Human review packet: GJ19OptimalBinaryRatingSystems

Generated from byte-pinned source excerpts, the expanded PaperInterface specifications, and hash-bound raw-source-to-Spec screening records.

Paper: GJ19OptimalBinaryRatingSystems
Paper id: GJ19OptimalBinaryRatingSystems
Source version: AISTATS 2019 / PMLR 89 version, 2019
Source record: audit/paper_statement_map.json
Rows: 33
Generated: 2026-09-07
Source URL: official source

How to use this packet

For each source claim, compare the exact source excerpts with the expanded PaperInterface specification. Mark up this PDF or fill the blank reviewer box in the TeX source; return the annotated file for reconciliation into the paper-local human-review log.

Open the interactive dashboard

The interactive dashboard is an optional alternative to using this PDF; it presents the same review surface rather than an additional review requirement. After forking and cloning (or pulling) AppliedModelingLib, run the following from the repository root. The cache command is needed only the first time in a new clone.

```
cd <your-repository-checkout>
lake exe cache get
python3 scripts/review_dashboard.py --paper GJ19OptimalBinaryRatingSystems --serve
```

Open http://127.0.0.1:8765/ in a browser and keep that terminal running while reviewing. The dashboard records saved annotations in this paper's local reviewer-owned review record.

Contents

- Governing Lean declarations (145; supporting context)
- Paper-specific semantic prerequisites (25; not paper claims)
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_large_deviation_rate
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_lexicographic_optimality
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_pairwise_accuracy_eq1
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_step_rule_partition_levels
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_weighted_objective_eq2
  - GJ19OptimalBinaryRatingSystems.ProofBridge.source_theoremB1_quantile_representation
  - GJ19OptimalBinaryRatingSystems.SourceKnownTypeExperiment
  - GJ19OptimalBinaryRatingSystems.sourceQualityDomain
  - GJ19OptimalBinaryRatingSystems.SourceMatchingFunction
  - GJ19OptimalBinaryRatingSystems.SourceQuestionDistribution
  - GJ19OptimalBinaryRatingSystems.SourceQuestionDesignSolution
  - GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
  - GJ19OptimalBinaryRatingSystems.SourceSystemState
  - GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment
  - GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate
  - GJ19OptimalBinaryRatingSystems.kendallConstantWeightOrderedPairIntervalObjective
  - GJ19OptimalBinaryRatingSystems.sourceMatchingCount
  - GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet
  - GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionKernel
  - GJ19OptimalBinaryRatingSystems.sourceBernoulliKL
  - GJ19OptimalBinaryRatingSystems.sourceEmpiricalReputationScore
  - GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
  - GJ19OptimalBinaryRatingSystems.sourceSystemStateMass
  - GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective
  - GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput

- Source claims (33)
  - source_theorem31_adjacent_rate_eq3Spec
  - theorem31_source_cell_matching_rate_eq_lower_cutpointSpec
  - source_lemma31_equalization_eq4Spec
  - lemma31_closed_adjacent_rate_formulaSpec
  - theorem32_weighted_nested_bisection_loss_and_runtimeSpec
  - section4_induced_binary_responseSpec
  - section4_l1_design_objectiveSpec
  - appendixB1_state_updateSpec
  - lemmaB1_matching_rate_shiftSpec
  - paper_theoremB1_uniform_subsequence_principle_to_of_quantile_floor_tendstoUniformlyOn_geometric_meshSpec
  - paper_lemmaB2_knownTypeExperiment_uniform_convergence_of_lipschitz_trackingSpec
  - lemmaB2_knownTypeExperiment_random_question_sllnSpec
  - paper_lemmaB3_unknownTypeExperiment_uniform_convergence_of_ranking_trackingSpec
  - lemmaB3_unknownTypeExperiment_random_question_response_and_rankSpec
  - source_lemmaC1_pairwise_error_rate_from_derivativesSpec
  - source_lemmaC2_binary_complement_rateSpec
  - source_theoremC1_laplace_principleSpec
  - paper_lemmaC3_partition_integral_hasExponentialRate_of_min_component_uniform_logRateSpec
  - paper_lemmaC4_finite_selected_pullback_positive_rate_and_nonfiniteStep_source_zero_rateSpec
  - source_lemmaC5_uniform_doubled_chainSpec
  - source_lemmaC5_refinement_equations24_25Spec
  - paper_corollaryC2_uniform_equalized_last_rate_tendsto_zeroSpec
  - paper_corollaryC2_uniform_equalized_adjacent_mesh_tendsto_zeroSpec
  - lemmaC6_monotone_matching_penultimate_level_boundSpec
  - lemmaC7_uniform_doubled_objective_rate_ge_one_fifth_old_objectiveSpec
  - source_lemmaC8_uniform_first_level_polynomial_lower_boundSpec
  - corollaryC3_monotone_scaled_first_level_ge_half_inv_adjacent_count_sqSpec
  - source_lemmaC9_nested_bisection_runtime_log_squaredSpec
  - source_theoremB1_proof_selector_nestingSpec
  - source_corollaryC4_kendall_spearman_subsequenceSpec
  - paper_lemmaC10_spearman_linear_weight_ordered_pair_interval_objective_eq_gap_objectiveSpec
  - paper_lemmaC11_kendall_constant_weight_ordered_pair_interval_objective_le_equispacedSpec
  - paper_lemmaC12_spearman_linear_weight_ordered_pair_interval_objective_le_equispacedSpec

Governing Lean declarations

These exact graph-bound declaration bodies are retained by the displayed specifications or reviewed prerequisites. They are supporting context and do not add source-result rows or semantic judgments.

AppliedModelingLib.Optimization.IsLexicographicMaximizerOn

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → (α → Prop) → (α → ℝ) → (α → ℝ) → α → Prop
```

value:
```
fun {α : Type u_1} (feasible : α → Prop) (primary secondary : α → ℝ) (x : α) =>
  feasible x ∧ ∀ (y : α), feasible y → primary y < primary x ∨ primary y = primary x ∧ secondary y ≤ secondary x
```

AppliedModelingLib.Optimization.IsMaximizerOn

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → (α → Prop) → (α → ℝ) → α → Prop
```

value:
```
fun {α : Type u_1} (feasible : α → Prop) (objective : α → ℝ) (x : α) =>
  feasible x ∧ ∀ (y : α), feasible y → objective y ≤ objective x
```

AppliedModelingLib.Optimization.IsMinimizerOn

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → (α → Prop) → (α → ℝ) → α → Prop
```

value:
```
fun {α : Type u_1} (feasible : α → Prop) (objective : α → ℝ) (x : α) =>
  feasible x ∧ ∀ (y : α), feasible y → objective x ≤ objective y
```

AppliedModelingLib.Optimization.nestedBisectionStepBound

Declaration location: reusable library

Lean declaration body

type:
```
ℕ → ℕ → ℕ
```

value:
```
fun (M L : ℕ) => (L + 1) * (1 + (M - 3) * L)
```

AppliedModelingLib.Optimization.realBisectionMidpoint

Declaration location: reusable library

Lean declaration body

type:
```
ℝ → ℝ → ℝ
```

value:
```
fun (lower upper : ℝ) => (lower + upper) / 2
```

AppliedModelingLib.Optimization.realBisectionStep

Declaration location: reusable library

Lean declaration body

type:
(ℝ → Bool) → ℝ → ℝ → ℝ × ℝ

value:
fun (above : ℝ → Bool) (lower upper : ℝ) =>
  have mid : ℝ := AppliedModelingLib.Optimization.realBisectionMidpoint lower upper;
  if above mid = true then (lower, mid) else (mid, upper)

Direct retained dependencies

- AppliedModelingLib.Optimization.realBisectionMidpoint

AppliedModelingLib.Optimization.realBisectionStepFn

Declaration location: reusable library

Lean declaration body

type:
(ℝ → Bool) → ℝ × ℝ → ℝ × ℝ

value:
fun (above : ℝ → Bool) (p : ℝ × ℝ) => AppliedModelingLib.Optimization.realBisectionStep above p.1 p.2

Direct retained dependencies

- AppliedModelingLib.Optimization.realBisectionStep

AppliedModelingLib.Optimization.realBisectionRun

Declaration location: reusable library

Lean declaration body

type:
(ℝ → Bool) → ℕ → ℝ → ℝ → ℝ × ℝ

value:
fun (above : ℝ → Bool) (n : ℕ) (lower upper : ℝ) =>
  (AppliedModelingLib.Optimization.realBisectionStepFn above)^[n] (lower, upper)

Direct retained dependencies

- AppliedModelingLib.Optimization.realBisectionStepFn

AppliedModelingLib.Probability.FiniteMeasurableSetPartition

Declaration location: reusable library

Lean declaration body

field support:
{α : Type u_1} →
  [inst : MeasurableSpace α] →
    {μ : MeasureTheory.Measure α} →
      {Piece : Type u_2} →
        [inst_1 : Fintype Piece] → AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece → Set α

field pieceSet:
{α : Type u_1} →
  [inst : MeasurableSpace α] →
    {μ : MeasureTheory.Measure α} →
      {Piece : Type u_2} →
        [inst_1 : Fintype Piece] → AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece → Piece → Set α

field measurable_piece:
∀ {α : Type u_1} [inst : MeasurableSpace α] {μ : MeasureTheory.Measure α} {Piece : Type u_2} [inst_1 : Fintype Piece]
  (self : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece) (piece : Piece),
  MeasurableSet (self.pieceSet piece)

field pairwiseDisjoint:
∀ {α : Type u_1} [inst : MeasurableSpace α] {μ : MeasureTheory.Measure α} {Piece : Type u_2} [inst_1 : Fintype Piece]
  (self : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece),
  Pairwise fun (piece₁ piece₂ : Piece) => Disjoint (self.pieceSet piece₁) (self.pieceSet piece₂)

field cover:
∀ {α : Type u_1} [inst : MeasurableSpace α] {μ : MeasureTheory.Measure α} {Piece : Type u_2} [inst_1 : Fintype Piece]
  (self : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece), Set.iUnion self.pieceSet = self.support

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints

Declaration location: reusable library

Lean declaration body

```
type:
(μ : MeasureTheory.Measure ℝ) →
  (n : ℕ) → (cut : ℕ → ℝ) → Monotone cut → AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ (Fin n)

value:
fun (μ : MeasureTheory.Measure ℝ) (n : ℕ) (cut : ℕ → ℝ) (hmono : Monotone cut) =>
  { support := Set.Ioc (cut 0) (cut n), pieceSet := fun (piece : Fin n) => Set.Ioc (cut ↑piece) (cut (↑piece + 1)),
    measurable_piece :=
      AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints._proof_1 n cut,
    pairwiseDisjoint :=
      AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints._proof_2 n cut hmono,
    cover :=
      AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints._proof_3 n cut hmono }
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsMidpoint

Declaration location: reusable library

Lean declaration body

```
type:
(ℕ → ℝ) → {n : ℕ} → Fin n → ℝ

value:
fun (cut : ℕ → ℝ) {n : ℕ} (piece : Fin n) => (cut ↑piece + cut (↑piece + 1)) / 2
```

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsProdMidpoint

Declaration location: reusable library

Lean declaration body

```
type:
(ℕ → ℝ) → (ℕ → ℝ) → {n m : ℕ} → Fin n × Fin m → ℝ × ℝ

value:
fun (cut₁ cut₂ : ℕ → ℝ) {n m : ℕ} (piece : Fin n × Fin m) =>
  (AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsMidpoint cut₁ piece.1,
    AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsMidpoint cut₂ piece.2)
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsMidpoint

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.piecewiseConstKernel

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} →
  {Piece : Type u_2} →
    [inst : MeasurableSpace α] →
      [inst_1 : Fintype Piece] →
        {μ : MeasureTheory.Measure α} →
          AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece → (Piece → ℕ → ℝ) → ℕ → α → ℝ

value:
fun {α : Type u_1} {Piece : Type u_2} [MeasurableSpace α] [Fintype Piece] {μ : MeasureTheory.Measure α}
    (P : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece) (p : Piece → ℕ → ℝ) (n : ℕ) (x : α) =>
  ∑ piece : Piece, (P.pieceSet piece).indicator (fun (x : α) => p piece n) x
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.prod

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} →
  {Piece : Type u_2} →
    [inst : MeasurableSpace α] →
      [inst_1 : Fintype Piece] →
        {μ : MeasureTheory.Measure α} →
          {β : Type u_4} →
            {Pieceβ : Type u_5} →
              [inst_2 : MeasurableSpace β] →
                [inst_3 : Fintype Pieceβ] →
                  {ν : MeasureTheory.Measure β} →
                    AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece →
                      AppliedModelingLib.Probability.FiniteMeasurableSetPartition ν Pieceβ →
                        AppliedModelingLib.Probability.FiniteMeasurableSetPartition (μ.prod ν) (Piece × Pieceβ)

value:
fun {α : Type u_1} {Piece : Type u_2} [MeasurableSpace α] [Fintype Piece] {μ : MeasureTheory.Measure α} {β : Type u_4}
    {Pieceβ : Type u_5} [MeasurableSpace β] [Fintype Pieceβ] {ν : MeasureTheory.Measure β}
    (P : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece)
    (Q : AppliedModelingLib.Probability.FiniteMeasurableSetPartition ν Pieceβ) =>
  { support := P.support ×ˢ Q.support,
    pieceSet := fun (piece : Piece × Pieceβ) => P.pieceSet piece.1 ×ˢ Q.pieceSet piece.2,
    measurable_piece := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.prod._proof_1 P Q,
    pairwiseDisjoint := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.prod._proof_2 P Q,
    cover := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.prod._proof_3 P Q }
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.subpartition

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} →
  {Piece : Type u_2} →
    [inst : MeasurableSpace α] →
      [inst_1 : Fintype Piece] →
        {μ : MeasureTheory.Measure α} →
          AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece →
            (selected : Piece → Prop) →
              [inst_2 : DecidablePred selected] →
                AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ { piece : Piece // selected piece }

value:
fun {α : Type u_1} {Piece : Type u_2} [MeasurableSpace α] [Fintype Piece] {μ : MeasureTheory.Measure α}
    (P : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece) (selected : Piece → Prop)
    [DecidablePred selected] =>
  { support := ⋃ (piece : { piece : Piece // selected piece }), P.pieceSet ↑piece,
    pieceSet := fun (piece : { piece : Piece // selected piece }) => P.pieceSet ↑piece,
    measurable_piece := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.subpartition._proof_1 P selected,
    pairwiseDisjoint := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.subpartition._proof_2 P selected,
    cover := AppliedModelingLib.Probability.FiniteMeasurableSetPartition.subpartition._proof_3 P selected }
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition

AppliedModelingLib.Probability.FiniteMeasurableSetPartition.selectedProduct

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} →
  {Piece : Type u_2} →
    [inst : MeasurableSpace α] →
      [inst_1 : Fintype Piece] →
        {μ : MeasureTheory.Measure α} →
          {β : Type u_4} →
            {Pieceβ : Type u_5} →
              [inst_2 : MeasurableSpace β] →
                [inst_3 : Fintype Pieceβ] →
```

{ν : MeasureTheory.Measure β} →
AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece →
AppliedModelingLib.Probability.FiniteMeasurableSetPartition ν Pieceβ →
(selected : Piece × Pieceβ → Prop) →
[inst_4 : DecidablePred selected] →
AppliedModelingLib.Probability.FiniteMeasurableSetPartition (μ.prod ν)
{ piece : Piece × Pieceβ // selected piece }

value:
fun {α : Type u_1} {Piece : Type u_2} [MeasurableSpace α] [Fintype Piece] {μ : MeasureTheory.Measure α} {β : Type u_4}
{Pieceβ : Type u_5} [MeasurableSpace β] [Fintype Pieceβ] {ν : MeasureTheory.Measure β}
(P : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Piece)
(Q : AppliedModelingLib.Probability.FiniteMeasurableSetPartition ν Pieceβ) (selected : Piece × Pieceβ → Prop)
[DecidablePred selected] =>
(P.prod Q).subpartition selected

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.prod
- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.subpartition

AppliedModelingLib.Probability.FiniteRatingLDPModel

Declaration location: reusable library

Lean declaration body

field typeLaw:
{θ : Type u_2} →
{Rating : Type u_3} →
[inst : Fintype Rating] →
[inst_1 : DecidableEq Rating] → AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → θ → PMF Rating

field score:
{θ : Type u_2} →
{Rating : Type u_3} →
[inst : Fintype Rating] →
[inst_1 : DecidableEq Rating] → AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → Rating → ℝ

AppliedModelingLib.Probability.HasAEEssentialInfimum

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [inst : MeasurableSpace α] → MeasureTheory.Measure α → (α → ℝ) → ℝ → Prop

value:
fun {α : Type u_1} [MeasurableSpace α] (μ : MeasureTheory.Measure α) (phi : α → ℝ) (rate : ℝ) =>
(∀ᵐ (x : α) ∂μ, rate ≤ phi x) ∧ ∀ ε > 0, ∃ (s : Set α), MeasurableSet s ∧ 0 < μ.real s ∧ ∀ x ∈ s, phi x ≤ rate + ε

AppliedModelingLib.Probability.HasPositiveWeightNearAEEssentialInfimum

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [inst : MeasurableSpace α] → MeasureTheory.Measure α → (α → ℝ) → (α → ℝ) → ℝ → Prop

value:
fun {α : Type u_1} [MeasurableSpace α] (μ : MeasureTheory.Measure α) (w phi : α → ℝ) (rate : ℝ) =>
∀ ε > 0, ∃ (s : Set α) (c : ℝ), 0 < μ.real s ∧ 0 < c ∧ ∀ᵐ (x : α) ∂μ.restrict s, c ≤ w x ∧ phi x ≤ rate + ε

AppliedModelingLib.Probability.bernoulliFirstEndpointEqualSplit

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ

value:
fun (pHi : ℝ) => (1 - √(1 - pHi)) / 2

AppliedModelingLib.Probability.bernoulliInteriorFailureSplitNumerator

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pLo pHi : ℝ) => (√(1 - pLo) - √(1 - pHi)) ^ 2

AppliedModelingLib.Probability.bernoulliInteriorSuccessSplitNumerator

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pLo pHi : ℝ) => (√pHi - √pLo) ^ 2

AppliedModelingLib.Probability.bernoulliInteriorSplitDenominator

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pLo pHi : ℝ) =>
  AppliedModelingLib.Probability.bernoulliInteriorFailureSplitNumerator pLo pHi +
    AppliedModelingLib.Probability.bernoulliInteriorSuccessSplitNumerator pLo pHi

Direct retained dependencies

- AppliedModelingLib.Probability.bernoulliInteriorFailureSplitNumerator
- AppliedModelingLib.Probability.bernoulliInteriorSuccessSplitNumerator

AppliedModelingLib.Probability.bernoulliInteriorEqualSplit

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (pLo pHi : ℝ) =>
  AppliedModelingLib.Probability.bernoulliInteriorFailureSplitNumerator pLo pHi /
    AppliedModelingLib.Probability.bernoulliInteriorSplitDenominator pLo pHi

Direct retained dependencies

- AppliedModelingLib.Probability.bernoulliInteriorFailureSplitNumerator
- AppliedModelingLib.Probability.bernoulliInteriorSplitDenominator

AppliedModelingLib.Probability.bernoulliKL

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (a b : ℝ) => a * Real.log (a / b) + (1 - a) * Real.log ((1 - a) / (1 - b))

AppliedModelingLib.Probability.bernoulliLastEndpointEqualSplit

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ

value:
fun (pLo : ℝ) => (1 + √pLo) / 2

AppliedModelingLib.Probability.binaryRatingScore

Declaration location: reusable library

Lean declaration body

type:
Bool → ℝ

value:
fun (b : Bool) => if b = true then 1 else 0

AppliedModelingLib.Probability.finiteIidScoreSum

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → {ι : Type u_2} → [Fintype ι] → (α → ℝ) → (ι → α) → ℝ

value:
fun {α : Type u_1} {ι : Type u_2} [Fintype ι] (score : α → ℝ) (sample : ι → α) => ∑ i : ι, score (sample i)

AppliedModelingLib.Probability.finiteMGF

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (z : ℝ) => ∑ a : α, (μ a).toReal * Real.exp (z * score a)

AppliedModelingLib.Probability.finiteLogMGF

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (z : ℝ) =>
  Real.log (AppliedModelingLib.Probability.finiteMGF μ score z)

Direct retained dependencies

- AppliedModelingLib.Probability.finiteMGF

AppliedModelingLib.Probability.FiniteRatingLDPModel.logMGF

Declaration location: reusable library

Lean declaration body

type:
{θ : Type u_2} →
  {Rating : Type u_3} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] → AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → θ → ℝ → ℝ

value:
fun {θ : Type u_2} {Rating : Type u_3} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating) (t : θ) (z : ℝ) =>
  AppliedModelingLib.Probability.finiteLogMGF (M.typeLaw t) M.score z

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.finiteLogMGF

AppliedModelingLib.Probability.finiteLegendreValue

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → ℝ → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (a z : ℝ) =>
  z * a - AppliedModelingLib.Probability.finiteLogMGF μ score z

Direct retained dependencies

- AppliedModelingLib.Probability.finiteLogMGF

AppliedModelingLib.Probability.finiteRateFunctionTop

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ → WithTop ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (score : α → ℝ) (a : ℝ) =>
  sSup (Set.range fun (z : ℝ) => ↑(AppliedModelingLib.Probability.finiteLegendreValue μ score a z))

Direct retained dependencies

- AppliedModelingLib.Probability.finiteLegendreValue

AppliedModelingLib.Probability.FiniteRatingLDPModel.rateFunctionTop

Declaration location: reusable library

Lean declaration body

type:
{θ : Type u_2} →
  {Rating : Type u_3} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] → AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → θ → ℝ → WithTop ℝ

value:
fun {θ : Type u_2} {Rating : Type u_3} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating) (t : θ) (a : ℝ) =>
  AppliedModelingLib.Probability.finiteRateFunctionTop (M.typeLaw t) M.score a

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.finiteRateFunctionTop

AppliedModelingLib.Probability.floorSampleCount

Declaration location: reusable library

Lean declaration body

type:
{Seller : Type u_1} → (Seller → ℝ) → Seller → ℕ → ℕ

value:
fun {Seller : Type u_1} (sampleRate : Seller → ℝ) (θ : Seller) (k : ℕ) => ⌊↑k * sampleRate θ⌋₊

AppliedModelingLib.Probability.logDecay

Declaration location: reusable library

Lean declaration body

type:
(ℕ → ℝ) → ℕ → ℝ

value:
fun (p : ℕ → ℝ) (n : ℕ) => -(↑n)⁻¹ * Real.log (p n)

AppliedModelingLib.Probability.HasExponentialRate

Declaration location: reusable library

Lean declaration body

type:
(ℕ → ℝ) → ℝ → Prop

value:
fun (p : ℕ → ℝ) (rate : ℝ) => Filter.Tendsto (AppliedModelingLib.Probability.logDecay p) Filter.atTop (nhds rate)

Direct retained dependencies

- AppliedModelingLib.Probability.logDecay

AppliedModelingLib.Probability.ExponentialRateCertificate

Declaration location: reusable library

Lean declaration body

field eventually_pos:
∀ {p : ℕ → ℝ} {rate : ℝ},
  AppliedModelingLib.Probability.ExponentialRateCertificate p rate → ∀ᶠ (n : ℕ) in Filter.atTop, 0 < p n

field has_rate:
∀ {p : ℕ → ℝ} {rate : ℝ},
  AppliedModelingLib.Probability.ExponentialRateCertificate p rate →
    AppliedModelingLib.Probability.HasExponentialRate p rate

Direct retained dependencies

- AppliedModelingLib.Probability.HasExponentialRate

AppliedModelingLib.Probability.realBernoulliPMF

Declaration location: reusable library

Lean declaration body

type:
(p : ℝ) → 0 ≤ p → p ≤ 1 → PMF Bool

value:
fun (p : ℝ) (hp0 : 0 ≤ p) (hp1 : p ≤ 1) =>
  PMF.bernoulli (p, hp0) (AppliedModelingLib.Probability.realBernoulliPMF._proof_1 p hp0 hp1)

AppliedModelingLib.Probability.realBinaryRatingLDPModel

Declaration location: reusable library

Lean declaration body

```
type:
{θ : Type u_1} →
  (successProb : θ → ℝ) →
    (∀ (t : θ), 0 ≤ successProb t) →
      (∀ (t : θ), successProb t ≤ 1) → AppliedModelingLib.Probability.FiniteRatingLDPModel θ Bool

value:
fun {θ : Type u_1} (successProb : θ → ℝ) (hprob0 : ∀ (t : θ), 0 ≤ successProb t)
    (hprob1 : ∀ (t : θ), successProb t ≤ 1) =>
  { typeLaw := fun (t : θ) => AppliedModelingLib.Probability.realBernoulliPMF (successProb t) (hprob0 t) (hprob1 t),
    score := AppliedModelingLib.Probability.binaryRatingScore }
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.binaryRatingScore
- AppliedModelingLib.Probability.realBernoulliPMF

AppliedModelingLib.Probability.twoBernoulliThresholdRate

Declaration location: reusable library

Lean declaration body

```
type:
ℝ → ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo tHi tLo a : ℝ) =>
  gHi * AppliedModelingLib.Probability.bernoulliKL a tHi + gLo * AppliedModelingLib.Probability.bernoulliKL a tLo
```

Direct retained dependencies

- AppliedModelingLib.Probability.bernoulliKL

AppliedModelingLib.Probability.twoSampleScoreGapSum

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating →
          {nHi nLo : ℕ} → ℝ → ℝ → (Fin nHi → Rating) × (Fin nLo → Rating) → ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) {nHi nLo : ℕ} (cHi cLo : ℝ)
    (sample : (Fin nHi → Rating) × (Fin nLo → Rating)) =>
  cHi * AppliedModelingLib.Probability.finiteIidScoreSum M.score sample.1 -
    cLo * AppliedModelingLib.Probability.finiteIidScoreSum M.score sample.2
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.finiteIidScoreSum

AppliedModelingLib.Probability.weightedBernoulliFailureBase

Declaration location: reusable library

Lean declaration body

```
type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo pHi pLo : ℝ) => (1 - pLo) ^ (gLo / (gHi + gLo)) * (1 - pHi) ^ (gHi / (gHi + gLo))
```

AppliedModelingLib.Probability.weightedBernoulliSuccessBase

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo pHi pLo : ℝ) => pLo ^ (gLo / (gHi + gLo)) * pHi ^ (gHi / (gHi + gLo))

AppliedModelingLib.Probability.weightedBernoulliClosedRateBase

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo pHi pLo : ℝ) =>
  AppliedModelingLib.Probability.weightedBernoulliFailureBase gHi gLo pHi pLo +
    AppliedModelingLib.Probability.weightedBernoulliSuccessBase gHi gLo pHi pLo

Direct retained dependencies

- AppliedModelingLib.Probability.weightedBernoulliFailureBase
- AppliedModelingLib.Probability.weightedBernoulliSuccessBase

AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate

Declaration location: reusable library

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo pHi pLo : ℝ) =>
  -(gHi + gLo) * Real.log (AppliedModelingLib.Probability.weightedBernoulliClosedRateBase gHi gLo pHi pLo)

Direct retained dependencies

- AppliedModelingLib.Probability.weightedBernoulliClosedRateBase

AppliedModelingLib.Probability.withTopRealScale

Declaration location: reusable library

Lean declaration body

type:
ℝ → WithTop ℝ → WithTop ℝ

value:
fun (c : ℝ) (x : WithTop ℝ) => WithTop.map (fun (r : ℝ) => c * r) x

AppliedModelingLib.Probability.FiniteRatingLDPModel.pairwiseRateObjectiveTop

Declaration location: reusable library

Lean declaration body

type:
{θ : Type u_2} →
  {Rating : Type u_3} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → (θ → ℝ) → θ → θ → ℝ → WithTop ℝ

value:
fun {θ : Type u_2} {Rating : Type u_3} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating) (sampleRate : θ → ℝ) (hi lo : θ) (a : ℝ) =>
  AppliedModelingLib.Probability.withTopRealScale (sampleRate hi) (M.rateFunctionTop hi a) +
    AppliedModelingLib.Probability.withTopRealScale (sampleRate lo) (M.rateFunctionTop lo a)

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.FiniteRatingLDPModel.rateFunctionTop
- AppliedModelingLib.Probability.withTopRealScale

AppliedModelingLib.Probability.FiniteRatingLDPModel.pairwiseThresholdRateTop

Declaration location: reusable library

Lean declaration body

```
type:
{θ : Type u_2} →
  {Rating : Type u_3} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating → (θ → ℝ) → θ → θ → WithTop ℝ

value:
fun {θ : Type u_2} {Rating : Type u_3} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel θ Rating) (sampleRate : θ → ℝ) (hi lo : θ) =>
  sInf (Set.range fun (a : ℝ) => M.pairwiseRateObjectiveTop sampleRate hi lo a)
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.FiniteRatingLDPModel.pairwiseRateObjectiveTop

AppliedModelingLib.Probability.pairwiseSellerThresholdRateTop

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → (Seller → ℝ) → Seller → Seller → WithTop ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (sampleRate : Seller → ℝ)
    (hi lo : Seller) =>
  M.pairwiseThresholdRateTop sampleRate hi lo
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.FiniteRatingLDPModel.pairwiseThresholdRateTop

AppliedModelingLib.finiteMax

Declaration location: reusable library

Lean declaration body

```
type:
{α : Type u_1} → [Fintype α] → [Nonempty α] → (α → ℝ) → ℝ

value:
fun {α : Type u_1} [Fintype α] [Nonempty α] (f : α → ℝ) => Finset.univ.sup' Finset.univ_nonempty f
```

AppliedModelingLib.finiteMin

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → [Nonempty α] → (α → ℝ) → ℝ

value:
fun {α : Type u_1} [Fintype α] [Nonempty α] (f : α → ℝ) => Finset.univ.inf' Finset.univ_nonempty f

AppliedModelingLib.pmfExp

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (α → ℝ) → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (f : α → ℝ) => ∑ a : α, (μ a).toReal * f a

AppliedModelingLib.pmfProb

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → [Fintype α] → PMF α → (p : α → Prop) → [DecidablePred p] → ℝ

value:
fun {α : Type u_1} [Fintype α] (μ : PMF α) (p : α → Prop) [DecidablePred p] =>
  AppliedModelingLib.pmfExp μ fun (a : α) => if p a then 1 else 0

