm) ≤ H (alpha := Fin n) (uniformDist n hn
          )
  grouping :
    ∀ {alpha : Type} [Fintype alpha] {beta : alpha → Type} [∀ a, Fintype (beta a)
        ],
      ∀ (p : Prob alpha) (q : ∀ a, Prob (beta a)),
        H (comp p q) = H p + ∑ a : alpha, p.1 a * H (q a)
```

Shannon–Khinchin (1957): standard textbook axiom interface

Definition 12.15 (`ShannonKhinchinEntropy`)**. Lines 60–81, ShannonEntropy/ShannonKhinchin.lean**

Shannon–Khinchin axiom interface (continuity, symmetry, maximality, expansibility, strong additivity, normalization).

Listing 71: Shannon–Khinchin axiom interface

```
structure ShannonKhinchinEntropy where
  H : ∀ {n : ℕ}, ProbVec n → ℝ
  continuity : ∀ {n : ℕ}, Continuous (H (n := n))
  symmetry : ∀ {n : ℕ} (p : ProbVec n) (sigma : Equiv.Perm (Fin n)), H (permute
      sigma p) = H p
  maximality : ∀ {n : ℕ} (hn : 0 < n) (p : ProbVec n), H p ≤ H (uniformDist n hn)
  expansibility : ∀ {n : ℕ} (p : ProbVec n), H (expandZero p) = H p
  strong_additivity : ∀ {n : ℕ} (p : ProbVec (n + 2)) (h : 0 < p.1 0 + p.1 1),
    H p = H (groupFirstTwo p h) + (p.1 0 + p.1 1) * H (normalizeBinary (p.1 0) (p
        .1 1)
      (p.nonneg 0) (p.nonneg 1) h)
  normalization : H binaryUniform = 1
```

Equivalence of axiom systems (glue theorems)

Theorem 12.16 (`faddeev_implies_shannonKhinchin`)**.** ***Lines 46–57, ShannonEntropy/Equivalence.lean***

Existence-level bridge from Faddeev to Shannon–Khinchin (constructs a ***`ShannonKhinchinEntropy`*** *with the same underlying function).*

Listing 72: Faddeev -¿ Shannon–Khinchin bridge

```
theorem faddeev_implies_shannonKhinchin (E : FaddeevEntropy) :
    ∃ (E' : ShannonKhinchinEntropy), ∀ (n : ℕ) (p : ProbVec n), E'.H p = E.H p :=
      by
  -- proof in file
  ...
```

Theorem 12.17 (`shannonKhinchin_implies_faddeev`)**.** ***Lines 63–72, ShannonEntropy/Equivalence.lean***

Existence-level bridge from Shannon–Khinchin to Faddeev.

Listing 73: Shannon–Khinchin -¿ Faddeev bridge

```
theorem shannonKhinchin_implies_faddeev (E : ShannonKhinchinEntropy) :
    ∃ (E' : FaddeevEntropy), ∀ (n : ℕ) (p : ProbVec n), E'.H p = E.H p := by
  -- proof in file
  ...
```

D. Shore–Johnson (1980): system independence ⇒ Cauchy ⇒ log ⇒ KL

Scope note (Uffink [12]). At the level of inference procedures, weaker "independence" axioms can admit families of update rules (e.g. Rényi-like families). In this project we isolate the core *product additivity* hypothesis at the atom level, and then separate the analytic *regularity gate* (measurability) that rules out Hamel solutions.

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/ShoreJohnson/SystemIndependence.lean`

Definition 12.18 (`SJSystemIndependenceAtom`)**. Lines 45–51, ShoreJohnson/SystemIndependence.lean**

Atomic system independence: additivity of $\sum d(p_i, q_i)$ over product distributions.

Listing 74: Shore–Johnson system independence (atom level)