Direct retained dependencies

- AppliedModelingLib.pmfExp

AppliedModelingLib.pmfProd

Declaration location: reusable library

Lean declaration body

type:
{α : Type u_1} → {β : Type u_2} → [Fintype α] → [Fintype β] → PMF α → PMF β → PMF (α × β)

value:
fun {α : Type u_1} {β : Type u_2} [Fintype α] [Fintype β] (μ : PMF α) (ν : PMF β) =>
  PMF.ofFintype (fun (p : α × β) => μ p.1 * ν p.2) (AppliedModelingLib.pmfProd._proof_1 μ ν)

AppliedModelingLib.pmfProduct

Declaration location: reusable library

Lean declaration body

type:
(ι : Type u_1) → (α : Type u_2) → [Fintype ι] → [DecidableEq ι] → [Fintype α] → PMF α → PMF (ι → α)

value:
fun (ι : Type u_1) (α : Type u_2) [Fintype ι] [DecidableEq ι] [Fintype α] (μ : PMF α) =>
  PMF.ofFintype (fun (f : ι → α) => ∏ i : ι, μ (f i)) (AppliedModelingLib.pmfProduct._proof_1 ι α μ)

AppliedModelingLib.Probability.twoSampleRatingLaw

Declaration location: reusable library

Lean declaration body

type:
```
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating →
          Seller → Seller → (nHi nLo : ℕ) → PMF ((Fin nHi → Rating) × (Fin nLo → Rating))
```

value:
```
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (hi lo : Seller) (nHi nLo : ℕ) =>
  AppliedModelingLib.pmfProd (AppliedModelingLib.pmfProduct (Fin nHi) Rating (M.typeLaw hi))
    (AppliedModelingLib.pmfProduct (Fin nLo) Rating (M.typeLaw lo))
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.pmfProd
- AppliedModelingLib.pmfProduct

AppliedModelingLib.Probability.twoSampleScoreGapLeftTailProb

Declaration location: reusable library

Lean declaration body

type:
```
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → Seller → Seller → ℕ → ℕ → ℝ → ℝ → ℝ
```

value:
```
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (hi lo : Seller) (nHi nLo : ℕ)
    (cHi cLo : ℝ) =>
  AppliedModelingLib.pmfProb (AppliedModelingLib.Probability.twoSampleRatingLaw M hi lo nHi nLo)
    fun (sample : (Fin nHi → Rating) × (Fin nLo → Rating)) =>
    AppliedModelingLib.Probability.twoSampleScoreGapSum M cHi cLo sample ≤ 0
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.twoSampleRatingLaw
- AppliedModelingLib.Probability.twoSampleScoreGapSum
- AppliedModelingLib.pmfProb

AppliedModelingLib.Probability.twoSampleFloorScoreGapLeftTailProb

Declaration location: reusable library

Lean declaration body

type:
```
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → (Seller → ℝ) → Seller → Seller → ℕ → ℝ
```

value:
```
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (sampleRate : Seller → ℝ) (hi lo : Seller)
    (k : ℕ) =>
  have nHi : ℕ := AppliedModelingLib.Probability.floorSampleCount sampleRate hi k;
  have nLo : ℕ := AppliedModelingLib.Probability.floorSampleCount sampleRate lo k;
  AppliedModelingLib.Probability.twoSampleScoreGapLeftTailProb M hi lo nHi nLo (↑nHi)⁻¹ (↑nLo)⁻¹
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.floorSampleCount
- AppliedModelingLib.Probability.twoSampleScoreGapLeftTailProb

AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate

Declaration location: reusable library

Lean declaration body

```
field rate:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    {Pair : Type u_3} →
      [inst : Fintype Rating] →
        [inst_1 : DecidableEq Rating] →
          {M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating} →
            {sampleRate : Seller → ℝ} →
              {pairHi pairLo : Pair → Seller} →
                AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate M sampleRate pairHi pairLo →
                  Pair → ℝ

field threshold_rate_top_eq:
∀ {Seller : Type u_1} {Rating : Type u_2} {Pair : Type u_3} [inst : Fintype Rating] [inst_1 : DecidableEq Rating]
  {M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating} {sampleRate : Seller → ℝ}
  {pairHi pairLo : Pair → Seller}
  (self : AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate M sampleRate pairHi pairLo) (p : Pair),
  AppliedModelingLib.Probability.pairwiseSellerThresholdRateTop M sampleRate (pairHi p) (pairLo p) = ↑(self.rate p)

field leftTail:
∀ {Seller : Type u_1} {Rating : Type u_2} {Pair : Type u_3} [inst : Fintype Rating] [inst_1 : DecidableEq Rating]
  {M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating} {sampleRate : Seller → ℝ}
  {pairHi pairLo : Pair → Seller}
  (self : AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate M sampleRate pairHi pairLo) (p : Pair),
  AppliedModelingLib.Probability.ExponentialRateCertificate
    (AppliedModelingLib.Probability.twoSampleFloorScoreGapLeftTailProb M sampleRate (pairHi p) (pairLo p)) (self.rate p)
```

Direct retained dependencies

- AppliedModelingLib.Probability.ExponentialRateCertificate
- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.pairwiseSellerThresholdRateTop
- AppliedModelingLib.Probability.twoSampleFloorScoreGapLeftTailProb

AppliedModelingLib.Probability.twoSampleScoreGapStrictLeftProb

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → Seller → Seller → ℕ → ℕ → ℝ → ℝ → ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (hi lo : Seller) (nHi nLo : ℕ)
    (cHi cLo : ℝ) =>
  AppliedModelingLib.pmfProb (AppliedModelingLib.Probability.twoSampleRatingLaw M hi lo nHi nLo)
    fun (sample : (Fin nHi → Rating) × (Fin nLo → Rating)) =>
    AppliedModelingLib.Probability.twoSampleScoreGapSum M cHi cLo sample < 0
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.twoSampleRatingLaw
- AppliedModelingLib.Probability.twoSampleScoreGapSum
- AppliedModelingLib.pmfProb

AppliedModelingLib.Probability.twoSampleScoreGapTieProb

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → Seller → Seller → ℕ → ℕ → ℝ → ℝ → ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (hi lo : Seller) (nHi nLo : ℕ)
    (cHi cLo : ℝ) =>
  AppliedModelingLib.pmfProb (AppliedModelingLib.Probability.twoSampleRatingLaw M hi lo nHi nLo)
    fun (sample : (Fin nHi → Rating) × (Fin nLo → Rating)) =>
    AppliedModelingLib.Probability.twoSampleScoreGapSum M cHi cLo sample = 0
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.twoSampleRatingLaw
- AppliedModelingLib.Probability.twoSampleScoreGapSum
- AppliedModelingLib.pmfProb

AppliedModelingLib.Probability.twoSamplePkComplementErrorProb

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → Seller → Seller → ℕ → ℕ → ℝ → ℝ → ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (hi lo : Seller) (nHi nLo : ℕ)
    (cHi cLo : ℝ) =>
  2 * AppliedModelingLib.Probability.twoSampleScoreGapStrictLeftProb M hi lo nHi nLo cHi cLo +
    AppliedModelingLib.Probability.twoSampleScoreGapTieProb M hi lo nHi nLo cHi cLo
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.twoSampleScoreGapStrictLeftProb
- AppliedModelingLib.Probability.twoSampleScoreGapTieProb

AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb

Declaration location: reusable library

Lean declaration body

```
type:
{Seller : Type u_1} →
  {Rating : Type u_2} →
    [inst : Fintype Rating] →
      [inst_1 : DecidableEq Rating] →
        AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating → (Seller → ℝ) → Seller → Seller → ℕ → ℝ

value:
fun {Seller : Type u_1} {Rating : Type u_2} [Fintype Rating] [DecidableEq Rating]
    (M : AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Rating) (sampleRate : Seller → ℝ) (hi lo : Seller)
    (k : ℕ) =>
  have nHi : ℕ := AppliedModelingLib.Probability.floorSampleCount sampleRate hi k;
  have nLo : ℕ := AppliedModelingLib.Probability.floorSampleCount sampleRate lo k;
  AppliedModelingLib.Probability.twoSamplePkComplementErrorProb M hi lo nHi nLo (↑nHi)⁻¹ (↑nLo)⁻¹
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.floorSampleCount
- AppliedModelingLib.Probability.twoSamplePkComplementErrorProb

AppliedModelingLib.strictUpperPairSet

Declaration location: reusable library

Lean declaration body

type:
ℝ × ℝ → Prop

value:
fun (a : ℝ × ℝ) => {q : ℝ × ℝ | q.2 < q.1} a

GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel

Declaration location: paper-local

Lean declaration body

field lo:
{μ : MeasureTheory.Measure ℝ} → GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℝ

field hi:
{μ : MeasureTheory.Measure ℝ} → GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℝ

field hlohi:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  self.lo < self.hi

field successProb:
{μ : MeasureTheory.Measure ℝ} → GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℝ → ℝ

field sampleRate:
{μ : MeasureTheory.Measure ℝ} → GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℝ → ℝ

field hprob_mono:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  Monotone self.successProb

field hprob0:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
  (θ : ℝ), 0 ≤ self.successProb θ

field hprob1:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
  (θ : ℝ), self.successProb θ ≤ 1

field hprob_meas:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  Measurable self.successProb

field hsample_meas:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  Measurable self.sampleRate

field weight:
{μ : MeasureTheory.Measure ℝ} →
  GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℝ × ℝ → ℝ

field hsource_weight_int:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  MeasureTheory.Integrable self.weight ((μ.prod μ).restrict AppliedModelingLib.strictUpperPairSet)

field hsource_weight_nonneg:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  ∀ᵐ (q : ℝ × ℝ) ∂(μ.prod μ).restrict AppliedModelingLib.strictUpperPairSet, 0 ≤ self.weight q

field hweight_cont:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
  (θ : ℝ), ContinuousAt self.weight (θ, θ)

field hweight_diag_cont:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
  (θ : ℝ), ContinuousAt (fun (x : ℝ) => self.weight (x, x)) θ

field hsample_cont:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
  (θ : ℝ), ContinuousAt self.sampleRate θ

field hweight_pos_on:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  ∀ θ ∈ Set.Ioo self.lo self.hi, 0 < self.weight (θ, θ)

field hsample_pos_on:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  ∀ θ ∈ Set.Ioo self.lo self.hi, 0 < self.sampleRate θ

field hprob_interior_on:
∀ {μ : MeasureTheory.Measure ℝ} (self : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ),
  ∀ θ ∈ Set.Ioo self.lo self.hi, 0 < self.successProb θ ∧ self.successProb θ < 1

Direct retained dependencies

- AppliedModelingLib.strictUpperPairSet

GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign

Declaration location: paper-local

Lean declaration body

field cutpoints:
{M : ℕ} → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → ℕ → ℝ

field endpointLevels:
{M : ℕ} → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → Fin (M + 2) → ℝ

GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRateBase

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gLo gHi tLo tHi : ℝ) =>
  (1 - tLo) ^ (gLo / (gLo + gHi)) * (1 - tHi) ^ (gHi / (gLo + gHi)) +
    tLo ^ (gLo / (gLo + gHi)) * tHi ^ (gHi / (gLo + gHi))

GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gLo gHi tLo tHi : ℝ) =>
  -(gLo + gHi) * Real.log (GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRateBase gLo gHi tLo tHi)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRateBase

GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo tHi tLo : ℝ) =>
  sInf (Set.range fun (a : ℝ) => AppliedModelingLib.Probability.twoBernoulliThresholdRate gHi gLo tHi tLo a)

Direct retained dependencies

- AppliedModelingLib.Probability.twoBernoulliThresholdRate

GJ19OptimalBinaryRatingSystems.adjacentHighIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 1) → Fin (m + 2)

value:
fun {m : ℕ} (i : Fin (m + 1)) => i.succ

GJ19OptimalBinaryRatingSystems.adjacentLowIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 1) → Fin (m + 2)

value:
fun {m : ℕ} (i : Fin (m + 1)) => i.castSucc

GJ19OptimalBinaryRatingSystems.assumption_lemmaC4_positive_support_interval_model

Declaration location: paper-local

Lean declaration body

type:
MeasureTheory.Measure ℝ → Type

value:
fun (μ : MeasureTheory.Measure ℝ) => GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel

GJ19OptimalBinaryRatingSystems.binaryEndpointAdjacentMaxWidth

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → ℝ

value:
fun {m : ℕ} (levels : Fin (m + 2) → ℝ) =>
  AppliedModelingLib.finiteMax fun (i : Fin (m + 1)) =>
    levels (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i) -
      levels (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i)

Direct retained dependencies

- AppliedModelingLib.finiteMax
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex

GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → (Fin (m + 2) → ℝ) → Prop

value:
fun {m : ℕ} (successProb sampleRate : Fin (m + 2) → ℝ) =>
  ∀ (i j : Fin (m + 1)),
    GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate successProb sampleRate i =
      GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate successProb sampleRate j

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate

GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → (Fin (m + 2) → ℝ) → ℝ

value:
fun {m : ℕ} (successProb sampleRate : Fin (m + 2) → ℝ) =>
  AppliedModelingLib.finiteMin (GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate successProb sampleRate)

Direct retained dependencies

- AppliedModelingLib.finiteMin
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → ℝ

value:
fun (M : ℕ) (design : GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M) =>
  GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective design.endpointLevels fun (x : Fin (M + 2)) =>
    1

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective

GJ19OptimalBinaryRatingSystems.binaryEndpointSampleRateNat

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → ℕ → ℝ

value:
fun {m : ℕ} (sampleRate : Fin (m + 2) → ℝ) (n : ℕ) => if h : n < m + 2 then sampleRate ⟨n, h⟩ else 0

GJ19OptimalBinaryRatingSystems.binaryRatingModel

Declaration location: paper-local

Lean declaration body

type:
{Seller : Type u_1} →
  (successProb : Seller → ℝ) →
    (∀ (θ : Seller), 0 ≤ successProb θ) →
      (∀ (θ : Seller), successProb θ ≤ 1) → AppliedModelingLib.Probability.FiniteRatingLDPModel Seller Bool

value:
fun {Seller : Type u_1} (successProb : Seller → ℝ) (hprob0 : ∀ (θ : Seller), 0 ≤ successProb θ)
    (hprob1 : ∀ (θ : Seller), successProb θ ≤ 1) =>
  AppliedModelingLib.Probability.realBinaryRatingLDPModel successProb hprob0 hprob1

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteRatingLDPModel
- AppliedModelingLib.Probability.realBinaryRatingLDPModel

GJ19OptimalBinaryRatingSystems.canonicalTinyEndpointLevels

Declaration location: paper-local

Lean declaration body

type:
(m : ℕ) → Fin (m + 2) → ℝ

value:
fun (m : ℕ) (a : Fin (m + 2)) => ↑↑a / (↑m + 1)

GJ19OptimalBinaryRatingSystems.clampedFloorLevelIndex

Declaration location: paper-local

Lean declaration body

type:
(m : ℕ) → ℝ → Fin (m + 2)

value:
fun (m : ℕ) (θ : ℝ) => ⟨min ⌊↑(m + 2) * θ⌋₊ (m + 1), GJ19OptimalBinaryRatingSystems.clampedFloorLevelIndex._proof_2 m θ⟩

GJ19OptimalBinaryRatingSystems.cutpointStepSuccessProb

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (ℕ → ℝ) → (Fin (m + 2) → ℝ) → ℝ → ℝ

value:
fun {m : ℕ} (cut : ℕ → ℝ) (levels : Fin (m + 2) → ℝ) (θ : ℝ) =>
  if h : ∃ (i : Fin (m + 2)), cut ↑i < θ ∧ θ ≤ cut (↑i + 1) then levels (Classical.choose h) else 0

GJ19OptimalBinaryRatingSystems.equispacedIntervalCutpoint

Declaration location: paper-local

Lean declaration body

type:
ℕ → ℕ → ℝ

value:
fun (M i : ℕ) => ↑i / ↑M

GJ19OptimalBinaryRatingSystems.firstAdjacentIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 1)

value:
fun {m : ℕ} => ⟨0, GJ19OptimalBinaryRatingSystems.firstAdjacentIndex._proof_2⟩

GJ19OptimalBinaryRatingSystems.firstLevelIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 2)

value:
fun {m : ℕ} => ⟨0, GJ19OptimalBinaryRatingSystems.firstLevelIndex._proof_2⟩

GJ19OptimalBinaryRatingSystems.intervalCutpointsEndpointFeasible

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℕ → ℝ) → Prop

value:
fun (M : ℕ) (s : ℕ → ℝ) => s 0 = 0 ∧ s M = 1

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignValue

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → ℝ

value:
fun (M : ℕ) (design : GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M) =>
  GJ19OptimalBinaryRatingSystems.kendallConstantWeightOrderedPairIntervalObjective M design.cutpoints

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.kendallConstantWeightOrderedPairIntervalObjective

GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 1)

value:
fun {m : ℕ} => ⟨m, GJ19OptimalBinaryRatingSystems.lastAdjacentIndex._proof_2⟩

GJ19OptimalBinaryRatingSystems.lastLevelIndex

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 2)

value:
fun {m : ℕ} => ⟨m + 1, GJ19OptimalBinaryRatingSystems.lastLevelIndex._proof_2⟩

GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → Prop

value:
fun {m : ℕ} (t : Fin (m + 2) → ℝ) =>
  t GJ19OptimalBinaryRatingSystems.firstLevelIndex = 0 ∧
    t GJ19OptimalBinaryRatingSystems.lastLevelIndex = 1 ∧
      ∀ (i : Fin (m + 1)),
        t (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i) < t (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.firstLevelIndex
- GJ19OptimalBinaryRatingSystems.lastLevelIndex

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignFeasible

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → Prop

value:
fun (M : ℕ) (design : GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M) =>
  GJ19OptimalBinaryRatingSystems.intervalCutpointsEndpointFeasible M design.cutpoints ∧
    GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector design.endpointLevels

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.intervalCutpointsEndpointFeasible

GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels

Declaration location: paper-local

Lean declaration body

type:
(m : ℕ) → Fin (m + 2) → ℝ

value:
fun (m : ℕ) (a : Fin (m + 2)) =>
  (if hm : 1 < m then
      Classical.choose (GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels._proof_1 m hm)
    else GJ19OptimalBinaryRatingSystems.canonicalTinyEndpointLevels m)
    a

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.canonicalTinyEndpointLevels

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M

value:
fun (M : ℕ) =>
  { cutpoints := GJ19OptimalBinaryRatingSystems.equispacedIntervalCutpoint M,
    endpointLevels := GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels M }

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels
- GJ19OptimalBinaryRatingSystems.equispacedIntervalCutpoint

GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedClampedFloorBetaSeq

Declaration location: paper-local

Lean declaration body

type:
ℕ → ℝ → ℝ

value:
fun (M : ℕ) (θ : ℝ) =>
  GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels (M - 2)
    (GJ19OptimalBinaryRatingSystems.clampedFloorLevelIndex (M - 2) θ)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedEndpointLevels
- GJ19OptimalBinaryRatingSystems.clampedFloorLevelIndex

GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo

Declaration location: paper-local

Lean declaration body

type:
(ℝ → ℝ) → ℝ → ℝ → Prop

value:
fun (successProb : ℝ → ℝ) (lo hi : ℝ) => (successProb '' Set.Ioo lo hi).Finite

GJ19OptimalBinaryRatingSystems.lemmaC4FiniteStepOnIoo

Declaration location: paper-local

Lean declaration body

type:
(ℝ → ℝ) → ℝ → ℝ → Prop

value:
fun (successProb : ℝ → ℝ) (lo hi : ℝ) =>
  GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo successProb lo hi ∧
    ∀ x ∈ Set.Ioo lo hi,
      ∀ y ∈ Set.Ioo lo hi,
        ∀ z ∈ Set.Ioo lo hi, x ≤ y → y ≤ z → successProb x = successProb z → successProb y = successProb x

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo

GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel

Declaration location: paper-local

Lean declaration body

type:
{μ : MeasureTheory.Measure ℝ} →
  GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ → ℕ → ℝ × ℝ → ℝ

value:
fun {μ : MeasureTheory.Measure ℝ} (R : GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel μ)
    (a : ℕ) (a_1 : ℝ × ℝ) =>
  AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
    (GJ19OptimalBinaryRatingSystems.binaryRatingModel R.successProb R.hprob0 R.hprob1) R.sampleRate a_1.1 a_1.2 a

Direct retained dependencies

- AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
- GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel
- GJ19OptimalBinaryRatingSystems.binaryRatingModel

GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar

Declaration location: paper-local

Lean declaration body

type:
MeasureTheory.Measure ℝ → (ℝ × ℝ → ℝ) → (ℕ → ℝ × ℝ → ℝ) → ℕ → ℝ

value:
fun (μ : MeasureTheory.Measure ℝ) (weight : ℝ × ℝ → ℝ) (sourcePbarKernel : ℕ → ℝ × ℝ → ℝ) (a : ℕ) =>
  ∫ (q : ℝ × ℝ) in AppliedModelingLib.strictUpperPairSet, weight q * sourcePbarKernel a q ∂μ.prod μ

Direct retained dependencies

- AppliedModelingLib.strictUpperPairSet

GJ19OptimalBinaryRatingSystems.monotoneIntervalCutpointsEndpointFeasible

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℕ → ℝ) → Prop

value:
fun (M : ℕ) (s : ℕ → ℝ) => Monotone s ∧ GJ19OptimalBinaryRatingSystems.intervalCutpointsEndpointFeasible M s

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.intervalCutpointsEndpointFeasible

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignFeasible

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → Prop

value:
fun (M : ℕ) (design : GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M) =>
  GJ19OptimalBinaryRatingSystems.monotoneIntervalCutpointsEndpointFeasible M design.cutpoints ∧
    GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector design.endpointLevels

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.monotoneIntervalCutpointsEndpointFeasible

GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount

Declaration location: paper-local

Lean declaration body

type:
ℕ → ℕ → ℕ → ℕ

value:
fun (M outerSteps innerSteps : ℕ) => outerSteps + outerSteps * ((M - 3) * innerSteps)

GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryClosedRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gLo gHi tLo tHi : ℝ) => GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRate gLo gHi tLo tHi

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingClosedRate

GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryRatingRate

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ

value:
fun (gHi gLo tHi tLo : ℝ) => GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingRate gHi gLo tHi tLo

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.adjacentBinaryRatingRate

GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator

Declaration location: paper-local

Lean declaration body

type:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3}
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (n : ℕ) (ω : Ω) =>
  if E.positiveResponse i n ω = true then 1 else 0

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment

GJ19OptimalBinaryRatingSystems.paperBernoulliKL

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ

value:
fun (a b : ℝ) => GJ19OptimalBinaryRatingSystems.sourceBernoulliKL a b

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.sourceBernoulliKL

GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (ℕ → ℝ) → (ℝ → ℝ) → Fin (m + 2) → ℝ

value:
fun {m : ℕ} (cut : ℕ → ℝ) (g : ℝ → ℝ) (i : Fin (m + 2)) => sInf (g '' Set.Icc (cut ↑i) (cut (↑i + 1)))

GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore

Declaration location: paper-local

Lean declaration body

type:
{Y : Type u_1} → [Fintype Y] → (ℝ → Y → ℝ) → (Y → ℝ) → ℝ → ℝ

value:
fun {Y : Type u_1} [Fintype Y] (ψ : ℝ → Y → ℝ) (H : Y → ℝ) (θ : ℝ) => ∑ y : Y, H y * ψ θ y

GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean

Declaration location: paper-local

Lean declaration body

type:
{Ω : Type u_1} → (ℕ → Ω → ℝ) → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} (observation : ℕ → Ω → ℝ) (N : ℕ) (ω : Ω) => (∑ n ∈ Finset.range N, observation n ω) / ↑N

GJ19OptimalBinaryRatingSystems.sourceInducedBinaryResponse

Declaration location: paper-local

Lean declaration body

type:
{Y : Type u_1} → [Fintype Y] → (ℝ → Y → ℝ) → (Y → ℝ) → ℝ → ℝ

value:
fun {Y : Type u_1} [Fintype Y] (ψ : ℝ → Y → ℝ) (H : Y → ℝ) (θ : ℝ) => ∑ y : Y, ψ θ y * H y

GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator

Declaration location: paper-local

Lean declaration body

type:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} →
      [DecidableEq Y] → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → Y → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3} [DecidableEq Y]
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (y : Y) (n : ℕ) (ω : Ω) =>
  if E.question i n ω = y then 1 else 0

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment

GJ19OptimalBinaryRatingSystems.sourceExperimentQuestionCount

Declaration location: paper-local

Lean declaration body

type:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} →
      [DecidableEq Y] → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → Y → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3} [DecidableEq Y]
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (y : Y) (N : ℕ) (ω : Ω) =>
  ∑ n ∈ Finset.range N, GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y n ω

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator

GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective

Declaration location: paper-local

Lean declaration body

```
type:
{Representative : Type u_1} →
  {Y : Type u_2} →
    [Fintype Representative] → [Fintype Y] → (Representative → ℝ) → (ℝ → ℝ) → (Representative → Y → ℝ) → (Y → ℝ) → ℝ

value:
fun {Representative : Type u_1} {Y : Type u_2} [Fintype Representative] [Fintype Y] (quality : Representative → ℝ)
    (β : ℝ → ℝ) (ψHat : Representative → Y → ℝ) (H : Y → ℝ) =>
  ∑ i : Representative, |β (quality i) - ∑ y : Y, ψHat i y * H y|
```

GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

Declaration location: paper-local

Lean declaration body

```
type:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} →
      [DecidableEq Y] → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → Y → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3} [DecidableEq Y]
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (y : Y) (n : ℕ) (ω : Ω) =>
  if E.question i n ω = y ∧ E.positiveResponse i n ω = true then 1 else 0
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment

GJ19OptimalBinaryRatingSystems.sourceExperimentPositiveQuestionCount

Declaration location: paper-local

Lean declaration body

```
type:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} →
      [DecidableEq Y] → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → Y → ℕ → Ω → ℝ

value:
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3} [DecidableEq Y]
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (y : Y) (N : ℕ) (ω : Ω) =>
  ∑ n ∈ Finset.range N, GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y n ω
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

GJ19OptimalBinaryRatingSystems.sourceScoreAfterRating

Declaration location: paper-local

Lean declaration body

```
type:
ℕ → ℝ → ℝ → ℝ

value:
fun (n : ℕ) (oldScore outcome : ℝ) => (↑n * oldScore + outcome) / (↑n + 1)
```

GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionMass

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℝ → ℝ) → ℝ → ℝ → Set ℝ → ENNReal

value:
fun (n : ℕ) (β : ℝ → ℝ) (θ oldScore : ℝ) (X : Set ℝ) =>
  (GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionKernel n β θ oldScore) X

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionKernel

GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℝ → ℝ) → (ℝ → ℝ) → GJ19OptimalBinaryRatingSystems.SourceSystemState → Set ℝ → Set ℝ → ENNReal

value:
fun (k : ℕ) (g β : ℝ → ℝ) (state : GJ19OptimalBinaryRatingSystems.SourceSystemState) (Θ X : Set ℝ) =>
  ∫⁻ (z : ℝ × ℝ) in (Θ ∩ GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet k g) ×ˢ Set.univ,
    GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionMass
      (GJ19OptimalBinaryRatingSystems.sourceMatchingCount k g z.1) β z.1 z.2 X ∂state +
  state ((Θ \ GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet k g) ×ˢ X)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceSystemState
- GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet
- GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionMass
- GJ19OptimalBinaryRatingSystems.sourceMatchingCount

GJ19OptimalBinaryRatingSystems.ProofBridge.appendixB1_state_update

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℝ → ℝ) → (ℝ → ℝ) → GJ19OptimalBinaryRatingSystems.SourceSystemState → Set ℝ → Set ℝ → ENNReal

value:
fun (k : ℕ) (g β : ℝ → ℝ) (state : GJ19OptimalBinaryRatingSystems.SourceSystemState) (Θ X : Set ℝ) =>
  GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass k g β state Θ X

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceSystemState
- GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass

GJ19OptimalBinaryRatingSystems.sourceStrictCutpointSet

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → (Fin (M + 1) → ℝ) → Prop

value:
fun (M : ℕ) (a : Fin (M + 1) → ℝ) => {s : Fin (M + 1) → ℝ | StrictMono s ∧ s 0 = 0 ∧ s (Fin.last M) = 1} a

GJ19OptimalBinaryRatingSystems.spearmanLinearWeightGapObjective

Declaration location: paper-local

Lean declaration body

type:
{M : ℕ} → (Fin M → ℝ) → ℝ

value:
fun {M : ℕ} (gap : Fin M → ℝ) => (1 - ∑ i : Fin M, gap i ^ 3) / 6

GJ19OptimalBinaryRatingSystems.spearmanLinearWeightIntervalObjective

Declaration location: paper-local

Lean declaration body

type:
ℕ → (ℕ → ℝ) → ℝ

value:
fun (M : ℕ) (s : ℕ → ℝ) =>
  GJ19OptimalBinaryRatingSystems.spearmanLinearWeightGapObjective fun (i : Fin M) => s (↑i + 1) - s ↑i

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.spearmanLinearWeightGapObjective

GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignValue

Declaration location: paper-local

Lean declaration body

type:
(M : ℕ) → GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M → ℝ

value:
fun (M : ℕ) (design : GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign M) =>
  GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective M design.cutpoints

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective

GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → Fin (m + 2) × Fin (m + 2) → Prop

value:
fun {m : ℕ} (piece : Fin (m + 2) × Fin (m + 2)) => ↑piece.1 + 1 ≤ ↑piece.2 ∧ (↑piece.1 ≠ 0 ∨ ↑piece.2 ≠ m + 1)

GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent

Declaration location: paper-local

Lean declaration body

type:
ℕ → Type

value:
fun (m : ℕ) =>
  { piece : Fin (m + 2) × Fin (m + 2) // GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece }

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected

GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected_decidablePred

Declaration location: paper-local

Lean declaration body

type:
```
{m : ℕ} →
  (a : Fin (m + 2) × Fin (m + 2)) → Decidable (GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected a)
```

value:
```
fun {m : ℕ} (a : Fin (m + 2) × Fin (m + 2)) => id inferInstance
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected

GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap

Declaration location: paper-local

Lean declaration body

type:
```
ℝ × ℝ ≃ᵐ ℝ × ℝ
```

value:
```
MeasurableEquiv.prodComm
```

GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceWeight

Declaration location: paper-local

Lean declaration body

type:
```
(ℝ × ℝ → ℝ) → ℝ × ℝ → ℝ
```

value:
```
fun (targetWeight : ℝ × ℝ → ℝ) (a : ℝ × ℝ) =>
  targetWeight (GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap a)
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap

GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_interval_partition

Declaration location: paper-local

Lean declaration body

type:
```
(μ : MeasureTheory.Measure ℝ) →
  (n : ℕ) → (cut : ℕ → ℝ) → Monotone cut → AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ (Fin n)
```

value:
```
fun (μ : MeasureTheory.Measure ℝ) (n : ℕ) (cut : ℕ → ℝ) (hmono : Monotone cut) =>
  AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints μ n cut hmono
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpoints

GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_component_midpoint

Declaration location: paper-local

Lean declaration body

type:
```
{m : ℕ} → (ℕ → ℝ) → GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent m → ℝ × ℝ
```

value:
```
fun {m : ℕ} (cut : ℕ → ℝ) (component : GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent m) =>
  AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsProdMidpoint cut cut ↑component
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.orderedRealIocNatCutpointsProdMidpoint
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected

GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition

Declaration location: paper-local

Lean declaration body

type:
```
(μ : MeasureTheory.Measure ℝ) →
  (n : ℕ) →
    (cut : ℕ → ℝ) →
      Monotone cut →
        (selected : Fin n × Fin n → Prop) →
          [inst : DecidablePred selected] →
            AppliedModelingLib.Probability.FiniteMeasurableSetPartition (μ.prod μ)
              { piece : Fin n × Fin n // selected piece }
```

value:
```
fun (μ : MeasureTheory.Measure ℝ) (n : ℕ) (cut : ℕ → ℝ) (hmono : Monotone cut) (selected : Fin n × Fin n → Prop)
    [DecidablePred selected] =>
  (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_interval_partition μ n cut hmono).selectedProduct
    (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_interval_partition μ n cut hmono) selected
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.selectedProduct
- GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_interval_partition

GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceKernel

Declaration location: paper-local

Lean declaration body

type:
```
MeasureTheory.Measure ℝ →
  {m : ℕ} →
    (cut : ℕ → ℝ) →
      Monotone cut →
        (Fin (m + 2) → ℝ) →
          (levels : Fin (m + 2) → ℝ) → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels → ℕ → ℝ × ℝ → ℝ
```

value:
```
fun (μ : MeasureTheory.Measure ℝ) {m : ℕ} (cut : ℕ → ℝ) (hmono : Monotone cut) (sampleRate levels : Fin (m + 2) → ℝ)
    (hlevels : GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels) (a : ℕ) (a_1 : ℝ × ℝ) =>
  (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
      GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).piecewiseConstKernel
    (fun
      (component :
        { piece : Fin (m + 2) × Fin (m + 2) //
          GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece })
      (k : ℕ) =>
    AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
      (GJ19OptimalBinaryRatingSystems.binaryRatingModel levels
        (GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector_nonneg hlevels)
        (GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector_le_one hlevels))
      sampleRate (↑component).2 (↑component).1 k)
    a a_1
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition.piecewiseConstKernel
- AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryRatingModel
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected_decidablePred
- GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition

GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceSupport

Declaration location: paper-local

Lean declaration body

```
type:
MeasureTheory.Measure ℝ → {m : ℕ} → (cut : ℕ → ℝ) → Monotone cut → ℝ × ℝ → Prop

value:
fun (μ : MeasureTheory.Measure ℝ) {m : ℕ} (cut : ℕ → ℝ) (hmono : Monotone cut) (a : ℝ × ℝ) =>
  (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
      GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).support
    a
```

Direct retained dependencies

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected_decidablePred
- GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition

GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceKernel

Declaration location: paper-local

Lean declaration body

```
type:
MeasureTheory.Measure ℝ →
  {m : ℕ} →
    (cut : ℕ → ℝ) →
      Monotone cut →
        (Fin (m + 2) → ℝ) →
          (levels : Fin (m + 2) → ℝ) → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels → ℕ → ℝ × ℝ → ℝ

value:
fun (μ : MeasureTheory.Measure ℝ) {m : ℕ} (cut : ℕ → ℝ) (hmono : Monotone cut) (sampleRate levels : Fin (m + 2) → ℝ)
    (hlevels : GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels) (a : ℕ) (a_1 : ℝ × ℝ) =>
  if
      GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap a_1 ∈
        GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceSupport μ cut hmono then
    GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceKernel μ cut hmono sampleRate levels hlevels a
      (GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap a_1)
  else 0
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceCoordinateMap
- GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceKernel
- GJ19OptimalBinaryRatingSystems.theorem31SelectedSourceSupport

GJ19OptimalBinaryRatingSystems.theorem32InnerSourceWeightedRateAbove

Declaration location: paper-local

Lean declaration body

type:
ℝ → ℝ → ℝ → ℝ → ℝ → Bool

value:
fun (gHi gLo pHi target x : ℝ) =>
  if AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate gHi gLo pHi x ≤ target then true else false

Direct retained dependencies

- AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate

GJ19OptimalBinaryRatingSystems.theorem32OuterSourceWeightedRateAbove

Declaration location: paper-local

Lean declaration body

type:
{n : ℕ} → (Fin (n + 2) → ℝ) → (Fin (n + 2) → ℝ) → Bool

value:
fun {n : ℕ} (candidate sampleRate : Fin (n + 2) → ℝ) =>
  if
      GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate candidate sampleRate
          GJ19OptimalBinaryRatingSystems.lastAdjacentIndex ≤
        GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate candidate sampleRate
          GJ19OptimalBinaryRatingSystems.firstAdjacentIndex then
    true
  else false

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate
- GJ19OptimalBinaryRatingSystems.firstAdjacentIndex
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionFromTop

Declaration location: paper-local

Lean declaration body

type:
(n : ℕ) → ℕ → (Fin (n + 2) → ℝ) → ℝ → ℝ → ℝ → ℕ → ℝ

value:
fun (n innerSteps : ℕ) (sampleRate : Fin (n + 2) → ℝ) (grid target lastLow : ℝ) (a : ℕ) =>
  Nat.brecOn a
    (GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionFromTop._f n innerSteps sampleRate grid
      target lastLow)

Direct retained dependencies

- AppliedModelingLib.Optimization.realBisectionRun
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.theorem32InnerSourceWeightedRateAbove

GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionLevels

Declaration location: paper-local

Lean declaration body

type:
(n : ℕ) → ℕ → (Fin (n + 2) → ℝ) → ℝ → ℝ → ℝ → Fin (n + 2) → ℝ

value:
fun (n innerSteps : ℕ) (sampleRate : Fin (n + 2) → ℝ) (grid target lastLow : ℝ) (a : Fin (n + 2)) =>
  if ↑a = 0 then 0
  else
    GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionFromTop n innerSteps sampleRate grid target
      lastLow (n + 1 - ↑a)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionFromTop

GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid

Declaration location: paper-local

Lean declaration body

```
type:
{m : ℕ} → (Fin (m + 2) → ℝ) → ℝ → ℝ

value:
fun {m : ℕ} (sampleRate : Fin (m + 2) → ℝ) (eps : ℝ) =>
  have sourceFirstLower : ℝ := (1 / ↑(m + 1)) ^ 2 / 2;
  have lastLower : ℝ := 1 - 1 / ↑(m + 1);
  have firstLower : ℝ := sourceFirstLower / 2;
  have gLast : ℝ :=
    sampleRate (GJ19OptimalBinaryRatingSystems.adjacentLowIndex GJ19OptimalBinaryRatingSystems.lastAdjacentIndex);
  have gapLowerHalf : ℝ := sourceFirstLower / (4 * gLast);
  min gapLowerHalf (eps / (gLast * (lastLower⁻¹ + firstLower⁻¹)))
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndex

Declaration location: paper-local

Lean declaration body

```
type:
ℕ → ℕ

value:
fun (M : ℕ) => 2 * (M - 1) + 1
```

GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate

Declaration location: paper-local

Lean declaration body

```
type:
ℕ → ℕ → ℕ

value:
fun (M a : ℕ) => Nat.brecOn a (GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate._f M)
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndex

GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex

Declaration location: paper-local

Lean declaration body

```
type:
ℕ → ℕ → ℕ

value:
fun (C N : ℕ) => C * 2 ^ N + 1
```

GJ19OptimalBinaryRatingSystems.theoremB1DyadicSubsequenceUniformConvergence

Declaration location: paper-local

Lean declaration body

type:
(ℕ → ℝ → ℝ) → Prop

value:
```
fun (betaSeq : ℕ → ℝ → ℝ) =>
  ∀ (C : ℕ),
    ∃ (betaLimit : ℝ → ℝ),
      TendstoUniformlyOn
        (fun (N : ℕ) (θ : ℝ) => betaSeq (GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex C N) θ) betaLimit
        Filter.atTop (Set.Icc 0 1)
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex

GJ19OptimalBinaryRatingSystems.theoremB1UniformOptimalSubsequencePrincipleTo

Declaration location: paper-local

Lean declaration body

type:
(ℕ → ℝ → ℝ) → (ℕ → ℝ → ℝ) → (ℝ → ℝ) → Prop

value:
```
fun (betaSeq quantileSeq : ℕ → ℝ → ℝ) (quantileLimit : ℝ → ℝ) =>
  TendstoUniformlyOn quantileSeq quantileLimit Filter.atTop (Set.Icc 0 1) →
    GJ19OptimalBinaryRatingSystems.theoremB1DyadicSubsequenceUniformConvergence betaSeq
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.theoremB1DyadicSubsequenceUniformConvergence

GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → Fin (2 * m + 1 + 2) → ℝ

value:
```
fun {m : ℕ} (oldLevels : Fin (m + 2) → ℝ) (a : Fin (2 * m + 1 + 2)) =>
  if hEven : ↑a % 2 = 0 then
    oldLevels ⟨↑a / 2, GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels._proof_4 a hEven⟩
  else
    if hFirst : ↑a = 1 then
      AppliedModelingLib.Probability.bernoulliFirstEndpointEqualSplit
        (oldLevels (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex))
    else
      if hLast : ↑a = 2 * m + 1 then
        AppliedModelingLib.Probability.bernoulliLastEndpointEqualSplit
          (oldLevels (GJ19OptimalBinaryRatingSystems.adjacentLowIndex GJ19OptimalBinaryRatingSystems.lastAdjacentIndex))
      else
        AppliedModelingLib.Probability.bernoulliInteriorEqualSplit
          (oldLevels ⟨↑a / 2, GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels._proof_5 a⟩)
          (oldLevels
            ⟨↑a / 2 + 1, GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels._proof_6 a hEven hFirst hLast⟩)
```

Direct retained dependencies

- AppliedModelingLib.Probability.bernoulliFirstEndpointEqualSplit
- AppliedModelingLib.Probability.bernoulliInteriorEqualSplit
- AppliedModelingLib.Probability.bernoulliLastEndpointEqualSplit
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.firstAdjacentIndex
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

GJ19OptimalBinaryRatingSystems.lemmaC5_uniform_doubled_endpoint_levels

Declaration location: paper-local

Lean declaration body

type:
{m : ℕ} → (Fin (m + 2) → ℝ) → Fin (2 * m + 1 + 2) → ℝ

value:
fun {m : ℕ} (oldLevels : Fin (m + 2) → ℝ) (a : Fin (2 * m + 1 + 2)) =>
  GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels a

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels

Paper-specific semantic prerequisites

GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_large_deviation_rate

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_large_deviation_rate
Source locator: cited publication, lines 321-360

Verbatim paper-source connection

Large deviations & discretization

Recall the question: What β yields the highest value
of Wk ?. We now refine objective Wk and constrain β
to form a non-degenerate, feasible optimization task.
Large deviation rate function. Wk is not one
objective: it has a different value per time k, and
no single β simultaneously optimizes Wk for all
3
We use θ1 θ2 (θ1 − θ2), (1 − θ1)(1 − θ2)(θ1 − θ2), and
1
(2 − θ1)2 (12 − θ2)2 (θ1 − θ2) as examples.

k.4 Considering asymptotic performance is also
insufficient: when β is strictly increasing in θ,
limk→∞ Pk (θ1 , θ2 |β) = 1 ∀θ1 , θ2 by the law of large
numbers. Thus, W , limk→∞ Wk = 1, and any such
β is asymptotically optimal.
For this reason, we consider maximization of the
rate at which Wk converges, i.e., how fast the estimated item ranking converges to the true item ranking. We use a large deviations approach (Dembo and Zeitouni, 2010) to quantify this convergence
rate. Formally, given sequence Yk ≤ limk→∞ Yk =
Y , the large deviations rate of convergence is
− limk→∞ k1 log(Y − Yk) = c. If c exists then Yk approaches Y exponentially fast: Y − Yk = e−kc+o(k) .
Then, we wish to choose β to maximize Wk 's large
deviations rate, r = − limk→∞ k1 log(W − Wk).
Discretizing β. Unfortunately, even this problem
is degenerate if we consider continuous β: for any β
that is not piecewise constant, the large deviations
rate of convergence is zero, i.e., convergence of Wk
to its limit is only polynomially fast, and characterizing the dependence of this convergence rate on β
is intractable. Thus, the rate of convergence for Wk
is not a satisfactory objective with continuous β.
We make progress by discretizing β; in particular,
we restrict attention to optimization over stepwise
increasing β functions.5 Among stepwise increasing
β, the large deviations rate of Wk to its limiting
value W can be shown to be nondegenerate, i.e. ∃c >
0 s.t. W − Wk = e−kc+o(k) . (See Lemma C.4 in the
Appendix for further discussion.)

Lean-expanded paper semantic target

type:
ℝ → (ℕ → ℝ) → ℝ → Prop

value:
fun (W : ℝ) (Wk : ℕ → ℝ) (rate : ℝ) => AppliedModelingLib.Probability.HasExponentialRate (fun (k : ℕ) => W - Wk k) rate

Direct retained dependencies

- AppliedModelingLib.Probability.HasExponentialRate

Recorded source-to-declaration screening **Verdict:** matches

Reason: Compared the source definition of the finite large-deviation limit with HasExponentialRate and logDecay: the actual predicate is Tendsto (fun k => -(k:ℝ)⁻¹ * log (W-Wk k)) atTop (nhds rate). It therefore gives the stated rate on source sequences approaching W from below whenever that finite logarithmic limit exists; the generalized sequence carrier and the irrelevant k=0 convention add no source restriction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_lexicographic_optimality

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_lexicographic_optimality
Source locator: cited publication, lines 459-476

Verbatim paper-source connection

Our optimization problem: Within the class of
stepwise increasing functions with M levels, find the
β that is optimal, i.e. is

For Kendall's τ and Spearman's ρ, the optimal intervals are simply equispaced in [0, 1], i.e. Si∗ =
[Mi , i+1
M), because the entire item quality distribution is equally important. For other objective weight
functions w, the difficulty of finding the optimal subsets S ∗ depends on the properties of w. Since S ∗ is
trivial for Kendall's τ and Spearman's ρ – and w
is just an analytic tool that formalizes a platform's
goals – we focus on finding the optimal levels t∗ .

(1) Asymptotically optimal. It yields the highest limiting value of Wk . AND
(2) Rate optimal. It yields the fastest large deviations rate r among asymptotically optimal β.
[U+0001 control character]
A remarkable result of our paper is a O M log2 M
[U+000F control character]
procedure to find an optimal β with M levels.

Lean-expanded paper semantic target

type:
{α : Type u_1} → (α → Prop) → (α → ℝ) → (α → ℝ) → α → Prop

value:
fun {α : Type u_1} (feasible : α → Prop) (primary secondary : α → ℝ) (x : α) =>
  AppliedModelingLib.Optimization.IsLexicographicMaximizerOn feasible primary secondary x

Direct retained dependencies

- AppliedModelingLib.Optimization.IsLexicographicMaximizerOn

Recorded source-to-declaration screening **Verdict:** matches

Reason: The expanded IsLexicographicMaximizerOn requires feasibility and, for every feasible competitor, either a strictly smaller primary value or an equal primary value with no larger secondary value. With limiting accuracy as primary and convergence rate as secondary, this is exactly the source's value-first, rate-second optimization definition.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_pairwise_accuracy_eq1

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_pairwise_accuracy_eq1
Source locator: cited publication, lines 238-256

Verbatim paper-source connection

Platform objective. The platform wishes to rank
the items accurately. Given β and θ1 > θ2 , define:
[U+0001 control character]
Pk (θ1 , θ2 |β) =μk xk (θ1) > xk (θ2)|θ1 , θ2
[U+0001 control character]
− μk xk (θ1) < xk (θ2)|θ1 , θ2
(1)

Items. The system consists of a set [0, 1] of items,
where each item is associated with a unique (but unknown) quality θ ∈ [0, 1]; i.e., the system consists of
a continuum of a unit mass of items whose unknown
qualities are uniform2 in [0, 1]. Below, we discretize
the continuous quality space [0, 1] into M types, to
calculate a stepwise increasing β. We will make clear
why we introduce a continuum but then discretize.

This expression captures observed score ranking's
accuracy. When θ1 > θ2 but xk (θ1) < xk (θ2), the
ranking mistakenly orders θ1 below θ2 . A good system has large Pk (θ1 , θ2 |β). Integrating across items

Lean-expanded paper semantic target

type:
ℝ → ℝ → ℝ

value:
fun (probabilityCorrect probabilityIncorrect : ℝ) => probabilityCorrect - probabilityIncorrect

Recorded source-to-declaration screening **Verdict:** matches

Reason: The actual function subtracts the probability of incorrect strict ordering from that of correct strict ordering. This is Equation (1) with its two probabilities exposed as arguments; score ties enter neither probability and contribute zero.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_step_rule_partition_levels

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_step_rule_partition_levels
Source locator: cited publication, lines 354-368

Verbatim paper-source connection

We make progress by discretizing β; in particular,
we restrict attention to optimization over stepwise
increasing β functions.5 Among stepwise increasing
β, the large deviations rate of Wk to its limiting
value W can be shown to be nondegenerate, i.e. ∃c >
0 s.t. W − Wk = e−kc+o(k) . (See Lemma C.4 in the
Appendix for further discussion.)
Notationally, we will calculate an optimal stepwise
increasing β with M levels, i.e. there are M intervals
Si ⊂ [0, 1] and levels ti such that when θ1 , θ2 ∈ Si ,
then β(θ1) = β(θ2) , ti . The challenge is calculating an optimal S ∗ = {Si } and t∗ = {ti }.
The physical interpretation is that we group the
items into M subsets (types) Si ⊂ [0, 1]. When
items θ1 , θ2 are in the same subset, then their asymptotic reputation scores are the same, limk xk (θ1) =
limk xk (θ2) , ti . These items cannot be distinguished from one another even asymptotically.

[Next verbatim source excerpt]

Algorithm to find the optimal levels We now
describe how to find t∗ , the maximizer of r(t).
The following lemma describes a system of equations
to find the t∗ that maximizes r. It states that t∗
equalizes the rates at which each interval i is separated from its neighbors. The proof involves manipulation of r and convexity, and is in the appendix.
Lemma 3.1. The unique solution t∗ to the following
system of equations maximizes r(t):
r(t0 , t1) = r(t1 , t2) = · · · = r(tM −2 , tM −1)

(4)

t0 = 0, tM −1 = 1

[Next verbatim source excerpt]

log(tM −2)−gM −2
and t∗ maximizes rw (t) ⇐⇒ all the terms inside the minimization rw (t∗) are equal. Further, the
optimal levels t∗ are unique. The result immediately follows, that {ti } is the unique solution that
equalizes the rates inside the minimization, by noting that the optimal r has t0 = 0, tM −1 = 1.
We first prove the alternative form for r. Note that {gi−1 KL(a||ti−1) + gi KL(a||ti)} is convex

Settled source reading The endpoint-normalized vector represents the optimal designs of Lemma 3.1; the paper also permits general step rules.

Lean-expanded paper semantic target

type:
(m : ℕ) → (Fin (m + 2 + 1) → ℝ) → (Fin (m + 2) → ℝ) → Prop

value:
fun (m : ℕ) (cutpoints : Fin (m + 2 + 1) → ℝ) (levels : Fin (m + 2) → ℝ) =>
  cutpoints ∈ GJ19OptimalBinaryRatingSystems.sourceStrictCutpointSet (m + 2) ∧
    GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.sourceStrictCutpointSet

Recorded source-to-declaration screening **Verdict:** matches
Reason: Under the displayed approved optimal-design-domain reading, the target represents the Lemma 3.1 step-rule designs with strict interval cutpoints from zero to one and strictly increasing level values whose first and last entries are zero and one; it does not claim this endpoint normalization for every source-admissible step rule or assert optimality itself.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_weighted_objective_eq2

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_definition_weighted_objective_eq2
Source locator: cited publication, lines 257-288

Verbatim paper-source connection

creates the following objective for each time k:
Z
Wk =
w(θ1 , θ2)Pk (θ1 , θ2 |β)dθ1 dθ2
(2)

Matching with buyers. Items accumulate ratings
over time by matching with buyers. We assume the

Weight function w(θ1 , θ2) > 0 indicates how much
the platform cares about not mistaking a quality θ1
item with Ra quality θ2 item. We consider scaled w
such that θ1 >θ2 w(θ1 , θ2)dθ1 dθ2 = 1.

1
Note that a rating is not the same as a reward; buyers
often give positive ratings after bad experiences.
2
Any distribution can be handled by considering θ to
be the item's quantile rather than its absolute quality.

θ1 >θ2

Our first question then becomes: What β yields the
highest value of Wk ? As discussed above, the plat-

[form-feed in source extraction]
Designing Optimal Binary Rating Systems

form influences β through the design of its rating
system. The optimal choice of β sets the benchmark.
Discussion. Objective function. The specification
(2) of the objective is quite rich. It contains scaled

Lean-expanded paper semantic target

type:
(ℝ → ℝ → ℝ) → (ℝ → ℝ → ℝ) → ℝ → Prop

value:
```
fun (weight pairwiseAccuracy : ℝ → ℝ → ℝ) (Wk : ℝ) =>
  (∀ θ1 ∈ Set.Icc 0 1, ∀ θ2 ∈ Set.Ico 0 θ1, 0 < weight θ1 θ2) ∧
    ∫ (θ1 : ℝ) in Set.Icc 0 1, ∫ (θ2 : ℝ) in Set.Ico 0 θ1, weight θ1 θ2 = 1 ∧
      Wk = ∫ (θ1 : ℝ) in Set.Icc 0 1, ∫ (θ2 : ℝ) in Set.Ico 0 θ1, weight θ1 θ2 * pairwiseAccuracy θ1 θ2
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The target requires pointwise positive weights on 0≤θ2<θ1≤1, unit weight integral on that triangle, and Wk equal to the integral of weight times pairwise accuracy. These are the domain, normalization and formula in Equation (2); the nested integral is the same ordered-quality integral.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

GJ19OptimalBinaryRatingSystems.ProofBridge.source_theoremB1_quantile_representation

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.ProofBridge.source_theoremB1_quantile_representation
Source locator: cited publication, lines 1951-1979

Verbatim paper-source connection

Limit of β as M → ∞

w denote the optimal β with M intervals for weight function w, with intervals {S wM } =
Let βM
i
wM
wM . Let q
wM , swM), i.e. the quantile of
{[si , swM
)}
and
levels
t
(θ)
=
i/M
when
θ
∈
[s
wM
i+1
i
i+1
interval item of type θ is in.
Then, we have the following convergence result for βM .

Theorem B.1. Let g be uniform. Suppose w such that qwM converges uniformly. Then, ∀C ∈

Lean-expanded paper semantic target

type:
(M : ℕ) → (ℝ → Fin M) → (Fin M → ℝ) → (ℝ → ℝ) → (ℝ → ℝ) → Prop

value:
fun (M : ℕ) (cellIndex : ℝ → Fin M) (levels : Fin M → ℝ) (quantile beta : ℝ → ℝ) =>
  (∀ (θ : ℝ), quantile θ = ↑↑(cellIndex θ) / ↑M) ∧ ∀ (θ : ℝ), beta θ = levels (cellIndex θ)

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source assigns q_M(θ)=i/M and β_M(θ)=t_i on the selected cell. The actual predicate makes that cell selector explicit as cellIndex:ℝ→Fin M and asserts both equalities for the same selector. Every source partition has this representation; arbitrary selectors are a broader vocabulary and do not remove any source instance.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceKnownTypeExperiment

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceKnownTypeExperiment
Source locator: cited publication, lines 2005-2027

Verbatim paper-source connection

Learning ψ(θ, y) through experiments

Now, we show how a platform would run an experiment to decide to learn ψ(θ, y). In particular, one potential issue is that the platform does not have any items with know quality that it can use as representative items in its optimization. In this case, we show that it can use ratings within the experiment itself to identify these representative items. The results essentially follow from the law of large numbers.
We assume that |Θ| = L representative items i ∈ {1 . . . L} are in the experiment, and each are matched N times. The experiment proceeds as follows: every time an item is matched, show the buyer a random question from Y. For each word y ∈ Y, track the empirical $\hat{\psi}(i, y)$, the proportion of times a positive response was given to question y. Alternatively, if Y is totally ordered (i.e. a positive rating for a given y also implies positive ratings would be given to all "easier" y 0), and can be phrased as a multiple choice question, data collection can be faster: e.g., as we do in our experiments: Y consists of a set of totally ordered adjectives that can describe the item; the rater is asked to pick an adjective out of the set; this is interpreted as the item receiving a positive answer to the questions induced by the chosen answer and all worse adjectives, and a negative answer to all better adjectives.
First, suppose the platform approximately knows the quality θi of each item i, and θi are evenly distributed in [0, 1]. Suppose the items are ordered by index, i.e. θ1 < θ2 < · · · < θL . Then let $\hat{\psi}(\theta, y) = \hat{\psi}(i, y)$ when θ ∈ [θi−1 , θi]. Call this procedure KnownTypeExperiment.
Lemma B.2. Suppose ψ(θ, y) is Lipschitz continuous in θ. With KnownTypeExperiment, $\hat{\psi}(i, y) \to \psi(\theta i , y) \forall y$ uniformly as N → ∞. As L → ∞, $\hat{\psi}(\theta, y) \to \psi(\theta, y) \forall \theta$ uniformly.

Lean-elaborated paper signature

```
field quality:
{Representative : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Representative] →
      [inst_1 : Fintype Y] →
        GJ19OptimalBinaryRatingSystems.SourceKnownTypeExperiment Representative Y → Representative → ℝ

field empiricalResponse:
{Representative : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Representative] →
      [inst_1 : Fintype Y] →
        GJ19OptimalBinaryRatingSystems.SourceKnownTypeExperiment Representative Y → ℕ → Representative → Y → ℝ
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The displayed structure stores a finite family of known representative qualities and empirical question-response frequencies indexed by sample count. These are the data of KnownTypeExperiment in the pinned excerpt. It is a data carrier, with no claim that arbitrary field values satisfy the separate sampling or convergence assertions, and includes every source experiment.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceQualityDomain

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceQualityDomain
Source locator: cited publication, lines 187-204

Verbatim paper-source connection

3

Model and optimization

We now formalize our model and show how to optimize the rating function to maximize the learning rate. We focus on finding an optimal β : [0, 1] 7→ [0, 1], a map from item quality θ to the probability it should receive a positive rating. This section requires no data: we characterize the optimal system.
3.1

Model and problem specification

Our model is constructed to emphasize the rating system's learning rate. Time is discrete (k = 0, 1, 2, . . .). Informally: there is a set of items. Each time step, buyers match with the items and leave a rating according to β(θ). The platform records the ratings and ranks the items. Formally:

Lean-expanded paper semantic target

type:
ℝ → Prop

value:
fun (a : ℝ) => Set.Icc 0 1 a

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source normalizes item quality to the continuum [0,1]; the Lean domain is exactly Set.Icc 0 1 with no added restriction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceMatchingFunction

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceMatchingFunction
Source locator: cited publication, lines 199-218

Verbatim paper-source connection

Our model is constructed to emphasize the rating system's learning rate. Time is discrete (k = 0, 1, 2, . . .). Informally: there is a set of items. Each time step, buyers match with the items and leave a rating according to β(θ). The platform records the ratings and ranks the items. Formally:

existence of a nondecreasing match function g(θ), where item θ receives nk (θ) = bkg(θ)c matches, and thus ratings, up to time k. In other words, item 1 time steps.
θ is matched approximately every g(θ)
g(θ) ≤ 1 and bounded away from 0, i.e. ∃c > 0: g(θ) > c. This accumulation captures the feature that better items may be more likely to match.
Ratings. The key quantity for our subsequent analysis is the probability of a positive rating for each θ, β(θ) , Pr(positive rating|θ). Let y` (θ) ∼ Bernoulli(β(θ)) be the rating an item of quality θ receives at the `th time it matches.

Lean-expanded paper semantic target

type:
(ℝ → ℝ) → Prop

value:
```
fun (g : ℝ → ℝ) =>
  MonotoneOn g GJ19OptimalBinaryRatingSystems.sourceQualityDomain ∧
    (∀ θ ∈ GJ19OptimalBinaryRatingSystems.sourceQualityDomain, g θ ≤ 1) ∧
      ∃ (c : ℝ), 0 < c ∧ ∀ θ ∈ GJ19OptimalBinaryRatingSystems.sourceQualityDomain, c < g θ
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.sourceQualityDomain

Recorded source-to-declaration screening **Verdict:** matches

Reason: The byte-pinned Section 2 source requires a nondecreasing matching function bounded by one and strictly away from zero on the quality domain; the full Lean predicate retains each clause.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceQuestionDistribution

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceQuestionDistribution
Source locator: cited publication, lines 642-668

Verbatim paper-source connection

Resources available to platform. We suppose
the platform has a set Y of possible binary questions that it could ask a buyer, e.g., “Are you satisfied with your experience” or “Is this experience
your best on our platform?”. Informally, at each
rating opportunity (i.e., match made), the rater can
be shown a single question y ∈ Y. Let ψ(θ, y) be the
probability an item quality θ would receive a positive
rating when the rater is asked question y ∈ Y.
We further suppose the platform has a set Θ of representative items for which it has access to item quality. Θ, then, is the granularity at which the platform
can collect data about historical performance, or
otherwise get expert labels. (We assume M [U+001D control character] |Θ|).
Using known set Θ, the platform can run an experiment to create an estimate $\hat{\psi}(\theta, y)$, ∀y ∈ Y, θ ∈ Θ.
Design heuristic. How can the platform build an
effective rating system using these primitives and
β? We consider the following heuristic design: the
platform randomly shows a question y ∈ Y to each
buyer. The choice of the platform is a distribution
H(y) over y ∈ Y; in other words, for the platform
the design of the system amounts to choosing the frequency with which each question is shown. At each
rating opportunity, y is chosen from Y according to
H, independently across opportunities.
Clearly, the probability that
P an item θ receives a
positive rating is: $\tilde{\beta}(\theta)$, y∈Y ψ(θ, y)H(y).
We want a distribution H such that $\tilde{\beta}(\theta) = \beta(\theta)$ for
all θ ∈ [0, 1], i.e., that the positive rating probability
for each item is exactly the optimal value. However,
there may not exist any set of questions Y with associated ψ and choice of H such that $\tilde{\beta} = \beta$.7

Lean-expanded paper semantic target

type:
{Y : Type u_1} → [Fintype Y] → (Y → ℝ) → Prop

value:
fun {Y : Type u_1} [Fintype Y] (H : Y → ℝ) => (∀ (y : Y), 0 ≤ H y) ∧ ∑ y : Y, H y = 1

Recorded source-to-declaration screening **Verdict:** matches

Reason: The target is exactly nonnegativity of every finite question weight and a sum of one. This is the probability distribution H in the response-mixture source passage. Its displayed minimization use employs the same feasible set; no full-support restriction has been imposed on the Section 4 design problem.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceQuestionDesignSolution

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceQuestionDesignSolution
Source locator: cited publication, lines 669-696

Verbatim paper-source connection

We propose the following heuristic to address this
difficulty. Choose a probability distribution H to
minimize the following L1 distance:
X
X
min
|β(θ) −
ψ̂(θ, y)H(y)|
(5)
H:kHk1 =1

θ∈Θ

y∈Y

This heuristic uses the data available to the platform, ψ̂(θ, y) for a set of items θ ∈ Θ, and designs H
so at least these items receive ratings close to their
optimal ratings β(θi). Then, as long as ψ is wellbehaved, and Θ is representative of the full set, one
can hope that β̃(θ) u β(θ), for all θ ∈ [0, 1].
Discussion. Real-world analogue & constraints. A
special case of this system is already in place on
7

There are special cases where an exact solution exists. For example, let Y = [0, 1], and ψ(θ, y) = I[θ ≥ y].

many platforms, where the same question is always
shown. Static systems can be designed by restricting
H to only have mass at one question y. More generally, constraints can be used in optimization (5).

Lean-expanded paper semantic target

```
type:
{Representative : Type u_1} →
  {Y : Type u_2} →
    [Fintype Representative] → [Fintype Y] → (Representative → ℝ) → (ℝ → ℝ) → (Representative → Y → ℝ) → (Y → ℝ) → Prop

value:
fun {Representative : Type u_1} {Y : Type u_2} [Fintype Representative] [Fintype Y] (quality : Representative → ℝ)
    (β : ℝ → ℝ) (ψHat : Representative → Y → ℝ) (H : Y → ℝ) =>
  AppliedModelingLib.Optimization.IsMinimizerOn GJ19OptimalBinaryRatingSystems.SourceQuestionDistribution
    (GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective quality β ψHat) H
```

Direct retained dependencies

- AppliedModelingLib.Optimization.IsMinimizerOn
- GJ19OptimalBinaryRatingSystems.SourceQuestionDistribution
- GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective

Recorded source-to-declaration screening **Verdict:** matches

Reason: Expanding IsMinimizerOn, SourceQuestionDistribution and sourceQuestionDesignL1Objective gives a nonnegative finite probability vector H of total mass one that globally minimizes the sum over representatives of |β(quality)-Σ_y ψHat·H_y|. This is the Section 4 optimization problem (5), with the representative index and empirical response table explicit.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
Source locator: cited publication, lines 2005-2027

Verbatim paper-source connection

Learning ψ(θ, y) through experiments

Now, we show how a platform would run an experiment to decide to learn ψ(θ, y). In particular, one potential issue is that the platform does not have any items with know quality that it can use as representative items in its optimization. In this case, we show that it can use ratings within the experiment itself to identify these representative items. The results essentially follow from the law of large numbers.
We assume that |Θ| = L representative items i ∈ {1 . . . L} are in the experiment, and each are matched N times. The experiment proceeds as follows: every time an item is matched, show the buyer a random question from Y. For each word y ∈ Y, track the empirical $\hat{\psi}(i, y)$, the proportion of times a positive response was given to question y. Alternatively, if Y is totally ordered (i.e. a positive rating for a given y also implies positive ratings would be given to all "easier" y 0), and can be phrased as a multiple choice question, data collection can be faster: e.g., as we do in our experiments: Y consists of a set of totally ordered adjectives that can describe the item; the rater is asked to pick an adjective out of the set; this is interpreted as the item receiving a positive answer to the questions induced by the chosen answer and all worse adjectives, and a negative answer to all better adjectives.
First, suppose the platform approximately knows the quality θi of each item i, and θi are evenly distributed in [0, 1]. Suppose the items are ordered by index, i.e. θ1 < θ2 < · · · < θL . Then let $\hat{\psi}(\theta, y) = \hat{\psi}(i, y)$ when θ ∈ [θi−1 , θi]. Call this procedure KnownTypeExperiment.
Lemma B.2. Suppose ψ(θ, y) is Lipschitz continuous in θ. With KnownTypeExperiment, $\hat{\psi}(i, y) \to \psi(\theta i , y) \forall y$ uniformly as N → ∞. As L → ∞, $\hat{\psi}(\theta, y) \to \psi(\theta, y) \forall \theta$ uniformly.

Lean-elaborated paper signature

```
field question:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → ℕ → Ω → Y

field positiveResponse:
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → ℕ → Ω → Bool
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The carrier records one selected question and one binary response for each item, match index and outcome. The displayed indicators and counts use those same fields. The four displayed B.2/B.3 uses add positive question mass, the appropriate repeated-sampling independence and identical-distribution conditions, and expectations H_y and H_yψ; hence the general raw-data carrier specializes to the source random-question experiment without changing its observations.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceSystemState

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceSystemState
Source locator: cited publication, lines 228-237

Verbatim paper-source connection

System state. The state of the system is given by a
joint distribution μk (Θ, X), which gives the mass of
items of quality θ ∈ Θ ⊂ [0, 1] with aggregate score
xk (θ) ∈ X ⊂ [0, 1] at time k. Because our model is
a continuum, the evolution of the system state μk
follows a deterministic dynamical system.
We have described these dynamics at the level of individual items; however, such statements should be
interpreted as describing the evolution of the joint
distribution μk . The state update for μk is determined by the mass of items that match and the distributions of their ratings. A formal description of
the state evolution is in Appendix Section B.1.

Lean-expanded paper semantic target

type:
Type

value:
MeasureTheory.Measure (ℝ × ℝ)

Recorded source-to-declaration screening **Verdict:** matches

Reason: The actual carrier is a measure on quality-score pairs, and the displayed mass and update operations evaluate or transport that same measure. A source joint probability distribution supported on $[0,1]^2$ is an instance of this broader carrier, with exactly the source rectangle masses; the carrier itself adds no assertion about arbitrary measures being source probability states.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment
Source locator: cited publication, lines 2032-2044

Verbatim paper-source connection

Suppose instead the platform has many items L of unknown quality who are expected to match N
times each over the experiment time period. For each item, the platform would again ask questions
from Y, drawn according to any distribution (with positive mass on each question). Then generate
$\hat{\psi}(\theta, y)$ as follows: first, rank the items according to their ratings during the experiment itself.
Then, for each y, $\hat{\psi}(\theta, y)$ is
the empirical
performance of the θth percentile item in the ranking, i.e.
[U+0002 control character] i−1
[U+0003 control character]
i
$\hat{\psi}(\theta, y) = \hat{\psi}(\theta i , y)$ for $\theta \in L , L$. Call this procedure UnknownTypeExperiment.
Lemma B.3. Suppose $\psi(\theta, y)$ is Lipschitz continuous in θ. With UnknownTypeExperiment, $\hat{\psi}(\theta, y) \to$
$\psi(\theta, y) \forall y, \theta$ uniformly as $L, N \to \infty$.

Lean-elaborated paper signature

```
field trueQuality:
{Item : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Item] →
      [inst_1 : Fintype Y] → GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment Item Y → Item → ℝ

field empiricalResponse:
{Item : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Item] →
      [inst_1 : Fintype Y] → GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment Item Y → ℕ → Item → Y → ℝ

field empiricalAverageScore:
{Item : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Item] →
      [inst_1 : Fintype Y] → GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment Item Y → ℕ → Item → ℝ

field rankedItem:
{Item : Type u_1} →
  {Y : Type u_2} →
    [inst : Fintype Item] →
      [inst_1 : Fintype Y] → GJ19OptimalBinaryRatingSystems.SourceUnknownTypeExperiment Item Y → ℕ → Item → Item
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The expanded record exposes the finite unknown-quality items, sample-indexed empirical question responses, aggregate scores and percentile-to-item rank map described by UnknownTypeExperiment. These are storage fields rather than a proposition asserting correct ranking for arbitrary supplied data. Every source data record is representable, with no extra admissibility premise.

Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate
Source locator: cited publication, lines 2670-2714

Verbatim paper-source connection

Additional necessary lemmas

Now, we begin the set-up that will lead to a proof for Theorem 3.2. It turns out that proving the
theorem requires, in the process, essentially proving our convergence result with M → ∞, Theorem B.1. For Theorem 3.2, we need a lower bound for t1 as a
↪ function of M . This seems hard
to do in general. Luckily, in our case, there is a property for how t∗ changes when M is doubled.
Using this property, we can derive that t∗1 ≥ O(M −3).
−1
Recall that step-wise increasing β with M intervals Si = [si , si+1) has levels {ti }M
i=0 , where
t0 = 1, tM −1 = 1, and s0 , 0, sM , 1.
Furthermore, we use the following notation for the large deviation rate
#
"
g
g

ri = −(gi−1 + gi) log (1 − ti−1)

gi−1
gi−1 +gi

(1 − ti)

gi
gi−1 +gi

i−1
gi−1 +gi

+ ti−1

g

i
+gi

ti i−1

(23)

for i ∈ {1 . . . M − 1}, which implies r1 = −g1 log(1 − t1) and rM −1 = −gM −2 log(tM −2).
We further use rM −1 to be the rate achieved by the optimal βM with M intervals.
−1
Lemma C.5. Suppose g uniform, i.e. gi = 1, ∀i and that βM has values√{ti }M

Lean-expanded paper semantic target

```
type:
{m : ℕ} → (Fin (m + 2) → ℝ) → (Fin (m + 2) → ℝ) → Fin (m + 1) → ℝ

value:
fun {m : ℕ} (successProb sampleRate : Fin (m + 2) → ℝ) (i : Fin (m + 1)) =>
  if ↑i = 0 then
    sampleRate (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i) *
      -Real.log (1 - successProb (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i))
  else
    if ↑i = m then
      sampleRate (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i) *
        -Real.log (successProb (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i))
    else
      AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate
        (sampleRate (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i))
        (sampleRate (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i))
        (successProb (GJ19OptimalBinaryRatingSystems.adjacentHighIndex i))
        (successProb (GJ19OptimalBinaryRatingSystems.adjacentLowIndex i))
```

Direct retained dependencies

- AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex

Recorded source-to-declaration screening **Verdict:** matches

Reason: The actual middle branch expands to -(g_lo+g_hi) log[(1-t_lo)^(g_lo/(g_lo+g_hi))(1-t_hi)^(g_hi/(g_lo+g_hi))+t_lo^(g_lo/(g_lo+g_hi))t_hi^(g_hi/(g_lo+g_hi))], exactly Equation (23), visually checked in the archive supplement p.17. The first and last branches are the two endpoint formulas explicitly printed after that equation. The displayed uses

supply the ordered 0-to-1 endpoint chain and positive sample rates for the quantitative claims; the only two-level use is an initial member of an asymptotic optimizer sequence. This verdict concerns the rate formulas, not equivalence of the broader step-rule feasible-set definition.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

GJ19OptimalBinaryRatingSystems.kendallConstantWeightOrderedPairIntervalObjective

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.kendallConstantWeightOrderedPairIntervalObjective
Source locator: cited publication, lines 3595-3614

Verbatim paper-source connection

Kendall’s tau and Spearman’s rho related proofs

Definition C.1 (see e.g. Embrechts et al. (2003); Nelsen (2007)). The population version of
Kendall-tau correlation between item true quality and rating scores is proportional to
Z
τ
Pk (θ1 , θ2)dθ1 dθ2
Wk , 2
θ1 >θ2

Similarly, given items with qualities θ1 , θ2 , θ3 , the population version of Spearman’s rho correlation
between item true quality and rating scores is
Z
ρ
Pk (θ1 , θ3)dθ1 dθ2 dθ3
Wk , 6
θ1 >θ2 ,θ3

Lemma C.10. Spearman’s ρ can also be written as being proportional to

[Next verbatim source excerpt]

Definition C.1 (see e.g. Embrechts et al. (2003); Nelsen (2007)). The population version of
Kendall-tau correlation between item true quality and rating scores is proportional to
Z
τ
Pk (θ1 , θ2)dθ1 dθ2
Wk , 2
θ1 >θ2

Similarly, given items with qualities θ1 , θ2 , θ3 , the population version of Spearman’s rho correlation
between item true quality and rating scores is
Z
ρ
Pk (θ1 , θ3)dθ1 dθ2 dθ3
Wk , 6
θ1 >θ2 ,θ3

[Next verbatim source excerpt]

Lemma C.10. Spearman’s ρ can also be written as being proportional to
i.e. with w(θ1 , θ2) = (θ1 − θ2).

[Next verbatim source excerpt]

R

θ1 >θ2 (θ1 −θ2)Pk (θ1 , θ2)dθ1 dθ2 ,

Lean-expanded paper semantic target

type:
ℕ → (ℕ → ℝ) → ℝ

value:
fun (M : ℕ) (s : ℕ → ℝ) => ∑ i : Fin M, ∑ j : Fin M, if i < j then (s (↑i + 1) - s ↑i) * (s (↑j + 1) - s ↑j) else 0

Recorded source-to-declaration screening **Verdict:** matches

Reason: For an ordered partition, the target sums the areas |S_i||S_j| of all cross-cell rectangles i<j. This is the constant-weight asymptotic Kendall objective from Definition C.1, up to its fixed positive normalization factor 2. The displayed finite-design use and Corollary C.4 use it as an optimization objective, so that factor preserves all maximizing designs and lexicographic primary comparisons.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceMatchingCount

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceMatchingCount
Source locator: cited publication, lines 199-218

Verbatim paper-source connection

Our model is constructed to emphasize the rating system’s learning rate. Time is discrete (k =
0, 1, 2, . . .). Informally: there is a set of items. Each
time step, buyers match with the items and leave a
rating according to β(θ). The platform records the
ratings and ranks the items. Formally:

existence of a nondecreasing match function g(θ),
where item θ receives nk (θ) = bkg(θ)c matches, and
thus ratings, up to time k. In other words, item
1
time steps.
θ is matched approximately every g(θ)
g(θ) ≤ 1 and bounded away from 0, i.e. ∃c > 0:
g(θ) > c. This accumulation captures the feature
that better items may be more likely to match.
Ratings. The key quantity for our subsequent
analysis is the probability of a positive rating for
each θ, β(θ) , Pr(positive rating|θ). Let y` (θ) ∼
Bernoulli(β(θ)) be the rating an item of quality θ
receives at the `th time it matches.

Lean-expanded paper semantic target

type:
ℕ → (ℝ → ℝ) → ℝ → ℕ

value:
fun (k : ℕ) (g : ℝ → ℝ) (θ : ℝ) => ⌊↑k * g θ⌋₊

Recorded source-to-declaration screening **Verdict:** matches

Reason: The byte-pinned Section 2 source states n_k(theta) = floor(k g(theta)); the Lean definition is exactly Nat.floor of the real product.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceAppendixBActiveSet
Source locator: cited publication, lines 1820-1828

Verbatim paper-source connection

Formal specification of system state update

Recall that μk (Θ, X) is the mass of items with true quality θ ∈ Θ ⊆ [0, 1] and a reputation score
x ∈ X ⊆ [0, 1] at time k. Let Ek = {θ : nk (θ) = nk−1 (θ) + 1}. These are the items who receive an
additional rating at time k; for all θ ∈ Ekc , nk (θ) = nk−1 (θ). Our system is completely deterministic,
and evolves according to the distributions of the individual seller dynamics.
For each θ ∈ Ek , x, x0 , define ω(θ, x, x0) as follows:
ω(θ, x, x0) = β(θ)I{nk (θ)x − nk−1 (θ)x0 = 1} + (1 − β(θ))I{nk (θ)x − nk−1 (θ)x0 = 0}.

Lean-expanded paper semantic target

type:
ℕ → (ℝ → ℝ) → ℝ → Prop

value:
```
fun (k : ℕ) (g : ℝ → ℝ) (a : ℝ) =>
  {θ : ℝ |
     GJ19OptimalBinaryRatingSystems.sourceMatchingCount (k + 1) g θ =
       GJ19OptimalBinaryRatingSystems.sourceMatchingCount k g θ + 1}
    a
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.sourceMatchingCount

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source active set selects qualities whose match count increases by one. The target uses floor(k·g(θ)) for that count and selects n_(k+1)=n_k+1. This is the same definition with the transition indexed from k to k+1; the displayed update uses the same old count and active set consistently.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionKernel

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceAppendixBTransitionKernel
Source locator: cited publication, lines 1828-1831

Verbatim paper-source connection

ω(θ, x, x0) = β(θ)I{nk (θ)x − nk−1 (θ)x0 = 1} + (1 − β(θ))I{nk (θ)x − nk−1 (θ)x0 = 0}.
Then ω gives the probability of transition from x0 to x when an item receives a rating. We then
have:
Z Z

Lean-expanded paper semantic target

type:
```
ℕ → (ℝ → ℝ) → ℝ → ℝ → MeasureTheory.Measure ℝ
```

value:
```
fun (n : ℕ) (β : ℝ → ℝ) (θ oldScore : ℝ) =>
  ENNReal.ofReal (β θ) •
      MeasureTheory.Measure.dirac (GJ19OptimalBinaryRatingSystems.sourceScoreAfterRating n oldScore 1) +
    ENNReal.ofReal (1 - β θ) •
      MeasureTheory.Measure.dirac (GJ19OptimalBinaryRatingSystems.sourceScoreAfterRating n oldScore 0)
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.sourceScoreAfterRating

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source explicitly describes omega as the probability of transition when one rating arrives. For old count n and score x, solving its two indicator equations gives new scores (n·x+1)/(n+1) and n·x/(n+1), with probabilities β(θ) and 1-β(θ). The expanded kernel is exactly that two-atom measure, and the displayed update uses the old count consistently. On source probabilities in [0,1], ofReal preserves both masses; this is an equivalent measure representation, not a materially corrected target.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceBernoulliKL

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceBernoulliKL
Source locator: cited publication, lines 519-526

Verbatim paper-source connection

We further note that the choice of M is not conse6
b
1−b
quential; in the Appendix Section B.4 we show that
KL(a||b) = a log a + (1 − a) log 1−a is the Kullbackin an appropriate sense, a sequence of optimal βM
Leibler (KL) divergence between Bernoulli random varifor
each M converges as M gets large.
ables with success probabilities a and b respectively.

[Next verbatim source excerpt]

Remark C.2. The KL divergence for two Bernoulli random variables is continuous and strictly
convex, with minima at a = b, when a, b ∈ (0, 1). Note that inf a {gi KL(a||ti) + g(i + 1)KL(a||ti+1)},
for all feasible g, is also continuous and strictly convex in ti , ti+1 , with minima at ti = ti+1 .
One consequence of the above fact is that fixing either ti or ti+1 and moving the other farther
away monotonically increases KL, while moving it closer decreases KL.

[Next verbatim source excerpt]

Proof. Recall r(t) , − limk→∞ k1 log(W −Wk) = min0≤i≤M −2 inf a∈R {gi+1 KL(a||ti+1) + gi KL(a||ti)}.
We show the following:
r(t) =
min

log(1 − t1)−g1 ,
"
log (1 − ti−1)

gi−1
gi−1 +gi

(1 − ti)

gi
gi−1 +gi

gi−1
gi−1 +gi

+ ti−1

gi
gi−1 +gi

ti

#−gi−1 −gi
for 1 < i < M − 1,

!
log(tM −2)−gM −2
and t∗ maximizes rw (t) ⇐⇒ all the terms inside the minimization rw (t∗) are equal. Further, the
optimal levels t∗ are unique. The result immediately follows, that {ti } is the unique solution that
equalizes the rates inside the minimization, by noting that the optimal r has t0 = 0, tM −1 = 1.
We first prove the alternative form for r. Note that {gi−1 KL(a||ti−1) + gi KL(a||ti)} is convex
in a, and so we can find an analytic form for the infinum over a.

Settled source reading In the KL footnote, reverse both logarithm ratios to obtain the standard nonnegative Bernoulli divergence used by the proof.

Lean-expanded paper semantic target

type:
ℝ → ℝ → ℝ

value:
fun (a b : ℝ) => AppliedModelingLib.Probability.bernoulliKL a b

Direct retained dependencies

- AppliedModelingLib.Probability.bernoulliKL

Recorded source-to-declaration screening **Verdict:** matches

Reason: Under the displayed local formula clarification, the source's named Bernoulli KL and its stated minimum at equality, convexity, and nonnegative rate use select the standard expression a log(a/b) plus (1-a) log((1-a)/(1-b)); the expanded target is exactly that expression and its selected adjacent-rate use preserves the same argument order.
Reviewer check: □ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceEmpiricalReputationScore

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceEmpiricalReputationScore
Source locator: cited publication, lines 219-237

Verbatim paper-source connection

Aggregating ratings and ranking sellers.
These ratings are aggregated into a reputation score,
xk (θ), at each time k. The score is the fraction
of positive ratings received up to time k: xk (θ) ,
Pnk (θ)
1
`=0 y` (θ) with x0 (θ) , 0 for all θ. Thus,
nk (θ)
xk (θ) ∼ nk1(θ) Binomial(β(θ), nk (θ)).
System state. The state of the system is given by a
joint distribution μk (Θ, X), which gives the mass of
items of quality θ ∈ Θ ⊂ [0, 1] with aggregate score
xk (θ) ∈ X ⊂ [0, 1] at time k. Because our model is
a continuum, the evolution of the system state μk
follows a deterministic dynamical system.
We have described these dynamics at the level of individual items; however, such statements should be
interpreted as describing the evolution of the joint
distribution μk . The state update for μk is determined by the mass of items that match and the distributions of their ratings. A formal description of
the state evolution is in Appendix Section B.1.

Lean-expanded paper semantic target

type:
(ℕ → Bool) → ℕ → ℝ

value:
fun (rating : ℕ → Bool) (n : ℕ) => if n = 0 then 0 else (∑ ell ∈ Finset.range n, if rating ell = true then 1 else 0) / ↑n

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source defines the score as the fraction of positive ratings received, specifies a Binomial(β,n)/n law, and initializes the score to zero. The target counts exactly n binary observations, divides by n, and returns zero at n=0. This matches that sample-count meaning; its range-n indexing makes the number of observations explicit rather than treating the printed zero-based summation label as an additional rating.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
Source locator: cited publication, lines 2005-2027

Verbatim paper-source connection

Learning ψ(θ, y) through experiments

Now, we show how a platform would run an experiment to decide to learn ψ(θ, y). In particular, one potential issue is that the platform does not have any items with know quality that it can use as representative items in its optimization. In this case, we show that it can use ratings within the experiment itself to identify these representative items. The results essentially follow from the law of large numbers.
We assume that |Θ| = L representative items i ∈ {1 . . . L} are in the experiment, and each are matched N times. The experiment proceeds as follows: every time an item is matched, show the buyer a random question from Y. For each word y ∈ Y, track the empirical $\hat{\psi}(i, y)$, the proportion of times a positive response was given to question y. Alternatively, if Y is totally ordered (i.e. a positive rating for a given y also implies positive ratings would be given to all "easier" y 0), and can be phrased as a multiple choice question, data collection can be faster: e.g., as we do in our experiments: Y consists of a set of totally ordered adjectives that can describe the item; the rater is asked to pick an adjective out of the set; this is interpreted as the item receiving a positive answer to the questions induced by the chosen answer and all worse adjectives, and a negative answer to all better adjectives.
First, suppose the platform approximately knows the quality θi of each item i, and θi are evenly distributed in [0, 1]. Suppose the items are ordered by index, i.e. θ1 < θ2 < · · · < θL . Then let $\hat{\psi}(\theta, y) = \hat{\psi}(i, y)$ when θ ∈ [θi−1 , θi]. Call this procedure KnownTypeExperiment.
Lemma B.2. Suppose ψ(θ, y) is Lipschitz continuous in θ. With KnownTypeExperiment, $\hat{\psi}(i, y) \to \psi(\theta i , y) \forall y$ uniformly as N → ∞. As L → ∞, $\hat{\psi}(\theta, y) \to \psi(\theta, y) \forall \theta$ uniformly.

Lean-expanded paper semantic target

type:
```
{Ω : Type u_1} →
  {Item : Type u_2} →
    {Y : Type u_3} →
      [DecidableEq Y] → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y → Item → Y → ℕ → Ω → ℝ
```

value:
```
fun {Ω : Type u_1} {Item : Type u_2} {Y : Type u_3} [DecidableEq Y]
    (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Item Y) (i : Item) (y : Y) (N : ℕ) (ω : Ω) =>
  GJ19OptimalBinaryRatingSystems.sourceExperimentPositiveQuestionCount E i y N ω /
    GJ19OptimalBinaryRatingSystems.sourceExperimentQuestionCount E i y N ω
```

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceExperimentPositiveQuestionCount
- GJ19OptimalBinaryRatingSystems.sourceExperimentQuestionCount

Recorded source-to-declaration screening **Verdict:** matches

Reason: The expanded numerator counts only matches with question y and a positive response; the denominator counts all matches with question y. Their quotient is exactly the empirical fraction in the source experiment. Its four displayed B.2/B.3 uses assume H_y>0 and the corresponding sampling laws, so the total-division convention before any y question has appeared does not alter the source convergence statement.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.sourceSystemStateMass

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.sourceSystemStateMass
Source locator: cited publication, lines 228-237

Verbatim paper-source connection

System state. The state of the system is given by a
joint distribution μk (Θ, X), which gives the mass of
items of quality θ ∈ Θ ⊂ [0, 1] with aggregate score
xk (θ) ∈ X ⊂ [0, 1] at time k. Because our model is
a continuum, the evolution of the system state μk
follows a deterministic dynamical system.
We have described these dynamics at the level of individual items; however, such statements should be
interpreted as describing the evolution of the joint
distribution μk . The state update for μk is determined by the mass of items that match and the distributions of their ratings. A formal description of
the state evolution is in Appendix Section B.1.

Lean-expanded paper semantic target

type:
GJ19OptimalBinaryRatingSystems.SourceSystemState → Set ℝ → Set ℝ → ENNReal

value:
fun (state : GJ19OptimalBinaryRatingSystems.SourceSystemState) (Θ X : Set ℝ) => state (Θ ×ˢ X)

Direct retained dependencies

- GJ19OptimalBinaryRatingSystems.SourceSystemState

Recorded source-to-declaration screening **Verdict:** matches

Reason: The source notation μ_k(Θ,X) denotes the mass of items with quality in Θ and score in X. The actual function evaluates the same state measure on the product set Θ×X. Using an extended-nonnegative result type accommodates measure values and agrees with every source probability-state rectangle mass.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective
Source locator: cited publication, lines 3595-3614

Verbatim paper-source connection

Kendall’s tau and Spearman’s rho related proofs

Definition C.1 (see e.g. Embrechts et al. (2003); Nelsen (2007)). The population version of
Kendall-tau correlation between item true quality and rating scores is proportional to
Z
τ
Pk (θ1 , θ2)dθ1 dθ2
Wk , 2
θ1 >θ2

Similarly, given items with qualities θ1 , θ2 , θ3 , the population version of Spearman’s rho correlation
between item true quality and rating scores is
Z
ρ
Pk (θ1 , θ3)dθ1 dθ2 dθ3
Wk , 6
θ1 >θ2 ,θ3

Lemma C.10. Spearman’s ρ can also be written as being proportional to

[Next verbatim source excerpt]

Definition C.1 (see e.g. Embrechts et al. (2003); Nelsen (2007)). The population version of
Kendall-tau correlation between item true quality and rating scores is proportional to
Z
τ
Pk (θ1 , θ2)dθ1 dθ2
Wk , 2
θ1 >θ2

Similarly, given items with qualities θ1 , θ2 , θ3 , the population version of Spearman’s rho correlation
between item true quality and rating scores is
Z
ρ
Pk (θ1 , θ3)dθ1 dθ2 dθ3
Wk , 6
θ1 >θ2 ,θ3

[Next verbatim source excerpt]

Lemma C.10. Spearman’s ρ can also be written as being proportional to
i.e. with w(θ1 , θ2) = (θ1 − θ2).