```
structure SJSystemIndependenceAtom (d : ℝ → ℝ → ℝ) : Prop where
  regularAtom : RegularAtom d
  add_over_products :
    ∀ {n m : ℕ} [NeZero n] [NeZero m], ∀ P Q : ProbDist n, ∀ R S : ProbDist m,
      ∑ ij : Fin n × Fin m, d ((P ⊗ R).p ij) ((Q ⊗ S).p ij) =
        (∑ i : Fin n, d (P.p i) (Q.p i)) + (∑ j : Fin m, d (R.p j) (S.p j))
```

Definition 12.19 (`SJSystemIndependenceAtomPos`)**. Lines 62–68, ShoreJohnson/SystemIndependence.lean**

Positivity-restricted variant (the right formulation for `klAtom` and KL divergence).

Listing 75: System independence with positivity hypotheses on references

```
structure SJSystemIndependenceAtomPos (d : ℝ → ℝ → ℝ) : Prop where
  regularAtom : RegularAtom d
  add_over_products_pos :
    ∀ {n m : ℕ} [NeZero n] [NeZero m], ∀ P Q : ProbDist n, ∀ R S : ProbDist m,
      (∀ i, 0 < Q.p i) → (∀ j, 0 < S.p j) →
        ∑ ij : Fin n × Fin m, d ((P ⊗ R).p ij) ((Q ⊗ S).p ij) =
          (∑ i : Fin n, d (P.p i) (Q.p i)) + (∑ j : Fin m, d (R.p j) (S.p j))
```

Theorem 12.20 (d_one_eq_const_mul_log_of_measurable)**.** ***Lines 80–84, ShoreJohnson/SystemIndependence.lean***

Dirac extraction: under system independence and measurability, $q \mapsto d(1, q)$ is logarithmic on $0 < q \leq 1$.

Listing 76: Dirac extraction: Cauchy -¿ log under measurability

```
theorem d_one_eq_const_mul_log_of_measurable
    (d : R → R → R) (hSJ : SJSystemIndependenceAtom d)
    (hMeas : Measurable (d 1)) :
    ∃ C : R, ∀ q : R, 0 < q → q ≤ 1 → d 1 q = C * log q :=
  dirac_extraction_log_of_measurable d hSJ.regularAtom hSJ.add_over_products hMeas
```

File: Mettapedia/ProbabilityTheory/KnuthSkilling/ShoreJohnson/KL.lean

Definition 12.21 (klAtom)**. Lines 74–75, ShoreJohnson/KL.lean**

The KL atom $w \log(w/u)$ (with Lean's standard boundary conventions for 0).

Listing 77: KL atom divergence

```
noncomputable def klAtom (w u : R) : R :=
  w * R.log (w / u)
```

Theorem 12.22 (ratioForm_eq_const_mul_klAtom)**.** ***Lines 93–108, ShoreJohnson/KL.lean***

Ratio-form rigidity: if $d(w, u) = w\, g(w/u)$ and g satisfies multiplicative Cauchy + measurability, then d is a constant multiple of ***klAtom****.*

Listing 78: Ratio-form rigidity -¿ constant multiple of klAtom

```
theorem ratioForm_eq_const_mul_klAtom
    (d : R → R → R) (g : R → R)
    (hd : ∀ w u : R, 0 < w → 0 < u → d w u = w * g (w / u))
    (hgMul : MulCauchyOnPos g)
    (hgMeas : Measurable g) :
    ∃ C : R, ∀ w u : R, 0 < w → 0 < u → d w u = C * klAtom w u := by
  -- proof in file
  ...
```

File: Mettapedia/ProbabilityTheory/KnuthSkilling/ShoreJohnson/Bridge.lean

Theorem 12.23 (sum_klAtom_eq_klDivergence)**.** ***Lines 34–37, ShoreJohnson/Bridge.lean***

Bridge back to the project's finite KL divergence on ***ProbDist****.*

Listing 79: Finite KL divergence is the sum of KL atoms