[Next verbatim source excerpt]

R

θ1 >θ2 (θ1 −θ2)Pk (θ1 , θ2)dθ1 dθ2 ,

Lean-expanded paper semantic target

type:
ℕ → (ℕ → ℝ) → ℝ

value:
```
fun (M : ℕ) (s : ℕ → ℝ) =>
  ∑ i ∈ Finset.range M,
    ∑ j ∈ Finset.range M,
      if i < j then ((s (j + 1) + s j) / 2 - (s (i + 1) + s i) / 2) * (s (i + 1) - s i) * (s (j + 1) - s j) else 0
```

Recorded source-to-declaration screening **Verdict:** matches

Reason: The now-bound complete Lemma C.10 statement identifies the Spearman objective as proportional to the ordered-pair integral with weight θ1-θ2. The actual finite objective sums, over i<j, the midpoint difference of S_j and S_i times both interval lengths; this is exactly the integral of θ1-θ2 over each cross-cell rectangle. The displayed finite-design objective and Corollary C.4 use it for optimization, where the fixed positive correlation normalization preserves maximizers; the displayed C.10 and C.12 identities use the same unnormalized interval objective. The added source sentence closes the previously missing formula without changing any target or premise.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput

Paper-local declaration:
GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput
Source locator: cited publication, lines 550-587

Verbatim paper-source connection

ALGORITHM 1: Nested Bisection
Data: Number of intervals M , match function g
Result: β levels, i.e. {t0 . . . tM −1 }
Function main (M , δ, g)
while Uncertainty region for tM −2 is bigger
than error tolerance do
Calculate r(tM −2 , 1), the rate between
current guess for tM −2 , and tM −1 = 1.
Fixing tM −2 , find t1 . . . tM −3 such that
r(t1 , t2) = r(t2 , t3) = · · · =
r(tM −3 , tM −2) = r(tM −2 , 1), which can be
done through a sequence of bisection
routines.
Calculate r(0, t1), the rate between current
guess for t1 , and t0 = 0.
Compare r(tM −2 , 1) and r(0, t1), adjust
uncertainty region for tM −2 accordingly.
return {ti }

[U+0013 control character]gi−1 +gi [U+0015 control character]

+ti−1 gi−1 +gi ti gi−1 +gi
We do not know of any algorithm that efficiently
and provably solves such convex equality systems
in general. However, we leverage some structure in
our setting to develop an algorithm, NestedBisection, with run-time and optimality guarantees. The
efficiency of our algorithm results from the property
that, given a rate, ti is uniquely determined by the
value of either of the adjacent levels ti−1 , ti+1 , reducing an exponentially large search space to an almost
linear one. Physically, i.e., we only need to separate
each type of item from its neighboring types.
Below we include pseudo-code. Akin to branch and
bound, the algorithm proceeds via bisection on the
optimal value of tM −2 . For each candidate value
of tM −2 , the other values can be found using a sequence of bisection subroutines. These values approximately obey all the equalities in the system (4)
except the first. The direction of the first equality's
violation reveals how to change the interval for the
next outer bisection iteration.

Lean-expanded paper semantic target

type:
(m : ℕ) → ℕ → ℕ → (Fin (m + 2) → ℝ) → ℝ → Fin (m + 2) → ℝ

value:
```
fun (m outerSteps innerSteps : ℕ) (sampleRate : Fin (m + 2) → ℝ) (grid : ℝ) (a : Fin (m + 2)) =>
  (have gLast : ℝ :=
     sampleRate (GJ19OptimalBinaryRatingSystems.adjacentLowIndex GJ19OptimalBinaryRatingSystems.lastAdjacentIndex);
   have candidate : ℝ → Fin (m + 2) → ℝ := fun (lastLow : ℝ) =>
     GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionLevels m innerSteps sampleRate grid
       (gLast * -Real.log lastLow) lastLow;
   have above : ℝ → Bool := fun (lastLow : ℝ) =>
     GJ19OptimalBinaryRatingSystems.theorem32OuterSourceWeightedRateAbove (candidate lastLow) sampleRate;
   have lastLow : ℝ :=
     (AppliedModelingLib.Optimization.realBisectionRun above outerSteps (1 - 1 / ↑(m + 1)) (1 - grid)).2;
   candidate lastLow)
   a
```

Direct retained dependencies

- AppliedModelingLib.Optimization.realBisectionRun
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex
- GJ19OptimalBinaryRatingSystems.theorem32OuterSourceWeightedRateAbove
- GJ19OptimalBinaryRatingSystems.theorem32WeightedBackwardGridRateBisectionLevels

Recorded source-to-declaration screening **Verdict:** matches

Reason: The actual output searches the penultimate level, computes the last-pair target rate, recursively bisects backward to obtain the intermediate levels, compares the last and first endpoint rates, and updates the outer bracket before returning the candidate with endpoints 0 and 1. The displayed bisection bodies implement those branch tests directly and take no optimizer as input. The displayed Theorem 3.2 use supplies positive ordered sample rates and chooses finite search budgets from the error tolerance, matching Algorithm 1's nested approximate-equalization procedure.

Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Source claims

Review row 1

Source locator: cited publication, lines 501-518

Verbatim source input

t = arg max r(t), where gi , inf θ∈Si g(θ) and
In other words, given a discrete set of M items
1
(equiv, given S ∗), calculating the optimal t is equivr(t) , − lim log(W − Wk)
(3)
alent to solving a maximin problem for the rates at
k→∞ k
which
each type is distinguished from each of the
= min inf {gi+1 KL(a||ti+1) + gi KL(a||ti)}
0≤i≤M −2 a∈R
others. Thus, the algorithm below also solves the
inverse bandits problem in which we wish to rank
The proof is in the Appendix. The main hurdle is
the M arms, and we can choose the structure of the
showing that the continuum of rates for Pk (θ1 , θ2)
(binary) observations at each arm.
for each pair θ1 , θ2 translates into a rate for Wk .

Expanded PaperInterface specification

```
∀ {m : ℕ} (cutpoints : ℕ → ℝ) (g : ℝ → ℝ) (j : Fin (m + 1)),
  Monotone cutpoints →
    Monotone g →
      ∀ (tHi tLo : ℝ),
        GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
            (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j) =
          g (cutpoints ↑(GJ19OptimalBinaryRatingSystems.adjacentLowIndex j)) ∧
        GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
            (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j) =
          g (cutpoints ↑(GJ19OptimalBinaryRatingSystems.adjacentHighIndex j)) ∧
        GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryRatingRate
            (GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
              (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j))
            (GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
              (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j))
            tHi tLo =
          sInf
            (Set.range fun (a : ℝ) =>
              GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
                  (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j) *
                GJ19OptimalBinaryRatingSystems.paperBernoulliKL a tHi +
              GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
                  (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j) *
                GJ19OptimalBinaryRatingSystems.paperBernoulliKL a tLo)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryRatingRate
- GJ19OptimalBinaryRatingSystems.paperBernoulliKL
- GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ → ℝ
3. parameter: ℝ → ℝ
4. parameter: Fin (m + 1)
5. assumption: Monotone cutpoints
6. assumption: Monotone g
7. parameter: ℝ
8. parameter: ℝ
9. conclusion: GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j) =
   g (cutpoints ↑(GJ19OptimalBinaryRatingSystems.adjacentLowIndex j)) ∧
  GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
      (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j) =
    g (cutpoints ↑(GJ19OptimalBinaryRatingSystems.adjacentHighIndex j)) ∧
  GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryRatingRate
      (GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
        (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j))
      (GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
        (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j))

```
  tHi tLo =
sInf
  (Set.range fun (a : ℝ) =>
    GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
          (GJ19OptimalBinaryRatingSystems.adjacentHighIndex j) *
        GJ19OptimalBinaryRatingSystems.paperBernoulliKL a tHi +
      GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cutpoints g
          (GJ19OptimalBinaryRatingSystems.adjacentLowIndex j) *
        GJ19OptimalBinaryRatingSystems.paperBernoulliKL a tLo)
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_theorem31_adjacent_rate_eq3

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 2

Source locator: cited publication, lines 501-518

Verbatim source input

t = arg max r(t), where gi , inf θ∈Si g(θ) and
In other words, given a discrete set of M items
1
(equiv, given S ∗), calculating the optimal t is equivr(t) , − lim log(W − Wk)
(3)
alent to solving a maximin problem for the rates at
k→∞ k
which
each type is distinguished from each of the
= min inf {gi+1 KL(a||ti+1) + gi KL(a||ti)}
0≤i≤M −2 a∈R
others. Thus, the algorithm below also solves the
inverse bandits problem in which we wish to rank
The proof is in the Appendix. The main hurdle is
the M arms, and we can choose the structure of the
showing that the continuum of rates for Pk (θ1 , θ2)
(binary) observations at each arm.
for each pair θ1 , θ2 translates into a rate for Wk .

Expanded PaperInterface specification

```
∀ {m : ℕ} (cut : ℕ → ℝ) (g : ℝ → ℝ) (i : Fin (m + 2)),
  cut ↑i ≤ cut (↑i + 1) →
    MonotoneOn g (Set.Icc (cut ↑i) (cut (↑i + 1))) →
      GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cut g i = g (cut ↑i)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ → ℝ
3. parameter: ℝ → ℝ
4. parameter: Fin (m + 2)
5. assumption: cut ↑i ≤ cut (↑i + 1)
6. assumption: MonotoneOn g (Set.Icc (cut ↑i) (cut (↑i + 1)))
7. conclusion: GJ19OptimalBinaryRatingSystems.sourceCellMatchingRate cut g i = g (cut ↑i)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.theorem31_source_cell_matching_rate_eq_lower_cutpoint

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 3

Source locator: cited publication, lines 530-549

Verbatim source input

Algorithm to find the optimal levels We now
describe how to find t∗ , the maximizer of r(t).
The following lemma describes a system of equations
to find the t∗ that maximizes r. It states that t∗
equalizes the rates at which each interval i is separated from its neighbors. The proof involves manipulation of r and convexity, and is in the appendix.
Lemma 3.1. The unique solution t∗ to the following
system of equations maximizes r(t):
r(t0 , t1) = r(t1 , t2) = · · · = r(tM −2 , tM −1)

(4)

t0 = 0, tM −1 = 1
[U+0014 control character][U+0012 control character]
gi−1
gi
r(ti−1 , ti) , − log (1 − ti−1) gi−1 +gi (1 − ti) gi−1 +gi
gi−1

gi

Expanded PaperInterface specification

```
∀ {m : ℕ},
  1 < m →
    ∀ (sampleRate : Fin (m + 2) → ℝ),
      (∀ k < m + 2, 0 < GJ19OptimalBinaryRatingSystems.binaryEndpointSampleRateNat sampleRate k) →
        ∃! levels : Fin (m + 2) → ℝ,
          GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels ∧
            GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate ∧
              AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
                (fun (xs : Fin (m + 2) → ℝ) =>
                  GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs sampleRate)
                levels
```

Retained governing declarations

- AppliedModelingLib.Optimization.IsMaximizerOn
- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
- GJ19OptimalBinaryRatingSystems.binaryEndpointSampleRateNat

Lean-elaborated claim roles

```
1. parameter: ℕ
2. assumption: 1 < m
3. parameter: Fin (m + 2) → ℝ
4. assumption: ∀ k < m + 2, 0 < GJ19OptimalBinaryRatingSystems.binaryEndpointSampleRateNat sampleRate k
5. conclusion: ∃ (x : Fin (m + 2) → ℝ),
  (fun (levels : Fin (m + 2) → ℝ) =>
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels ∧
        GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate ∧
          AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
            (fun (xs : Fin (m + 2) → ℝ) =>
              GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs sampleRate)
            levels)
    x ∧
    ∀ (y : Fin (m + 2) → ℝ),
      (fun (levels : Fin (m + 2) → ℝ) =>
          GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels ∧
            GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate ∧
              AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
                (fun (xs : Fin (m + 2) → ℝ) =>
                  GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs sampleRate)
                levels)
        y →
        y = x
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemma31_equalization_eq4

Recorded source-to-Spec screening **Verdict:** matches
Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 4

Source locator: cited publication, lines 542-571

Verbatim source input

[U+0014 control character][U+0012 control character]
gi−1
gi
r(ti−1 , ti) , − log (1 − ti−1) gi−1 +gi (1 − ti) gi−1 +gi
gi−1

gi

ALGORITHM 1: Nested Bisection
Data: Number of intervals M , match function g
Result: β levels, i.e. {t0 . . . tM −1 }
Function main (M , δ, g)
while Uncertainty region for tM −2 is bigger
than error tolerance do
Calculate r(tM −2 , 1), the rate between
current guess for tM −2 , and tM −1 = 1.
Fixing tM −2 , find t1 . . . tM −3 such that
r(t1 , t2) = r(t2 , t3) = · · · =
r(tM −3 , tM −2) = r(tM −2 , 1), which can be
done through a sequence of bisection
routines.
Calculate r(0, t1), the rate between current
guess for t1 , and t0 = 0.
Compare r(tM −2 , 1) and r(0, t1), adjust
uncertainty region for tM −2 accordingly.
return {ti }

[U+0013 control character]gi−1 +gi [U+0015 control character]

+ti−1 gi−1 +gi ti gi−1 +gi

Expanded PaperInterface specification

```
∀ (gLo gHi tLo tHi : ℝ),
  GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryClosedRate gLo gHi tLo tHi =
    -(gLo + gHi) *
      Real.log
        ((1 - tLo) ^ (gLo / (gLo + gHi)) * (1 - tHi) ^ (gHi / (gLo + gHi)) +
          tLo ^ (gLo / (gLo + gHi)) * tHi ^ (gHi / (gLo + gHi)))
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryClosedRate

Lean-elaborated claim roles

1. parameter: ℝ
2. parameter: ℝ
3. parameter: ℝ
4. parameter: ℝ
5. conclusion: GJ19OptimalBinaryRatingSystems.paperAdjacentBinaryClosedRate gLo gHi tLo tHi =
  -(gLo + gHi) *
    Real.log
      ((1 - tLo) ^ (gLo / (gLo + gHi)) * (1 - tHi) ^ (gHi / (gLo + gHi)) +
        tLo ^ (gLo / (gLo + gHi)) * tHi ^ (gHi / (gLo + gHi)))

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemma31_closed_adjacent_rate_formula

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 5

Source locator: cited publication, lines 588-598

Verbatim source input

Theorem 3.2. [U+0001 control character]NestedBisection finds an [U+000F control character]-optimal t
in O M log2 M
operations, where [U+000F control character]-optimal means
[U+000F control character]
that r(t) is within additive constant [U+000F control character] of optimal.
The proof is in the appendix. The main difficulty
is finding a Lipschitz constant [U+000F control character](δ) for how much
the rate changes with a shift δ in a level ti . This
requires lower bounding t1 as a function of M . In
practice, the algorithm runs instantaneously on a
modern machine (e.g. for M = 200).

Expanded PaperInterface specification

```
∀ {m : ℕ},
  1 < m →
    ∀ (optimal sampleRate : Fin (m + 2) → ℝ) (eps : ℝ),
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector optimal →
        GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize optimal sampleRate →
          (∀ (idx : Fin (m + 2)), 0 < sampleRate idx) →
            (∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b) →
              sampleRate
                  (GJ19OptimalBinaryRatingSystems.adjacentHighIndex
                    GJ19OptimalBinaryRatingSystems.firstAdjacentIndex) =
                1 →
                0 < eps →
                  have grid : ℝ := GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps;
                  have lastLower : ℝ := 1 - 1 / ↑(m + 1);
                  have sourceDepthBudget : ℝ := max ((1 - grid - lastLower) / 2) 1;
                  have runtimeLog : ℝ := Real.logb 2 (max 1 (sourceDepthBudget / grid)) + 2;
                  0 < grid ∧
                    ∃ (L : ℕ),
                      ↑(L + 1) ≤ runtimeLog ∧
                        GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective optimal sampleRate -
                            GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
                              (GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput m (L + 1) L
                                sampleRate grid)
                              sampleRate ≤
                          eps ∧
                        GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount (m + 2) (L + 1) L ≤
                          AppliedModelingLib.Optimization.nestedBisectionStepBound (m + 2) L ∧
                          ↑(GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount (m + 2) (L + 1) L) ≤
                            ↑(m + 2) * runtimeLog ^ 2
```

Retained governing declarations

- AppliedModelingLib.Optimization.nestedBisectionStepBound
- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
- GJ19OptimalBinaryRatingSystems.firstAdjacentIndex
- GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount
- GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput
- GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: 1 < m
3. parameter: Fin (m + 2) → ℝ
4. parameter: Fin (m + 2) → ℝ
5. parameter: ℝ
6. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector optimal
7. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize optimal sampleRate
8. assumption: ∀ (idx : Fin (m + 2)), 0 < sampleRate idx
9. assumption: ∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b
10. assumption: sampleRate (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex) = 1
11. assumption: 0 < eps
12. conclusion: 0 < GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps ∧
  ∃ (L : ℕ),
    ↑(L + 1) ≤

```
  Real.logb 2
      (max 1
        (max
           ((1 - GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps -
              (1 - 1 / ↑(m + 1))) /
             2)
           1 /
          GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps)) +
    2 ∧
GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective optimal sampleRate -
      GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
        (GJ19OptimalBinaryRatingSystems.theorem32WeightedNestedBisectionOutput m (L + 1) L sampleRate
          (GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps))
        sampleRate ≤
    eps ∧
  GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount (m + 2) (L + 1) L ≤
      AppliedModelingLib.Optimization.nestedBisectionStepBound (m + 2) L ∧
    ↑(GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount (m + 2) (L + 1) L) ≤
      ↑(m + 2) *
        (Real.logb 2
            (max 1
              (max
                 ((1 - GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps -
                    (1 - 1 / ↑(m + 1))) /
                   2)
                 1 /
                GJ19OptimalBinaryRatingSystems.theorem32WeightedSourceGrid sampleRate eps)) +
          2) ^
        2
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.theorem32_weighted_nested_bisection_loss_and_runtime

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 6

Source locator: cited publication, lines 652-668

Verbatim source input

Using known set Θ, the platform can run an experiment to create an estimate $\hat{\psi}(\theta, y)$, $\forall y \in Y, \theta \in \Theta$.
Design heuristic. How can the platform build an
effective rating system using these primitives and
β? We consider the following heuristic design: the
platform randomly shows a question $y \in Y$ to each
buyer. The choice of the platform is a distribution
$H(y)$ over $y \in Y$; in other words, for the platform
the design of the system amounts to choosing the frequency with which each question is shown. At each
rating opportunity, y is chosen from Y according to
H, independently across opportunities.
Clearly, the probability that
P an item θ receives a
positive rating is: $\tilde{\beta}(\theta)$, $y \in Y \psi(\theta, y)H(y)$.
We want a distribution H such that $\tilde{\beta}(\theta) = \beta(\theta)$ for
all $\theta \in [0, 1]$, i.e., that the positive rating probability
for each item is exactly the optimal value. However,
there may not exist any set of questions Y with associated ψ and choice of H such that $\tilde{\beta} = \beta$.7

Expanded PaperInterface specification

```
∀ {Y : Type u_1} [inst : Fintype Y] (ψ : ℝ → Y → ℝ) (H : Y → ℝ) (θ : ℝ),
  GJ19OptimalBinaryRatingSystems.sourceInducedBinaryResponse ψ H θ = ∑ y : Y, ψ θ y * H y
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.sourceInducedBinaryResponse

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype Y
3. parameter: ℝ → Y → ℝ
4. parameter: Y → ℝ
5. parameter: ℝ
6. conclusion: GJ19OptimalBinaryRatingSystems.sourceInducedBinaryResponse ψ H θ = ∑ y : Y, ψ θ y * H y

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.section4_induced_binary_response_realizes_spec

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 7

Source locator: cited publication, lines 669-696

Verbatim source input

We propose the following heuristic to address this
difficulty. Choose a probability distribution H to
minimize the following L1 distance:
X
X
min
|β(θ) −
ψ̂(θ, y)H(y)|
(5)
H:kHk1 =1

θ∈Θ

y∈Y

This heuristic uses the data available to the platform, ψ̂(θ, y) for a set of items θ ∈ Θ, and designs H
so at least these items receive ratings close to their
optimal ratings β(θi). Then, as long as ψ is wellbehaved, and Θ is representative of the full set, one
can hope that β̂(θ) u β(θ), for all θ ∈ [0, 1].
Discussion. Real-world analogue & constraints. A
special case of this system is already in place on
7

There are special cases where an exact solution exists. For example, let Y = [0, 1], and ψ(θ, y) = I[θ ≥ y].

many platforms, where the same question is always
shown. Static systems can be designed by restricting
H to only have mass at one question y. More generally, constraints can be used in optimization (5).

Expanded PaperInterface specification

```
∀ {Representative : Type u_1} {Y : Type u_2} [inst : Fintype Representative] [inst_1 : Fintype Y]
  (quality : Representative → ℝ) (β : ℝ → ℝ) (ψHat : Representative → Y → ℝ) (H : Y → ℝ),
  GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective quality β ψHat H =
    ∑ i : Representative, |β (quality i) - ∑ y : Y, ψHat i y * H y|
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Fintype Representative
4. parameter: Fintype Y
5. parameter: Representative → ℝ
6. parameter: ℝ → ℝ
7. parameter: Representative → Y → ℝ
8. parameter: Y → ℝ
9. conclusion: GJ19OptimalBinaryRatingSystems.sourceQuestionDesignL1Objective quality β ψHat H =
  ∑ i : Representative, |β (quality i) - ∑ y : Y, ψHat i y * H y|

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.section4_l1_design_objective_realizes_spec

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 8

Source locator: cited publication, lines 1832-1852

Verbatim source input

Z
Z Z
1

μk+1 (Θ, X) =
Ek

x0 =0

ω(θ, x, x0)dxμk (dx0 , dθ) +

x∈X

Ekc

μk (dx, dθ).
x∈X

It is straightforward but tedious to check that the preceding dynamics are well defined, given our primitives.

Expanded PaperInterface specification

GJ19OptimalBinaryRatingSystems.ProofBridge.appendixB1_state_update =
  GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.ProofBridge.appendixB1_state_update
- GJ19OptimalBinaryRatingSystems.SourceSystemState
- GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass

Lean-elaborated claim roles

1. conclusion: GJ19OptimalBinaryRatingSystems.ProofBridge.appendixB1_state_update =
  GJ19OptimalBinaryRatingSystems.sourceAppendixBStateUpdateMass

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.appendixB1_state_update_realizes_spec

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 9

Source locator: cited publication, lines 1927-1949

Verbatim source input

Formalization of effect of matching rates shifting

Matching concentrating at the top items moves mass of β(θ) away from high θ, and subsequently
mass of H(y) away from the questions that help distinguish the top items, as observed in Figures 1b and 3b above. Informally, this occurs because when
↪ matching concentrates, top items are
accumulating many ratings more ratings comparatively, and so the amount of information needed
per rating is comparatively less. We formalize this intuition in Lemma B.1 below.
The lemma states that if matching rates shift such that there is an index k above which matching
rates increase and below which they decrease, then correspondingly the levels of β, (i.e. ti) become
closer together above k.
Lemma B.1. Suppose k, g, g̃ such that ∀j ∈ {k+1 . . . M −1}, gj ≥ g̃j , and ∀j ∈ {0 . . . k−1}, gj ≤ g̃j ,
and gk = g̃k . Then, t∗k ≥ t̃∗k .
Proof. This proof is similar to that of Lemma 3.1, except that with the matching function changing
the overall rate function can either increase, decrease, or stay the same. Suppose the overall rate
function decreased or stayed the same when the matching function changed from g̃ to g. Then
gM −2 > g̃M −2 and the target rate is no larger, and so t∗M −2 > t̃∗M −2 . Then, t∗M −3 > t̃∗M −3 (a smaller
width is needed because the matching rates are higher and the rate is no larger, and the next value
also increased). This shifting continues until t∗k+1 > t̃∗k+1 . Then, t∗k > t̃∗k .
Suppose instead that the overall rate function increased when the matching function changed
from g̃ to g. Then g1 < g̃1 and the target rate is larger, and so t∗1 > t̃∗1 . Then, t∗2 > t̃∗2 (a larger
width is needed and the previous value also increased). This shifting continues until t∗k−1 > t̃∗k−1 .
Then, t∗k > t̃∗k .

Expanded PaperInterface specification

```
∀ {m k : ℕ},
  0 < k →
    ∀ (hkm : k < m + 1) (sampleRate shiftedRate levels shiftedLevels : Fin (m + 2) → ℝ),
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels →
        GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector shiftedLevels →
          GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate →
            GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize shiftedLevels shiftedRate →
              (∀ (i : Fin (m + 2)), 0 < sampleRate i) →
                (∀ (i : Fin (m + 2)), 0 < shiftedRate i) →
                  (∀ (i : Fin (m + 2)), k < ↑i → shiftedRate i ≤ sampleRate i) →
                    (∀ (i : Fin (m + 2)), ↑i < k → sampleRate i ≤ shiftedRate i) →
                      sampleRate
                          ⟨k,
                            Decidable.byContradiction fun (a : ¬k < m + 2) =>
                              GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_1
                                hkm a) =
                        shiftedRate
                          ⟨k,
                            Decidable.byContradiction fun (a : ¬k < m + 2) =>
                              GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_1
                                hkm a) →
                      shiftedLevels
                          ⟨k,
                            GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_2
                              hkm⟩ ≤
                        levels
                          ⟨k,
                            GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_2
                              hkm⟩
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ
3. assumption: 0 < k
4. assumption: k < m + 1
5. parameter: Fin (m + 2) → ℝ
6. parameter: Fin (m + 2) → ℝ
7. parameter: Fin (m + 2) → ℝ
8. parameter: Fin (m + 2) → ℝ
9. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels
10. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector shiftedLevels
11. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate
12. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize shiftedLevels shiftedRate
13. assumption: ∀ (i : Fin (m + 2)), 0 < sampleRate i
14. assumption: ∀ (i : Fin (m + 2)), 0 < shiftedRate i
15. assumption: ∀ (i : Fin (m + 2)), k < ↑i → shiftedRate i ≤ sampleRate i
16. assumption: ∀ (i : Fin (m + 2)), ↑i < k → sampleRate i ≤ shiftedRate i
17. assumption: sampleRate
    ⟨k,

```
    Decidable.byContradiction fun (a : ¬k < m + 2) =>
      GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_1 hkm a) =
  shiftedRate
    (k,
      Decidable.byContradiction fun (a : ¬k < m + 2) =>
        GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_1 hkm a)
18. conclusion: Real.le† (shiftedLevels (k, GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_2 hkm))
  (levels (k, GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shiftSpec._proof_2 hkm))
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB1_matching_rate_shift

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 10

Source locator: cited publication, lines 1980-2003

Verbatim source input

w
w uniformly as N → ∞.
N, ∃β w s.t. βC2
N +1 → β
The proof is technical and is below. We leverage the fact that, for g uniform, the levels of β2M can be analytically written as a function of the levels of βM . We believe (numerically observe) that this theorem holds for the entire sequence as opposed to the each such subsequence, and for general matching functions g. However, our proof technique does not carry over, and the proof would leverage more global properties of the optimal βM .
Furthermore, the condition on w is light. For example, it holds for Kendall's τ , Spearman's ρ, and all other examples mentioned in this work.
This convergence result suggests that the choice of M when calculating a asymptotic and rate optimal β is not consequential. As M increases, the limiting value of Wk increases to 1 (i.e. the asymptotic value increases), but the optimal rate decreases to 0. As discussed above, with strictly increasing and continuous β, the asymptotic value is 1 but the large deviations rate does not exist, i.e. convergence is polynomial.
This result could potentially be strengthened as follows: first, show convergence on the entire sequence as opposed to these exponential subsequences, as conjectured; second, show desirable properties of the limiting function itself. It is conceivable but not necessarily true that the limiting
9

[form-feed in source extraction]
function is "better" than other strictly continuous increasing functions in some rate sense, even though the comparison through large deviations rate is degenerate.

[Next verbatim source excerpt]

Theorem B.1. Let g be uniform. Suppose w such that qwM converges uniformly. Then, ∀C ∈
w
w uniformly as N → ∞.
N, ∃β w s.t. βC2
N +1 → β
The proof is technical and is below. We leverage the fact that, for g uniform, the levels of β2M can be analytically written as a function of the levels of βM . We believe (numerically observe) that this theorem holds for the entire sequence as opposed to the each such subsequence, and for general matching functions g. However, our proof technique does not carry over, and the proof would leverage more global properties of the optimal βM .
Furthermore, the condition on w is light. For example, it holds for Kendall's τ , Spearman's ρ, and all other examples mentioned in this work.
This convergence result suggests that the choice of M when calculating a asymptotic and rate optimal β is not consequential. As M increases, the limiting value of Wk increases to 1 (i.e. the asymptotic value increases), but the optimal rate decreases to 0. As discussed above, with strictly increasing and continuous β, the asymptotic value is 1 but the large deviations rate does not exist, i.e. convergence is polynomial.
This result could potentially be strengthened as follows: first, show convergence on the entire sequence as opposed to these exponential subsequences, as conjectured; second, show desirable properties of the limiting function itself. It is conceivable but not necessarily true that the limiting
9

[form-feed in source extraction]
function is "better" than other strictly continuous increasing functions in some rate sense, even though the comparison through large deviations rate is degenerate.

B.5

Learning ψ(θ, y) through experiments

Now, we show how a platform would run an experiment to decide to learn ψ(θ, y). In particular, one potential issue is that the platform does not have any items with know quality that it can use as representative items in its optimization. In this case, we show that it can use ratings within the experiment itself to identify these representative items. The results essentially follow from the law of large numbers.
We assume that |Θ| = L representative items i ∈ {1 . . . L} are in the experiment, and each are matched N times. The experiment proceeds as follows: every time an item is matched, show the buyer a random question from Y. For each word y ∈ Y, track the empirical ψ̂(i, y), the proportion of times a positive response was given to question y. Alternatively, if Y is totally ordered (i.e. a positive rating for a given y also implies positive ratings would be given to all "easier" y 0), and can be phrased as a multiple choice question, data collection can be faster: e.g., as we do in our experiments: Y consists of a set of totally ordered adjectives that can describe the item; the rater is asked to pick an adjective out of the set; this is interpreted as the item receiving a positive answer to the questions induced by the chosen answer and all worse adjectives, and a negative answer to all better adjectives.
First, suppose the platform approximately knows the quality θi of each item i, and θi are evenly distributed in [0, 1]. Suppose the items are ordered by index, i.e. θ1 < θ2 < · · · < θL . Then let ψ̂(θ, y) = ψ̂(i, y) when θ ∈ [θi−1 , θi]. Call this procedure KnownTypeExperiment.