```
theorem sum_klAtom_eq_klDivergence {n : N} (P Q : ProbDist n)
    (hQ_pos : ∀ i, P.p i ≠ 0 → 0 < Q.p i) :
    (∑ i : Fin n, klAtom (P.p i) (Q.p i)) = klDivergence P Q hQ_pos := by
  simp [klAtom, klDivergence]
```

File: Mettapedia/ProbabilityTheory/KnuthSkilling/Bridges/ShoreJohnsonVariationalBridge.lean

Theorem 12.24 (mulCauchyOnPos_eq_const_mul_log_of_variationalEquation_solution_measurable)**.** ***Lines 33–35, Bridges/ShoreJohnsonVariationalBridge.lean***

Formal bridge: the Shore–Johnson multiplicative-Cauchy log lemma is a corollary of the Appendix C lemma ***variationalEquation_solution_measurable****.*

Listing 80: SJ log rigidity is a corollary of Appendix C rigidity

```
theorem mulCauchyOnPos_eq_const_mul_log_of_variationalEquation_solution_measurable
    (g : R → R) (hg : MulCauchyOnPos g) (hMeas : Measurable g) :
    ∃ C : R, ∀ x : R, 0 < x → g x = C * R.log x := by
  -- proof in file
  ...
```

E. Cox (1946/1961): plausibility axioms ⇒ product/complement rules (plus event-level integration)

Entry point: `Mettapedia/ProbabilityTheory/Cox.lean`

File: `Mettapedia/ProbabilityTheory/Cox/Basic.lean`

Definition 12.25 (`PlausibilityFunction`)**. Lines 65–70, Cox/Basic.lean**
Basic object: a real-valued plausibility function $p(A \mid B) \in [0, 1]$.

Listing 81: Cox plausibility function interface

```
structure PlausibilityFunction (Prop' : Type*) where
  p : Prop' → Prop' → R
  p_nonneg : ∀ A B, 0 ≤ p A B
  p_le_one : ∀ A B, p A B ≤ 1
```

Definition 12.26 (`ConjunctionRule`)**. Lines 74–88, Cox/Basic.lean**
Conjunction functional equation packaging: a binary operation F on plausibilities, with unit interval range and continuity.

Listing 82: Cox conjunction rule interface

```
structure ConjunctionRule where
  F : R → R → R
  F_range : ∀ x y, 0 ≤ x → x ≤ 1 → 0 ≤ y → y ≤ 1 → 0 ≤ F x y ∧ F x y ≤ 1
  F_continuous : Continuous (Function.uncurry F)
  F_one_left : ∀ y, F 1 y = y
  F_one_right : ∀ x, F x 1 = x
  F_zero_left : ∀ y, F 0 y = 0
  F_zero_right : ∀ x, F x 0 = 0
```

Definition 12.27 (`NegationRule`)**. Lines 91–103, Cox/Basic.lean**
Negation functional equation packaging: a unary operation G on plausibilities, with continuity and strict antitonicity.

Listing 83: Cox negation rule interface

```
structure NegationRule where
  G : R → R
  G_range : ∀ x, 0 ≤ x → x ≤ 1 → 0 ≤ G x ∧ G x ≤ 1
  G_continuous : Continuous G
  G_strictAnti : StrictAnti G
  G_zero : G 0 = 1
  G_one : G 1 = 0
```

Definition 12.28 (`AssociativityEquation`)**. Lines 105–108, Cox/Basic.lean**

Associativity equation on F (coming from $(A \wedge B) \wedge C = A \wedge (B \wedge C)$).

Listing 84: Associativity functional equation

```
def AssociativityEquation (F : R → R → R) : Prop :=
  ∀ x y z : R, F (F x y) z = F x (F y z)
```

File: `Mettapedia/ProbabilityTheory/Cox/ProductRuleDerivation.lean`

Theorem 12.29 (`cox_productRule`)**.** ***Lines 2349–2351, Cox/ProductRuleDerivation.lean***

Cox's main theorem: under Cox's axioms, there exists a reparametrization p such that $p(F(x,y)) = p(x)p(y)$ on $(0,1]$.

Listing 85: Cox product rule theorem (reparametrized)

```
theorem cox_productRule (C : CoxFullAxioms) :
    ∃ p : R → R, StrictMonoOn p (Set.Ioc 0 1) ∧ ContinuousOn p (Set.Ioc 0 1) ∧
      (∀ x y, 0 < x → x ≤ 1 → 0 < y → y ≤ 1 → p (C.F x y) = p x * p y) ∧ p 1 = 1
        := by
  -- proof in file
  ...
```

Theorem 12.30 (`cox_commutativity`)**.** ***Lines 2391–2402, Cox/ProductRuleDerivation.lean***

Commutativity is derived: $F(x,y) = F(y,x)$ on the probability domain, as a consequence of the product rule.

Listing 86: Cox commutativity derived from multiplicative representation

```
theorem cox_commutativity (C : CoxFullAxioms)
    (x y : R) (hx_pos : 0 < x) (hx_le : x ≤ 1) (hy_pos : 0 < y) (hy_le : y ≤ 1) :
    C.F x y = C.F y x := by
  -- proof in file
  ...
```

Theorem 12.31 (`cox_negationRule`)**.** ***Lines 3374–3376, Cox/ProductRuleDerivation.lean***

Negation rule (power-family form): there exists $r > 0$ such that $G(x)^r + x^r = 1$ on $(0,1)$.

Listing 87: Cox negation rule: power-family form

```
theorem cox_negationRule (N : CoxNegationAxioms) :
    ∃ r : R, 0 < r ∧ ∀ x ∈ Set.Ioo (0 : R) 1, (N.G x) ^ r + x ^ r = 1 := by
  -- proof in file
  ...
```

File: `Mettapedia/ProbabilityTheory/Cox/ProbabilityCalculusBridge.lean`

Theorem 12.32 (`cox_productRule_regrade`)**.** ***Lines 18–28, Cox/ProbabilityCalculusBridge.lean***

Cox reparametrization matches the K&S probability-calculus regraduation interface.

Listing 88: Bridge: Cox product rule as a K&S-style regrade

```
theorem cox_productRule_regrade (C : CoxFullAxioms) :
    ∃ p : R → R, StrictMonoOn p (Ioc (0 : R) 1) ∧ ContinuousOn p (Ioc (0 : R) 1) ∧
      (∀ x y, 0 < x → x ≤ 1 → 0 < y → y ≤ 1 → p (C.F x y) = p x * p y) ∧ p 1 = 1
        :=
  cox_productRule C
```

File: `Mettapedia/ProbabilityTheory/Cox/ProbabilityCalculusIntegration.lean`

Definition 12.33 (`CoxProbabilityModel`)**. Lines 31–52, Cox/ProbabilityCalculusIntegration.lean**

Event-level output: after regrading, Cox yields a probability-like valuation on a Boolean lattice of events satisfying the product and complement rules.

Listing 89: Event-level Cox model (product + complement rules)

```
structure CoxProbabilityModel (alpha : Type*)
    [PlausibilitySpace alpha] [ComplementedLattice alpha] where
  P : alpha → alpha → ℝ
  P_nonneg : ∀ a c, 0 ≤ P a c
  P_le_one : ∀ a c, P a c ≤ 1
  P_top : ∀ c, P ⊤ c = 1
  P_bot : ∀ c, P ⊥ c = 0
  P_mono_uncond : ∀ {a b}, a ≤ b → P a ⊤ ≤ P b ⊤
  product_rule : ∀ a b c, P (a ⊓ b) c = P a c * P b (a ⊓ c)
  compl : alpha → alpha
  compl_isCompl : ∀ a, IsCompl a (compl a)
  complement_rule : ∀ a c, P (compl a) c = 1 - P a c
```

Theorem 12.34 (`sum_rule_disjoint`)**.** ***Lines 70–113, Cox/ProbabilityCalculusIntegration.lean***

Finite additivity for disjoint events, derived from product + complement rules.

Listing 90: Disjoint additivity derived from Cox rules

```
theorem sum_rule_disjoint (M : CoxProbabilityModel alpha) {a b : alpha} (h :
    Disjoint a b) :
    (M.valuation).val (a ⊔ b) = (M.valuation).val a + (M.valuation).val b := by
  -- proof in file
  ...
```

Theorem 12.35 (`ProbabilityCalculusClass instance`)**.** ***Lines 117–129, Cox/ProbabilityCalculusIntegration.lean***

The event-level Cox model instantiates the shared `ProbabilityCalculusClass` *interface.*

Listing 91: ProbabilityCalculusClass instance for CoxProbabilityModel