Expanded PaperInterface specification

∀ (betaSeq quantileSeq : ℕ → ℝ → ℝ) (quantileLimit : ℝ → ℝ) (levels : (m : ℕ) → Fin (m + 2) → ℝ)
  (levelIndex : (m : ℕ) → ℝ → Fin (m + 2)),
  (∀ (m : ℕ) (θ : ℝ), betaSeq (m + 2) θ = levels m (levelIndex m θ)) →
    (∀ (m : ℕ),
      AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector

```
    (fun (xs : Fin (m + 2) → ℝ) =>
      GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs fun (x : Fin (m + 2)) => 1)
    (levels m)) →
(∀ (m : ℕ), ∀ θ ∈ Set.Icc 0 1, ↑(levelIndex m θ) = min ⌊↑(m + 2) * quantileSeq (m + 2) θ⌋₊ (m + 1)) →
  (∀ (m : ℕ), ∀ θ ∈ Set.Icc 0 1, quantileSeq (m + 2) θ ∈ Set.Icc 0 1) →
    TendstoUniformlyOn quantileSeq quantileLimit Filter.atTop (Set.Icc 0 1) →
      GJ19OptimalBinaryRatingSystems.theoremB1UniformOptimalSubsequencePrincipleTo betaSeq quantileSeq
        quantileLimit
```

Retained governing declarations

- AppliedModelingLib.Optimization.IsMaximizerOn
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
- GJ19OptimalBinaryRatingSystems.theoremB1UniformOptimalSubsequencePrincipleTo

Lean-elaborated claim roles

1. parameter: ℕ → ℝ → ℝ
2. parameter: ℕ → ℝ → ℝ
3. parameter: ℝ → ℝ
4. parameter: (m : ℕ) → Fin (m + 2) → ℝ
5. parameter: (m : ℕ) → ℝ → Fin (m + 2)
6. assumption: ∀ (m : ℕ) (θ : ℝ), betaSeq (m + 2) θ = levels m (levelIndex m θ)
7. assumption: ∀ (m : ℕ),
   AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
     (fun (xs : Fin (m + 2) → ℝ) =>
       GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs fun (x : Fin (m + 2)) => 1)
     (levels m)
8. assumption: ∀ (m : ℕ), ∀ θ ∈ Set.Icc 0 1, ↑(levelIndex m θ) = min ⌊↑(m + 2) * quantileSeq (m + 2) θ⌋₊ (m + 1)
9. assumption: ∀ (m : ℕ), ∀ θ ∈ Set.Icc 0 1, quantileSeq (m + 2) θ ∈ Set.Icc 0 1
10. assumption: TendstoUniformlyOn quantileSeq quantileLimit Filter.atTop (Set.Icc 0 1)
11. assumption: TendstoUniformlyOn quantileSeq quantileLimit Filter.atTop (Set.Icc 0 1)
12. conclusion: GJ19OptimalBinaryRatingSystems.theoremB1DyadicSubsequenceUniformConvergence betaSeq

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_theoremB1_uniform_subsequence_principle_to_of_quantile_floor_tendstoUniformlyOn_geometric_⌋
↪ mesh

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 11

Source locator: cited publication, lines 2027-2031

Verbatim source input

ψ(θi , y)∀y uniformly as N → ∞. As L → ∞, ψ̂(θ, y) → ψ(θ, y)∀θ uniformly.
Proof. The proof follows directly from the Strong Law of Large Numbers. As N → ∞, ∀i, ψ̂(i, x) →
ψ(θi , x) uniformly. Now, let L → ∞. ∀[U+000F control character], ∃L0 s.t. ∀L > L0 , ∀θ, ∃i s.t. |θ −θi | < [U+000F control character]. ψ(θ, x) is Lipschitz in θ by assumption, and so ψ̂(θ, x) → ψ(θ, x) uniformly.
We now relax the assumption that the platform has an existing set of items with known qualities.

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} {Y : Type u_2} [inst : MeasurableSpace Ω] [Fintype Y] [inst_2 : DecidableEq Y]
  (P : MeasureTheory.Measure Ω)
  (E : (L : ℕ) → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin (L + 1)) Y)
  (quality : (L : ℕ) → Fin (L + 1) → ℝ) (representative : (L : ℕ) → ℝ → Fin (L + 1)) (psi : ℝ → Y → ℝ) (H : Y → ℝ),
  (∀ (y : Y), 0 < H y) →
    (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
       MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P) →
     (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
        Pairwise
          (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
            (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y))) →
      (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
         ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y n)
           (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P P) →
       (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
          ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0 ω ∂P = H y) →
        (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
           MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0)
             P) →
         (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
            Pairwise
              (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y))) →
           (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
              ProbabilityTheory.IdentDistrib
                (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y n)
                (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P P) →
            (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
               ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0 ω ∂P =
                 H y * psi (quality L i) y) →
              (∀ δ > 0,
                 ∀ᶠ (L : ℕ) in Filter.atTop, ∀ θ ∈ Set.Icc 0 1, dist (quality L (representative L θ)) θ < δ) →
               ∀ {K : ℝ},
                 0 ≤ K →
                   (∀ (θ θ' : ℝ) (y : Y), dist (psi θ y) (psi θ' y) ≤ K * dist θ θ') →
                     ∀ᵐ (ω : Ω) ∂P,
                       ∀ ε > 0,
                         ∀ᶠ (L : ℕ) (N : ℕ) in Filter.atTop,
                           ∀ θ ∈ Set.Icc 0 1,
                             ∀ (y : Y),
                               dist
                                   (GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
                                     (E L) (representative L θ) y N ω)
                                   (psi θ y) <
                                 ε
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
- GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator
- GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: MeasurableSpace Ω
4. parameter: Fintype Y
5. parameter: DecidableEq Y
6. parameter: MeasureTheory.Measure Ω
7. parameter: (L : ℕ) → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin (L + 1)) Y
8. parameter: (L : ℕ) → Fin (L + 1) → ℝ
9. parameter: (L : ℕ) → ℝ → Fin (L + 1)
10. parameter: ℝ → Y → ℝ
11. parameter: Y → ℝ
12. assumption: ∀ (y : Y), 0 < H y
13. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
  MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P
14. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),

```
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y))
15. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P P
16. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0 ω ∂P = H y
17. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
  MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P
18. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y))
19. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P P
20. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0 ω ∂P = H y * psi (quality L i) y
21. assumption: ∀ δ > 0, ∀ᶠ (L : ℕ) in Filter.atTop, ∀ θ ∈ Set.Icc 0 1, dist (quality L (representative L θ)) θ < δ
22. parameter: ℝ
23. assumption: 0 ≤ K
24. assumption: ∀ (θ θ' : ℝ) (y : Y), dist (psi θ y) (psi θ' y) ≤ K * dist θ θ'
25. conclusion: P
    {x : Ω |
      (fun (ω : Ω) =>
        ∀ ε > 0,
          ∀ᶠ (L : ℕ) (N : ℕ) in Filter.atTop,
            ∀ θ ∈ Set.Icc 0 1,
              ∀ (y : Y),
                dist
                  (GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse (E L)
                    (representative L θ) y N ω)
                  (psi θ y) <
                ε)
        x}ᶜ =
0
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaB2_knownTypeExperiment_uniform_convergence_of_lipschitz_tracking

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 12

Source locator: cited publication, lines 2005-2031

Verbatim source input

Learning ψ(θ, y) through experiments

Now, we show how a platform would run an experiment to decide to learn ψ(θ, y). In particular, one potential issue is that the platform does not have any items with know quality that it can use as representative items in its optimization. In this case, we show that it can use ratings within the experiment itself to identify these representative items. The results essentially follow from the law of large numbers.
We assume that |Θ| = L representative items i ∈ {1 . . . L} are in the experiment, and each are matched N times. The experiment proceeds as follows: every time an item is matched, show the buyer a random question from Y. For each word y ∈ Y, track the empirical ψ̂(i, y), the proportion of times a positive response was given to question y. Alternatively, if Y is totally ordered (i.e. a positive rating for a given y also implies positive ratings would be given to all “easier” y 0), and can be phrased as a multiple choice question, data collection can be faster: e.g., as we do in our experiments: Y consists of a set of totally ordered adjectives that can describe the item; the rater is asked to pick an adjective out of the set; this is interpreted as the item receiving a positive answer to the questions induced by the chosen answer and all worse adjectives, and a negative answer to all better adjectives.
First, suppose the platform approximately knows the quality θi of each item i, and θi are evenly distributed in [0, 1]. Suppose the items are ordered by index, i.e. θ1 < θ2 < · · · < θL . Then let ψ̂(θ, y) = ψ̂(i, y) when θ ∈ [θi−1 , θi]. Call this procedure KnownTypeExperiment.
Lemma B.2. Suppose ψ(θ, y) is Lipschitz continuous in θ. With KnownTypeExperiment, ψ̂(i, y) → ψ(θi , y)∀y uniformly as N → ∞. As L → ∞, ψ̂(θ, y) → ψ(θ, y)∀θ uniformly.
Proof. The proof follows directly from the Strong Law of Large Numbers. As N → ∞, ∀i, ψ̂(i, x) → ψ(θi , x) uniformly. Now, let L → ∞. ∀[U+000F control character], ∃L0 s.t. ∀L > L0 , ∀θ, ∃i s.t. |θ −θi | < [U+000F control character]. ψ(θ, x) is Lipschitz in θ by assumption, and so ψ̂(θ, x) → ψ(θ, x) uniformly.
We now relax the assumption that the platform has an existing set of items with known qualities.

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} {Representative : Type u_2} {Y : Type u_3} [inst : MeasurableSpace Ω] [Fintype Representative]
  [Fintype Y] [inst_3 : DecidableEq Y] (P : MeasureTheory.Measure Ω)
  (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Representative Y) (quality : Representative → ℝ)
  (ψ : ℝ → Y → ℝ) (H : Y → ℝ),
  (∀ (y : Y), 0 < H y) →
    (∀ (i : Representative) (y : Y),
       MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P) →
     (∀ (i : Representative) (y : Y),
        Pairwise
          (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
            (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y))) →
      (∀ (i : Representative) (y : Y) (n : ℕ),
         ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y n)
           (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P P) →
       (∀ (i : Representative) (y : Y),
          ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0 ω ∂P = H y) →
        (∀ (i : Representative) (y : Y),
           MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P) →
         (∀ (i : Representative) (y : Y),
            Pairwise
              (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y))) →
          (∀ (i : Representative) (y : Y) (n : ℕ),
             ProbabilityTheory.IdentDistrib
               (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y n)
               (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P P) →
           (∀ (i : Representative) (y : Y),
              ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0 ω ∂P =
                H y * ψ (quality i) y) →
            ∀ᵐ (ω : Ω) ∂P,
              ∀ (i : Representative) (y : Y),
                Filter.Tendsto
                  (fun (N : ℕ) =>
                    GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse E i y N ω)
                  Filter.atTop (nhds (ψ (quality i) y))
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
- GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator
- GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: MeasurableSpace Ω

```
5. parameter: Fintype Representative
6. parameter: Fintype Y
7. parameter: DecidableEq Y
8. parameter: MeasureTheory.Measure Ω
9. parameter: GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω Representative Y
10. parameter: Representative → ℝ
11. parameter: ℝ → Y → ℝ
12. parameter: Y → ℝ
13. assumption: ∀ (y : Y), 0 < H y
14. assumption: ∀ (i : Representative) (y : Y),
  MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P
15. assumption: ∀ (i : Representative) (y : Y),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y))
16. assumption: ∀ (i : Representative) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P P
17. assumption: ∀ (i : Representative) (y : Y),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0 ω ∂P = H y
18. assumption: ∀ (i : Representative) (y : Y),
  MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P
19. assumption: ∀ (i : Representative) (y : Y),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y))
20. assumption: ∀ (i : Representative) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P P
21. assumption: ∀ (i : Representative) (y : Y),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0 ω ∂P = H y * ψ (quality i) y
22. conclusion: P
    {x : Ω |
      (fun (ω : Ω) =>
        ∀ (i : Representative) (y : Y),
          Filter.Tendsto
            (fun (N : ℕ) => GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse E i y N ω)
            Filter.atTop (nhds (ψ (quality i) y)))
        x}ᶜ =
  0
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB2_knownTypeExperiment_random_question_slln

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 13

Source locator: cited publication, lines 2044-2060

Verbatim source input

ψ(θ, y)∀y, θ uniformly as L, N → ∞.
Proof. Fix L. Denote each item in the experiment as i ∈ {1 . . . L} (with true quality θi 6= θj),
and each item has N samples. Without loss of generality, assume the items are indexed according
to their rank on the average of their scores on the samples, defined as the percentage of positive
ratings received. i = 1 is then the worst item, and i = L is the best item according to scores in the
experiment.
For ψ(θ, x) increasing in θ, as N → ∞, P r(θi > θj |i < j) → 0 almost surely by SLLN, and for
a fixed L, {θi } this convergence[U+0002 control character]is uniform.
Furthermore, by SLLN, ψ̂(i, x) → ψ(θi , x) as N → ∞.
[U+0003 control character]
i−1 i
Recall ψ̂(θ, x) = ψ̂(i, x) for θ ∈ L , L .
10

[form-feed in source extraction]
Now, let L → ∞. ∀[U+000F control character], ∃L0 s.t. ∀L > L0 , ∀θ, ∃i s.t. |θ − θi | < [U+000F control character]. ψ(θ, x) is Lipschitz in θ by assumption, and so ψ̂(θ, x) → ψ(θ, x) uniformly.

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} {Y : Type u_2} [inst : MeasurableSpace Ω] [inst_1 : Fintype Y] [inst_2 : DecidableEq Y]
  (P : MeasureTheory.Measure Ω)
  (E : (L : ℕ) → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin (L + 1)) Y)
  (quality : (L : ℕ) → Fin (L + 1) → ℝ) (rankRepresentative : (L : ℕ) → ℝ → Fin (L + 1)) (psi : ℝ → Y → ℝ) (H : Y → ℝ),
  (∀ (y : Y), 0 < H y) →
    (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
       MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P) →
     (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
        Pairwise
          (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
            (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y))) →
      (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
         ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y n)
           (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P P) →
       (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
          ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0 ω ∂P = H y) →
        (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
           MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0)
             P) →
         (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
            Pairwise
              (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y))) →
          (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
             ProbabilityTheory.IdentDistrib
               (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y n)
               (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P P) →
           (∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
              ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0 ω ∂P =
                H y * psi (quality L i) y) →
            (∀ (L : ℕ) (i : Fin (L + 1)),
               MeasureTheory.Integrable
                 (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0) P) →
             (∀ (L : ℕ) (i : Fin (L + 1)),
                Pairwise
                  (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                    (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i))) →
              (∀ (L : ℕ) (i : Fin (L + 1)) (n : ℕ),
                 ProbabilityTheory.IdentDistrib
                   (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i n)
                   (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0) P P) →
               (∀ (L : ℕ) (i : Fin (L + 1)),
                  ∫ (ω : Ω),
                    GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0 ω ∂P =
                  GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore psi H (quality L i)) →
                (∀ (L : ℕ),
                   StrictMono fun (i : Fin (L + 1)) =>
                     GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore psi H (quality L i)) →
                 (∀ δ > 0,
                    ∀ᶠ (L : ℕ) in Filter.atTop,
                      ∀ θ ∈ Set.Icc 0 1, dist (quality L (rankRepresentative L θ)) θ < δ) →
                  ∀ {K : ℝ},
                    0 ≤ K →
                      (∀ (θ θ' : ℝ) (y : Y), dist (psi θ y) (psi θ' y) ≤ K * dist θ θ') →
                        ∀ᵐ (ω : Ω) ∂P,
                          ∀ ε > 0,
                            ∀ᶠ (L : ℕ) (N : ℕ) in Filter.atTop,
                              (StrictMono fun (i : Fin (L + 1)) =>
                                  GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
                                    (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L)
                                      i)
                                    N ω) ∧
                                ∀ θ ∈ Set.Icc 0 1,
                                  ∀ (y : Y),
                                    dist
```

(GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
(E L) (rankRepresentative L θ) y N ω)
(psi θ y) <
ε

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator
- GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
- GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator
- GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: MeasurableSpace Ω
4. parameter: Fintype Y
5. parameter: DecidableEq Y
6. parameter: MeasureTheory.Measure Ω
7. parameter: (L : ℕ) → GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin (L + 1)) Y
8. parameter: (L : ℕ) → Fin (L + 1) → ℝ
9. parameter: (L : ℕ) → ℝ → Fin (L + 1)
10. parameter: ℝ → Y → ℝ
11. parameter: Y → ℝ
12. assumption: ∀ (y : Y), 0 < H y
13. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P
14. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
Pairwise
(Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
(GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y))
15. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y n)
(GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0) P P
16. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator (E L) i y 0 ω ∂P = H y
17. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P
18. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
Pairwise
(Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
(GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y))
19. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y) (n : ℕ),
ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y n)
(GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0) P P
20. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (y : Y),
∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator (E L) i y 0 ω ∂P = H y * psi (quality L i) y
21. assumption: ∀ (L : ℕ) (i : Fin (L + 1)),
MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0) P
22. assumption: ∀ (L : ℕ) (i : Fin (L + 1)),
Pairwise
(Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
(GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i))
23. assumption: ∀ (L : ℕ) (i : Fin (L + 1)) (n : ℕ),
ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i n)
(GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0) P P
24. assumption: ∀ (L : ℕ) (i : Fin (L + 1)),
∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i 0 ω ∂P =
GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore psi H (quality L i)
25. assumption: ∀ (L : ℕ),
StrictMono fun (i : Fin (L + 1)) => GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore psi H (quality L i)
26. assumption: ∀ δ > 0, ∀ᶠ (L : ℕ) in Filter.atTop, ∀ θ ∈ Set.Icc 0 1, dist (quality L (rankRepresentative L θ)) θ < δ
27. parameter: ℝ
28. assumption: 0 ≤ K
29. assumption: ∀ (θ θ' : ℝ) (y : Y), dist (psi θ y) (psi θ' y) ≤ K * dist θ θ'
30. conclusion: P
{x : Ω |
(fun (ω : Ω) =>
∀ ε > 0,
∀ᶠ (L : ℕ) (N : ℕ) in Filter.atTop,
(StrictMono fun (i : Fin (L + 1)) =>
GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
(GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator (E L) i) N ω) ∧
∀ θ ∈ Set.Icc 0 1,
∀ (y : Y),
dist
(GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse (E L)
(rankRepresentative L θ) y N ω)
(psi θ y) <
ε)
x}ᶜ =
0

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaB3_unknownTypeExperiment_uniform_convergence_of_ranking_tracking

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 14

Source locator: cited publication, lines 2032-2060

Verbatim source input

Suppose instead the platform has many items L of unknown quality who are expected to match N
times each over the experiment time period. For each item, the platform would again ask questions
from Y, drawn according to any distribution (with positive mass on each question). Then generate
ψ̂(θ, y) as follows: first, rank the items according to their ratings during the experiment itself.
Then, for each y, ψ̂(θ, y) is
the empirical
performance of the θth percentile item in the ranking, i.e.
[U+0002 control character] i−1
[U+0003 control character]
i
ψ̂(θ, y) = ψ̂(θi , y) for θ ∈ L , L . Call this procedure UnknownTypeExperiment.
Lemma B.3. Suppose ψ(θ, y) is Lipschitz continuous in θ. With UnknownTypeExperiment, ψ̂(θ, y) →
ψ(θ, y)∀y, θ uniformly as L, N → ∞.
Proof. Fix L. Denote each item in the experiment as i ∈ {1 . . . L} (with true quality θi 6= θj),
and each item has N samples. Without loss of generality, assume the items are indexed according
to their rank on the average of their scores on the samples, defined as the percentage of positive
ratings received. i = 1 is then the worst item, and i = L is the best item according to scores in the
experiment.
For ψ(θ, x) increasing in θ, as N → ∞, P r(θi > θj |i < j) → 0 almost surely by SLLN, and for
a fixed L, {θi } this convergence[U+0002 control character]is uniform.
Furthermore, by SLLN, ψ̂(i, x) → ψ(θi , x) as N → ∞.
[U+0003 control character]
i−1 i
Recall ψ̂(θ, x) = ψ̂(i, x) for θ ∈ L , L .
10

[form-feed in source extraction]
Now, let L → ∞. ∀[U+000F control character], ∃L0 s.t. ∀L > L0 , ∀θ, ∃i s.t. |θ − θi | < [U+000F control character]. ψ(θ, x) is Lipschitz in θ by
assumption, and so ψ̂(θ, x) → ψ(θ, x) uniformly.

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} {Y : Type u_2} {L : ℕ} [inst : MeasurableSpace Ω] [inst_1 : Fintype Y] [inst_2 : DecidableEq Y]
  (P : MeasureTheory.Measure Ω) (E : GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin L) Y)
  (quality : Fin L → ℝ) (ψ : ℝ → Y → ℝ) (H : Y → ℝ),
  (∀ (y : Y), 0 < H y) →
    (∀ (i : Fin L) (y : Y),
       MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P) →
      (∀ (i : Fin L) (y : Y),
         Pairwise
           (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
             (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y))) →
        (∀ (i : Fin L) (y : Y) (n : ℕ),
           ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y n)
             (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P P) →
          (∀ (i : Fin L) (y : Y),
             ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0 ω ∂P = H y) →
            (∀ (i : Fin L) (y : Y),
               MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P) →
              (∀ (i : Fin L) (y : Y),
                 Pairwise
                   (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                     (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y))) →
                (∀ (i : Fin L) (y : Y) (n : ℕ),
                   ProbabilityTheory.IdentDistrib
                     (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y n)
                     (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P P) →
                  (∀ (i : Fin L) (y : Y),
                     ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0 ω ∂P =
                       H y * ψ (quality i) y) →
                    (∀ (i : Fin L),
                       MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0)
                         P) →
                      (∀ (i : Fin L),
                         Pairwise
                           (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
                             (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i))) →
                        (∀ (i : Fin L) (n : ℕ),
                           ProbabilityTheory.IdentDistrib
                             (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i n)
                             (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0) P P) →
                          (∀ (i : Fin L),
                             ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0 ω ∂P =
                               GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore ψ H (quality i)) →
                            (StrictMono fun (i : Fin L) =>
                               GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore ψ H (quality i)) →
                              ∀ᵐ (ω : Ω) ∂P,
                                TendstoUniformlyOn
                                  (fun (N : ℕ) (p : Fin L × Y) =>
                                    GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse E p.1 p.2
                                      N ω)
                                  (fun (p : Fin L × Y) => ψ (quality p.1) p.2) Filter.atTop Set.univ ∧
                                ∀ᶠ (N : ℕ) in Filter.atTop,
                                  StrictMono fun (i : Fin L) =>
                                    GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
                                      (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i) N ω
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment
- GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator
- GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
- GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse
- GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator
- GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator

Lean-elaborated claim roles

```
1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: ℕ
4. parameter: MeasurableSpace Ω
5. parameter: Fintype Y
6. parameter: DecidableEq Y
7. parameter: MeasureTheory.Measure Ω
8. parameter: GJ19OptimalBinaryRatingSystems.SourceRandomQuestionExperiment Ω (Fin L) Y
9. parameter: Fin L → ℝ
10. parameter: ℝ → Y → ℝ
11. parameter: Y → ℝ
12. assumption: ∀ (y : Y), 0 < H y
13. assumption: ∀ (i : Fin L) (y : Y), MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P
14. assumption: ∀ (i : Fin L) (y : Y),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y))
15. assumption: ∀ (i : Fin L) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0) P P
16. assumption: ∀ (i : Fin L) (y : Y), ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionAskedIndicator E i y 0 ω ∂P = H y
17. assumption: ∀ (i : Fin L) (y : Y),
  MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P
18. assumption: ∀ (i : Fin L) (y : Y),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y))
19. assumption: ∀ (i : Fin L) (y : Y) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y n)
    (GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0) P P
20. assumption: ∀ (i : Fin L) (y : Y),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceQuestionPositiveIndicator E i y 0 ω ∂P = H y * ψ (quality i) y
21. assumption: ∀ (i : Fin L), MeasureTheory.Integrable (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0) P
22. assumption: ∀ (i : Fin L),
  Pairwise
    (Function.onFun (fun (x1 x2 : Ω → ℝ) => ProbabilityTheory.IndepFun x1 x2 P)
      (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i))
23. assumption: ∀ (i : Fin L) (n : ℕ),
  ProbabilityTheory.IdentDistrib (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i n)
    (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0) P P
24. assumption: ∀ (i : Fin L),
  ∫ (ω : Ω), GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i 0 ω ∂P =
    GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore ψ H (quality i)
25. assumption: StrictMono fun (i : Fin L) => GJ19OptimalBinaryRatingSystems.sourceExpectedAggregateScore ψ H (quality i)
26. conclusion: P
    {x : Ω |
      (fun (ω : Ω) =>
        TendstoUniformlyOn
          (fun (N : ℕ) (p : Fin L × Y) =>
            GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalQuestionResponse E p.1 p.2 N ω)
          (fun (p : Fin L × Y) => ψ (quality p.1) p.2) Filter.atTop Set.univ ∧
        ∀ᶠ (N : ℕ) in Filter.atTop,
          StrictMono fun (i : Fin L) =>
            GJ19OptimalBinaryRatingSystems.sourceExperimentEmpiricalMean
              (GJ19OptimalBinaryRatingSystems.sourceAggregatePositiveIndicator E i) N ω)
        x}ᶜ =
  0
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaB3_unknownTypeExperiment_random_question_response_and_rank

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 15

Source locator: cited publication, lines 2083-2131

Verbatim source input

Rate functions for Pk (θ1 , θ2)

Lemma C.1.
1
lim − log [μ((xk (θ1) − xk (θ2)) ≤ 0|θ1 , θ2)] = inf {g(θ1)I(a|θ1) + g(θ2)I(a|θ2)}
a∈R
k→∞ k
where I(a|`) = supz {za − Λ(z|θ)}, and Λ(z|θ) is the log moment generating function of a single
sample from x(θ1) and g(θ) is the sampling rate.
Proof. limk→∞ − k1 log [μ((xk (θ1) − xk (θ2)) ≤ 0|θ1 , θ2)]
[U+0015 control character]
[U+0014 control character]Z
1
μ((xk (θ1) = a|θ1)μ(xk (θ2) ≥ a|θ2)da
= lim − log
k→∞ k
[U+0014 control character]Za∈R
[U+0015 control character]
1
−kg(θ1)I(a|θ1) −kg(θ2)I(a|θ2)
= lim − log
e
e
da
k→∞ k
a∈R
= inf {g(θ1)I(a|θ1) + g(θ2)I(a|θ2)}

(1)
(2)
Laplace principle

a∈R

(3)

Where (2) is a basic result from large deviations, where kg(θi) is the number of samples item
of quality θi has received.
Note that this lemma also appears in Glynn and Juneja (2004), which uses the Gartner-Ellis
Theorem in the proof. Our proof is conceptually similar but instead uses Laplace's principle.
1−b
Recall that KL(a||b) = a log ab + (1 − a) log 1−a
is the Kullback-Leibler (KL) divergence between
Bernoulli random variables with success probabilities a and b respectively. It is well known that
for a Bernoulli random variable with success probability t,
I(a|t) = KL(a||t)
11

Expanded PaperInterface specification

```
∀ {Seller : Type u_1} {Pair : Type u_2} (successProb : Seller → ℝ) (hprob0 : ∀ (θ : Seller), 0 ≤ successProb θ)
  (hprob1 : ∀ (θ : Seller), successProb θ ≤ 1),
  (∀ (θ : Seller), 0 < successProb θ) →
    (∀ (θ : Seller), successProb θ < 1) →
      ∀ (sampleRate : Seller → ℝ) (pairHi pairLo : Pair → Seller),
        (∀ (p : Pair), 0 < sampleRate (pairHi p)) →
          (∀ (p : Pair), 0 < sampleRate (pairLo p)) →
            ∀ (a z : Pair → ℝ),
              (∀ (p : Pair), z p ≤ 0) →
                (∀ (p : Pair),
                   HasDerivAt
                     (fun (t : ℝ) =>
                       (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1).logMGF (pairHi p)
                         t)
                     (a p) (z p * (sampleRate (pairHi p))⁻¹)) →
                  (∀ (p : Pair),
                     HasDerivAt
                       (fun (t : ℝ) =>
                         (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1).logMGF (pairLo p)
                           t)
                       (a p) (-(z p * (sampleRate (pairLo p))⁻¹))) →
                    Nonempty
                      (AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate
                        (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1) sampleRate pairHi
                        pairLo)
```

Retained governing declarations

- AppliedModelingLib.Probability.FiniteRatingLDPModel.logMGF
- AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate
- GJ19OptimalBinaryRatingSystems.binaryRatingModel

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Seller → ℝ
4. assumption: ∀ (θ : Seller), 0 ≤ successProb θ
5. assumption: ∀ (θ : Seller), successProb θ ≤ 1
6. assumption: ∀ (θ : Seller), 0 < successProb θ
7. assumption: ∀ (θ : Seller), successProb θ < 1
8. parameter: Seller → ℝ
9. parameter: Pair → Seller
10. parameter: Pair → Seller
11. assumption: ∀ (p : Pair), 0 < sampleRate (pairHi p)
12. assumption: ∀ (p : Pair), 0 < sampleRate (pairLo p)
13. parameter: Pair → ℝ
14. parameter: Pair → ℝ
15. assumption: ∀ (p : Pair), z p ≤ 0
16. assumption: ∀ (p : Pair),
  HasDerivAt
    (fun (t : ℝ) => (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1).logMGF (pairHi p) t)
    (a p) (z p * (sampleRate (pairHi p))⁻¹)
17. assumption: ∀ (p : Pair),
  HasDerivAt
    (fun (t : ℝ) => (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1).logMGF (pairLo p) t)
    (a p) (-(z p * (sampleRate (pairLo p))⁻¹))
18. conclusion: Nonempty
  (AppliedModelingLib.Probability.PairwiseThresholdRateTopLdpCertificate
    (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1) sampleRate pairHi pairLo)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC1_pairwise_error_rate_from_derivatives

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 16

Source locator: cited publication, lines 2133-2161

Verbatim source input

Lemma C.2. Let θ1 > θ2 and I(a|θ) = KL(a||β(θ)). Further, Let P k (θ1 , θ2) = 1 − Pk (θ1 , θ2).
Then,
1
− lim log P k (θ1 , θ2) = inf {g(θ1)I(a|θ1) + g(θ2)I(a|θ2)} ,
(4)
a∈R
k→∞ k
Proof. Follows directly from Lemma C.1.
− lim

1

k→∞ k

log P k (θ1 , θ2 |β)

1
= lim − log [1 + µk (xk (θ1) − xk (θ2) < 0|θ1 , θ2) − µk (xk (θ1) − xk (θ2) > 0|θ1 , θ2)]
k→∞ k
1
= lim − log [2µk (xk (θ1) − xk (θ2) < 0|θ1 , θ2) + µk (xk (θ1) − xk (θ2) = 0|θ1 , θ2)]
k→∞ k
1
= lim − log [µk (xk (θ1) − xk (θ2) ≤ 0|θ1 , θ2)]
k→∞ k
= inf {g(θ1)I(a|θ1) + g(θ2)I(a|θ2)}
Lemma C.1
a∈R

Expanded PaperInterface specification

```
∀ {Seller : Type u_1} (successProb : Seller → ℝ) (hprob0 : ∀ (θ : Seller), 0 ≤ successProb θ)
  (hprob1 : ∀ (θ : Seller), successProb θ ≤ 1) (sampleRate : Seller → ℝ) (hi lo : Seller),
  0 < sampleRate hi →
    0 < sampleRate lo →
      0 < successProb hi →
        successProb hi < 1 →
          0 < successProb lo →
            successProb lo < 1 →
              successProb lo ≤ successProb hi →
                AppliedModelingLib.Probability.ExponentialRateCertificate
                  (AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
                    (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1) sampleRate hi lo)
                  (AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate (sampleRate hi) (sampleRate lo)
                    (successProb hi) (successProb lo))
```

Retained governing declarations

- AppliedModelingLib.Probability.ExponentialRateCertificate
- AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
- AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate
- GJ19OptimalBinaryRatingSystems.binaryRatingModel

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Seller → ℝ
3. assumption: ∀ (θ : Seller), 0 ≤ successProb θ
4. assumption: ∀ (θ : Seller), successProb θ ≤ 1
5. parameter: Seller → ℝ
6. parameter: Seller
7. parameter: Seller
8. assumption: 0 < sampleRate hi
9. assumption: 0 < sampleRate lo
10. assumption: 0 < successProb hi
11. assumption: successProb hi < 1
12. assumption: 0 < successProb lo
13. assumption: successProb lo < 1
14. assumption: successProb lo ≤ successProb hi
15. conclusion: AppliedModelingLib.Probability.ExponentialRateCertificate
  (AppliedModelingLib.Probability.twoSampleFloorPkComplementErrorProb
    (GJ19OptimalBinaryRatingSystems.binaryRatingModel successProb hprob0 hprob1) sampleRate hi lo)
  (AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate (sampleRate hi) (sampleRate lo) (successProb hi)
    (successProb lo))

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC2_binary_complement_rate

Recorded source-to-Spec screening **Verdict:** matches
Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 17

Source locator: cited publication, lines 2163-2223

Verbatim source input

Laplace’s principle with sequence of rate functions

In order to derive a rate function for W k = (limk Wk) − Wk , we need to be able to relate its rate to that of P k (θ1 , θ2). The following theorem, related to Laplace’s principle for large deviations allows us to do so.
Theorem C.1. Suppose that X is compact with finite Lebesgue measure μ(X) < ∞. Suppose that ϕ(x) has an essential infimum ϕ on X, that ϕn (x) has an essential infimum ϕn , that both ϕ and all ϕn are nonnegative, and that ϕn → ϕ uniformly:
lim sup |ϕn (x) − ϕ(x)| = 0.

n→∞ x∈X

Then:

1
log
n→∞ n

Z

lim

e−nϕn (x) dx = −ϕ.