```
instance (M : CoxProbabilityModel alpha) :
    Mettapedia.ProbabilityTheory.KnuthSkilling.Probability.ProbabilityCalculus.
        ProbabilityCalculusClass alpha
      M.valuation where
  sum_rule' := fun {a b} h => M.sum_rule_disjoint h
  complement_rule' := by
    -- proof in file
    ...
```

Appendix: Small Example Files

These files are optional and are not imported by the reviewer entrypoint. They are intended as lightweight sanity checks and modeling notes.

1. Coin flip and die (`ProbDist` examples)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Examples/CoinDie.lean`

Definition 12.36 (`fairCoin`, `fairDie`)**. Lines 20–51, Examples/CoinDie.lean**
Concrete finite distributions `ProbDist 2` and `ProbDist 6` with constant probabilities.

Definition 12.37 (`eventProb`)**. Lines 55–60, Examples/CoinDie.lean**
Event probability for `Finset`-events: `eventProb P E` is the finite sum of atom probabilities over `E`.

Theorem 12.38 (`eventProb_union_disjoint`)**.** ***Lines 77–82, Examples/CoinDie.lean***
Finite additivity for disjoint events: if A and B are disjoint, then $\mathrm{eventProb}(P, A \cup B) = \mathrm{eventProb}(P, A) + \mathrm{eventProb}(P, B)$.

Listing 92: Event probability and disjoint additivity (Examples/CoinDie.lean, excerpt)

```
/-- Probability of an event (subset of outcomes) under a distribution.

For `ProbDist n`, an event is a `Finset (Fin n)` and its probability
is the sum of individual outcome probabilities. -/
noncomputable def eventProb {n : ℕ} (P : ProbDist n) (E : Finset (Fin n)) : ℝ :=
  ∑ i ∈ E, P.p i

/-- Event probability is non-negative -/
theorem eventProb_nonneg {n : ℕ} (P : ProbDist n) (E : Finset (Fin n)) :
    0 ≤ eventProb P E := by
  apply Finset.sum_nonneg
  intro i _
  exact P.nonneg i

/-- Probability of the entire sample space is 1 -/
theorem eventProb_univ {n : ℕ} (P : ProbDist n) :
    eventProb P Finset.univ = 1 := P.sum_one

/-- Probability of empty event is 0 -/
theorem eventProb_empty {n : ℕ} (P : ProbDist n) :
    eventProb P ∅ = 0 := Finset.sum_empty