(5)

X

Proof. First, we note that for all x and n, e−nϕn (x) ≤ e−nϕn . Therefore, letting (∗) denote the LHS of (5), we have:
Z
1
(∗) ≤ lim log
e−nϕn dx = −ϕ,
n→∞ n
X
where the last limit follows from the fact that ϕn converges uniformly to ϕ, so that ϕn → ϕ.
Next, for [U+000F control character] > 0 let An ([U+000F control character]) = {x : ϕn (x) ≤ ϕn + [U+000F control character]} and let A([U+000F control ↪ character]) = {x : ϕ(x) ≤ ϕ + [U+000F control character]}. It
follows (again by uniform convergence) that for all sufficiently large n, A([U+000F control character]/2) ⊆ An ([U+000F control character]), so that μ(A([U+000F control character]/2)) ≤ μ(An ([U+000F control character])) for all sufficiently large n. Further, μ(A([U+000F control character]/2)) > 0, since ↪ ϕ is the essential
infimum.
Since:
Z
e−nϕn (x) dx ≥ μ(An ([U+000F control character]))e−n(ϕn +[U+000F control character]) ,
X

12

[form-feed in source extraction]
it follows that:

1
log μ(An ([U+000F control character])).
n→∞ n
To complete the proof, observe that since μ(An ([U+000F control character])) is bounded below by a positive constant for all sufficiently large n, the last limit is zero. Therefore:
(∗) ≥ −ϕ − [U+000F control character] + lim

(∗) ≥ −ϕ − [U+000F control character].
Since [U+000F control character] was arbitrary, this completes the proof.
Remark C.1. Let X = [0, 1] × [0, 1], ϕn (θ1 , θ2) = − n1 log P n (θ1 , θ2). Then, all the conditions for

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} [inst : MeasurableSpace Ω] (μ : MeasureTheory.Measure Ω) [MeasureTheory.IsFiniteMeasure μ]
  (phiSeq : ℕ → Ω → ℝ) (phi : Ω → ℝ) {rate : ℝ},
  (∀ (k : ℕ), MeasureTheory.Integrable (fun (x : Ω) => Real.exp (-↑k * phiSeq k x)) μ) →
    AppliedModelingLib.Probability.HasAEEssentialInfimum μ phi rate →
      (∀ ε > 0, ∀ᶠ (k : ℕ) in Filter.atTop, ∀ (x : Ω), |phiSeq k x - phi x| ≤ ε) →
        AppliedModelingLib.Probability.HasExponentialRate (fun (k : ℕ) => ∫ (x : Ω), Real.exp (-↑k * phiSeq k x) ∂μ)
          rate
```

Retained governing declarations

- AppliedModelingLib.Probability.HasAEEssentialInfimum
- AppliedModelingLib.Probability.HasExponentialRate

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: MeasurableSpace Ω
3. parameter: MeasureTheory.Measure Ω
4. assumption: MeasureTheory.IsFiniteMeasure μ
5. parameter: ℕ → Ω → ℝ
6. parameter: Ω → ℝ
7. parameter: ℝ
8. assumption: ∀ (k : ℕ), MeasureTheory.Integrable (fun (x : Ω) => Real.exp (-↑k * phiSeq k x)) μ
9. assumption: AppliedModelingLib.Probability.HasAEEssentialInfimum μ phi rate
10. assumption: ∀ ε > 0, ∀ᶠ (k : ℕ) in Filter.atTop, ∀ (x : Ω), |phiSeq k x - phi x| ≤ ε
11. parameter: Set ℝ
12. assumption: U ∈ nhds rate
13. conclusion: U ∈
  Filter.map (AppliedModelingLib.Probability.logDecay fun (k : ℕ) => ∫ (x : Ω), Real.exp (-↑k * phiSeq k x) ∂μ)
    Filter.atTop

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_theoremC1_laplace_principle

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 18

Source locator: cited publication, lines 2227-2364

Verbatim source input

Rate function for Wk

Our next lemma shows that we can obtain a nontrivial large deviations rate for Wk when β is a
step-wise increasing
R function.
Recall Wk = θ1 >θ2 w(θ1 , θ2)Pk (θ1 , θ2 |β)d(θ1 , θ2).
Let P k (θ1 , θ2) = 1 − Pk (θ1 , θ2).
R
Further, let W k = (limk Wk) − Wk = θ1 >θ2 w(θ1 , θ2)P k (θ1 , θ2 |β)d(θ1 , θ2). (recall we assumed
w integrates to 1 without loss of generality).
Lemma C.3. Suppose β is piecewise constant with M levels {ti }. Let gi , inf θ∈Si g(θ) = g(si)
Then,
1
− lim log W k = min inf {gi+1 I(a|ti+1) + gi I(a|ti)} , r,
(6)
0≤i≤M −2 a∈R
k→∞ k
where I(a|t) = KL(a||t) as defined in Lemma C.2.
Proof. When β is step-wise increasing with M levels {ti }, then
X Z
Wk =
w(θ1 , θ2)P k (θ1 , θ2 |β)d(θ1 , θ2)
0≤i<j<M

θ2 ∈Si ,θ1 ∈Sj

as P k (θ1 , θ2) = 0 when β(θ1) = β(θ2).

13

[form-feed in source extraction]
− limk→∞ k1 log W k
1
= − lim log
k→∞ k
1
= − lim log
k→∞ k

Z
w(θ1 , θ2)P k (θ1 , θ2 |β)d(θ1 , θ2)
X Z
w(θ1 , θ2)P k (θ1 , θ2 |β)d(θ1 , θ2)

0≤i<j<M

=
=

min

inf

(8)

θ2 ∈Si ,θ1 ∈Sj

" Z
#!
1
= − max
lim
log
w(θ1 , θ2)P k (θj , θi |β)d(θ1 , θ2)
0≤i<j<M k→∞ k
θ2 ∈Si ,θ1 ∈Sj
[U+0013 control character]
[U+0012 control character]
[U+0003 control character]
1[U+0002 control character]
= − max
sup
log w(θ1 , θ2)P k (θj , θi |β)
lim
0≤i<j<M θ1 ∈Sj ,θ2 ∈Si k→∞ k
[U+0012 control character]
[U+0013 control character]
1
= − max
sup
lim log P k (θj , θi |β)
0≤i<j<M θ1 ∈Sj ,θ2 ∈Si k→∞ k
[U+0012 control character]
[U+0013 control character]
1
− lim log P k (θj , θi |β)
= min
inf

0≤i<j<M θ1 ∈Sj ,θ2 ∈Si
k→∞ k
=

(7)

θ1 >θ2

inf {g(θ1)I(a|tj) + g(θ2)I(a|ti)}

0≤i<j<M θ1 ∈Sj ,θ2 ∈Si a∈R

(9)
(10)
(11)
(12)
(13)

min

inf {gj I(a|tj) + gi I(a|ti)}

(14)

min

inf {gi+1 I(a|ti+1) + gi I(a|ti)}

(15)

0≤i<j<M a∈R

0≤i<M −1 a∈R

The last line follows from adjacent ti , ti+1 dominating the rate due to monotonicity properties.
Line (10) follows from Theorem C.1.
[U+0011 control character]i
h
[U+0010 control character]P
N [U+000F control character]
[U+000F control character]
= maxN
Line (9) follows from: ∀a[U+000F control character]i ≥ 0, lim sup[U+000F control character]→0 [U+000F control character] log
a
i lim sup[U+000F control character]→0 [U+000F control character] log(ai), which
i
i
is a finite case version (with fewer assumptions) of Theorem C.1. See, e.g., Lemma 1.2.15 in (Dembo
and Zeitouni, 2010) for a proof of this property.

Expanded PaperInterface specification

```
∀ {Ω : Type u_1} {Component : Type u_2} [inst : MeasurableSpace Ω] [inst_1 : Fintype Component] [DecidableEq Component]
  [Nonempty Component] (μ : MeasureTheory.Measure Ω) [MeasureTheory.IsFiniteMeasure μ]
  (P : AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Component) (weight : Ω → ℝ) (kernel : ℕ → Ω → ℝ)
  (phi : Component → Ω → ℝ) (rate W : Component → ℝ),
  (∀ (k : ℕ) (component : Component),
      MeasureTheory.IntegrableOn (fun (x : Ω) => weight x * kernel k x) (P.pieceSet component) μ) →
    (∀ (component : Component), MeasureTheory.IntegrableOn weight (P.pieceSet component) μ) →
      (∀ (component : Component), 0 < W component) →
        (∀ (component : Component), ∀ᵐ (x : Ω) ∂μ.restrict (P.pieceSet component), 0 ≤ weight x) →
          (∀ (component : Component), ∀ᵐ (x : Ω) ∂μ.restrict (P.pieceSet component), weight x ≤ W component) →
            (∀ (component : Component),
                AppliedModelingLib.Probability.HasAEEssentialInfimum (μ.restrict (P.pieceSet component)) (phi component)
                  (rate component)) →
              (∀ (component : Component),
                  AppliedModelingLib.Probability.HasPositiveWeightNearAEEssentialInfimum
                    (μ.restrict (P.pieceSet component)) weight (phi component) (rate component)) →
                (∀ (k : ℕ) (x : Ω), 0 < kernel k x) →
                  (∀ (component : Component),
                      ∀ ε > 0,
                        ∀ᶠ (k : ℕ) in Filter.atTop, ∀ (x : Ω), |-Real.log (kernel k x) / ↑k - phi component x| ≤ ε) →
                    ∃ (minComponent : Component),
                      (∀ (component : Component), rate minComponent ≤ rate component) ∧
                        AppliedModelingLib.Probability.HasExponentialRate
                          (fun (k : ℕ) => ∫ (x : Ω) in P.support, weight x * kernel k x ∂μ) (rate minComponent)
```

Retained governing declarations

- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- AppliedModelingLib.Probability.HasAEEssentialInfimum
- AppliedModelingLib.Probability.HasExponentialRate
- AppliedModelingLib.Probability.HasPositiveWeightNearAEEssentialInfimum

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: MeasurableSpace Ω
4. parameter: Fintype Component
5. parameter: DecidableEq Component
6. assumption: Nonempty Component
7. parameter: MeasureTheory.Measure Ω
8. assumption: MeasureTheory.IsFiniteMeasure μ
9. parameter: AppliedModelingLib.Probability.FiniteMeasurableSetPartition μ Component
10. parameter: Ω → ℝ
11. parameter: ℕ → Ω → ℝ
12. parameter: Component → Ω → ℝ
13. parameter: Component → ℝ
14. parameter: Component → ℝ
15. assumption: ∀ (k : ℕ) (component : Component),
  MeasureTheory.IntegrableOn (fun (x : Ω) => weight x * kernel k x) (P.pieceSet component) μ
16. assumption: ∀ (component : Component), MeasureTheory.IntegrableOn weight (P.pieceSet component) μ
17. assumption: ∀ (component : Component), 0 < W component
18. assumption: ∀ (component : Component), ∀ᵐ (x : Ω) ∂μ.restrict (P.pieceSet component), 0 ≤ weight x
19. assumption: ∀ (component : Component), ∀ᵐ (x : Ω) ∂μ.restrict (P.pieceSet component), weight x ≤ W component
20. assumption: ∀ (component : Component),
  AppliedModelingLib.Probability.HasAEEssentialInfimum (μ.restrict (P.pieceSet component)) (phi component)
    (rate component)
21. assumption: ∀ (component : Component),
  AppliedModelingLib.Probability.HasPositiveWeightNearAEEssentialInfimum (μ.restrict (P.pieceSet component)) weight
    (phi component) (rate component)
22. assumption: ∀ (k : ℕ) (x : Ω), 0 < kernel k x
23. assumption: ∀ (component : Component),
  ∀ ε > 0, ∀ᶠ (k : ℕ) in Filter.atTop, ∀ (x : Ω), |-Real.log (kernel k x) / ↑k - phi component x| ≤ ε
24. conclusion: ∃ (minComponent : Component),
  (∀ (component : Component), rate minComponent ≤ rate component) ∧
    AppliedModelingLib.Probability.HasExponentialRate (fun (k : ℕ) => ∫ (x : Ω) in P.support, weight x * kernel k x ∂μ)
      (rate minComponent)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaC3_partition_integral_hasExponentialRate_of_min_component_uniform_logRate

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 19

Source locator: cited publication, lines 2366-2399

Verbatim source input

Proof. =⇒ follows directly from Lemma C.3: inf a∈R {gi+1 I(a|ti+1) + gi I(a|ti)} > 0 when ti 6= ti+1 ,
which holds when β is piece-wise constant with the appropriate number of levels.
⇐= Consider β that is not piece-wise constant. Recall that we further assume that β is nondecreasing, and discontinuous only on a measure 0 set.
↪ Following algebra steps similar to those in
Lemma C.3, but for general β:
Z
1
1
log W k = − lim log
w(θ1 , θ2)P k (θ1 , θ2 |β)d(θ1 , θ2)
k→∞ k
k→∞ k
θ1 >θ2
[U+0012 control character]
[U+0013 control character]
1
= inf − lim log P k (θj , θi |β)
θ1 >θ2
k→∞ k
=0

− lim

(16)
(17)
(18)

Where the last line follows from β continuous at some θ1 , and so limθ2 →θ1 P k (θ1 , θ2 |β) = 1.
Intuitively, what goes wrong with continuous β is that P k (θ1 , θ2 |β) does not converge uniformly:
∀[U+000F control character], k, ∃θ2 6= θ1 P k (θ1 , θ2) > [U+000F control character]
14

[form-feed in source extraction]
i.e. close by items are very hard to distinguish from one another. Then, because the large deviations
rate of W k is dominated by the worst rates under the integral, we don’t get a positive rate.

[Next verbatim source excerpt]

Lemma C.4. β(θ) is piecewise constant ⇐⇒ ∃c(β) > 0 s.t. − limk→∞ k1 log(W k) = c(β).

Expanded PaperInterface specification

```
∀ (μ : MeasureTheory.Measure ℝ) [MeasureTheory.SFinite μ] [MeasureTheory.IsFiniteMeasure (μ.prod μ)]
  [(μ.prod μ).IsOpenPosMeasure] {m : ℕ},
  0 < m →
    ∀ (cut : ℕ → ℝ) (hmono : Monotone cut),
      (∀ i < m + 2, cut i < cut (i + 1)) →
        ∀ (levels sampleRate : Fin (m + 2) → ℝ)
          (hlevels : GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels),
          (∀ (idx : Fin (m + 2)), 0 < sampleRate idx) →
            (∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b) →
              ∀ (weight : ℝ × ℝ → ℝ),
                (∀
                   (component :
                     { piece : Fin (m + 2) × Fin (m + 2) //
                       GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece }),
                   MeasureTheory.IntegrableOn weight
                     ((GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
                           GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).pieceSet
                       component)
                     (μ.prod μ)) →
                 (∀
                    (component :
                      { piece : Fin (m + 2) × Fin (m + 2) //
                        GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece }),
                    ∀ᵐ (x :
                      ℝ ×
                        ℝ) ∂(μ.prod μ).restrict
                        ((GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
                              GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).pieceSet
                          component),
                      0 ≤ weight x) →
                   Continuous weight →
                    (∀ (component : GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent m),
                       0 <
                         weight
                           (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_component_midpoint cut
                             component)) →
                     ∀ (lo hi : ℝ)
                       (R : GJ19OptimalBinaryRatingSystems.assumption_lemmaC4_positive_support_interval_model μ),
                       ¬GJ19OptimalBinaryRatingSystems.lemmaC4FiniteStepOnIoo R.successProb R.lo R.hi →
                         (GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo
                               (GJ19OptimalBinaryRatingSystems.cutpointStepSuccessProb cut levels) lo hi ∧
                           ∃ (rate : ℝ),
```

```
    0 < rate ∧
      AppliedModelingLib.Probability.ExponentialRateCertificate
        (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ
          (GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceWeight weight)
          (GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceKernel μ cut
            hmono sampleRate levels hlevels))
        rate) ∧
  AppliedModelingLib.Probability.HasExponentialRate
    (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ R.weight
      (GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel R))
    0 ∧
   ∀ (rate : ℝ),
    0 < rate →
      ¬AppliedModelingLib.Probability.ExponentialRateCertificate
        (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ R.weight
          (GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel R))
        rate
```

Retained governing declarations

- AppliedModelingLib.Probability.ExponentialRateCertificate
- AppliedModelingLib.Probability.FiniteMeasurableSetPartition
- AppliedModelingLib.Probability.HasExponentialRate
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.LemmaC4RawSourcePositiveSupportIntervalModel
- GJ19OptimalBinaryRatingSystems.assumption_lemmaC4_positive_support_interval_model
- GJ19OptimalBinaryRatingSystems.cutpointStepSuccessProb
- GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo
- GJ19OptimalBinaryRatingSystems.lemmaC4FiniteStepOnIoo
- GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel
- GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected
- GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected_decidablePred
- GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceKernel
- GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceWeight
- GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_component_midpoint
- GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition

Lean-elaborated claim roles

```
1. parameter: MeasureTheory.Measure ℝ
2. assumption: MeasureTheory.SFinite μ
3. assumption: MeasureTheory.IsFiniteMeasure (μ.prod μ)
4. assumption: (μ.prod μ).IsOpenPosMeasure
5. parameter: ℕ
6. assumption: 0 < m
7. parameter: ℕ → ℝ
8. assumption: Monotone cut
9. assumption: ∀ i < m + 2, cut i < cut (i + 1)
10. parameter: Fin (m + 2) → ℝ
11. parameter: Fin (m + 2) → ℝ
12. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels
13. assumption: ∀ (idx : Fin (m + 2)), 0 < sampleRate idx
14. assumption: ∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b
15. parameter: ℝ × ℝ → ℝ
16. assumption: ∀
  (component :
    { piece : Fin (m + 2) × Fin (m + 2) //
      GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece }),
  MeasureTheory.IntegrableOn weight
    ((GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
        GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).pieceSet
      component)
    (μ.prod μ)
17. assumption: ∀
  (component :
    { piece : Fin (m + 2) × Fin (m + 2) //
      GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected piece }),
  ∀ᵐ (x :
    ℝ ×
      ℝ) ∂(μ.prod μ).restrict
```

```
    ((GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_partition μ (m + 2) cut hmono
        GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairSelected).pieceSet
      component),
  0 ≤ weight x
18. assumption: Continuous weight
19. assumption: ∀ (component : GJ19OptimalBinaryRatingSystems.theorem31OrderedNontrivialPairComponent m),
  0 < weight (GJ19OptimalBinaryRatingSystems.theorem31_ordered_quality_pair_component_midpoint cut component)
20. parameter: ℝ
21. parameter: ℝ
22. parameter: GJ19OptimalBinaryRatingSystems.assumption_lemmaC4_positive_support_interval_model μ
23. assumption: ¬GJ19OptimalBinaryRatingSystems.lemmaC4FiniteStepOnIoo R.successProb R.lo R.hi
24. conclusion: (GJ19OptimalBinaryRatingSystems.lemmaC4FiniteRangeOnIoo
    (GJ19OptimalBinaryRatingSystems.cutpointStepSuccessProb cut levels) lo hi ∧
  ∃ (rate : ℝ),
    0 < rate ∧
      AppliedModelingLib.Probability.ExponentialRateCertificate
        (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ
          (GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceWeight weight)
          (GJ19OptimalBinaryRatingSystems.theorem31SelectedPullbackSourceKernel μ cut hmono sampleRate levels
            hlevels))
        rate) ∧
AppliedModelingLib.Probability.HasExponentialRate
    (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ R.weight
      (GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel R))
    0 ∧
  ∀ (rate : ℝ),
    0 < rate →
      ¬AppliedModelingLib.Probability.ExponentialRateCertificate
        (GJ19OptimalBinaryRatingSystems.lemmaC4TieErasedSourceWbar μ R.weight
          (GJ19OptimalBinaryRatingSystems.lemmaC4RawSourcePbarKernel R))
        rate
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaC4_finite_selected_pullback_positive_rate_and_nonfiniteStep_source_zero_rate

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 20

Source locator: cited publication, lines 2715-2895

Verbatim source input

i=0[U+0001 control character] . Then β2M −1
2M −2
1
0
0
0
has values {ti }i=0 , where t2i = ti , ∀i ∈ {0 . . . M − 1}, t1 = 2 1 − 1 − t1 and t02M −3 =
√
1
2 (1 + tM −2).

Proof. We first set the values t02i = ti and then optimally choose the remaining values t0k , k odd. Then, we show that the resulting large deviation rates between all adjacent pairs are equal. Then, by the proof of Lemma 3.1, which showed that equalizing the rates between adjacent intervals is a
−2
sufficient condition for optimality, β2M −1 has the levels {t0i }2M
i=0 .
Let r0 denote rates between adjacent t0 as r does for t. Supposing t02 = t1 , we find t01 such that
0
r1 = r20 and t01 < t02 .
[U+0014 control character]q
[U+0015 control character]
q
0
0
0
0
0
− log(1 − t1) = −2 log
(1 − t1)(1 − t2) + t1 t2
q
=⇒ 1 − t01 = (1 − t01)(1 − t02) + t01 t02 + 2 (1 − t01)(1 − t02)t01 t02
[U+0012 control character]
[U+0013 control character]
q
[U+0001 control character]
√
1
1
0
0
=⇒ t1 =
1 − 1 − t2 =
1 − 1 − t1
2
2
1
0
0
0
Similarly, r2M
−3 = r2M −2 when t2M −3 = 2 (1 +
by choosing such t01 , t02M −3 .

√

0
0
tM −2). It follows that r10 = r20 = r2M
−3 = r2M −2

17

[form-feed in source extraction]
0
Next, we find t0k ∈ (t0k−1 , t0k+1) for k ∈ {3, 5, . . . 2M − 5} such that the rates rk0 = rk+1
.

q
q
hq
i
hq
i
0
0
0
0
0
0
(1 − tk)(1 − tk−1) + tk tk−1 = −2 log
(1 − tk)(1 − tk+1) + t0k t0k+1
−2 log
q
[U+F8F9 private-use glyph]2
[U+F8EE private-use glyph]q
0
0

1
−
t
−
1
−
t
k+1
k−1
c
[U+F8FB private-use glyph]
q
q
=⇒ t0k =
, where c = [U+F8F0 private-use glyph]
1+c
0
0
− t
t
k−1

k+1

Now, we show that rk0 = rj0 , ∀j, k by showing that the difference between each rate ri and its
0 is constant. r = r , ∀j, k by assumption and so r 0 = r 0 , ∀j, k follows.
analogous rate r2i
j
k
j
k
√
1
0
(1
+
t
).
Thus
if
r
=
− log x for some x, then
rM −1 = − log tM −2 and r2M
=
−
log
i
M
−2
−2
2
√
r2i = − log 21 (1 + x) would imply that all the rates are equal. Thus, it is sufficient to show that
q
i2 1 h
i
hq
p
p
(1 − t02i−1)(1 − t02i) + t02i−1 t02i =
1 + (1 − ti−1)(1 − ti) + ti−1 ti
2
"s[U+0012 control character]
#2
r
[U+0013 control character]
i
p
p
c
c
1h
≡
1−
1 + (1 − ti−1)(1 − ti) + ti−1 ti
(1 − ti) +
ti =
1+c
1+c
2
[U+0015 control character]2
[U+0014 control character]√
√
1 − ti − 1 − ti−1
√
where c =
√
ti−1 − ti

(24)
(25)

The proof for (25) is algebraically tedious and is shown in Remark C.3 below.

[Next verbatim source excerpt]

Lemma C.5. Suppose g uniform, i.e. gi = 1, ∀i and that βM has values√{ti }M
i=0[U+0001 control character] . Then β2M −1
2M −2
1
0
0
0
has values {ti }i=0 , where t2i = ti , ∀i ∈ {0 . . . M − 1}, t1 = 2 1 − 1 − t1 and t02M −3 =
√
1
2 (1 + tM −2).

Expanded PaperInterface specification

```
∀ {m : ℕ},
  0 < m →
    ∀ {oldLevels : Fin (m + 2) → ℝ},
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector oldLevels →
        (GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize oldLevels fun (x : Fin (m + 2)) => 1) →
          GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
              (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels) ∧
            (GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
               (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels) fun (x : Fin (2 * m + 1 + 2)) =>
               1) ∧
             AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
               (fun (xs : Fin (2 * m + 1 + 2) → ℝ) =>
                 GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs
                   fun (x : Fin (2 * m + 1 + 2)) => 1)
               (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels)
```

Retained governing declarations

- AppliedModelingLib.Optimization.IsMaximizerOn
- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
- GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: 0 < m
3. parameter: Fin (m + 2) → ℝ
4. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector oldLevels
5. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize oldLevels fun (x : Fin (m + 2)) => 1
6. conclusion: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
   (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels) ∧
  (GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
     (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels) fun (x : Fin (2 * m + 1 + 2)) => 1) ∧
   AppliedModelingLib.Optimization.IsMaximizerOn GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
     (fun (xs : Fin (2 * m + 1 + 2) → ℝ) =>
       GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective xs fun (x : Fin (2 * m + 1 + 2)) => 1)
     (GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_uniform_doubled_chain

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 21

Source locator: cited publication, lines 2884-3038

Verbatim source input

[U+0014 control character]√
√
1 − ti − 1 − ti−1
√
where c =
√
ti−1 − ti

(24)
(25)

The proof for (25) is algebraically tedious and is shown in Remark C.3 below.
Then, by the proof of Lemma 3.1, which shows that equalizing the rates inside the minimization
−2
terms implies an optimal {ti }, β2M −1 has the levels {t0i }2M
i=0 .
Remark C.3.
"s[U+0012 control character]

c
1−
1+c

r

[U+0013 control character]
(1 − ti) +

#2

i
p
p
1h
1 + (1 − ti−1)(1 − ti) + ti−1 ti
2
[U+0015 control character]2
[U+0014 control character]√
√
1 − ti − 1 − ti−1
√
where c =
√
ti−1 − ti

c
ti
1+c

=

√
√
√
√
Proof. Let x = ti , y = 1 − ti , z = ti−1 , and w = 1 − ti−1 . Note that x > z, w > y, y =
1 − x2 , w = 1 − z 2 . Then,
c
(y − w)2
1
(x − z)2
=
, and
=
c+1
2 − 2xz − 2yw
c+1
2 − 2xz − 2yw
1
(To show the above two equalities, factor out (x−z)
2 from numerator and denominator, and substi2
2
tute y = 1 − x , w = 1 − z).