/-- Finite additivity: P(A ∪ B) = P(A) + P(B) for disjoint A, B -/
theorem eventProb_union_disjoint {n : ℕ} (P : ProbDist n)
    (A B : Finset (Fin n)) (hDisj : Disjoint A B) :
    eventProb P (A ∪ B) = eventProb P A + eventProb P B := by
  unfold eventProb
  exact Finset.sum_union hDisj
```

2. 7-element lattice modeling caution

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Examples/ImpreciseOn7Element.lean`

This file accompanies the 7-element distributive lattice in `Counterexamples/NonModularDistributive.lean`. It is a modeling caution: the ideal-lattice meet/join are not the same as Boolean conjunction/disjunction for feature events. Earlier drafts of the file mistakenly presented this as evidence for imprecise probability; the intended lesson is about choosing the correct event semantics.

Theorem 12.39 (`modular_law_holds`)**.** ***Lines 115–123, Examples/ImpreciseOn7Element.lean***
In the Boolean event model (using Boolean $\wedge/\vee$), modularity holds.

Theorem 12.40 (`lattice_modular_fails_iff`)**.** ***Lines 186–201, Examples/ImpreciseOn7Element.lean***

If one instead interprets meet as lattice-meet (so that $abc \wedge abd = ab$*), then the modular identity fails whenever the "cores-only" state has positive mass.*

3. Toy decision example (non-FOI)

File: `Mettapedia/ProbabilityTheory/KnuthSkilling/Examples/PreciseVsImprecise.lean`

Theorem 12.41 (`decisions_differ`)**.** ***Lines 212–216, Examples/PreciseVsImprecise.lean***

Exhibits a simple payoff table where expected utility and a maximin-style rule give different choices under interval uncertainty.

For a version that constructs K&S representations on a Boolean event algebra and then compares decision rules, see `Examples/PreciseVsImpreciseGrounded.lean`.

Appendix: The KSSeparation Discovery

This appendix documents the discovery that the separation axiom is **independent** of the base K&S axioms—a concrete example of formalization exposing hidden assumptions in informal mathematics.

Timeline

Date	Event	Source
Nov 30, 2025	Initial `Algebra.lean` created	Commit 53f7a46
Dec 2, 2025	AI agents (likely Claude Code) introduce `KSSeparation`	Commit af3847d
Dec 4, 2025	1382-line proof attempt (`SeparationProof.lean`)	Commit d8ed2fb
Dec 19–20, 2025	Ben Goertzel provides "sandwich" framing; collaborative debugging	v1–v2 documents
Jan 8, 2026	Ben Goertzel suggests adopting separation as axiom	Private comm.
Jan 8, 2026	Counterexample proves independence (necessity)	Commit 6592c37
Jan 13, 2026	`SandwichSeparation.lean` consolidates results	Commit 9f150fa

Phase 1: Initial Optimism (Dec 2, 2025)

The original comment in `Algebra.lean` was optimistic:

> *"This is NOT a primitive axiom—it is derivable from the Knuth-Skilling axioms above. However, for organizational clarity, we factor it into a typeclass..."*

This matched the K&S paper's informal presentation, which treats separation as following naturally from the base axioms.

Phase 2: Failed Derivation Attempts (Dec 4, 2025)

A 1382-line file `SeparationProof.lean` attempted to prove `KSSeparation` from the base axioms. The proof never completed—gaps remained unfilled despite substantial effort.

Phase 3: Collaborative Debugging with Ben Goertzel (Dec 19–20, 2025)

Ben Goertzel's documents "Foundations of Inference: New Proofs" provided:

- The "sandwich" terminology for the separation property
- Detailed analysis of the A/B/C separation sets
- The "base-indexing" insight (Remark 4, Lemma 7)
- Categorical framing via monoidal functors

This initiated a rapid back-and-forth: Ben noted that Lemma 7 was where the theorem prover was "flailing." The next day (Dec 20), Codex constructed a counterexample to the Lemma 7 approach, and Ben responded with an attempted fix. This collaborative debugging helped clarify what additional structure the proof required.

Phase 4: Ben's Axiom Suggestion (Jan 8, 2026)

On January 8, 2026, Ben Goertzel suggested elevating separation to an explicit axiom:

> *"You could just assume something like: For any strictly positive atom a and any two values $x < y$, there exist integers n, m such that the scaled atom $n \cdot a$ falls 'between' the scaled values $m \cdot x$ and $m \cdot y$."*

Phase 5: The Counterexample (Jan 8, 2026)

Shortly after Ben's suggestion, the formalization proved his intuition was not merely convenient but **necessary**. Rather than continuing proof attempts, we constructed a **countermodel**: the semidirect product $\mathrm{SD} = \mathrm{WithBot}(\mathbb{N}^+ \times_{\mathrm{lex}} \mathbb{N})$ satisfies all base K&S axioms but fails separation.

This formally proved: **no derivation of separation from the base axioms can exist**. Ben's intuition and the formal verification converged independently on the same conclusion.

Attribution

- **KSSeparation typeclass**: Introduced by AI agents (likely Claude Code), Dec 2, 2025
- **"Sandwich" terminology & categorical framing**: Ben Goertzel, Dec 19–20, 2025
- **Suggestion to adopt separation as axiom**: Ben Goertzel, Jan 8, 2026
- **Semidirect product countermodel**: AI agents (likely Claude Code or Codex), Jan 8, 2026
- **Consolidation**: Formalized Jan 13, 2026, combining contributions

Note: Git commits are authored by the human user (Zar Goertzel), so the specific AI agent cannot be definitively identified from version control. Attribution to "Claude Code" is based on project records showing Claude Code assistance throughout this period.

Mathematical Significance

The countermodel proves that the K&S base axioms (associativity, strict monotonicity, identity-as-minimum) do **NOT** imply:

1. Commutativity
2. The separation property ("sandwich" / rational approximation)

Both must be **postulated as additional axioms** or **derived from stronger assumptions** (e.g., density, Archimedean properties on the real line, or the equivalent No Anomalous Pairs condition from ordered semigroup theory).

Lesson for Formalization

This is exactly what formalization is for: finding gaps that informal proofs skip over. The K&S paper's presentation *suggests* separation follows naturally, but rigorous formalization reveals it requires explicit justification. The countermodel construction took the project from "we can't prove it" to "we know it's unprovable."

13 Build Instructions

From the Mettapedia project root:

```
export LAKE_JOBS=3
nice -n 19 lake build Mettapedia.ProbabilityTheory.KnuthSkilling
```

For the FOI reviewer entrypoint only:

```
export LAKE_JOBS=3
nice -n 19 lake build Mettapedia.ProbabilityTheory.KnuthSkilling.
    FoundationsOfInference
```

For memory-intensive grid/induction files (e.g. ThetaPrime.lean):

```
ulimit -v 6291456
export LAKE_JOBS=1
nice -n 19 lake build Mettapedia.ProbabilityTheory.KnuthSkilling.Additive.Proofs.
    GridInduction.Main
```

Acknowledgments

Thanks to Ben Goertzel for discussion, brainstorming to unstick the formalization, for proposing to axiomatize separation, and for initially pointing me to the Knuth-Skilling paper years ago. Thanks to Eray Özkural for engaging in discussions about the interesting progress of these foundational explorations with me. Thanks also to the AI systems (Claude, ChatGPT Pro, Codex) for their tireless patience and enthusiasm in contributing to mathematical developments—ChatGPT Pro notably discovered the Alimov theorem that provided the classical foundation for the representation proof. The formalization builds on Mathlib, the Lean mathematical library maintained by the Lean community.

References

[1] Knuth, K. H. and Skilling, J. (2012). "Foundations of Inference." *Axioms* 1(1), 38–73. arXiv:1008.4831.

[2] Hölder, O. (1901). "Die Axiome der Quantität und die Lehre vom Mass." *Ber. Verh. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Cl.* 53, 1–64.

[3] Alimov, N. G. (1950). "On ordered semigroups." *Izv. Akad. Nauk SSSR Ser. Mat.* 14, 569–576.

[4] Fuchs, L. (1963). *Partially Ordered Algebraic Systems.* Pergamon Press.

[5] Klazar, M. (2016). "Alimov's theorem: any ordered semigroup without infinitesimals is commutative." Preprint.

[6] Paul, E. (2024). "OrderedSemigroups: Formalization of Ordered Semigroups in Lean 4." `https://github.com/ericluap/OrderedSemigroups` See also: `https://ericluap.github.io/`

[7] Aczél, J. (1966). *Lectures on Functional Equations and Their Applications.* Academic Press.

[8] Shannon, C. E. (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal* 27, 379–423, 623–656.

[9] Kolmogorov, A. N. (1933). *Grundbegriffe der Wahrscheinlichkeitsrechnung.* Springer. English translation: *Foundations of the Theory of Probability* (1956).

[10] Cox, R. T. (1946). "Probability, Frequency and Reasonable Expectation." *American Journal of Physics* 14(1), 1–13.

[11] Shore, J. E. and Johnson, R. W. (1980). "Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy." *IEEE Transactions on Information Theory* 26(1), 26–37.

[12] Uffink, J. (1995). "Can the Maximum Entropy Principle Be Explained as a Consistency Requirement?" *Studies in History and Philosophy of Modern Physics* 26(3), 223–261. doi: 10.1016/1355-2198(95)00015-1.

[13] Faddeev, D. K. (1956). "On the concept of entropy of a finite probabilistic scheme." *Uspekhi Mat. Nauk* 11, 227–231.

[14] Solomonoff, R. J. (1964). "A Formal Theory of Inductive Inference." *Information and Control* 7, 1–22, 224–254.