18

[form-feed in source extraction]
Now, the left hand side:
"s[U+0012 control character]

c
1−
1+c

#2
c
(1 − ti) +

ti
1+c
i2
hp
p
(x − z)2 y 2 + (y − w)2 x2
r

[U+0013 control character]

1
2 − 2xz − 2yw
(x − z)2 y 2 + (y − w)2 x2 + 2xy(x − z)(w − y)
=
2 − 2xz − 2yw
2
2
2
2
z y + w x − 2wxyz
=
2 − 2xz − 2yw
=

p
p
(y − w)2 = w − y, (x − z)2 = x − z

The right hand side:
i
p
p
1h
1 + (1 − ti−1)(1 − ti) + ti−1 ti
2
1
= [1 + (wy + xz)]
2
Multiplying both sides by 2 − 2xz − 2yw, we have:
"s[U+0012 control character]

c
1−
1+c

r

[U+0013 control character]
(1 − ti) +

c
ti
1+c

#2
=

i
p
p
1h
1 + (1 − ti−1)(1 − ti) + ti−1 ti
2

≡ z 2 y 2 + w2 x2 − 2wxyz = 1 − (wy + xz)2
≡ z 2 (1 − x2) + (1 − z 2)x2 − 2wxyz = 1 − w2 y 2 − x2 z 2 − 2wxyz
≡ z 2 − 2x2 z 2 + x2 = 1 − (1 − z 2)(1 − x2) − x2 z 2
≡0=0

Expanded PaperInterface specification

```
(∀ {m k : ℕ},
   0 < k →
     ∀ (hkm : k < m) (oldLevels : Fin (m + 2) → ℝ),
       GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels
           ⟨2 * k + 1,
             GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_4 hkm⟩ =
         AppliedModelingLib.Probability.bernoulliInteriorEqualSplit
           (oldLevels
             ⟨k,
               GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_5
                 hkm⟩)
           (oldLevels
             ⟨k + 1,
               GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_6
                 hkm⟩)) ∧
 ∀ {pLo pHi : ℝ},
   0 ≤ pLo →
     pHi ≤ 1 →
       pLo < pHi →
         AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate 1 1
             (AppliedModelingLib.Probability.bernoulliInteriorEqualSplit pLo pHi) pLo =
           AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate 1 1 pHi
             (AppliedModelingLib.Probability.bernoulliInteriorEqualSplit pLo pHi)
```

Retained governing declarations

- AppliedModelingLib.Probability.bernoulliInteriorEqualSplit
- AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate
- GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels

Lean-elaborated claim roles

```
1. conclusion: (∀ {m k : ℕ},
    0 < k →
      ∀ (hkm : k < m) (oldLevels : Fin (m + 2) → ℝ),
        GJ19OptimalBinaryRatingSystems.uniformDoubledEndpointLevels oldLevels
            ⟨2 * k + 1,
              GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_4 hkm⟩ =
          AppliedModelingLib.Probability.bernoulliInteriorEqualSplit
            (oldLevels
              ⟨k,
                GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_5
                  hkm⟩)
            (oldLevels
              ⟨k + 1,
                GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25Spec._proof_6
                  hkm⟩)) ∧
  ∀ {pLo pHi : ℝ},
    0 ≤ pLo →
      pHi ≤ 1 →
        pLo < pHi →
          AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate 1 1
              (AppliedModelingLib.Probability.bernoulliInteriorEqualSplit pLo pHi) pLo =
            AppliedModelingLib.Probability.weightedBernoulliClosedThresholdRate 1 1 pHi
              (AppliedModelingLib.Probability.bernoulliInteriorEqualSplit pLo pHi)
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC5_refinement_equations24_25_realizes_spec

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 22

Source locator: cited publication, lines 3041-3042

Verbatim source input

Corollary C.1. Suppose g uniform, i.e. gi = 1, ∀i. ∀[U+000F control character] > 0, ∃M s.t. ∀M 0 ≥ M , rM < [U+000F control character].
0

Expanded PaperInterface specification

```
∀ (levels : (N : ℕ) → Fin (N + 1 + 2) → ℝ),
  (∀ (N : ℕ), GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels N)) →
    (∀ (N : ℕ),
       GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels N) fun (x : Fin (N + 1 + 2)) =>
         1) →
      Filter.Tendsto
        (fun (N : ℕ) =>
          GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate (levels N) (fun (x : Fin (N + 1 + 2)) => 1)
            GJ19OptimalBinaryRatingSystems.lastAdjacentIndex)
        Filter.atTop (nhds 0)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

Lean-elaborated claim roles

1. parameter: (N : ℕ) → Fin (N + 1 + 2) → ℝ
2. assumption: ∀ (N : ℕ), GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels N)
3. assumption: ∀ (N : ℕ),
   GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels N) fun (x : Fin (N + 1 + 2)) => 1
4. parameter: Set ℝ
5. assumption: U ∈ nhds 0
6. conclusion: U ∈
   Filter.map
     (fun (N : ℕ) =>
       GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRate (levels N) (fun (x : Fin (N + 1 + 2)) => 1)
         GJ19OptimalBinaryRatingSystems.lastAdjacentIndex)
     Filter.atTop

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_corollaryC2_uniform_equalized_last_rate_tendsto_zero

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 23

Source locator: cited publication, lines 3094-3125

Verbatim source input

k − tk−1 < δ.

Proof. This corollary follows directly from Corollary C.1. If the rates are upper bounded, then so
are the level differences.
We first find where the rate is minimized given a width between levels of δ
hp
i
p
xm = arg min −2 log
(1 − x − δ)(1 − x) + x(x + δ)
x

1 1
= − δ
2 2
Then given an upper bound of [U+000F control character] on the rate, there is a bound on δ determined by the largest
possible difference at levels symmetric around 21 .
" r

1
1
r = −2 log 2 (− δ)(+ δ)
2
2
[U+0002 control character]
[U+0003 control character]
= − log 1 − 4δ 2
1p
≥ [U+000F control character] when δ >
1 − e−[U+000F control character]
2

[Next verbatim source excerpt]

Corollary C.2. Suppose g uniform, i.e. gi = 1, ∀i. ∀δ > 0, ∃N s.t. ∀M ≥ N , maxk tM
k − tk−1 < δ.

Expanded PaperInterface specification

```
∀ (levels : (N : ℕ) → Fin (N + 1 + 2) → ℝ),
  (∀ (N : ℕ), GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels N)) →
    (∀ (N : ℕ),
       GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels N) fun (x : Fin (N + 1 + 2)) =>
         1) →
      Filter.Tendsto (fun (N : ℕ) => GJ19OptimalBinaryRatingSystems.binaryEndpointAdjacentMaxWidth (levels N))
        Filter.atTop (nhds 0)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAdjacentMaxWidth

Lean-elaborated claim roles

1. parameter: (N : ℕ) → Fin (N + 1 + 2) → ℝ
2. assumption: ∀ (N : ℕ), GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels N)
3. assumption: ∀ (N : ℕ),
   GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels N) fun (x : Fin (N + 1 + 2)) => 1
4. parameter: Set ℝ
5. assumption: U ∈ nhds 0
6. conclusion: U ∈ Filter.map (fun (N : ℕ) => GJ19OptimalBinaryRatingSystems.binaryEndpointAdjacentMaxWidth (levels N)) Filter.atTop

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_corollaryC2_uniform_equalized_adjacent_mesh_tendsto_zero

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 24

Source locator: cited publication, lines 3131-3159

Verbatim source input

Proof. Note that, with uniform matching, ∀x ∈ (0, 1], y ∈ [0, 1 − x] the rate with values ti−1 =
y, ti = y + x is no more than the last with tM −2 = 1 − x. With width x, in other words, the extreme
points have a larger rates than the middle points. For i ∈
/ {1, M − 1}:
ri = inf {gi−1 KL(a||ti−1) + gi KL(a||ti)}
a

= inf {KL(a||y) + KL(a||y + x)}
uniform matching
a
h
i
1
1
1
1
= −2 log (1 − y) 2 (1 − y − x) 2 + y 2 (y + x) 2
(26)
h
i
= − log (1 − y)(1 − y − x) + y(y + x) + 2 [(1 − y)(1 − y − x)y(y + x)]1/2
≤ − log(1 − x)
Where line (26) follows from line (22).
By the proof of Lemma 3.1, the optimal levels equalize the rates between each level. Then,
when g is non-decreasing, gM −2 ≥ g` , ∀` ∈ {1 . . . M − 3}. Then, at the same level differences, the
rate corresponding to the last level is no smaller. Thus, to equalize the rates, the last width must
be no larger than any other width. Thus, tM −2 ≥ 1 − L1 .
N +1 −1

[Next verbatim source excerpt]

Lemma C.6. Suppose g is non-decreasing in θ. Then, tM −2 ≥ 1 − M1−1 .

Expanded PaperInterface specification

```
∀ {m : ℕ},
  0 < m →
    ∀ {levels sampleRate : Fin (m + 2) → ℝ},
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels →
        GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate →
          (∀ (idx : Fin (m + 2)), 0 < sampleRate idx) →
            (∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b) →
              1 - 1 / ↑(m + 1) ≤
                levels
                  (GJ19OptimalBinaryRatingSystems.adjacentLowIndex GJ19OptimalBinaryRatingSystems.lastAdjacentIndex)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.adjacentLowIndex
- GJ19OptimalBinaryRatingSystems.lastAdjacentIndex

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: 0 < m
3. parameter: Fin (m + 2) → ℝ
4. parameter: Fin (m + 2) → ℝ
5. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector levels
6. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize levels sampleRate
7. assumption: ∀ (idx : Fin (m + 2)), 0 < sampleRate idx
8. assumption: ∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate a ≤ sampleRate b
9. conclusion: Real.le↑ (1 - 1 / ↑(m + 1))
  (levels (GJ19OptimalBinaryRatingSystems.adjacentLowIndex GJ19OptimalBinaryRatingSystems.lastAdjacentIndex))

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaC6_monotone_matching_penultimate_level_bound

Recorded source-to-Spec screening **Verdict:** matches
Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 25

Source locator: cited publication, lines 3160-3168

Verbatim source input

Lemma C.7. With uniform matching (gi = 1), r2

20

N

≥ 15 r2 .

[form-feed in source extraction]
1

Expanded PaperInterface specification

```
∀ {m : ℕ},
  0 < m →
    ∀ {oldLevels : Fin (m + 2) → ℝ},
      GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector oldLevels →
        (GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize oldLevels fun (x : Fin (m + 2)) => 1) →
          (1 / 5 *
              GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective oldLevels fun (x : Fin (m + 2)) =>
                1) ≤
            GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
              (GJ19OptimalBinaryRatingSystems.lemmaC5_uniform_doubled_endpoint_levels oldLevels)
              fun (x : Fin (2 * m + 1 + 2)) => 1
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
- GJ19OptimalBinaryRatingSystems.lemmaC5_uniform_doubled_endpoint_levels

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: 0 < m
3. parameter: Fin (m + 2) → ℝ
4. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector oldLevels
5. assumption: GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize oldLevels fun (x : Fin (m + 2)) => 1
6. conclusion: Real.le✝
   (1 / 5 * GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective oldLevels fun (x : Fin (m + 2)) => 1)
   (GJ19OptimalBinaryRatingSystems.binaryEndpointAwareAdjacentRateObjective
     (GJ19OptimalBinaryRatingSystems.lemmaC5_uniform_doubled_endpoint_levels oldLevels) fun (x : Fin (2 * m + 1 + 2)) =>
     1)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.lemmaC7_uniform_doubled_objective_rate_ge_one_fifth_old_objective

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 26

Source locator: cited publication, lines 3234-3267

Verbatim source input

1 ≥ CM

Proof. By Lemma C.7, ∃C2 > 0 s.t. rM ≥ C2 5−dlog2 M e . Then
M
− log(1 − tM
1)=r

≥ C2 5−dlog2 M e
h
i
−dlog2 M e
=⇒ tM
1 ≥ 1 − exp −C2 5
h
i
− 1
≥ 1 − exp −C3 M log5 2
e−1
− 1
C3 M log5 2
e
M
=⇒ ∃C > 0 s.t. t1 ≥ CM −3

e−x ≤ 1 −

≥

e−1
x for x ∈ [0, 1]
e

−3 .
Corollary C.3. With monotonically non-decreasing g, ∃C > 0 s.t. ∀M, tM

[Next verbatim source excerpt]

Lemma C.8. With uniform matching (gi = 1), ∃C > 0 s.t. ∀M, tM
1 ≥ CM

Expanded PaperInterface specification

```
∀ (levels : (m : ℕ) → Fin (m + 2) → ℝ),
  (∀ (m : ℕ), 0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels m)) →
    (∀ (m : ℕ),
       0 < m →
         GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels m) fun (x : Fin (m + 2)) =>
           1) →
      ∃ (C : ℝ),
        0 < C ∧
          ∀ (m : ℕ),
            0 < m →
              C / ↑(m + 2) ^ 3 ≤
                levels m
                  (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.firstAdjacentIndex

Lean-elaborated claim roles

```
1. parameter: (m : ℕ) → Fin (m + 2) → ℝ
2. assumption: ∀ (m : ℕ), 0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels m)
3. assumption: ∀ (m : ℕ),
  0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels m) fun (x : Fin (m + 2)) => 1
4. conclusion: ∃ (C : ℝ),
  0 < C ∧
    ∀ (m : ℕ),
      0 < m →
        C / ↑(m + 2) ^ 3 ≤
          levels m (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex)
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC8_uniform_first_level_polynomial_lower_bound

Recorded source-to-Spec screening **Verdict:** matches
Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 27

Source locator: cited publication, lines 3268-3274

Verbatim source input

1 ≥ CM

Proof. The result follows from noting that tM
1 with uniform matching lower bounds the first value
with any other monotonically non-decreasing g, which is a direct application of Lemma B.1 – scale
g such that g1 = 1. Then, gj ≥ 1, j > 1 and g0 ≤ 1. Then, the condition of the lemma holds.
Lemma C.9. The run-time of NestedBisection is O(M log2 1δ), where δ is the bisection grid width

[Next verbatim source excerpt]

Corollary C.3. With monotonically non-decreasing g, ∃C > 0 s.t. ∀M, tM
1 ≥ CM

Expanded PaperInterface specification

```
∀ (levels sampleRate : (m : ℕ) → Fin (m + 2) → ℝ),
  (∀ (m : ℕ), 0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels m)) →
    (∀ (m : ℕ),
       0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels m) (sampleRate m)) →
      (∀ (m : ℕ), 0 < m → ∀ (idx : Fin (m + 2)), 0 < sampleRate m idx) →
        (∀ (m : ℕ), 0 < m → ∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate m a ≤ sampleRate m b) →
          (∀ (m : ℕ),
             0 < m →
               sampleRate m
                   (GJ19OptimalBinaryRatingSystems.adjacentHighIndex
                     GJ19OptimalBinaryRatingSystems.firstAdjacentIndex) =
                 1) →
            ∃ (C : ℝ),
              0 < C ∧
                ∀ (m : ℕ),
                  0 < m →
                    C / ↑(m + 2) ^ 3 ≤
                      levels m
                        (GJ19OptimalBinaryRatingSystems.adjacentHighIndex
                          GJ19OptimalBinaryRatingSystems.firstAdjacentIndex)
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize
- GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector
- GJ19OptimalBinaryRatingSystems.adjacentHighIndex
- GJ19OptimalBinaryRatingSystems.firstAdjacentIndex

Lean-elaborated claim roles

```
1. parameter: (m : ℕ) → Fin (m + 2) → ℝ
2. parameter: (m : ℕ) → Fin (m + 2) → ℝ
3. assumption: ∀ (m : ℕ), 0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointLevelVector (levels m)
4. assumption: ∀ (m : ℕ), 0 < m → GJ19OptimalBinaryRatingSystems.BinaryEndpointAwareAdjacentRatesEqualize (levels m) (sampleRate m)
5. assumption: ∀ (m : ℕ), 0 < m → ∀ (idx : Fin (m + 2)), 0 < sampleRate m idx
6. assumption: ∀ (m : ℕ), 0 < m → ∀ {a b : Fin (m + 2)}, ↑a ≤ ↑b → sampleRate m a ≤ sampleRate m b
7. assumption: ∀ (m : ℕ),
  0 < m →
    sampleRate m (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex) =
      1
8. conclusion: ∃ (C : ℝ),
  0 < C ∧
    ∀ (m : ℕ),
      0 < m →
        C / ↑(m + 2) ^ 3 ≤
          levels m (GJ19OptimalBinaryRatingSystems.adjacentHighIndex GJ19OptimalBinaryRatingSystems.firstAdjacentIndex)
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.corollaryC3_monotone_scaled_first_level_ge_half_inv_adjacent_count_sq

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 28

Source locator: cited publication, lines 3275-3279

Verbatim source input

and M is the number of intervals.
Proof. The outer bisection, in main, runs at most log2 2δ + 1 iterations. Each outer iteration calls
BisectNextLevel M − 3 times, and the inner bisection in each call runs for at most log2 2δ iterations.
Thus the run-time of algorithm is O(M log2 1δ).

[Next verbatim source excerpt]

Lemma C.9. The run-time of NestedBisection is O(M log2 1δ), where δ is the bisection grid width
and M is the number of intervals.

Expanded PaperInterface specification

```
∀ {M : ℕ},
  0 < M →
    ∀ {delta : ℝ},
      0 < delta →
        ∃ (L : ℕ),
          1 ≤ delta * 2 ^ L ∧
            ↑(L + 1) ≤ Real.logb 2 (max 1 (1 / delta)) + 2 ∧
              ↑(GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount M (L + 1) L) ≤
                ↑M * (Real.logb 2 (max 1 (1 / delta)) + 2) ^ 2
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: 0 < M
3. parameter: ℝ
4. assumption: 0 < delta
5. conclusion: ∃ (L : ℕ),
  1 ≤ delta * 2 ^ L ∧
    ↑(L + 1) ≤ Real.logb 2 (max 1 (1 / delta)) + 2 ∧
      ↑(GJ19OptimalBinaryRatingSystems.nestedBisectionOperationCount M (L + 1) L) ≤
        ↑M * (Real.logb 2 (max 1 (1 / delta)) + 2) ^ 2

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_lemmaC9_nested_bisection_runtime_log_squared

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 29

Source locator: cited publication, lines 3467-3589

Verbatim source input

Proof of Theorem B.1

w denote the optimal β with M intervals for weight function w, with intervals swM and levels
Let βM
twM . Let qwM (θ) = i/M when θ ∈ [swM
, swM
i
i+1), i.e. the quantile of interval item of type θ is in.
Then we have the following convergence result for βM .

Theorem B.1. Let g be uniform. Suppose w such that qwM converges uniformly. Then, ∀C ∈
w uniformly as N → ∞.
w
N, ∃β w s.t. βC2
N +1 → β
hProof. Note that
[U+0011 control character] the condition on q implies that ∃M s.t.
M
M
sbxθ M c , sdxθ M e .

∀M > M , ∀θ, ∃xθ such that θ ∈

h
[U+0011 control character]
M
Let M 0 = 2M − 1, M 00 = 4M − 3, M q = 2q M − 2q + 1. θ ∈ sM
,
s
=⇒ βM (θ) =
bxθ M c dxθ M e
h
i
M
M
tM
bxθ M c ∈ tbxθ M c−1 , tbxθ M c+1 . Then,
0

βM 0 (θ) = tM
bxθ M 0 c
0

= tM
bx (2M −1)c
h θ0
i
M0
∈ tM
,
t
2bxθ M c−2 2bxθ M c+2
i
h
M
⊂ tM
,
t
bxθ M c−1 bxθ M c+1

23

Lemma C.5

[form-feed in source extraction]
And, for general q,
q

βM q (θ) = tM
bx (2q M −2q +1)c
i
h θq
Mq
,
t
∈ tM
q
q
q
bxθ (2 M)c+1
bxθ 2 M c−2
h q
i
Mq
⊂ tM
,
t
q

q
q
2 bxθ M c−2
2 bxθ M c+1
h
i
M
⊂ tM
,
t
bxθ M c−1 bxθ M c+1

Lemma C.5

h
i
M
Then, ∀N 0 > 1, θ: β2N 0 M −2N 0 +1 (θ) ∈ tM
,
t
bxθ M c−1 bxθ M c+1 and
M
|β2N 0 M −2N 0 +1 (θ) − βM (θ)| ≤ tM
bxθ M c+1 − tbxθ M c−1
0

0

K
By Corollary C.2, ∀δ > 0, ∃K s.t. ∀K 0 > K, tK
bxθ K 0 c+1 − tbxθ K 0 c−1 < 2δ.
By the Cauchy criterion, ∃β s.t. β(C−1)2N +1 → β uniformly.
By change of variables, ∃β s.t. βC2N +1 → β uniformly.

Corollary C.4. For Kendall's tau and Spearman's rho correlation measures, ∃β s.t. β2N → β

Expanded PaperInterface specification

```
∀ {M q : ℕ},
  0 < M →
    ∀ {θ : ℝ},
      0 ≤ θ →
        θ ≤ 1 →
          have old : ℕ := ⌊↑M * θ⌋₊;
          have refined : ℕ := ⌊↑(GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate M q) * θ⌋₊;
          refined ≤ 2 ^ q * old + 2 ^ q ∧ 2 ^ q * old ≤ refined + 2 ^ q
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ
3. assumption: 0 < M
4. parameter: ℝ
5. assumption: 0 ≤ θ
6. assumption: θ ≤ 1
7. conclusion: ⌊↑(GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate M q) * θ⌋₊ ≤ 2 ^ q * ⌊↑M * θ⌋₊ + 2 ^ q ∧
   2 ^ q * ⌊↑M * θ⌋₊ ≤ ⌊↑(GJ19OptimalBinaryRatingSystems.theoremB1SourceDoubledIndexIterate M q) * θ⌋₊ + 2 ^ q

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_theoremB1_proof_selector_nesting

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 30

Source locator: cited publication, lines 3590-3593

Verbatim source input

uniformly as N → ∞.
Proof. For Kendall’s tau and Spearman’s rho, {si } is spaced such that ∀i, j, si − si−1 = sj − sj−1 .
Thus, xθ = θ meets the criterion.

[Next verbatim source excerpt]

Corollary C.4. For Kendall’s tau and Spearman’s rho correlation measures, ∃β s.t. β2N → β
uniformly as N → ∞.

Expanded PaperInterface specification

```
∀ (C : ℕ),
  (∀ (M : ℕ),
     Nonempty (Fin M) →
       0 < M →
         AppliedModelingLib.Optimization.IsLexicographicMaximizerOn
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignFeasible M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignValue M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign M)) ∧
    (∀ (M : ℕ),
       Nonempty (Fin M) →
         0 < M →
           AppliedModelingLib.Optimization.IsLexicographicMaximizerOn
             (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignFeasible M)
             (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignValue M)
             (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate M)
             (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign M)) ∧
     ∃ (betaLimit : ℝ → ℝ),
       TendstoUniformlyOn
         (fun (N : ℕ) (θ : ℝ) =>
           GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedClampedFloorBetaSeq
             (GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex C N) θ)
         betaLimit Filter.atTop (Set.Icc 0 1)
```

Retained governing declarations

- AppliedModelingLib.Optimization.IsLexicographicMaximizerOn
- GJ19OptimalBinaryRatingSystems.ProofBridge.Theorem31FiniteBinaryRatingDesign
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignFeasible
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignValue
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignFeasible
- GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignValue
- GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedClampedFloorBetaSeq
- GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex

Lean-elaborated claim roles

```
1. parameter: ℕ
2. conclusion: (∀ (M : ℕ),
    Nonempty (Fin M) →
      0 < M →
        AppliedModelingLib.Optimization.IsLexicographicMaximizerOn
          (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignFeasible M)
          (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31KendallFiniteDesignValue M)
          (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate M)
          (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign M)) ∧
  (∀ (M : ℕ),
     Nonempty (Fin M) →
       0 < M →
         AppliedModelingLib.Optimization.IsLexicographicMaximizerOn
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignFeasible M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31SpearmanFiniteDesignValue M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31FiniteDesignEndpointRate M)
           (GJ19OptimalBinaryRatingSystems.ProofBridge.theorem31CanonicalUniformEndpointDesign M)) ∧
   ∃ (betaLimit : ℝ → ℝ),
     TendstoUniformlyOn
       (fun (N : ℕ) (θ : ℝ) =>
         GJ19OptimalBinaryRatingSystems.canonicalUniformEqualizedClampedFloorBetaSeq
           (GJ19OptimalBinaryRatingSystems.theoremB1SubsequenceIndex C N) θ)
       betaLimit Filter.atTop (Set.Icc 0 1)
```

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.source_corollaryC4_kendall_spearman_subsequence

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 31

Source locator: cited publication, lines 3615-3697

Verbatim source input

i.e. with w(θ1 , θ2) = (θ1 − θ2).
Proof. Recall Pk (θ1 , θ3) =
P r((θ1 − θ2)(xk1 − xk3) > 0)
Z
Z
=
P r(xk1 − xk3 > 0)dθ1 dθ2 dθ3 +
θ1 >θ2 ,θ3

Z
=

P r(xk1 − xk3 > 0)

R

θ1 >θ2 (θ1 −θ2)Pk (θ1 , θ2)dθ1 dθ2 ,

P r(xk1 − xk3 < 0)dθ1 dθ2 dθ3

θ1 <θ2 ,θ3

[U+0014 control character]Z θ1

[U+0015 control character]

Z

P r(xk1 − xk3 < 0)

[U+0014 control character]Z 1

[U+0015 control character]

dθ2 dθ1 dθ3 +
dθ2 dθ1 dθ3
θ ,θ
θ2 =θ
θ1 ,θ3
θ2 =θ1
Z 1 3h
i
=
P r(xk1 − xk3 > 0)θ1] + P r(xk1 − xk3 < 0)(1 − θ1) dθ1 dθ3
θ ,θ
Z 1 3h
h
ii
=
P r(xk1 − xk3 < 0) + θ1 P r(xk1 − xk3 > 0) − P r(xk1 − xk3 < 0) dθ1 dθ3
θ1 ,θ3

24

[form-feed in source extraction]
Similarly,
P r((θ1 − θ2)(xk1 − xk3) < 0) =
Z
h
h
ii
=
P r(xk1 − xk3 > 0) + θ1 P r(xk1 − xk3 < 0) − P r(xk1 − xk3 > 0) dθ1 dθ3
θ ,θ
Z 1 3h
h
ii
=
P r(xk3 − xk1 > 0) + θ3 P r(xk3 − xk1 < 0) − P r(xk3 − xk1 > 0) dθ1 dθ3
θ1 ,θ3

Where the second equality follows from θ1 , θ3 interchangeable. Then
Z
Wkρ = 3
(θ1 − θ2)Pk (θ1 , θ2)dθ1 dθ2
θ1 ,θ2
Z
6(θ1 − θ2)Pk (θ1 , θ2)dθ1 dθ2
=
θ1 >θ2

Note that Spearman’s ρ is similar to Kendall’s τ with an additional weighting for how far apart the two values that are flipped are.
Lemma C.11. When w is constant, i.e. for Kendall’s τ rank correlation, the intervals s that

[Next verbatim source excerpt]

Lemma C.10. Spearman’s ρ can also be written as being proportional to
i.e. with w(θ1 , θ2) = (θ1 − θ2).

Expanded PaperInterface specification

```
∀ (M : ℕ) (s : ℕ → ℝ),
  s 0 = 0 →
    s M = 1 →
      GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective M s =
        GJ19OptimalBinaryRatingSystems.spearmanLinearWeightIntervalObjective M s
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.spearmanLinearWeightIntervalObjective
- GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective

Lean-elaborated claim roles

1. parameter: ℕ
2. parameter: ℕ → ℝ
3. assumption: s 0 = 0
4. assumption: s M = 1
5. conclusion: GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective M s =
  GJ19OptimalBinaryRatingSystems.spearmanLinearWeightIntervalObjective M s

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaC10_spearman_linear_weight_ordered_pair_interval_objective_eq_gap_objective

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 32

Source locator: cited publication, lines 3698-3712

Verbatim source input

maximize (20),
X
X Z
w(θ1 , θ2)d(θ1 , θ2) =
(si+1 − si)(sj+1 − sj)
(27)
0≤i<j<M

θ2 ∈Si ,θ1 ∈Sj

0≤i<j<M

, are {si = Mi }M
i=0 .
Lemma C.12. When w is (θ1 − θ2), i.e. for Spearman's ρ rank correlation, the intervals s that

[Next verbatim source excerpt]

Lemma C.11. When w is constant, i.e. for Kendall's τ rank correlation, the intervals s that
maximize (20),
X
X Z
w(θ1 , θ2)d(θ1 , θ2) =
(si+1 − si)(sj+1 − sj)
(27)
0≤i<j<M

θ2 ∈Si ,θ1 ∈Sj

0≤i<j<M

, are {si = Mi }M
i=0 .

Expanded PaperInterface specification

```
∀ {M : ℕ} [Nonempty (Fin M)] (s : ℕ → ℝ),
  s 0 = 0 →
    s M = 1 →
      (∑ i : Fin M, ∑ j : Fin M, if i < j then (s (↑i + 1) - s ↑i) * (s (↑j + 1) - s ↑j) else 0) ≤ (1 - (↑M)⁻¹) / 2
```

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: Nonempty (Fin M)
3. parameter: ℕ → ℝ
4. assumption: s 0 = 0
5. assumption: s M = 1
6. conclusion: Real.le† (∑ i : Fin M, ∑ j : Fin M, if i < j then (s (↑i + 1) - s ↑i) * (s (↑j + 1) - s ↑j) else 0) ((1 - (↑M)⁻¹) / 2)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaC11_kendall_constant_weight_ordered_pair_interval_objective_le_equispaced

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 33

Source locator: cited publication, lines 3713-3760

Verbatim source input

maximize (20),
X Z
w(θ1 , θ2)d(θ1 , θ2)
(28)
0≤i<j<M

θ2 ∈Si ,θ1 ∈Sj

are {si = Mi }M
i=0 , i.e. the same as those for Kendall's τ .
Proof.
X
0≤i<j<M

Z
w(θ1 , θ2)d(θ1 , θ2) =
θ2 ∈Si ,θ1 ∈Sj

Z

X

(θ1 − θ2)d(θ1 , θ2)

0≤i<j<M

=

θ2 ∈Si ,θ1 ∈Sj

[U+0012 control character]

X
0<i<j≤M

sj + sj−1 si + si−1
−
2
2

[U+0013 control character]
(si − si−1)(sj − sj−1)

Finding an asymptotically optimal {si } then is a constrained third order polynomial maximization
i=M , as for Kendall's tau
problem with M variables. The maximum is achieved at {si = Mi }i=0
correlation.

[Next verbatim source excerpt]

Lemma C.12. When w is (θ1 − θ2), i.e. for Spearman's ρ rank correlation, the intervals s that
maximize (20),
X Z
w(θ1 , θ2)d(θ1 , θ2)
(28)
0≤i<j<M

θ2 ∈Si ,θ1 ∈Sj

are {si = Mi }M
i=0 , i.e. the same as those for Kendall's τ .

Expanded PaperInterface specification

```
∀ {M : ℕ} [Nonempty (Fin M)] (s : ℕ → ℝ),
  Monotone s →
    s 0 = 0 →
      s M = 1 →
        GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective M s ≤ (1 - (↑M)⁻¹ ^ 2) / 6
```

Retained governing declarations

- GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective

Lean-elaborated claim roles

1. parameter: ℕ
2. assumption: Nonempty (Fin M)
3. parameter: ℕ → ℝ
4. assumption: Monotone s
5. assumption: s 0 = 0
6. assumption: s M = 1
7. conclusion: Real.le† (GJ19OptimalBinaryRatingSystems.spearmanLinearWeightOrderedPairIntervalObjective M s) ((1 - (↑M)⁻¹ ^ 2) / 6)

Lean proof endpoint (built separately)

GJ19OptimalBinaryRatingSystems.PaperInterface.paper_lemmaC12_spearman_linear_weight_ordered_pair_interval_objective_le_equispaced

Recorded source-to-Spec screening **Verdict:** matches

Reason: Adversarial comparison of the byte-pinned source bundle and complete Lean-expanded target found the same displayed definition or hypotheses and conclusion, with no unrecorded correction.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation