Human review packet: LG21TestOptionalPolicies

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

Paper: LG21TestOptionalPolicies
Paper id: LG21TestOptionalPolicies
Source version: EAAMO 2021 / arXiv:2107.08922
Source record: audit/paper_statement_map.json
Rows: 15
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 LG21TestOptionalPolicies --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 (174; supporting context)
- Semantic library prerequisites (1; not paper claims)
  - AppliedModelingLib.Probability.GaussianSignalFamily
- Paper-specific semantic prerequisites (15; not paper claims)
  - LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
  - LG21TestOptionalPolicies.LG21RequirementPolicy
  - LG21TestOptionalPolicies.LG21Definition1ExactEquilibrium
  - LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy
  - LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses
  - LG21TestOptionalPolicies.LG21RequirementPolicy.feasibleAction
  - LG21TestOptionalPolicies.PaperInterface.distributional_equality
  - LG21TestOptionalPolicies.PaperInterface.lg21GaussianSignalFamilyWithTest
  - LG21TestOptionalPolicies.PaperInterface.test_taking_probability_preference
  - LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel
  - LG21TestOptionalPolicies.lg21Definition6ExactObservedAccessEstimateLaw
  - LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
  - LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
  - LG21TestOptionalPolicies.lg21SourceLawObservablyFair
  - LG21TestOptionalPolicies.lg21SourceLawTestBlank
- Source claims (15)
  - fairness_implication_chainSpec
  - lemma4_1_observed_access_strategy_proofness_source_coreSpec
  - proposition4_2_all_observed_access_requirement_protocols_source_coreSpec
  - proposition4_3_each_requirement_protocol_has_unfair_pbo_equilibriumSpec
  - reporting_conditioned_generalizationSpec
  - theorem3_1_hidden_access_pbo_fails_all_fairness_definitionsSpec

- theorem3_1_optional_reporting_source_timedSpec
- theorem3_1_report_required_source_timedSpec
- theorem3_2_optional_reporting_clarified_modelSpec
- theorem3_2_report_required_clarified_modelSpec
- theorem4_4_all_observed_access_requirement_protocolsSpec
- thompson_acceptance_interpretationSpec
- report_decision_skill_independenceSpec
- ignoring_test_scores_achieves_fairnessSpec
- theorem3_2_observable_summary_clarifiedSpec

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.NoProfitableBinaryChoiceDeviation

Declaration location: reusable library

Lean declaration body

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

value:
```
fun {α : Type u_1} (chooses : α → Prop) (choosePayoff otherPayoff : α → ℝ) =>
  (∀ (a : α), chooses a → otherPayoff a ≤ choosePayoff a) ∧ ∀ (a : α), ¬chooses a → choosePayoff a ≤ otherPayoff a
```

AppliedModelingLib.Probability.gaussianLocationKernel

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} → [inst : MeasurableSpace α] → (mean : α → ℝ) → Measurable mean → NNReal → ProbabilityTheory.Kernel α ℝ
```

value:
```
fun {α : Type u_1} [MeasurableSpace α] (mean : α → ℝ) (hmean : Measurable mean) (variance : NNReal) =>
  ((ProbabilityTheory.Kernel.deterministic mean hmean).prod
      (ProbabilityTheory.Kernel.const α (ProbabilityTheory.gaussianReal 0 variance))).map
    fun (pair : ℝ × ℝ) => pair.1 + pair.2
```

AppliedModelingLib.Probability.gaussianSignalJointKernel

Declaration location: reusable library

Lean declaration body

type:
```
{α : Type u_1} →
  [inst : MeasurableSpace α] → (mean : α → ℝ) → Measurable mean → ℝ → ℝ → ProbabilityTheory.Kernel α (ℝ × ℝ)
```

value:
```
fun {α : Type u_1} [MeasurableSpace α] (mean : α → ℝ) (hmean : Measurable mean) (priorVariance noiseVariance : ℝ) =>
  ((AppliedModelingLib.Probability.gaussianLocationKernel mean hmean priorVariance.toNNReal).prod
      (ProbabilityTheory.Kernel.const α (ProbabilityTheory.gaussianReal 0 noiseVariance.toNNReal))).map
    fun (pair : ℝ × ℝ) => (pair.1 + pair.2, pair.1)
```

Direct retained dependencies

- AppliedModelingLib.Probability.gaussianLocationKernel

GaussianConvolutionRigidity.gaussianConvolution

Declaration location: paper-local

Lean declaration body

type:
```
(ℝ → ℝ) → NNReal → ℝ → ℝ
```

value:
```
fun (f : ℝ → ℝ) (noiseVariance : NNReal) (mean : ℝ) =>
  ∫ (score : ℝ), f score ∂ProbabilityTheory.gaussianReal mean noiseVariance
```

LG21TestOptionalPolicies.LG21AccessAction

Declaration location: paper-local

Lean declaration body

field takesTest:
```
LG21TestOptionalPolicies.LG21AccessAction → Bool
```

field reportsScore:
```
LG21TestOptionalPolicies.LG21AccessAction → Bool
```

LG21TestOptionalPolicies.LG21AccessAction.noTake

Declaration location: paper-local

Lean declaration body

type:
LG21TestOptionalPolicies.LG21AccessAction

value:
{ takesTest := false, reportsScore := false }

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21AccessAction.optionalReportingRequirement

Declaration location: paper-local

Lean declaration body

type:
LG21TestOptionalPolicies.LG21AccessAction → Prop

value:
fun (_a : LG21TestOptionalPolicies.LG21AccessAction) => True

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21AccessAction.reportImpliesTake

Declaration location: paper-local

Lean declaration body

type:
LG21TestOptionalPolicies.LG21AccessAction → Prop

value:
fun (a : LG21TestOptionalPolicies.LG21AccessAction) => a.reportsScore = true → a.takesTest = true

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21AccessAction.feasible

Declaration location: paper-local

Lean declaration body

type:
(LG21TestOptionalPolicies.LG21AccessAction → Prop) → LG21TestOptionalPolicies.LG21AccessAction → Prop

value:
fun (requirement : LG21TestOptionalPolicies.LG21AccessAction → Prop) (a : LG21TestOptionalPolicies.LG21AccessAction) =>
  a.reportImpliesTake ∧ requirement a

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.reportImpliesTake

LG21TestOptionalPolicies.LG21AccessAction.reportRequiredAfterTaking

Declaration location: paper-local

Lean declaration body

type:
LG21TestOptionalPolicies.LG21AccessAction → Prop

value:
fun (a : LG21TestOptionalPolicies.LG21AccessAction) => a.takesTest = a.reportsScore

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21AccessAction.takeAndReport

Declaration location: paper-local

Lean declaration body

type:
LG21TestOptionalPolicies.LG21AccessAction

value:
{ takesTest := true, reportsScore := true }

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21AccessStatus

Declaration location: paper-local

Lean declaration body

type:
Type

constructor LG21TestOptionalPolicies.LG21AccessStatus.noAccess:
LG21TestOptionalPolicies.LG21AccessStatus

constructor LG21TestOptionalPolicies.LG21AccessStatus.access:
LG21TestOptionalPolicies.LG21AccessStatus

LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource

Declaration location: paper-local

Lean declaration body

field baseLaw:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → MeasureTheory.Measure Base

field baseLaw_isProbability:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base),
  MeasureTheory.IsProbabilityMeasure self.baseLaw

field posteriorBaseMean:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → Base → ℝ

field posteriorBaseMean_measurable:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base), Measurable self.posteriorBaseMean

field posteriorBaseVariance:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → NNReal

field posteriorBaseVariance_pos:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base), 0 < ↑self.posteriorBaseVariance

field testNoiseVariance:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → NNReal

field testNoiseVariance_pos:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base), 0 < ↑self.testNoiseVariance

LG21TestOptionalPolicies.LG21NonTestFeature

Declaration location: paper-local

Lean declaration body

type:
```
(Feature : Type u_1) → Feature → Type (max 0 u_1)
```

value:
```
fun (Feature : Type u_1) (testFeature : Feature) => { feature : Feature // feature ≠ testFeature }
```

LG21TestOptionalPolicies.LG21ObservedAccessAllReportPBO

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        MeasureTheory.Measure Omega → (Omega → Base) → (Omega → ℝ) → (Omega → ℝ) → (Base → ℝ → ℝ) → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (law : MeasureTheory.Measure Omega) (base : Omega → Base) (score skill : Omega → ℝ)
    (reportedPayoff : Base → ℝ → ℝ) =>
  (fun (omega : Omega) => reportedPayoff (base omega) (score omega)) =ᵐ[law]
    law[skill | MeasurableSpace.comap (fun (omega : Omega) => (base omega, score omega)) inferInstance]
```

LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Estimate : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [MeasurableSpace Estimate] → MeasureTheory.Measure Omega → MeasureTheory.Measure Omega → (Omega → Estimate) → Prop
```

value:
```
fun {Omega : Type u_1} {Estimate : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Estimate]
    (accessLaw noAccessLaw : MeasureTheory.Measure Omega) (output : Omega → Estimate) =>
  MeasureTheory.Measure.map output accessLaw = MeasureTheory.Measure.map output noAccessLaw
```

LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Estimate : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Estimate] →
            [StandardBorelSpace Estimate] →
              [Nonempty Estimate] →
                MeasureTheory.Measure Base →
                  (Omega → Base) →
                    (accessLaw noAccessLaw : MeasureTheory.Measure Omega) →
                      [MeasureTheory.IsFiniteMeasure accessLaw] →
                        [MeasureTheory.IsFiniteMeasure noAccessLaw] → (Omega → Estimate) → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} {Estimate : Type u_3} [MeasurableSpace Omega] [MeasurableSpace Base]
    [MeasurableSpace Estimate] [StandardBorelSpace Estimate] [Nonempty Estimate] (baseLaw : MeasureTheory.Measure Base)
    (base : Omega → Base) (accessLaw noAccessLaw : MeasureTheory.Measure Omega)
    [MeasureTheory.IsFiniteMeasure accessLaw] [MeasureTheory.IsFiniteMeasure noAccessLaw] (output : Omega → Estimate) =>
  MeasureTheory.Measure.map base accessLaw = baseLaw ∧
    MeasureTheory.Measure.map base noAccessLaw = baseLaw ∧
      ⇑(ProbabilityTheory.condDistrib output base accessLaw) =ᵐ[baseLaw]
        ⇑(ProbabilityTheory.condDistrib output base noAccessLaw)
```

LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource

Declaration location: paper-local

Lean declaration body

```
field population:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature →
        LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature

field access_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (self : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature), 0 < self.population.accessLaw {true}

field test_feature:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] → LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature → Feature

field prior_variance_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (self : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature), 0 < ↑self.population.priorVariance

field non_test_noise_variance_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (self : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature)
  (feature : LG21TestOptionalPolicies.LG21NonTestFeature Feature self.test_feature),
  0 < ↑(self.population.noiseVariance ↑feature)

field test_noise_variance_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (self : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature),
  0 < ↑(self.population.noiseVariance self.test_feature)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput

Declaration location: paper-local

Lean declaration body

```
field accessOutput:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {testFeature : Feature} →
        LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature → Bool × ℝ × (Feature → ℝ) → ℝ

field noAccessKernel:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {testFeature : Feature} →
        LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature →
          ProbabilityTheory.Kernel (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ

field noAccessKernel_isMarkov:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature] {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature),
  ProbabilityTheory.IsMarkovKernel self.noAccessKernel
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.LG21OptionalCandidateBranchData

Declaration location: paper-local

Lean declaration body

```
field testLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test →
          Skill → Base → MeasureTheory.Measure Test

field testLaw_isProbability:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
```

```
  (self : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test) (skill : Skill) (base : Base),
  MeasureTheory.IsProbabilityMeasure (self.testLaw skill base)

field reportDecision:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test → Base → Test → Bool

field reportedValue:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test → Base → Test → ℝ

field noReportValue:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test → Base → ℝ

field continuationValue_integrable:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test) (skill : Skill) (base : Base),
  MeasureTheory.Integrable
    (fun (test : Test) =>
      if self.reportDecision base test = true then self.reportedValue base test else self.noReportValue base)
    (self.testLaw skill base)
```

LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank

Declaration location: paper-local

Lean declaration body

```
type:
{Score : Type u_1} →
  [inst : MeasurableSpace Score] →
    MeasureTheory.Measure Score → MeasureTheory.Measure ℝ → Set Score → ProbabilityTheory.Kernel Score ℝ → Prop

value:
fun {Score : Type u_1} [MeasurableSpace Score] (scoreLaw : MeasureTheory.Measure Score)
    (baseLaw : MeasureTheory.Measure ℝ) (reporterSet : Set Score)
    (operationalKernel : ProbabilityTheory.Kernel Score ℝ) =>
  scoreLaw reporterSet = 0 ∨ ∀ᵐ (score : Score) ∂scoreLaw, operationalKernel score = baseLaw
```

LG21TestOptionalPolicies.LG21OptionalSourceTimedActions

Declaration location: paper-local

Lean declaration body

```
field testLaw:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → ℝ → Base → MeasureTheory.Measure ℝ

field testLaw_isProbability:
∀ {Base : Type u_1} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base)
  (skill : ℝ) (base : Base), MeasureTheory.IsProbabilityMeasure (self.testLaw skill base)

field takeDecision:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → ℝ → Base → Bool

field reportDecision:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Base → ℝ → Bool

field reportedPayoff:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Base → ℝ → ℝ

field noReportPayoff:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Base → ℝ

field continuationPayoff_integrable:
∀ {Base : Type u_1} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base)
  (skill : ℝ) (base : Base),
  MeasureTheory.Integrable
    (fun (score : ℝ) =>
      if self.reportDecision base score = true then self.reportedPayoff base score else self.noReportPayoff base)
    (self.testLaw skill base)
```

LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel

Declaration location: paper-local

Lean declaration body

```
field basePosteriorMean:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → Base → ℝ

field basePosteriorMean_measurable:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base), Measurable self.basePosteriorMean

field basePosteriorVariance:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → NNReal

field basePosteriorVariance_pos:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base), 0 < ↑self.basePosteriorVariance

field latentSkillGivenBaseLaw:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → Base → MeasureTheory.Measure ℝ

field latentSkillGivenBaseLaw_eq_gaussian:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base) (base : Base),
  self.latentSkillGivenBaseLaw base =
    ProbabilityTheory.gaussianReal (self.basePosteriorMean base) self.basePosteriorVariance

field testNoiseVariance:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → NNReal

field testNoiseVariance_pos:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base), 0 < ↑self.testNoiseVariance

field testScoreGivenSkillLaw:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → ℝ → MeasureTheory.Measure ℝ

field testScoreGivenSkillLaw_eq_gaussian:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base) (skill : ℝ),
  self.testScoreGivenSkillLaw skill = ProbabilityTheory.gaussianReal skill self.testNoiseVariance

field noAccessEstimateKernel:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → ProbabilityTheory.Kernel Base ℝ

field noAccessEstimateKernel_isMarkov:
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base),
  ProbabilityTheory.IsMarkovKernel self.noAccessEstimateKernel
```

LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData

Declaration location: paper-local

Lean declaration body

```
field testLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test →
          Skill → Base → MeasureTheory.Measure Test

field testLaw_isProbability:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test) (skill : Skill)
  (base : Base), MeasureTheory.IsProbabilityMeasure (self.testLaw skill base)

field takeDecision:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test → Skill → Base → Bool

field reportedPayoff:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
```

```
    [inst : MeasurableSpace Test] →
      LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test → Base → Test → ℝ

field noReportPayoff:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test → Base → ℝ

field reportedPayoff_integrable:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test) (skill : Skill)
  (base : Base), MeasureTheory.Integrable (self.reportedPayoff base) (self.testLaw skill base)

field estimationConsistent:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test → Prop
```

LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData

Declaration location: paper-local

Lean declaration body

```
field testLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test →
          Skill → Base → MeasureTheory.Measure Test

field testLaw_isProbability:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test) (skill : Skill) (base : Base),
  MeasureTheory.IsProbabilityMeasure (self.testLaw skill base)

field takeDecision:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test → Skill → Base → Bool

field reportedPayoff:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test → Base → Test → ℝ

field noReportPayoff:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test → Base → ℝ

field reportedPayoff_integrable:
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test) (skill : Skill) (base : Base),
  MeasureTheory.Integrable (self.reportedPayoff base) (self.testLaw skill base)
```

LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface

Declaration location: paper-local

Lean declaration body

```
field Equilibrium:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law → Type u_5

field latentReporterLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Skill → Base → Law

field latentNonreporterLaw:
```

```
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Skill → Base → Law

field observableReporterLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Base → Law

field observableNonreporterLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Base → Law

field demographicReporterLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Law

field demographicNonreporterLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Law

field baseOnlyLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Base → Law

field fullFeatureLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Base → Test → Law
```

LG21TestOptionalPolicies.LG21RequirementPolicy.accessRequirement

Declaration location: paper-local

Lean declaration body

```
type:
LG21TestOptionalPolicies.LG21RequirementPolicy → LG21TestOptionalPolicies.LG21AccessAction → Prop

value:
fun (a : LG21TestOptionalPolicies.LG21RequirementPolicy) (a_1 : LG21TestOptionalPolicies.LG21AccessAction) =>
  (match (motive := LG21TestOptionalPolicies.LG21RequirementPolicy → LG21TestOptionalPolicies.LG21AccessAction → Prop)
      a with
    | LG21TestOptionalPolicies.LG21RequirementPolicy.noRequirements =>
      LG21TestOptionalPolicies.LG21AccessAction.optionalReportingRequirement
    | LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredConditionalTaking =>
      LG21TestOptionalPolicies.LG21AccessAction.reportRequiredAfterTaking
    | LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredGivenAccess =>
      fun (action : LG21TestOptionalPolicies.LG21AccessAction) =>
      action = LG21TestOptionalPolicies.LG21AccessAction.takeAndReport)
    a_1
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.optionalReportingRequirement
- LG21TestOptionalPolicies.LG21AccessAction.reportRequiredAfterTaking
- LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
- LG21TestOptionalPolicies.LG21RequirementPolicy

LG21TestOptionalPolicies.LG21SourceLawPolicySurface

Declaration location: paper-local

Lean declaration body

```
field Equilibrium:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law → Type u_5

field latentAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Skill → Base → Law

field latentNoAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Skill → Base → Law

field observableAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) → self.Equilibrium → Base → Law

field observableNoAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) → self.Equilibrium → Base → Law

field demographicAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) → self.Equilibrium → Law

field demographicNoAccessLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) → self.Equilibrium → Law

field baseOnlyLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) → self.Equilibrium → Base → Law

field fullFeatureLaw:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        (self : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) →
          self.Equilibrium → Base → Test → Law
```

LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface.toSourceLawSurface

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} →
        LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law →
          LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) =>
  { Equilibrium := R.Equilibrium, latentAccessLaw := R.latentReporterLaw, latentNoAccessLaw := R.latentNonreporterLaw,
    observableAccessLaw := R.observableReporterLaw, observableNoAccessLaw := R.observableNonreporterLaw,
    demographicAccessLaw := R.demographicReporterLaw, demographicNoAccessLaw := R.demographicNonreporterLaw,
    baseOnlyLaw := R.baseOnlyLaw, fullFeatureLaw := R.fullFeatureLaw }
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface
- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

LG21TestOptionalPolicies.PaperInterface.report_decision_skill_independent

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} → {Base : Type u_2} → {Test : Type u_3} → (Skill → Base → Test → Bool) → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} (report : Skill → Base → Test → Bool) =>
  ∀ (q q' : Skill) (base : Base) (score : Test), report q base score = report q' base score
```

LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface

Declaration location: paper-local

Lean declaration body

```
type:
(Skill : Type u_1) →
  (Base : Type u_2) →
    (Test : Type u_3) →
      (Law : Type u_4) → Type u_5 → Law → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law

value:
fun (Skill : Type u_1) (Base : Type u_2) (Test : Type u_3) (Law : Type u_4) (Equilibrium : Type u_5) (law : Law) =>
  { Equilibrium := Equilibrium, latentAccessLaw := fun (x : Equilibrium) (x_1 : Skill) (x_2 : Base) => law,
    latentNoAccessLaw := fun (x : Equilibrium) (x_1 : Skill) (x_2 : Base) => law,
    observableAccessLaw := fun (x : Equilibrium) (x_1 : Base) => law,
    observableNoAccessLaw := fun (x : Equilibrium) (x_1 : Base) => law,
    demographicAccessLaw := fun (x : Equilibrium) => law, demographicNoAccessLaw := fun (x : Equilibrium) => law,
    baseOnlyLaw := fun (x : Equilibrium) (x_1 : Base) => law,
    fullFeatureLaw := fun (x : Equilibrium) (x_1 : Base) (x_2 : Test) => law }
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond

Declaration location: paper-local

Lean declaration body

```
type:
{Ω : Type u_1} →
  {Candidate : Type u_2} →
    [inst : MeasurableSpace Ω] → MeasureTheory.Measure Ω → Set Ω → Candidate → (Candidate → Ω → Prop) → Prop

value:
fun {Ω : Type u_1} {Candidate : Type u_2} [MeasurableSpace Ω] (μ : MeasureTheory.Measure Ω) (branch : Set Ω)
    (candidate : Candidate) (memberBestResponse : Candidate → Ω → Prop) =>
  ∀ᵐ (omega : Ω) ∂μ.restrict branch, memberBestResponse candidate omega
```

LG21TestOptionalPolicies.lg21ContinuousGaussianNoiseLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (Feature → ℝ)

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  MeasureTheory.Measure.pi fun (feature : Feature) => ProbabilityTheory.gaussianReal 0 (M.noiseVariance feature)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation

LG21TestOptionalPolicies.lg21ContinuousGaussianNonTestNoiseLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature) =>
  MeasureTheory.Measure.pi fun (feature : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature) =>
    ProbabilityTheory.gaussianReal 0 (M.noiseVariance ↑feature)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          MeasureTheory.Measure ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ)

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature) =>
  MeasureTheory.Measure.map
    (fun (primitive : ℝ × (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
      (fun (feature : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature) =>
        primitive.1 + primitive.2 feature, primitive.1))
    ((ProbabilityTheory.gaussianReal M.priorMean M.priorVariance).prod
      (LG21TestOptionalPolicies.lg21ContinuousGaussianNonTestNoiseLaw M testFeature))
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianNonTestNoiseLaw

LG21TestOptionalPolicies.lg21ContinuousGaussianStudentPrimitiveLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  (ProbabilityTheory.gaussianReal M.priorMean M.priorVariance).prod
    (LG21TestOptionalPolicies.lg21ContinuousGaussianNoiseLaw M)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.lg21ContinuousGaussianNoiseLaw

LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  M.accessLaw.prod (LG21TestOptionalPolicies.lg21ContinuousGaussianStudentPrimitiveLaw M)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.lg21ContinuousGaussianStudentPrimitiveLaw

LG21TestOptionalPolicies.lg21ContinuousPopulationAccess

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} → Bool × ℝ × (Feature → ℝ) → Bool

value:
fun {Feature : Type u_1} (a : Bool × ℝ × (Feature → ℝ)) => a.1

LG21TestOptionalPolicies.lg21ContinuousPopulationAccessStatus

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} → Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21AccessStatus

value:
fun {Feature : Type u_1} (a : Bool × ℝ × (Feature → ℝ)) =>
  if LG21TestOptionalPolicies.lg21ContinuousPopulationAccess a = true then
    LG21TestOptionalPolicies.LG21AccessStatus.access
  else LG21TestOptionalPolicies.LG21AccessStatus.noAccess

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessStatus
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccess

LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium

Declaration location: paper-local

Lean declaration body

field action:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
      LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium M →
        Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21AccessAction

field feasible:
∀ {Feature : Type u_1} [inst : Fintype Feature] {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature}
  (self : LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium M)
  (student : Bool × ℝ × (Feature → ℝ)),
  LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredGivenAccess.feasibleAction
    (LG21TestOptionalPolicies.lg21ContinuousPopulationAccessStatus student) (self.action student)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21RequirementPolicy
- LG21TestOptionalPolicies.LG21RequirementPolicy.feasibleAction
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccessStatus

LG21TestOptionalPolicies.lg21ContinuousPopulationNoise

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} → Feature → Bool × ℝ × (Feature → ℝ) → ℝ

value:
fun {Feature : Type u_1} (feature : Feature) (a : Bool × ℝ × (Feature → ℝ)) => a.2.2 feature
```

LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} → Bool × ℝ × (Feature → ℝ) → ℝ

value:
fun {Feature : Type u_1} (a : Bool × ℝ × (Feature → ℝ)) => a.2.1
```

LG21TestOptionalPolicies.lg21ContinuousPopulationFeature

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} → Feature → Bool × ℝ × (Feature → ℝ) → ℝ

value:
fun {Feature : Type u_1} (feature : Feature) (a : Bool × ℝ × (Feature → ℝ)) =>
  LG21TestOptionalPolicies.lg21ContinuousPopulationSkill a +
    LG21TestOptionalPolicies.lg21ContinuousPopulationNoise feature a
```

Direct retained dependencies

- LG21TestOptionalPolicies.lg21ContinuousPopulationNoise
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.lg21ContinuousPopulationBase

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  (testFeature : Feature) →
    Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ

value:
fun {Feature : Type u_1} (testFeature : Feature) (a : Bool × ℝ × (Feature → ℝ))
    (a_1 : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature) =>
  LG21TestOptionalPolicies.lg21ContinuousPopulationFeature (↑a_1) a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature

LG21TestOptionalPolicies.lg21D6ConditionalTestVariance

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → NNReal

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  S.posteriorBaseVariance + S.testNoiseVariance
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource

LG21TestOptionalPolicies.lg21D6GaussianTestNoiseKernel

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → ProbabilityTheory.Kernel Base ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  ProbabilityTheory.Kernel.const Base
    (ProbabilityTheory.gaussianReal 0 (LG21TestOptionalPolicies.lg21D6ConditionalTestVariance S))
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.lg21D6ConditionalTestVariance

LG21TestOptionalPolicies.lg21D6ConditionalGaussianTestKernel

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → ProbabilityTheory.Kernel Base ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  (ProbabilityTheory.Kernel.id.prod (LG21TestOptionalPolicies.lg21D6GaussianTestNoiseKernel S)).map
    fun (pair : Base × ℝ) => S.posteriorBaseMean pair.1 + pair.2
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.lg21D6GaussianTestNoiseKernel

LG21TestOptionalPolicies.lg21D6ActualAccessTestJointKernel

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → ProbabilityTheory.Kernel Base (Base × ℝ)

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  ProbabilityTheory.Kernel.id.prod (LG21TestOptionalPolicies.lg21D6ConditionalGaussianTestKernel S)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.lg21D6ConditionalGaussianTestKernel

LG21TestOptionalPolicies.lg21D6PosteriorTestWeight

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} → [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  ↑S.posteriorBaseVariance / (↑S.posteriorBaseVariance + ↑S.testNoiseVariance)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource

LG21TestOptionalPolicies.lg21D6GaussianPBOEstimate

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → Base × ℝ → ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base)
    (a : Base × ℝ) =>
  S.posteriorBaseMean a.1 + LG21TestOptionalPolicies.lg21D6PosteriorTestWeight S * (a.2 - S.posteriorBaseMean a.1)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.lg21D6PosteriorTestWeight

LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base → ProbabilityTheory.Kernel Base ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (S : LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource Base) =>
  (ProbabilityTheory.Kernel.deterministic (LG21TestOptionalPolicies.lg21D6GaussianPBOEstimate S)
      (LG21TestOptionalPolicies.lg21D6GaussianPBOEstimate_measurable S)).comp
    (LG21TestOptionalPolicies.lg21D6ActualAccessTestJointKernel S)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.lg21D6ActualAccessTestJointKernel
- LG21TestOptionalPolicies.lg21D6GaussianPBOEstimate

LG21TestOptionalPolicies.lg21HiddenAccessNoAccessEvent

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} → Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} (a : Bool × ℝ × (Feature → ℝ)) => {student : Bool × ℝ × (Feature → ℝ) | student.1 = false} a

LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) → ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature) (student : ℝ × (Feature → ℝ))
    (a : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature) =>
  student.1 + student.2 ↑a

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
      LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2 ∈ region}
    a

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.lg21HiddenAccessBaseSkillObservation

Declaration location: paper-local

Lean declaration body

type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        Bool × ℝ × (Feature → ℝ) → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (student : Bool × ℝ × (Feature → ℝ)) =>
  (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2,
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.lg21HiddenAccessStudentScore

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} → [Fintype Feature] → Feature → ℝ × (Feature → ℝ) → ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] (testFeature : Feature) (student : ℝ × (Feature → ℝ)) =>
  student.1 + student.2 testFeature
```

LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        Bool × ℝ × (Feature → ℝ) → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature a.2,
    LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature a.2,
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill a)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21HiddenAccessStudentScore

LG21TestOptionalPolicies.lg21HiddenAccessStudentReport

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) → ℝ × (Feature → ℝ) → Bool

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (student : ℝ × (Feature → ℝ)) =>
  reportDecision (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student)
    (LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21HiddenAccessStudentScore

LG21TestOptionalPolicies.lg21HiddenAccessStudentTake

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) → ℝ × (Feature → ℝ) → Bool

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (student : ℝ × (Feature → ℝ)) =>
  takeDecision student.1 (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            Bool × ℝ × (Feature → ℝ) → Bool

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (student : Bool × ℝ × (Feature → ℝ)) =>
  if student.1 = true then
    if LG21TestOptionalPolicies.lg21HiddenAccessStudentTake testFeature takeDecision student.2 = true then
      LG21TestOptionalPolicies.lg21HiddenAccessStudentReport testFeature reportDecision student.2
    else false
  else false
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessStudentReport
- LG21TestOptionalPolicies.lg21HiddenAccessStudentTake

LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature takeDecision reportDecision student =
      false}
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction

LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicObservation

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            Bool × ℝ × (Feature → ℝ) → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × Bool × ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (student : Bool × ℝ × (Feature → ℝ)) =>
  have observedAction : Bool :=
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature takeDecision reportDecision student;
  (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2, observedAction,
    if observedAction = true then LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student.2 else 0)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21HiddenAccessStudentScore

LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ) →
              ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) → Bool × ℝ × (Feature → ℝ) → ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (reportedPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ)
    (noReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ)
    (student : Bool × ℝ × (Feature → ℝ)) =>
  if
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature takeDecision reportDecision student =
        true then
    reportedPayoff (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
      (LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student.2)
  else noReportPayoff (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21HiddenAccessStudentScore

LG21TestOptionalPolicies.LG21HiddenAccessActualPublicPBO

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
            ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
              ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ) →
                ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (reportedPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ)
    (noReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) =>
  have law : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
  have observation :
    Bool × ℝ × (Feature → ℝ) → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × Bool × ℝ :=
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicObservation testFeature takeDecision reportDecision;
  LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff testFeature takeDecision reportDecision reportedPayoff
      noReportPayoff =ᵐ[law]
    law[LG21TestOptionalPolicies.lg21ContinuousPopulationSkill | MeasurableSpace.comap observation inferInstance]
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicObservation
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff

LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature candidateTake candidateReport
          student =
        false}
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction

LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
            Bool × ℝ × (Feature → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature candidateTake candidateReport
        student =
        true}
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction

LG21TestOptionalPolicies.lg21NormalizedRestriction

Declaration location: paper-local

Lean declaration body

```
type:
{Outcome : Type u_1} →
  [inst : MeasurableSpace Outcome] → MeasureTheory.Measure Outcome → Set Outcome → MeasureTheory.Measure Outcome
```

```
value:
fun {Outcome : Type u_1} [MeasurableSpace Outcome] (law : MeasureTheory.Measure Outcome) (event : Set Outcome) =>
  (law event)⁻¹ • law.restrict event
```

LG21TestOptionalPolicies.LG21FullPublicReportRequiredNoTakePBO

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        MeasureTheory.Measure Omega → (Omega → Base) → (Omega → ℝ) → (Base → ℝ → Bool) → (Base → ℝ) → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (rawLaw : MeasureTheory.Measure Omega) (base : Omega → Base) (skill : Omega → ℝ) (take : Base → ℝ → Bool)
    (noTakePayoff : Base → ℝ) =>
  have noTakeLaw : MeasureTheory.Measure Omega :=
    LG21TestOptionalPolicies.lg21NormalizedRestriction rawLaw {omega : Omega | take (base omega) (skill omega) = false};
  (fun (omega : Omega) => noTakePayoff (base omega)) =ᵐ[noTakeLaw]
    noTakeLaw[skill | MeasurableSpace.comap base inferInstance]
```

Direct retained dependencies

- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21FullPublicReportRequiredReportedPBO

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        MeasureTheory.Measure Omega →
          (Omega → Base) → (Omega → ℝ) → (Omega → ℝ) → (Base → ℝ → Bool) → (Base → ℝ → ℝ) → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (rawLaw : MeasureTheory.Measure Omega) (base : Omega → Base) (score skill : Omega → ℝ) (take : Base → ℝ → Bool)
    (reportedPayoff : Base → ℝ → ℝ) =>
  have reporterLaw : MeasureTheory.Measure Omega :=
    LG21TestOptionalPolicies.lg21NormalizedRestriction rawLaw {omega : Omega | take (base omega) (skill omega) = true};
  (fun (omega : Omega) => reportedPayoff (base omega) (score omega)) =ᵐ[reporterLaw]
    reporterLaw[skill | MeasurableSpace.comap (fun (omega : Omega) => (base omega, score omega)) inferInstance]
```

Direct retained dependencies

- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  LG21TestOptionalPolicies.lg21NormalizedRestriction (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
    {student : Bool × ℝ × (Feature → ℝ) | LG21TestOptionalPolicies.lg21ContinuousPopulationAccess student = true}
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21ObservedAccessObservableFair

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          [MeasureTheory.IsFiniteMeasure (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)] →
            LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    [MeasureTheory.IsFiniteMeasure (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)]
    (policy : LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature) =>
  ⇑(ProbabilityTheory.condDistrib policy.accessOutput
      (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
        M)) =ᵐ[MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)]
    ⇑policy.noAccessKernel
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase

LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile

Declaration location: paper-local

Lean declaration body

```
field mandatory:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source →
          LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium source.population

field reportedPayoff:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source →
          (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ → ℝ

field reportedPBO:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature}
  (self : LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source),
  LG21TestOptionalPolicies.LG21ObservedAccessAllReportPBO
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.reportedPayoff
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessAllReportPBO
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  LG21TestOptionalPolicies.lg21NormalizedRestriction (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
    {student : Bool × ℝ × (Feature → ℝ) | LG21TestOptionalPolicies.lg21ContinuousPopulationAccess student = false}
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        0 < M.accessLaw {false} → Feature → (Bool × ℝ × (Feature → ℝ) → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (hnoAccess : 0 < M.accessLaw {false})
    (testFeature : Feature) (noAccessOutput : Bool × ℝ × (Feature → ℝ) → ℝ) =>
  let law : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw M;
  have base : Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ :=
    LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature;
  have skill : Bool × ℝ × (Feature → ℝ) → ℝ := LG21TestOptionalPolicies.lg21ContinuousPopulationSkill;
  noAccessOutput =ᵐ[law] fun (student : Bool × ℝ × (Feature → ℝ)) =>
    ∫ (latentSkill : ℝ), latentSkill ∂(ProbabilityTheory.condDistrib skill base law) (base student)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.LG21ObservedAccessDemographicallyFair

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) → LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (policy : LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature) =>
  MeasureTheory.Measure.map policy.accessOutput (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M) =
    (MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw M)).bind
      ⇑policy.noAccessKernel
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase

LG21TestOptionalPolicies.LG21ObservedAccessFair

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          [MeasureTheory.IsFiniteMeasure (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)] →
            LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    [MeasureTheory.IsFiniteMeasure (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)]
    (policy : LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput testFeature) =>
  LG21TestOptionalPolicies.LG21ObservedAccessObservableFair M testFeature policy ∧
    LG21TestOptionalPolicies.LG21ObservedAccessDemographicallyFair M testFeature policy
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21ObservedAccessDemographicallyFair
- LG21TestOptionalPolicies.LG21ObservedAccessObservableFair
- LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw

LG21TestOptionalPolicies.LG21ObservedAccessMandatoryPBODefinition1

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        0 < M.accessLaw {true} →
          0 < M.accessLaw {false} →
            (testFeature : Feature) →
              LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium M →
                ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ) →
                  (Bool × ℝ × (Feature → ℝ) → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (haccess : 0 < M.accessLaw {true})
    (hnoAccess : 0 < M.accessLaw {false}) (testFeature : Feature)
    (mandatory : LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium M)
    (reportedPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ)
    (noAccessOutput : Bool × ℝ × (Feature → ℝ) → ℝ) =>
  (∀ (student : Bool × ℝ × (Feature → ℝ)),
      LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredGivenAccess.feasibleAction
        (LG21TestOptionalPolicies.lg21ContinuousPopulationAccessStatus student) (mandatory.action student)) ∧
    (∀ (student : Bool × ℝ × (Feature → ℝ)),
        mandatory.action student =
          if LG21TestOptionalPolicies.lg21ContinuousPopulationAccess student = true then
            LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
          else LG21TestOptionalPolicies.LG21AccessAction.noTake) ∧
      (∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M,
          mandatory.action student = LG21TestOptionalPolicies.LG21AccessAction.takeAndReport) ∧
        LG21TestOptionalPolicies.LG21ObservedAccessAllReportPBO
          (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill reportedPayoff ∧
        LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO M hnoAccess testFeature noAccessOutput
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.noTake
- LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
- LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessAllReportPBO
- LG21TestOptionalPolicies.LG21RequirementPolicy
- LG21TestOptionalPolicies.LG21RequirementPolicy.feasibleAction
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccessStatus
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile

Declaration location: paper-local

Lean declaration body

```
field mandatory:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        {hnoAccess : 0 < source.population.accessLaw {false}} →
          LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess →
            LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium source.population

field reportedPayoff:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        {hnoAccess : 0 < source.population.accessLaw {false}} →
          LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ → ℝ

field noAccessOutput:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        {hnoAccess : 0 < source.population.accessLaw {false}} →
          LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess → Bool × ℝ × (Feature → ℝ) → ℝ

field pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature}
  {hnoAccess : 0 < source.population.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess),
  LG21TestOptionalPolicies.LG21ObservedAccessMandatoryPBODefinition1 source.population source.access_positive hnoAccess
    source.test_feature self.mandatory self.reportedPayoff self.noAccessOutput
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessMandatoryPBODefinition1

LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) →
            MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
  LG21TestOptionalPolicies.lg21NormalizedRestriction (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
    (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateNoReportPBOOn

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) →
            0 <
              (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
                (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region) →
              (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
                (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
                  LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
                    (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ →
                  0 <
                    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
                      (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent testFeature candidateTake
                        candidateReport) →
                  Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
    (hregionPositive :
      0 <
        (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
          (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region))
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (candidate :
      LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
        (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ)
    (hnoReportPositive :
      0 <
        (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
          (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent testFeature candidateTake
            candidateReport)) =>
  have rawLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
  let localLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region;
  let noReportEvent : Set (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent testFeature candidateTake candidateReport;
  let actionLaw : MeasureTheory.Measure ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) :=
    MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessBaseSkillObservation testFeature)
      (LG21TestOptionalPolicies.lg21NormalizedRestriction localLaw noReportEvent);
  ∀ᵐ (publicBase :
    LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ∂MeasureTheory.Measure.map Prod.fst actionLaw,
    candidate.noReportValue publicBase =
      ∫ (skill : ℝ), skill ∂(ProbabilityTheory.condDistrib Prod.snd Prod.fst actionLaw) publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessBaseSkillObservation
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateReportPBOOn

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) →
            0 <
                (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
                  (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region) →
              (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
                (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
                  LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
                      (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ →
                    0 <
                        (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
                          (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature candidateTake
                            candidateReport) →
                      Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
    (hregionPositive :
      0 <
        (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
          (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region))
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (candidate :
      LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
        (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ)
    (hreportPositive :
      0 <
        (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
          (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature candidateTake
            candidateReport)) =>
  have rawLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
  let localLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region;
  let reportEvent : Set (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature candidateTake candidateReport;
  let actionLaw :
    MeasureTheory.Measure ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) :=
    MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation testFeature)
      (LG21TestOptionalPolicies.lg21NormalizedRestriction localLaw reportEvent);
  ∀ᵐ (publicObservation :
    (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ×
      ℝ) ∂MeasureTheory.Measure.map
      (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
        (scoreSkill.1, scoreSkill.2.1))
      actionLaw,
    candidate.reportedValue publicObservation.1 publicObservation.2 =
      ∫ (latentSkill : ℝ),
        latentSkill ∂(ProbabilityTheory.condDistrib
            (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
              scoreSkill.2.2)
            (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
              (scoreSkill.1, scoreSkill.2.1))
            actionLaw)
          publicObservation
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.lg21HiddenAccessNoAccessLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) =>
  LG21TestOptionalPolicies.lg21NormalizedRestriction (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
    LG21TestOptionalPolicies.lg21HiddenAccessNoAccessEvent
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessNoAccessEvent
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
            ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
              MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ))

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) =>
  LG21TestOptionalPolicies.lg21NormalizedRestriction (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
    (LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportEvent testFeature takeDecision reportDecision)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportEvent
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21HiddenAccessActualNoReportPBO

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature →
        (testFeature : Feature) →
          (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
            ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
              ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (takeDecision : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (reportDecision : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (noReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) =>
  0 <
      (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
        (LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportEvent testFeature takeDecision reportDecision) →
    have law : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
      LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
    have noReportLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportLaw M testFeature takeDecision reportDecision;
    have base : Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ :=
      fun (student : Bool × ℝ × (Feature → ℝ)) =>
      LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2;
    (fun (student : Bool × ℝ × (Feature → ℝ)) => noReportPayoff (base student)) =ᵐ[noReportLaw]
      noReportLaw[LG21TestOptionalPolicies.lg21ContinuousPopulationSkill | MeasurableSpace.comap base inferInstance]
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportEvent
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoReportLaw
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE

Declaration location: paper-local

Lean declaration body

```
field takeDecision:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool
```

```
field reportDecision:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool
```

```
field reportedPayoff:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ
```

```
field noReportPayoff:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
```

```
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ

field testLaw:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → MeasureTheory.Measure ℝ

field testLaw_isProbability:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
  (base : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  MeasureTheory.IsProbabilityMeasure (self.testLaw skill base)

field takeDecision_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  Measurable fun (pair : ℝ × (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
    self.takeDecision pair.1 pair.2

field reportDecision_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  Measurable fun (pair : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) =>
    self.reportDecision pair.1 pair.2

field reportedPayoff_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  Measurable fun (pair : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) =>
    self.reportedPayoff pair.1 pair.2

field noReportPayoff_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  Measurable self.noReportPayoff

field continuationPayoff_integrable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
  (base : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  MeasureTheory.Integrable
    (fun (score : ℝ) =>
      if self.reportDecision base score = true then self.reportedPayoff base score else self.noReportPayoff base)
    (self.testLaw skill base)

field access_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature), 0 < M.accessLaw {true}

field take_best_response:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
  (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  AppliedModelingLib.NoProfitableBinaryChoiceDeviation
    (fun (latentSkill : ℝ) => self.takeDecision latentSkill publicBase = true)
    (fun (latentSkill : ℝ) =>
      ∫ (score : ℝ),
        if self.reportDecision publicBase score = true then self.reportedPayoff publicBase score
        else self.noReportPayoff publicBase ∂self.testLaw latentSkill publicBase)
    fun (_latentSkill : ℝ) => self.noReportPayoff publicBase

field raw_test_law:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
  (base : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  self.testLaw skill base = ProbabilityTheory.gaussianReal skill (M.noiseVariance testFeature)

field test_noise_nonzero:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  M.noiseVariance testFeature ≠ 0

field public_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  LG21TestOptionalPolicies.LG21HiddenAccessActualPublicPBO M testFeature self.takeDecision self.reportDecision
    self.reportedPayoff self.noReportPayoff
```

```
field noReport_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
  LG21TestOptionalPolicies.LG21HiddenAccessActualNoReportPBO M testFeature self.takeDecision self.reportDecision
    self.noReportPayoff
```

Direct retained dependencies

- AppliedModelingLib.NoProfitableBinaryChoiceDeviation
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessActualNoReportPBO
- LG21TestOptionalPolicies.LG21HiddenAccessActualPublicPBO
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE.OptionalReportBestResponse

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  ∀ (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
    AppliedModelingLib.NoProfitableBinaryChoiceDeviation (fun (score : ℝ) => E.reportDecision publicBase score = true)
      (E.reportedPayoff publicBase) fun (_score : ℝ) => E.noReportPayoff publicBase
```

Direct retained dependencies

- AppliedModelingLib.NoProfitableBinaryChoiceDeviation
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE.OptionalReportBestResponseAE

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  E.OptionalReportBestResponse
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE.OptionalReportBestResponse

LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE

Declaration location: paper-local

Lean declaration body

```
field source:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature

field report_required:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature)
  (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (score : ℝ),
  self.source.reportDecision publicBase score = true
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessActualReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction testFeature E.takeDecision E.reportDecision
          student =
        true}
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservedAction

LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedTesterEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
              Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
```

```
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
  (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
  (a : Bool × ℝ × (Feature → ℝ)) =>
{student : Bool × ℝ × (Feature → ℝ) |
   student.1 = true ∧
    E.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
        (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
      false ∧
     candidateTake (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
        (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
      true}
  a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedToReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
              ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
                Bool × ℝ × (Feature → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
  (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
  (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
  (a : Bool × ℝ × (Feature → ℝ)) =>
(LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature candidateTake candidateReport ∩
    (LG21TestOptionalPolicies.lg21HiddenAccessActualReportEvent E)ᶜ)
  a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessActualReportEvent
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent

LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedActionEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
              ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool) →
                Bool × ℝ × (Feature → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (candidateReport : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  (LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedTesterEvent E candidateTake ∪
     LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedToReportEvent E candidateTake candidateReport)
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedTesterEvent
- LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedToReportEvent

LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  MeasureTheory.Measure.map
    (LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff testFeature E.takeDecision E.reportDecision
      E.reportedPayoff E.noReportPayoff)
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff

LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicNoAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  MeasureTheory.Measure.map
    (LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff testFeature E.takeDecision E.reportDecision
      E.reportedPayoff E.noReportPayoff)
    (LG21TestOptionalPolicies.lg21HiddenAccessNoAccessLaw M)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessNoAccessLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalPublicPayoff

LG21TestOptionalPolicies.lg21HiddenAccessOptionalDemographicallyFairAt

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicAccessOutputLaw E =
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicNoAccessOutputLaw E
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicNoAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) =>
  MeasureTheory.Measure.dirac (E.noReportPayoff publicBase)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessOptionalSourceOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (score : ℝ) =>
  if E.takeDecision skill publicBase = true then
    if E.reportDecision publicBase score = true then E.reportedPayoff publicBase score else E.noReportPayoff publicBase
  else E.noReportPayoff publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) =>
  MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessOptionalSourceOutput E skill publicBase)
    (E.testLaw skill publicBase)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalSourceOutput

LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentSkillFairAt

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature) =>
  ∀ (latentSkill : ℝ) (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentAccessOutputLaw E latentSkill publicBase =
      LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoAccessOutputLaw E publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservablyFairAt

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) →
              Measurable baseMean → ℝ → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
    (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ)
    (hbaseMean : Measurable baseMean) (baseVariance : ℝ) =>
  ∀ (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
    MeasureTheory.Measure.map
      (fun (scoreSkill : ℝ × ℝ) =>
        LG21TestOptionalPolicies.lg21HiddenAccessOptionalSourceOutput E scoreSkill.2 publicBase scoreSkill.1)
      ((AppliedModelingLib.Probability.gaussianSignalJointKernel baseMean hbaseMean baseVariance
          ↑(M.noiseVariance testFeature))
        publicBase) =
    LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoAccessOutputLaw E publicBase
```

Direct retained dependencies

- AppliedModelingLib.Probability.gaussianSignalJointKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalNoAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalSourceOutput

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) =>
  LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicAccessOutputLaw E.source
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicNoAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) =>
  LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicNoAccessOutputLaw E.source
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalActualDemographicNoAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredDemographicallyFairAt

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) =>
  LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicAccessOutputLaw E =
    LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicNoAccessOutputLaw E
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredActualDemographicNoAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredNoAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) =>
  MeasureTheory.Measure.dirac (E.source.noReportPayoff publicBase)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredSourceOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (score : ℝ) =>
  if E.source.takeDecision skill publicBase = true then E.source.reportedPayoff publicBase score
  else E.source.noReportPayoff publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) (skill : ℝ)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) =>
  MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredSourceOutput E skill publicBase)
    (E.source.testLaw skill publicBase)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredSourceOutput

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentSkillFair

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature) =>
  ∀ (skill : ℝ) (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
    LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentAccessOutputLaw E skill publicBase =
      LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredNoAccessOutputLaw E publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredNoAccessOutputLaw

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservableAccessOutputLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) →
              (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) →
                Measurable baseMean → ℝ → MeasureTheory.Measure ℝ

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature)
    (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)
    (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ)
    (hbaseMean : Measurable baseMean) (baseVariance : ℝ) =>
  MeasureTheory.Measure.map
    (fun (scoreSkill : ℝ × ℝ) =>
      LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredSourceOutput E scoreSkill.2 publicBase scoreSkill.1)
    ((AppliedModelingLib.Probability.gaussianSignalJointKernel baseMean hbaseMean baseVariance
        ↑(M.noiseVariance testFeature))
      publicBase)
```

Direct retained dependencies

- AppliedModelingLib.Probability.gaussianSignalJointKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredSourceOutput

LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservablyFairAt

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature →
            (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ) →
              Measurable baseMean → ℝ → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature)
    (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ)
    (hbaseMean : Measurable baseMean) (baseVariance : ℝ) =>
  ∀ (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
    LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservableAccessOutputLaw E publicBase baseMean hbaseMean
      baseVariance =
      LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredNoAccessOutputLaw E publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredNoAccessOutputLaw
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservableAccessOutputLaw

LG21TestOptionalPolicies.lg21ObservedAccessActualOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → (Omega → LG21TestOptionalPolicies.LG21AccessAction) → (Base → ℝ → ℝ) → (Base → ℝ) → Omega → ℝ
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score : Omega → ℝ) (action : Omega → LG21TestOptionalPolicies.LG21AccessAction) (reportedPayoff : Base → ℝ → ℝ)
    (noReportPayoff : Base → ℝ) (a : Omega) =>
  if (action a).reportsScore = true then reportedPayoff (base a) (score a) else noReportPayoff (base a)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction

LG21TestOptionalPolicies.LG21T44MandatoryGivenAccessResamplingFairnessCertificate

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature) →
        0 < source.population.accessLaw {false} →
          LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source →
            ((LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature)
    (hnoAccess : 0 < source.population.accessLaw {false})
    (profile : LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source)
    (noReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) =>
  ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ))
    (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ)
    (hbaseMean : Measurable baseMean) (hbaseLaw : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseVariance :
    0 < baseVariance),
```

```
LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population source.test_feature =
  baseLaw.compProd
    (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
let S :
  LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
    (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) :=
  { baseLaw := baseLaw,
    baseLaw_isProbability :=
      LG21TestOptionalPolicies.LG21Definition6ObservedAccessSourceWitness._proof_1 source baseLaw hbaseLaw,
    posteriorBaseMean := baseMean, posteriorBaseMean_measurable := hbaseMean,
    posteriorBaseVariance := baseVariance.toNNReal,
    posteriorBaseVariance_pos :=
      LG21TestOptionalPolicies.LG21Definition6ObservedAccessSourceWitness._proof_2 baseVariance hbaseVariance,
    testNoiseVariance := source.population.noiseVariance source.test_feature,
    testNoiseVariance_pos := source.test_noise_variance_positive };
LG21TestOptionalPolicies.LG21ObservedAccessFair source.population source.test_feature
  {
    accessOutput :=
      LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature) profile.mandatory.action
        profile.reportedPayoff noReportPayoff,
    noAccessKernel := LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel S,
    noAccessKernel_isMarkov :=
      LG21TestOptionalPolicies.LG21T44MandatoryGivenAccessResamplingFairnessCertificate._proof_1 source baseLaw
        baseMean baseVariance hbaseMean hbaseLaw hbaseVariance }
```

Direct retained dependencies

- AppliedModelingLib.Probability.gaussianLocationKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessFair
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput
- LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel
- LG21TestOptionalPolicies.lg21ObservedAccessActualOutput

LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} → {Estimate : Type u_2} → (Omega → Bool) → (Omega → Estimate) → (Omega → Estimate) → Omega → Estimate

value:
fun {Omega : Type u_1} {Estimate : Type u_2} (hasAccess : Omega → Bool) (accessOutput noAccessOutput : Omega → Estimate)
    (a : Omega) =>
  if hasAccess a = true then accessOutput a else noAccessOutput a
```

LG21TestOptionalPolicies.lg21OptionalCandidateContinuationValue

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test → Base → Test → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test) (base : Base)
    (test : Test) =>
  if candidate.reportDecision base test = true then candidate.reportedValue base test else candidate.noReportValue base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData

LG21TestOptionalPolicies.lg21OptionalCandidateTestExpectedValue

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test → Skill → Base → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData Skill Base Test) (skill : Skill)
    (base : Base) =>
  ∫ (test : Test),
    LG21TestOptionalPolicies.lg21OptionalCandidateContinuationValue candidate base test ∂candidate.testLaw skill base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.lg21OptionalCandidateContinuationValue

LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry

Declaration location: paper-local

Lean declaration body

```
field region:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature} →
            {hnoAccess : 0 < M.accessLaw {false}} →
              LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess →
                Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)

field region_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess), MeasurableSet self.region

field region_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
      (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature self.region)

field candidateTake:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
```

```
        {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature} →
          {hnoAccess : 0 < M.accessLaw {false}} →
            LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess →
              ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool

field candidateReport:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature} →
            {hnoAccess : 0 < M.accessLaw {false}} →
              LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess →
                (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ → Bool

field candidate:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature} →
            {hnoAccess : 0 < M.accessLaw {false}} →
              LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess →
                LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
                  (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ

field candidate_test_law:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess) (latentSkill : ℝ)
  (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  self.candidate.testLaw latentSkill publicBase = E.testLaw latentSkill publicBase

field candidate_report_action:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  self.candidate.reportDecision = self.candidateReport

field candidate_take_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  Measurable fun (pair : ℝ × (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
    self.candidateTake pair.1 pair.2

field candidate_report_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  Measurable fun (pair : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) =>
    self.candidateReport pair.1 pair.2

field candidate_agrees_outside_region:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  ∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M,
    LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2 ∉ self.region →
      self.candidateTake (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
        E.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) ∧
      self.candidateReport (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
          (LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student.2) =
        E.reportDecision (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
          (LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student.2)

field candidate_changed_action_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  0 <
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedActionEvent E self.candidateTake self.candidateReport)

field candidate_report_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  0 <
```

```
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature self.candidateTake
        self.candidateReport)

field candidate_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  0 <
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent testFeature self.candidateTake
        self.candidateReport)

field candidate_report_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateReportPBOOn M testFeature self.region
    self.region_positive self.candidateTake self.candidateReport self.candidate self.candidate_report_positive

field candidate_noReport_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateNoReportPBOOn M testFeature self.region
    self.region_positive self.candidateTake self.candidateReport self.candidate self.candidate_noReport_positive

field candidate_report_members_best_respond:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
    (LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature self.candidateTake
      self.candidateReport)
    self.candidate
    fun
      (P :
        LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ
          (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ)
      (student : Bool × ℝ × (Feature → ℝ)) =>
    P.noReportValue (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) ≤
      P.reportedValue (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
        (LG21TestOptionalPolicies.lg21HiddenAccessStudentScore testFeature student.2)

field candidate_tester_strict_gain:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature}
  {hnoAccess : 0 < M.accessLaw {false}}
  (self : LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess),
  ∀ᵐ (student :
    Bool ×
      ℝ ×
        (Feature →
          ℝ)) ∂(LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region).restrict
    (LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedTesterEvent E self.candidateTake),
  self.candidate.noReportValue (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) <
    LG21TestOptionalPolicies.lg21OptionalCandidateTestExpectedValue self.candidate
      (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
      (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateNoReportPBOOn
- LG21TestOptionalPolicies.LG21HiddenAccessSourceTimedCandidateReportPBOOn
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedActionEvent

- LG21TestOptionalPolicies.lg21HiddenAccessCandidateChangedTesterEvent
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21HiddenAccessStudentScore
- LG21TestOptionalPolicies.lg21OptionalCandidateTestExpectedValue

LG21TestOptionalPolicies.LG21HiddenAccessSourceStableAgainstLocalCandidateEntry

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature →
            0 < M.accessLaw {false} → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (E : LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature)
    (hnoAccess : 0 < M.accessLaw {false}) =>
  ¬Nonempty (LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry E hnoAccess)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessSourceLocalCandidateEntry

LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel

Declaration location: paper-local

Lean declaration body

```
type:
{Score : Type u_1} →
  [inst : MeasurableSpace Score] →
    (reporterSet : Set Score) →
      MeasurableSet reporterSet →
        (reportedValue : Score → ℝ) →
          Measurable reportedValue → MeasureTheory.Measure ℝ → ProbabilityTheory.Kernel Score ℝ

value:
fun {Score : Type u_1} [MeasurableSpace Score] (reporterSet : Set Score) (hReporterSet : MeasurableSet reporterSet)
    (reportedValue : Score → ℝ) (hReportedValue : Measurable reportedValue) (baseLaw : MeasureTheory.Measure ℝ) =>
  ProbabilityTheory.Kernel.piecewise hReporterSet (ProbabilityTheory.Kernel.deterministic reportedValue hReportedValue)
    (ProbabilityTheory.Kernel.const Score baseLaw)
```

LG21TestOptionalPolicies.lg21OptionalSourceChangedTesterEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] → (Omega → Base) → (Omega → ℝ) → (ℝ → Base → Bool) → (ℝ → Base → Bool) → Omega → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (skill : Omega → ℝ) (currentTake candidateTake : ℝ → Base → Bool) (a : Omega) =>
  {omega : Omega | currentTake (skill omega) (base omega) = false ∧ candidateTake (skill omega) (base omega) = true} a
```

LG21TestOptionalPolicies.lg21OptionalSourceChangedToReportForActionEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) →
            (Omega → ℝ) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → Omega → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (currentTake : ℝ → Base → Bool) (currentReport : Base → ℝ → Bool)
    (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool) (a : Omega) =>
  {omega : Omega |
      candidateTake (skill omega) (base omega) = true ∧
        candidateReport (base omega) (score omega) = true ∧
          (currentTake (skill omega) (base omega) = false ∨ currentReport (base omega) (score omega) = false)}
    a
```

LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} → {Test : Type u_2} → (Base → Test → Bool) → (Base → Test → ℝ) → (Base → ℝ) → Base → Test → ℝ

value:
fun {Base : Type u_1} {Test : Type u_2} (reportDecision : Base → Test → Bool) (reportedPayoff : Base → Test → ℝ)
    (noReportPayoff : Base → ℝ) (base : Base) (test : Test) =>
  if reportDecision base test = true then reportedPayoff base test else noReportPayoff base
```

LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) → (Omega → ℝ) → (Omega → ℝ) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → Omega → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (take : ℝ → Base → Bool) (report : Base → ℝ → Bool) (a : Omega) =>
  {omega : Omega | take (skill omega) (base omega) = false ∨ report (base omega) (score omega) = false} a
```

LG21TestOptionalPolicies.LG21OptionalSequentialCandidateNoReportPBOForAction

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (candidateTake : ℝ → Base → Bool) →
                    (candidateReport : Base → ℝ → Bool) →
                      LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ →
                        0 <
                            sourceLaw
                              (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill candidateTake
                                candidateReport) →
                          Prop

value:
```

```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ)
    (hpositive :
      0 <
        sourceLaw
          (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill candidateTake candidateReport)) =>
  let actionLaw : MeasureTheory.Measure (Base × ℝ × ℝ) :=
    MeasureTheory.Measure.map (fun (omega : Omega) => (base omega, score omega, skill omega))
      (LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw
        (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill candidateTake candidateReport));
  ∀ᵐ (publicBase : Base) ∂MeasureTheory.Measure.map Prod.fst actionLaw,
    candidate.noReportValue publicBase =
      ∫ (scoreSkill : ℝ × ℝ), scoreSkill.2 ∂(ProbabilityTheory.condDistrib Prod.snd Prod.fst actionLaw) publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent

LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [inst : MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → (Omega → ℝ) → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Omega → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (A : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base) (a : Omega) =>
  LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill A.takeDecision A.reportDecision a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent

LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) → (Omega → ℝ) → (Omega → ℝ) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → Omega → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (take : ℝ → Base → Bool) (report : Base → ℝ → Bool) (a : Omega) =>
  {omega : Omega | take (skill omega) (base omega) = true ∧ report (base omega) (score omega) = true} a
```

LG21TestOptionalPolicies.LG21OptionalSequentialCandidateReportPBOForAction

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
```

```
          (base : Omega → Base) →
            (score skill : Omega → ℝ) →
              (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                (candidateTake : ℝ → Base → Bool) →
                  (candidateReport : Base → ℝ → Bool) →
                    LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ →
                      0 <
                          sourceLaw
                            (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake
                              candidateReport) →
                        Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ)
    (hpositive :
      0 <
        sourceLaw
          (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake candidateReport)) =>
  let actionLaw : MeasureTheory.Measure (Base × ℝ × ℝ) :=
    MeasureTheory.Measure.map (fun (omega : Omega) => (base omega, score omega, skill omega))
      (LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw
        (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake candidateReport));
  ∀ᵐ (publicObservation :
    Base × ℝ) ∂MeasureTheory.Measure.map (fun (scoreSkill : Base × ℝ × ℝ) => (scoreSkill.1, scoreSkill.2.1)) actionLaw,
    candidate.reportedValue publicObservation.1 publicObservation.2 =
      ∫ (latentSkill : ℝ),
        latentSkill ∂(ProbabilityTheory.condDistrib (fun (scoreSkill : Base × ℝ × ℝ) => scoreSkill.2.2)
            (fun (scoreSkill : Base × ℝ × ℝ) => (scoreSkill.1, scoreSkill.2.1)) actionLaw)
          publicObservation
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassLocalRecalibratedEntry

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → Bool) →
                    (Base → ℝ → Bool) →
                      Set Base →
                        (ℝ → Base → Bool) →
                          (Base → ℝ → Bool) → LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (currentTake : ℝ → Base → Bool) (currentReport : Base → ℝ → Bool) (region : Set Base)
    (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ) =>
  MeasurableSet region ∧
    ∃ (hregionPositive : 0 < sourceLaw (base ⁻¹' region)),
      sourceLaw
          (base ⁻¹' region ∩
            LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill currentTake currentReport) =
        0 ∧
        (Measurable fun (omega : Omega) => candidateTake (skill omega) (base omega)) ∧
          (Measurable fun (omega : Omega) => candidateReport (base omega) (score omega)) ∧
            (∀ᵐ (omega : Omega) ∂sourceLaw,
              base omega ∉ region →
                candidateTake (skill omega) (base omega) = currentTake (skill omega) (base omega) ∧
                  candidateReport (base omega) (score omega) = currentReport (base omega) (score omega)) ∧
              0 <
                  sourceLaw
                    (LG21TestOptionalPolicies.lg21OptionalSourceChangedTesterEvent base skill currentTake
                        candidateTake ∪
                      LG21TestOptionalPolicies.lg21OptionalSourceChangedToReportForActionEvent base score skill
                        currentTake currentReport candidateTake candidateReport) ∧
                (∀
```

```
    omega ∈
      LG21TestOptionalPolicies.lg21OptionalSourceChangedTesterEvent base skill currentTake
        candidateTake,
    candidate.noReportValue (base omega) <
      LG21TestOptionalPolicies.lg21OptionalCandidateTestExpectedValue candidate (skill omega)
        (base omega)) ∧
  let localLaw : MeasureTheory.Measure Omega :=
    LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw (base ⁻¹' region);
  ((fun (omega : Omega) => candidateReport (base omega) (score omega)) =ᵐ[localLaw]
      fun (omega : Omega) => candidate.reportDecision (base omega) (score omega)) ∧
    ∃ (hreportPositive :
      0 <
        localLaw
          (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake
            candidateReport))
      (hnoReportPositive :
      0 <
        localLaw
          (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill candidateTake
            candidateReport)),
      LG21TestOptionalPolicies.LG21OptionalSequentialCandidateReportPBOForAction localLaw base score
        skill hpublic candidateTake candidateReport candidate hreportPositive ∧
      LG21TestOptionalPolicies.LG21OptionalSequentialCandidateNoReportPBOForAction localLaw base score
          skill hpublic candidateTake candidateReport candidate hnoReportPositive ∧
        LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond localLaw
          (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake
            candidateReport)
          candidate
          fun (P : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ)
            (omega : Omega) =>
          P.noReportValue (base omega) ≤ P.reportedValue (base omega) (score omega)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.LG21OptionalSequentialCandidateNoReportPBOForAction
- LG21TestOptionalPolicies.LG21OptionalSequentialCandidateReportPBOForAction
- LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21OptionalCandidateTestExpectedValue
- LG21TestOptionalPolicies.lg21OptionalSourceChangedTesterEvent
- LG21TestOptionalPolicies.lg21OptionalSourceChangedToReportForActionEvent
- LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
- LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassRecalibratedReportEntry

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → Bool) →
                    (Base → ℝ → Bool) →
                      (ℝ → Base → Bool) →
                        (Base → ℝ → Bool) → LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (currentTake : ℝ → Base → Bool) (currentReport : Base → ℝ → Bool) (candidateTake : ℝ → Base → Bool)
    (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ) =>
  (Measurable fun (omega : Omega) => candidateTake (skill omega) (base omega)) ∧
    (Measurable fun (omega : Omega) => candidateReport (base omega) (score omega)) ∧
      ((fun (omega : Omega) => candidateTake (skill omega) (base omega)) =ᵐ[sourceLaw] fun (omega : Omega) =>
          currentTake (skill omega) (base omega)) ∧
        0 <
          sourceLaw
            (LG21TestOptionalPolicies.lg21OptionalSourceChangedToReportForActionEvent base score skill currentTake
```

```
      currentReport candidateTake candidateReport) ∧
  ∃ (hreportPositive :
    0 <
      sourceLaw
        (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake candidateReport))
    (hnoReportPositive :
    0 <
      sourceLaw
        (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent base score skill candidateTake
          candidateReport)),
    LG21TestOptionalPolicies.LG21OptionalSequentialCandidateReportPBOForAction sourceLaw base score skill
      hpublic candidateTake candidateReport candidate hreportPositive ∧
    LG21TestOptionalPolicies.LG21OptionalSequentialCandidateNoReportPBOForAction sourceLaw base score skill
      hpublic candidateTake candidateReport candidate hnoReportPositive ∧
    LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond sourceLaw
      (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill candidateTake
        candidateReport)
      candidate
      fun (P : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ) (omega : Omega) =>
      P.noReportValue (base omega) ≤ P.reportedValue (base omega) (score omega)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.LG21OptionalSequentialCandidateNoReportPBOForAction
- LG21TestOptionalPolicies.LG21OptionalSequentialCandidateReportPBOForAction
- LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond
- LG21TestOptionalPolicies.lg21OptionalSourceChangedToReportForActionEvent
- LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
- LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → MeasureTheory.Measure ℝ) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (fixedTestLaw : ℝ → Base → MeasureTheory.Measure ℝ) (currentTake : ℝ → Base → Bool)
    (currentReport : Base → ℝ → Bool) =>
  ∀ (region : Set Base) (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ),
    (∀ (latentSkill : ℝ) (publicBase : Base),
      candidate.testLaw latentSkill publicBase = fixedTestLaw latentSkill publicBase) →
    ¬LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassLocalRecalibratedEntry sourceLaw base score skill hpublic
        currentTake currentReport region candidateTake candidateReport candidate
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassLocalRecalibratedEntry

LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassRecalibratedReportEntryForTestLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → MeasureTheory.Measure ℝ) → (ℝ → Base → Bool) → (Base → ℝ → Bool) → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (fixedTestLaw : ℝ → Base → MeasureTheory.Measure ℝ) (currentTake : ℝ → Base → Bool)
    (currentReport : Base → ℝ → Bool) =>
  ∀ (candidateTake : ℝ → Base → Bool) (candidateReport : Base → ℝ → Bool)
    (candidate : LG21TestOptionalPolicies.LG21OptionalCandidateBranchData ℝ Base ℝ),
    (∀ (latentSkill : ℝ) (publicBase : Base),
       candidate.testLaw latentSkill publicBase = fixedTestLaw latentSkill publicBase) →
      ¬LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassRecalibratedReportEntry sourceLaw base score skill hpublic
        currentTake currentReport candidateTake candidateReport candidate
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalCandidateBranchData
- LG21TestOptionalPolicies.LG21OptionalSourcePositiveMassRecalibratedReportEntry

LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium

Declaration location: paper-local

Lean declaration body

```
field actions:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {haccess : 0 < M.accessLaw {true}} →
          {testFeature : Feature} →
            LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature →
              LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
                (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)
```

```
field takeDecision_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  Measurable fun (pair : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) =>
    self.actions.takeDecision pair.2 pair.1
```

```
field reportDecision_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  Measurable fun (pair : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) =>
    self.actions.reportDecision pair.1 pair.2
```

```
field local_recalibrated_entry_stable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
    (Measurable.prodMk (LG21TestOptionalPolicies.lg21ContinuousPopulationBase_measurable testFeature)
      (Measurable.prodMk (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature_measurable testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill_measurable))
    self.actions.testLaw self.actions.takeDecision self.actions.reportDecision
```

```
field recalibrated_report_entry_stable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassRecalibratedReportEntryForTestLaw
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
```

```
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
    (Measurable.prodMk (LG21TestOptionalPolicies.lg21ContinuousPopulationBase_measurable testFeature)
      (Measurable.prodMk (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature_measurable testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill_measurable))
    self.actions.testLaw self.actions.takeDecision self.actions.reportDecision

field test_law_gaussian:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature)
  (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (latentSkill : ℝ),
  self.actions.testLaw latentSkill publicBase = ProbabilityTheory.gaussianReal latentSkill (M.noiseVariance testFeature)

field actual_report_integrable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
        self.actions.reportDecision) →
    MeasureTheory.Integrable LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
      (LG21TestOptionalPolicies.lg21NormalizedRestriction
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision self.actions.reportDecision))

field actual_report_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
        self.actions.reportDecision) →
    (fun (student : Bool × ℝ × (Feature → ℝ)) =>
      self.actions.reportedPayoff (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature
          student)) =ᵐ[LG21TestOptionalPolicies.lg21NormalizedRestriction
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision self.actions.reportDecision)]
      (LG21TestOptionalPolicies.lg21NormalizedRestriction
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        (LG21TestOptionalPolicies.lg21OptionalSourceReportEvent
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
          self.actions.reportDecision))[LG21TestOptionalPolicies.lg21ContinuousPopulationSkill |
        MeasurableSpace.comap
          (fun (student : Bool × ℝ × (Feature → ℝ)) =>
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student,
              LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student))
          inferInstance]

field actual_noReport_integrable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
        self.actions.reportDecision) →
    MeasureTheory.Integrable LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
      (LG21TestOptionalPolicies.lg21NormalizedRestriction
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision self.actions.reportDecision))

field actual_noReport_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
```

```
      (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
      LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
      self.actions.reportDecision) →
  (fun (student : Bool × ℝ × (Feature → ℝ)) =>
    self.actions.noReportPayoff
      (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature
        student)) =ᵐ[LG21TestOptionalPolicies.lg21NormalizedRestriction
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
    (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
      (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
      (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
      LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision self.actions.reportDecision)]
  (LG21TestOptionalPolicies.lg21NormalizedRestriction
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.actions.takeDecision
        self.actions.reportDecision))[LG21TestOptionalPolicies.lg21ContinuousPopulationSkill |
    MeasurableSpace.comap (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature) inferInstance]
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw
- LG21TestOptionalPolicies.LG21OptionalSourceStableAgainstPositiveMassRecalibratedReportEntryForTestLaw
- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21OptionalSourceNoReportEvent
- LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

LG21TestOptionalPolicies.lg21OptionalCandidateReportBranch

Declaration location: paper-local

Lean declaration body

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [inst : MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → (Omega → ℝ) → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Omega → Prop
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (A : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base) (a : Omega) =>
  LG21TestOptionalPolicies.lg21OptionalSourceReportEvent base score skill A.takeDecision A.reportDecision a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.lg21OptionalSourceReportEvent

LG21TestOptionalPolicies.lg21OptionalSourceTakeExpectedPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        (Skill → Base → MeasureTheory.Measure Test) →
          (Base → Test → Bool) → (Base → Test → ℝ) → (Base → ℝ) → Skill → Base → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (testLaw : Skill → Base → MeasureTheory.Measure Test) (reportDecision : Base → Test → Bool)
    (reportedPayoff : Base → Test → ℝ) (noReportPayoff : Base → ℝ) (skill : Skill) (base : Base) =>
  ∫ (test : Test),
    LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff reportDecision reportedPayoff noReportPayoff base
      test ∂testLaw skill base
```

Direct retained dependencies

- LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff

LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [inst : MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → (Omega → ℝ) → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Omega → ℝ

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (A : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base) (a : Omega) =>
  if A.takeDecision (skill a) (base a) = true ∧ A.reportDecision (base a) (score a) = true then
    A.reportedPayoff (base a) (score a)
  else A.noReportPayoff (base a)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions

LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → Base → ℝ → ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (A : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base)
    (publicBase : Base) (observedScore : ℝ) =>
  max (A.reportedPayoff publicBase observedScore) (A.noReportPayoff publicBase)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions

LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base → ℝ → Base → ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (A : LG21TestOptionalPolicies.LG21OptionalSourceTimedActions Base)
    (latentSkill : ℝ) (publicBase : Base) =>
  ∫ (observedScore : ℝ),
    LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationPayoff A publicBase
      observedScore ∂A.testLaw latentSkill publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationPayoff

LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate

Declaration location: paper-local

Lean declaration body

```
field source_timed:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {haccess : 0 < M.accessLaw {true}} →
          {testFeature : Feature} →
            LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature →
              LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature

field active_report_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions)

field best_continuation_integrable_if_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ (latentSkill : ℝ) (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
      MeasureTheory.Integrable
        (fun (observedScore : ℝ) =>
          LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationPayoff self.source_timed.actions publicBase
            observedScore)
        (self.source_timed.actions.testLaw latentSkill publicBase)

field report_members_weakly_respond_if_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        (LG21TestOptionalPolicies.lg21OptionalCandidateReportBranch
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions),
      self.source_timed.actions.noReportPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        self.source_timed.actions.reportedPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student)

field withholding_members_weakly_respond_if_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
```

```
        (Feature →
          ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
      {student : Bool × ℝ × (Feature → ℝ) |
        self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          true ∧
        self.source_timed.actions.reportDecision
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student) =
          false},
    self.source_timed.actions.reportedPayoff
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student) ≤
      self.source_timed.actions.noReportPayoff
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field take_members_weakly_respond_if_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            true},
      self.source_timed.actions.noReportPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff self.source_timed.actions
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field noTake_members_weakly_respond_if_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            false},
      LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff self.source_timed.actions
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        self.source_timed.actions.noReportPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field report_outsiders_closed_under_strict_gain_if_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    (∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            true ∧
          self.source_timed.actions.reportDecision
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student) =
            false},
      self.source_timed.actions.reportedPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature student) ≤
        self.source_timed.actions.noReportPayoff
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)) ∧
```

```
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          false},
      LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff self.source_timed.actions
        (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
      self.source_timed.actions.noReportPayoff
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field noReport_outsiders_closed_under_strict_gain_if_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
        LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.source_timed.actions) →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.source_timed.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          true},
      self.source_timed.actions.noReportPayoff
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
      LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff self.source_timed.actions
        (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21OptionalCandidateNoReportBranch
- LG21TestOptionalPolicies.lg21OptionalCandidateReportBranch
- LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationExpectedPayoff
- LG21TestOptionalPolicies.lg21OptionalSourceTimedBestContinuationPayoff

LG21TestOptionalPolicies.LG21OptionalFibrewiseActiveBranchSelection

Declaration location: paper-local

Lean declaration body

```
field selected_self_enforcing:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  {selected : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature},
  LG21TestOptionalPolicies.LG21OptionalFibrewiseActiveBranchSelection M haccess testFeature selected →
    ∃ (candidate :
      LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
      candidate.source_timed = selected

field maximal_active_branch_ae:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  {selected : LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium M haccess testFeature},
```

```
LG21TestOptionalPolicies.LG21OptionalFibrewiseActiveBranchSelection M haccess testFeature selected →
  let law : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M;
  let base : Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ :=
    LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature;
  let score : Bool × ℝ × (Feature → ℝ) → ℝ := LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature;
  let skill : Bool × ℝ × (Feature → ℝ) → ℝ := LG21TestOptionalPolicies.lg21ContinuousPopulationSkill;
  ∀
    (candidate :
      LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)),
    MeasurableSet region →
      0 < law (base ⁻¹' region) →
        (∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂law.restrict (base ⁻¹' region),
          selected.actions.takeDecision (skill student) (base student) = true ∧
            selected.actions.reportDecision (base student) (score student) = true →
          candidate.source_timed.actions.takeDecision (skill student) (base student) = true ∧
            candidate.source_timed.actions.reportDecision (base student) (score student) = true) →
        ∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂law.restrict (base ⁻¹' region),
          candidate.source_timed.actions.takeDecision (skill student) (base student) = true ∧
            candidate.source_timed.actions.reportDecision (base student) (score student) = true →
          selected.actions.takeDecision (skill student) (base student) = true ∧
            selected.actions.reportDecision (base student) (score student) = true
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.LG21OptionalSourceTimedPositiveMassSelfEnforcingCandidate
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile

Declaration location: paper-local

Lean declaration body

```
field selected:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source →
          LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium source.population
            source.access_positive source.test_feature

field active_branch_selection:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature}
  (self : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
  LG21TestOptionalPolicies.LG21OptionalFibrewiseActiveBranchSelection source.population source.access_positive
    source.test_feature self.selected
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21OptionalFibrewiseActiveBranchSelection

LG21TestOptionalPolicies.lg21P42SourceAccessEstimateMean

Declaration location: paper-local

Lean declaration body

type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → Base → ℝ → ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (M : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base)
    (base : Base) (skill : ℝ) =>
  (M.basePosteriorMean base / ↑M.basePosteriorVariance + skill / ↑M.testNoiseVariance) /
    (1 / ↑M.basePosteriorVariance + 1 / ↑M.testNoiseVariance)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel

LG21TestOptionalPolicies.lg21P42SourceDisplayedAccessEstimateVariance

Declaration location: paper-local

Lean declaration body

type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → NNReal

value:
fun {Base : Type u_1} [MeasurableSpace Base] (M : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base) =>
  NNReal.mk
    ((1 / ↑M.basePosteriorVariance + 2 / ↑M.testNoiseVariance) /
      (1 / ↑M.basePosteriorVariance + 1 / ↑M.testNoiseVariance) ^ 2)
    (LG21TestOptionalPolicies.lg21P42SourceDisplayedAccessEstimateVariance._proof_2 M)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel

LG21TestOptionalPolicies.lg21P42SourceAccessEstimateLaw

Declaration location: paper-local

Lean declaration body

type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → Base → ℝ → MeasureTheory.Measure ℝ

value:
fun {Base : Type u_1} [MeasurableSpace Base] (M : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base)
    (base : Base) (skill : ℝ) =>
  ProbabilityTheory.gaussianReal (LG21TestOptionalPolicies.lg21P42SourceAccessEstimateMean M base skill)
    (LG21TestOptionalPolicies.lg21P42SourceDisplayedAccessEstimateVariance M)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel
- LG21TestOptionalPolicies.lg21P42SourceAccessEstimateMean
- LG21TestOptionalPolicies.lg21P42SourceDisplayedAccessEstimateVariance

LG21TestOptionalPolicies.lg21P42ObservedScoreLatentSkillFair

Declaration location: paper-local

Lean declaration body

type:
{Base : Type u_1} →
  [inst : MeasurableSpace Base] → LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base → Prop

value:
fun {Base : Type u_1} [MeasurableSpace Base] (M : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base) =>
  ∀ (base : Base) (skill : ℝ),
    LG21TestOptionalPolicies.lg21P42SourceAccessEstimateLaw M base skill = M.noAccessEstimateKernel base

Direct retained dependencies

- LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel
- LG21TestOptionalPolicies.lg21P42SourceAccessEstimateLaw

LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ → Omega → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (skill : Omega → ℝ) (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (a : Omega) =>
  {omega : Omega | candidate.takeDecision (skill omega) (base omega) = false} a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData

LG21TestOptionalPolicies.LG21ReportRequiredCandidateNoTakePBO

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ) →
                    0 <
                        sourceLaw
                          (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent base skill candidate) →
                      Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (hpositive :
      0 < sourceLaw (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent base skill candidate)) =>
  let actionLaw : MeasureTheory.Measure (Base × ℝ × ℝ) :=
    MeasureTheory.Measure.map (fun (omega : Omega) => (base omega, score omega, skill omega))
      (LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw
        (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent base skill candidate));
  ∀ᵐ (publicBase : Base) ∂MeasureTheory.Measure.map Prod.fst actionLaw,
    candidate.noReportPayoff publicBase =
      ∫ (latentSkill : ℝ),
        latentSkill ∂(ProbabilityTheory.condDistrib (fun (scoreSkill : Base × ℝ × ℝ) => scoreSkill.2.2) Prod.fst
            actionLaw)
          publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent

LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) → LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ → Omega → Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (skill : Omega → ℝ) (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (a : Omega) =>
  {omega : Omega | candidate.takeDecision (skill omega) (base omega) = true} a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData

LG21TestOptionalPolicies.LG21ReportRequiredCandidateReportedPBO

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ) →
                    0 <
                      sourceLaw
                        (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent base skill candidate) →
                    Prop

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (hpositive :
      0 < sourceLaw (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent base skill candidate)) =>
  let actionLaw : MeasureTheory.Measure (Base × ℝ × ℝ) :=
    MeasureTheory.Measure.map (fun (omega : Omega) => (base omega, score omega, skill omega))
      (LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw
        (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent base skill candidate));
  ∀ᵐ (publicObservation :
    Base × ℝ) ∂MeasureTheory.Measure.map (fun (scoreSkill : Base × ℝ × ℝ) => (scoreSkill.1, scoreSkill.2.1)) actionLaw,
    candidate.reportedPayoff publicObservation.1 publicObservation.2 =
      ∫ (latentSkill : ℝ),
        latentSkill ∂(ProbabilityTheory.condDistrib (fun (scoreSkill : Base × ℝ × ℝ) => scoreSkill.2.2)
            (fun (scoreSkill : Base × ℝ × ℝ) => (scoreSkill.1, scoreSkill.2.1)) actionLaw)
          publicObservation
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent

LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel

Declaration location: paper-local

Lean declaration body

```
type:
(takerSet : Set ℝ) →
  MeasurableSet takerSet → ProbabilityTheory.Kernel ℝ ℝ → MeasureTheory.Measure ℝ → ProbabilityTheory.Kernel ℝ ℝ

value:
fun (takerSet : Set ℝ) (hTakerSet : MeasurableSet takerSet) (reportedKernel : ProbabilityTheory.Kernel ℝ ℝ)
    (baseLaw : MeasureTheory.Measure ℝ) =>
  ProbabilityTheory.Kernel.piecewise hTakerSet reportedKernel (ProbabilityTheory.Kernel.const ℝ baseLaw)
```

LG21TestOptionalPolicies.lg21ReportRequiredSequentialAccessAction

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) →
            LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ →
              Omega → LG21TestOptionalPolicies.LG21AccessAction

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (skill : Omega → ℝ) (E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (a : Omega) =>
  if E.takeDecision (skill a) (base a) = true then LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
  else LG21TestOptionalPolicies.LG21AccessAction.noTake
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.noTake
- LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData

LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [MeasurableSpace Omega] →
      [MeasurableSpace Base] →
        (Omega → Base) →
          (Omega → ℝ) →
            (Omega → ℝ) → LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ → Omega → ℝ

value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base] (base : Omega → Base)
    (score skill : Omega → ℝ) (E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ)
    (a : Omega) =>
  LG21TestOptionalPolicies.lg21ObservedAccessActualOutput base score
    (LG21TestOptionalPolicies.lg21ReportRequiredSequentialAccessAction base skill E) E.reportedPayoff E.noReportPayoff a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
- LG21TestOptionalPolicies.lg21ReportRequiredSequentialAccessAction

LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test → Skill → Base → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData Skill Base Test) (skill : Skill)
    (base : Base) =>
  ∫ (test : Test), E.reportedPayoff base test ∂E.testLaw skill base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData

LG21TestOptionalPolicies.LG21ReportRequiredRecalibratedSourceBranchEntry

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ) =>
  ∃ (htakePositive :
    0 < sourceLaw (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent base skill candidate))
    (hnoTakePositive :
    0 < sourceLaw (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent base skill candidate)),
    LG21TestOptionalPolicies.LG21ReportRequiredCandidateReportedPBO sourceLaw base score skill hpublic candidate
        htakePositive ∧
      LG21TestOptionalPolicies.LG21ReportRequiredCandidateNoTakePBO sourceLaw base score skill hpublic candidate
          hnoTakePositive ∧
        LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond sourceLaw
          (LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent base skill candidate) candidate
          fun (C : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ) (omega : Omega) =>
          C.noReportPayoff (base omega) ≤
            LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff C (skill omega) (base omega)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredCandidateNoTakePBO
- LG21TestOptionalPolicies.LG21ReportRequiredCandidateReportedPBO
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.PositiveMassBranchMembersBestRespond
- LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceNoTakeEvent
- LG21TestOptionalPolicies.lg21ReportRequiredCandidateSourceTakeEvent
- LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff

LG21TestOptionalPolicies.LG21ReportRequiredSourcePositiveMassLocalRecalibratedEntry

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → Bool) →
                    Set Base → LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (currentTake : ℝ → Base → Bool) (region : Set Base)
    (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ) =>
  MeasurableSet region ∧
    ∃ (hregionPositive : 0 < sourceLaw (base ⁻¹' region)),
      sourceLaw (base ⁻¹' region ∩ {omega : Omega | currentTake (skill omega) (base omega) = true}) = 0 ∧
```

```
let localLaw : MeasureTheory.Measure Omega :=
  LG21TestOptionalPolicies.lg21NormalizedRestriction sourceLaw (base ⁻¹' region);
(∀ᵐ (omega : Omega) ∂localLaw, currentTake (skill omega) (base omega) = false) ∧
  LG21TestOptionalPolicies.LG21ReportRequiredRecalibratedSourceBranchEntry localLaw base score skill hpublic
    candidate
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredRecalibratedSourceBranchEntry
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21ReportRequiredSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw

Declaration location: paper-local

Lean declaration body

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    [inst : MeasurableSpace Omega] →
      [inst_1 : MeasurableSpace Base] →
        (sourceLaw : MeasureTheory.Measure Omega) →
          [MeasureTheory.IsProbabilityMeasure sourceLaw] →
            (base : Omega → Base) →
              (score skill : Omega → ℝ) →
                (Measurable fun (omega : Omega) => (base omega, score omega, skill omega)) →
                  (ℝ → Base → MeasureTheory.Measure ℝ) → (ℝ → Base → Bool) → Prop
```

```
value:
fun {Omega : Type u_1} {Base : Type u_2} [MeasurableSpace Omega] [MeasurableSpace Base]
    (sourceLaw : MeasureTheory.Measure Omega) [MeasureTheory.IsProbabilityMeasure sourceLaw] (base : Omega → Base)
    (score skill : Omega → ℝ) (hpublic : Measurable fun (omega : Omega) => (base omega, score omega, skill omega))
    (fixedTestLaw : ℝ → Base → MeasureTheory.Measure ℝ) (currentTake : ℝ → Base → Bool) =>
  ∀ (region : Set Base) (candidate : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ),
    (∀ (latentSkill : ℝ) (publicBase : Base),
       candidate.testLaw latentSkill publicBase = fixedTestLaw latentSkill publicBase) →
     ¬LG21TestOptionalPolicies.LG21ReportRequiredSourcePositiveMassLocalRecalibratedEntry sourceLaw base score skill
        hpublic currentTake region candidate
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.LG21ReportRequiredSourcePositiveMassLocalRecalibratedEntry

LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium

Declaration location: paper-local

Lean declaration body

```
field base_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
    testNoiseVariance →
    Measurable base
```

```
field score_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
    testNoiseVariance →
    Measurable score
```

```
field skill_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
    testNoiseVariance →
    Measurable skill
```

```
field action_measurable:
```

∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    Measurable fun (profileSkill : Base × ℝ) => E.takeDecision profileSkill.2 profileSkill.1

field reportedPayoff_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    Measurable fun (publicScore : Base × ℝ) => E.reportedPayoff publicScore.1 publicScore.2

field noReportPayoff_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    Measurable E.noReportPayoff

field test_law_gaussian:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    ∀ (publicBase : Base) (latentSkill : ℝ),
      E.testLaw latentSkill publicBase = ProbabilityTheory.gaussianReal latentSkill testNoiseVariance

field reported_integrable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    0 < rawLaw {omega : Omega | E.takeDecision (skill omega) (base omega) = true} →
      MeasureTheory.Integrable skill
        (LG21TestOptionalPolicies.lg21NormalizedRestriction rawLaw
          {omega : Omega | E.takeDecision (skill omega) (base omega) = true})

field reported_pbo:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    0 < rawLaw {omega : Omega | E.takeDecision (skill omega) (base omega) = true} →
      LG21TestOptionalPolicies.LG21FullPublicReportRequiredReportedPBO rawLaw base score skill
        (fun (publicBase : Base) (latentSkill : ℝ) => E.takeDecision latentSkill publicBase) E.reportedPayoff

field noTake_integrable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    0 < rawLaw {omega : Omega | E.takeDecision (skill omega) (base omega) = false} →
      MeasureTheory.Integrable skill
        (LG21TestOptionalPolicies.lg21NormalizedRestriction rawLaw
          {omega : Omega | E.takeDecision (skill omega) (base omega) = false})

field noTake_pbo:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal},
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance →
    0 < rawLaw {omega : Omega | E.takeDecision (skill omega) (base omega) = false} →
      LG21TestOptionalPolicies.LG21FullPublicReportRequiredNoTakePBO rawLaw base skill
        (fun (publicBase : Base) (latentSkill : ℝ) => E.takeDecision latentSkill publicBase) E.noReportPayoff

field positive_mass_recalibrated_stable:
∀ {Omega : Type u_1} {Base : Type u_2} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  {rawLaw : MeasureTheory.Measure Omega} [inst_2 : MeasureTheory.IsProbabilityMeasure rawLaw]
  [inst_3 : MeasureTheory.IsFiniteMeasure rawLaw] {base : Omega → Base} {score skill : Omega → ℝ}
  {E : LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ Base ℝ} {testNoiseVariance : NNReal}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium rawLaw base score skill E
      testNoiseVariance),
  LG21TestOptionalPolicies.LG21ReportRequiredSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw rawLaw base
    score skill (Measurable.prodMk self.base_measurable (Measurable.prodMk self.score_measurable self.skill_measurable))
    E.testLaw fun (latentSkill : ℝ) (publicBase : Base) => E.takeDecision latentSkill publicBase

Direct retained dependencies

- LG21TestOptionalPolicies.LG21FullPublicReportRequiredNoTakePBO
- LG21TestOptionalPolicies.LG21FullPublicReportRequiredReportedPBO
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.LG21ReportRequiredSourceStableAgainstPositiveMassLocalRecalibratedEntryForTestLaw
- LG21TestOptionalPolicies.lg21NormalizedRestriction

LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate

Declaration location: paper-local

Lean declaration body

```
field selected:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {haccess : 0 < M.accessLaw {true}} →
          {testFeature : Feature} →
            LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess
              testFeature →
              LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ
                (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ

field source_timed:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.selected (M.noiseVariance testFeature)

field active_take_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      {student : Bool × ℝ × (Feature → ℝ) |
        self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          true}

field take_members_weakly_respond_if_noTake_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      {student : Bool × ℝ × (Feature → ℝ) |
        self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          false} →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
      {student : Bool × ℝ × (Feature → ℝ) |
        self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          true},
      self.selected.noReportPayoff (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff self.selected
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field noTake_members_weakly_respond_if_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      {student : Bool × ℝ × (Feature → ℝ) |
        self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
          false} →
```

```
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            false},
      LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff self.selected
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        self.selected.noReportPayoff (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field take_outsiders_closed_under_strict_gain_if_noTake_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        {student : Bool × ℝ × (Feature → ℝ) |
          self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            false} →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            false},
      LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff self.selected
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        self.selected.noReportPayoff (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)

field noTake_outsiders_closed_under_strict_gain_if_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  (self :
    LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
  0 <
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
        {student : Bool × ℝ × (Feature → ℝ) |
          self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            false} →
    ∀ᵐ (student :
      Bool ×
        ℝ ×
          (Feature →
            ℝ)) ∂(LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M).restrict
        {student : Bool × ℝ × (Feature → ℝ) |
          self.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) =
            true},
      self.selected.noReportPayoff (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student) ≤
        LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff self.selected
          (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature student)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21ReportRequiredSequentialTakeExpectedPayoff

LG21TestOptionalPolicies.LG21ReportRequiredFibrewiseActiveBranchSelection

Declaration location: paper-local

Lean declaration body

```
field selected_self_enforcing:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  {selected :
    LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ
      (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ}
  {selectedSource :
    LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
      (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
      LG21TestOptionalPolicies.lg21ContinuousPopulationSkill selected (M.noiseVariance testFeature)},
  LG21TestOptionalPolicies.LG21ReportRequiredFibrewiseActiveBranchSelection M haccess testFeature selected
      selectedSource →
    ∃ (candidate :
      LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature),
      candidate.selected = selected

field maximal_active_branch_ae:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {haccess : 0 < M.accessLaw {true}}
  {testFeature : Feature}
  {selected :
    LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ
      (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ}
  {selectedSource :
    LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
      (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M)
      (LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature)
      (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature testFeature)
      LG21TestOptionalPolicies.lg21ContinuousPopulationSkill selected (M.noiseVariance testFeature)},
  LG21TestOptionalPolicies.LG21ReportRequiredFibrewiseActiveBranchSelection M haccess testFeature selected
      selectedSource →
    let law : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
      LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw M;
    let base : Bool × ℝ × (Feature → ℝ) → LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ :=
      LG21TestOptionalPolicies.lg21ContinuousPopulationBase testFeature;
    let skill : Bool × ℝ × (Feature → ℝ) → ℝ := LG21TestOptionalPolicies.lg21ContinuousPopulationSkill;
    ∀
      (candidate :
        LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate M haccess testFeature)
      (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)),
      MeasurableSet region →
        0 < law (base ⁻¹' region) →
          (∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂law.restrict (base ⁻¹' region),
              selected.takeDecision (skill student) (base student) = true →
                candidate.selected.takeDecision (skill student) (base student) = true) →
            ∀ᵐ (student : Bool × ℝ × (Feature → ℝ)) ∂law.restrict (base ⁻¹' region),
              candidate.selected.takeDecision (skill student) (base student) = true →
                selected.takeDecision (skill student) (base student) = true
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.LG21ReportRequiredSourceTimedPositiveMassSelfEnforcingCandidate
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile

Declaration location: paper-local

Lean declaration body

```
field selected:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature} →
        LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source →
          LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData ℝ
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) ℝ
```

```
field positive_branch_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
  LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
    (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
    (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
    LG21TestOptionalPolicies.lg21ContinuousPopulationSkill self.selected
    (source.population.noiseVariance source.test_feature)

field active_branch_selection:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
  LG21TestOptionalPolicies.LG21ReportRequiredFibrewiseActiveBranchSelection source.population source.access_positive
    source.test_feature self.selected self.positive_branch_pbo
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ReportRequiredFibrewiseActiveBranchSelection
- LG21TestOptionalPolicies.LG21ReportRequiredPositiveMassRefinedSourceEquilibrium
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.PaperInterface.lemma4_1_observed_access_strategy_proofnessSpec

Declaration location: paper-local

Lean declaration body

type:
Prop

value:
```
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature),
  Nonempty (LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source) ∧
    Nonempty (LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source) ∧
      (∀ (mandatory : LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium source.population),
        ∀ᵐ (student :
          Bool ×
            ℝ × (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
          mandatory.action student = LG21TestOptionalPolicies.LG21AccessAction.takeAndReport) ∧
        (∀ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
          ∀ᵐ (student :
            Bool ×
              ℝ ×
                (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
            optionalProfile.selected.actions.takeDecision
                (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
                (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
              true ∧
            optionalProfile.selected.actions.reportDecision
                (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
                (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student) =
              true) ∧
          (∀ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
            ∀ᵐ (student :
              Bool ×
                ℝ ×
                  (Feature →
                    ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
              reportRequiredProfile.selected.takeDecision
                (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
                (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
              true) ∧
            (∀ (left right : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
              ∀ᵐ (student :
                Bool ×
                  ℝ ×
                    (Feature →
                      ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
                left.selected.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
```

(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
right.selected.actions.takeDecision
(LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) ∧
left.selected.actions.reportDecision
(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
(LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student) =
right.selected.actions.reportDecision
(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
(LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student)) ∧
∀ (left right : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
∀ᵐ (student :
Bool ×
ℝ ×
(Feature →
ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
left.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
right.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
(LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.takeAndReport
- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile
- LG21TestOptionalPolicies.LG21OptionalSourceTimedActions
- LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile
- LG21TestOptionalPolicies.LG21ReportRequiredSequentialEquilibriumData
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill

LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff

Declaration location: paper-local

Lean declaration body

type:
{Skill : Type u_1} →
{Base : Type u_2} →
{Test : Type u_3} →
[inst : MeasurableSpace Test] → (Skill → Base → MeasureTheory.Measure Test) → (Base → Test → ℝ) → Skill → Base → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
(testLaw : Skill → Base → MeasureTheory.Measure Test) (reportedPayoff : Base → Test → ℝ) (skill : Skill)
(base : Base) =>
MeasureTheory.integral (testLaw skill base) (reportedPayoff base)

LG21TestOptionalPolicies.LG21ReportRequiredPointwiseBestResponse

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        (Skill → Base → MeasureTheory.Measure Test) → (Skill → Base → Bool) → (Base → Test → ℝ) → (Base → ℝ) → Prop
```

```
value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (testLaw : Skill → Base → MeasureTheory.Measure Test) (takeDecision : Skill → Base → Bool)
    (reportedPayoff : Base → Test → ℝ) (noReportPayoff : Base → ℝ) =>
  ∀ (base : Base),
    AppliedModelingLib.NoProfitableBinaryChoiceDeviation (fun (skill : Skill) => takeDecision skill base = true)
      (fun (skill : Skill) =>
        LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff testLaw reportedPayoff skill base)
      fun (_skill : Skill) => noReportPayoff base
```

Direct retained dependencies

- AppliedModelingLib.NoProfitableBinaryChoiceDeviation
- LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff

LG21TestOptionalPolicies.lg21ReportRequiredTailChangedTesterEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
            Bool × ℝ × (Feature → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (currentTake candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  {student : Bool × ℝ × (Feature → ℝ) |
      student.1 = true ∧
        currentTake (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
            (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
          false ∧
        candidateTake (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
            (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
          true}
    a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase

LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          Bool × ℝ × (Feature → ℝ) → Prop
```

```
value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent testFeature candidateTake
    (fun (x : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (x_1 : ℝ) => true) a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateNoReportEvent

LG21TestOptionalPolicies.LG21ReportRequiredTailNoReportPBOOn

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) →
            0 <
                (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
                  (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region) →
              (candidate :
                  LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData ℝ
                    (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ) →
                0 <
                    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
                      (LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent testFeature
                        candidate.takeDecision) →
                  Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
    (hregionPositive :
      0 <
        (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
          (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region))
    (candidate :
      LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData ℝ
        (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ)
    (hnoReportPositive :
      0 <
        (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
          (LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent testFeature candidate.takeDecision)) =>
  have rawLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
  let localLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region;
  let noReportEvent : Set (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent testFeature candidate.takeDecision;
  let actionLaw : MeasureTheory.Measure ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ) :=
    MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessBaseSkillObservation testFeature)
      (LG21TestOptionalPolicies.lg21NormalizedRestriction localLaw noReportEvent);
  ∀ᵐ (publicBase :
    LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ∂MeasureTheory.Measure.map Prod.fst actionLaw,
    candidate.noReportPayoff publicBase =
      ∫ (latentSkill : ℝ), latentSkill ∂(ProbabilityTheory.condDistrib Prod.snd Prod.fst actionLaw) publicBase
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessBaseSkillObservation
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent

LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [Fintype Feature] →
    [DecidableEq Feature] →
      (testFeature : Feature) →
        (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) →
          Bool × ℝ × (Feature → ℝ) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature] (testFeature : Feature)
    (candidateTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool)
    (a : Bool × ℝ × (Feature → ℝ)) =>
  LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent testFeature candidateTake
    (fun (x : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) (x_1 : ℝ) => true) a
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.lg21HiddenAccessRawCandidateReportEvent

LG21TestOptionalPolicies.LG21ReportRequiredTailReportPBOOn

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) →
        (testFeature : Feature) →
          (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) →
            0 <
                (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
                  (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region) →
              (candidate :
                  LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData ℝ
                    (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ) →
                0 <
                    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
                      (LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent testFeature candidate.takeDecision) →
                  Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature) (testFeature : Feature)
    (region : Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
    (hregionPositive :
      0 <
        (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
          (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature region))
    (candidate :
      LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData ℝ
        (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ)
    (hreportPositive :
      0 <
        (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region)
          (LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent testFeature candidate.takeDecision)) =>
  have rawLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M;
  let localLaw : MeasureTheory.Measure (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature region;
  let reportEvent : Set (Bool × ℝ × (Feature → ℝ)) :=
    LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent testFeature candidate.takeDecision;
  let actionLaw :
    MeasureTheory.Measure ((LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) :=
    MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation testFeature)
      (LG21TestOptionalPolicies.lg21NormalizedRestriction localLaw reportEvent);
  ∀ᵐ (publicObservation :
    (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ×
      ℝ) ∂MeasureTheory.Measure.map
      (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
        (scoreSkill.1, scoreSkill.2.1))
      actionLaw,
    candidate.reportedPayoff publicObservation.1 publicObservation.2 =
      ∫ (latentSkill : ℝ),
        latentSkill ∂(ProbabilityTheory.condDistrib
            (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
              scoreSkill.2.2)
            (fun (scoreSkill : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) × ℝ × ℝ) =>
              (scoreSkill.1, scoreSkill.2.1))
            actionLaw)
          publicObservation
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21NormalizedRestriction
- LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent

LG21TestOptionalPolicies.lg21ReportRequiredTailTakeExpectedPayoff

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test → Skill → Base → ℝ

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (candidate : LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData Skill Base Test) (skill : Skill)
    (base : Base) =>
  ∫ (test : Test), candidate.reportedPayoff base test ∂candidate.testLaw skill base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData

LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry

Declaration location: paper-local

Lean declaration body

```
field region:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool} →
            LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake →
              Set (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)

field region_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake), MeasurableSet self.region

field region_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  0 <
    (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M)
      (LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent testFeature self.region)

field current_take_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  Measurable fun (pair : ℝ × (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
    currentTake pair.1 pair.2

field current_take_zero:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
```

```
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
     {student : Bool × ℝ × (Feature → ℝ) |
      student.1 = true ∧
        currentTake (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
           (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) =
         true} =
   0

field candidate:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [inst_1 : DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool} →
            LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake →
              LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData ℝ
                (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) ℝ

field candidate_test_law:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake) (latentSkill : ℝ)
  (publicBase : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ),
  self.candidate.testLaw latentSkill publicBase =
    ProbabilityTheory.gaussianReal latentSkill (M.noiseVariance testFeature)

field candidate_take_measurable:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  Measurable fun (pair : ℝ × (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)) =>
    self.candidate.takeDecision pair.1 pair.2

field candidate_report_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  0 <
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent testFeature self.candidate.takeDecision)

field candidate_noReport_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  0 <
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent testFeature self.candidate.takeDecision)

field candidate_report_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  LG21TestOptionalPolicies.LG21ReportRequiredTailReportPBOOn M testFeature self.region self.region_positive
    self.candidate self.candidate_report_positive

field candidate_noReport_pbo:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  LG21TestOptionalPolicies.LG21ReportRequiredTailNoReportPBOOn M testFeature self.region self.region_positive
    self.candidate self.candidate_noReport_positive

field candidate_changed_taker_positive:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  0 <
    (LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region)
      (LG21TestOptionalPolicies.lg21ReportRequiredTailChangedTesterEvent testFeature currentTake
        self.candidate.takeDecision)

field candidate_changed_taker_strict_gain:
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
  {currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool}
  (self : LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake),
  ∀ᵐ (student :
    Bool ×
      ℝ ×
        (Feature →
          ℝ)) ∂(LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw M testFeature self.region).restrict
    (LG21TestOptionalPolicies.lg21ReportRequiredTailChangedTesterEvent testFeature currentTake
      self.candidate.takeDecision),
   self.candidate.noReportPayoff (LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2) <
     LG21TestOptionalPolicies.lg21ReportRequiredTailTakeExpectedPayoff self.candidate
       (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
```

(LG21TestOptionalPolicies.lg21HiddenAccessStudentBase testFeature student.2)

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredTailCandidateData
- LG21TestOptionalPolicies.LG21ReportRequiredTailNoReportPBOOn
- LG21TestOptionalPolicies.LG21ReportRequiredTailReportPBOOn
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21HiddenAccessBaseRegionEvent
- LG21TestOptionalPolicies.lg21HiddenAccessLocalRawLaw
- LG21TestOptionalPolicies.lg21HiddenAccessStudentBase
- LG21TestOptionalPolicies.lg21ReportRequiredTailChangedTesterEvent
- LG21TestOptionalPolicies.lg21ReportRequiredTailNoReportEvent
- LG21TestOptionalPolicies.lg21ReportRequiredTailReportEvent
- LG21TestOptionalPolicies.lg21ReportRequiredTailTakeExpectedPayoff

LG21TestOptionalPolicies.LG21ReportRequiredStableAgainstLocalTailEntry

Declaration location: paper-local

Lean declaration body

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    [DecidableEq Feature] →
      {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} →
        {testFeature : Feature} →
          (ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) → Prop

value:
fun {Feature : Type u_1} [Fintype Feature] [DecidableEq Feature]
    {M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature} {testFeature : Feature}
    (currentTake : ℝ → (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → Bool) =>
  ¬Nonempty (LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry currentTake)
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredLocalTailEntry

LG21TestOptionalPolicies.lg21ReportingConditionedDemographicallyFair

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) =>
  ∀ (e : R.Equilibrium), R.demographicReporterLaw e = R.demographicNonreporterLaw e
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface

LG21TestOptionalPolicies.lg21ReportingConditionedLatentSkillFair

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) =>
  ∀ (e : R.Equilibrium) (skill : Skill) (base : Base),
    R.latentReporterLaw e skill base = R.latentNonreporterLaw e skill base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface

LG21TestOptionalPolicies.lg21ReportingConditionedObservablyFair

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) =>
  ∀ (e : R.Equilibrium) (base : Base), R.observableReporterLaw e base = R.observableNonreporterLaw e base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface

LG21TestOptionalPolicies.lg21ReportingConditionedTestBlank

Declaration location: paper-local

Lean declaration body

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law) =>
  ∀ (e : R.Equilibrium) (base : Base) (test : Test), R.baseOnlyLaw e base = R.fullFeatureLaw e base test
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface

LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel

Declaration location: paper-local

Lean declaration body

```
field policy:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base →
      LG21TestOptionalPolicies.LG21SourceLawPolicySurface ℝ Base ℝ (MeasureTheory.Measure ℝ)

field latentSkillFair_implies_observablyFair:
```

```
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base),
  LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair self.policy →
    LG21TestOptionalPolicies.lg21SourceLawObservablyFair self.policy

field scoreLaw:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) →
      self.policy.Equilibrium → Base → MeasureTheory.Measure ℝ

field scoreLaw_probability:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), MeasureTheory.IsProbabilityMeasure (self.scoreLaw e base)

field reporterSet:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) → self.policy.Equilibrium → Base → Set ℝ

field reporterSet_measurable:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), MeasurableSet (self.reporterSet e base)

field reporterOutput:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) → self.policy.Equilibrium → Base → ℝ → ℝ

field reporterOutput_measurable:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), Measurable (self.reporterOutput e base)

field fullFeatureLaw_is_dirac:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium) (base : Base)
  (score : ℝ), self.policy.fullFeatureLaw e base score = MeasureTheory.Measure.dirac (self.reporterOutput e base score)

field baseOnlyLaw_probability:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), MeasureTheory.IsProbabilityMeasure (self.policy.baseOnlyLaw e base)

field baseOnlyLaw_integrable:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), MeasureTheory.Integrable id (self.policy.baseOnlyLaw e base)

field operationalMean_integrable:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base),
  MeasureTheory.Integrable
    (fun (score : ℝ) =>
      ∫ (estimate : ℝ),
        estimate ∂(LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (self.reporterSet e base)
            (self.reporterSet_measurable e base) (self.reporterOutput e base) (self.reporterOutput_measurable e base)
            (self.policy.baseOnlyLaw e base))
          score)
    (self.scoreLaw e base)

field mean_tower:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base),
  ∫ (score : ℝ),
      ∫ (estimate : ℝ),
        estimate ∂(LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (self.reporterSet e base)
            (self.reporterSet_measurable e base) (self.reporterOutput e base) (self.reporterOutput_measurable e base)
            (self.policy.baseOnlyLaw e base))
          score ∂self.scoreLaw e base =
    ∫ (estimate : ℝ),
      estimate ∂(self.scoreLaw e base).bind
        ⇑(LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (self.reporterSet e base)
            (self.reporterSet_measurable e base) (self.reporterOutput e base) (self.reporterOutput_measurable e base)
            (self.policy.baseOnlyLaw e base))

field reporter_bestResponse:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base),
  ∀ᵐ (score : ℝ) ∂(self.scoreLaw e base).restrict (self.reporterSet e base),
    ∫ (estimate : ℝ), estimate ∂self.policy.baseOnlyLaw e base ≤ self.reporterOutput e base score

field operationalLaw_eq_observableAccess:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base),
  (self.scoreLaw e base).bind
      ⇑(LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (self.reporterSet e base)
          (self.reporterSet_measurable e base) (self.reporterOutput e base) (self.reporterOutput_measurable e base)
          (self.policy.baseOnlyLaw e base)) =
    self.policy.observableAccessLaw e base
```

field noAccessLaw_eq_baseOnly:
∀ {Base : Type u} [inst : MeasurableSpace Base]
  (self : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base) (e : self.policy.Equilibrium)
  (base : Base), self.policy.observableNoAccessLaw e base = self.policy.baseOnlyLaw e base

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel

Declaration location: paper-local

Lean declaration body

field policy:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base →
      LG21TestOptionalPolicies.LG21SourceLawPolicySurface ℝ Base ℝ (MeasureTheory.Measure ℝ)

field latentSkillFair_implies_observablyFair:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base),
  LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair self.policy →
    LG21TestOptionalPolicies.lg21SourceLawObservablyFair self.policy

field populationMean:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) → self.policy.Equilibrium → Base → ℝ

field populationVariance:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) → self.policy.Equilibrium → Base → NNReal

field populationVariance_ne_zero:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), self.populationVariance e base ≠ 0

field noiseVariance:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) → self.policy.Equilibrium → Base → NNReal

field noiseVariance_ne_zero:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), self.noiseVariance e base ≠ 0

field takerSet:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) → self.policy.Equilibrium → Base → Set ℝ

field takerSet_measurable:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), MeasurableSet (self.takerSet e base)

field reporterOutput:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) → self.policy.Equilibrium → Base → ℝ → ℝ

field reporterOutput_measurable:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), Measurable (self.reporterOutput e base)

field reporterOutput_integrable_all_gaussian_shifts:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base) (mean : ℝ),
  MeasureTheory.Integrable (self.reporterOutput e base)
    (ProbabilityTheory.gaussianReal mean (self.noiseVariance e base))

field fullFeatureLaw_is_dirac:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base) (score : ℝ),
  self.policy.fullFeatureLaw e base score = MeasureTheory.Measure.dirac (self.reporterOutput e base score)

field baseOnlyLaw_probability:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), MeasureTheory.IsProbabilityMeasure (self.policy.baseOnlyLaw e base)

field baseOnlyLaw_integrable:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)

```
  (e : self.policy.Equilibrium) (base : Base), MeasureTheory.Integrable id (self.policy.baseOnlyLaw e base)

field reportedKernel:
{Base : Type u} →
  [inst : MeasurableSpace Base] →
    (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base) →
      self.policy.Equilibrium → Base → ProbabilityTheory.Kernel ℝ ℝ

field reportedKernel_eq_gaussian_map:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base) (skill : ℝ),
  (self.reportedKernel e base) skill =
    MeasureTheory.Measure.map (self.reporterOutput e base)
      (ProbabilityTheory.gaussianReal skill (self.noiseVariance e base))

field operationalMean_integrable:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base),
  MeasureTheory.Integrable
    (fun (skill : ℝ) =>
      ∫ (estimate : ℝ),
        estimate ∂(LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (self.takerSet e base)
            (self.takerSet_measurable e base) (self.reportedKernel e base) (self.policy.baseOnlyLaw e base))
          skill)
    (ProbabilityTheory.gaussianReal (self.populationMean e base) (self.populationVariance e base))

field mean_tower:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base),
  ∫ (skill : ℝ),
      ∫ (estimate : ℝ),
        estimate ∂(LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (self.takerSet e base)
            (self.takerSet_measurable e base) (self.reportedKernel e base) (self.policy.baseOnlyLaw e base))
          skill ∂ProbabilityTheory.gaussianReal (self.populationMean e base) (self.populationVariance e base) =
    ∫ (estimate : ℝ),
      estimate ∂(ProbabilityTheory.gaussianReal (self.populationMean e base) (self.populationVariance e base)).bind
        ⇑(LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (self.takerSet e base)
            (self.takerSet_measurable e base) (self.reportedKernel e base) (self.policy.baseOnlyLaw e base))

field taker_bestResponse:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base),
  ∀ᵐ (skill :
    ℝ) ∂(ProbabilityTheory.gaussianReal (self.populationMean e base) (self.populationVariance e base)).restrict
      (self.takerSet e base),
    ∫ (estimate : ℝ), estimate ∂self.policy.baseOnlyLaw e base ≤
      GaussianConvolutionRigidity.gaussianConvolution (self.reporterOutput e base) (self.noiseVariance e base) skill

field operationalLaw_eq_observableAccess:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base),
  (ProbabilityTheory.gaussianReal (self.populationMean e base) (self.populationVariance e base)).bind
      ⇑(LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (self.takerSet e base)
          (self.takerSet_measurable e base) (self.reportedKernel e base) (self.policy.baseOnlyLaw e base)) =
    self.policy.observableAccessLaw e base

field noAccessLaw_eq_baseOnly:
∀ {Base : Type u} [inst : MeasurableSpace Base] (self : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base)
  (e : self.policy.Equilibrium) (base : Base), self.policy.observableNoAccessLaw e base = self.policy.baseOnlyLaw e base
```

Direct retained dependencies

- GaussianConvolutionRigidity.gaussianConvolution
- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Material library prerequisites

AppliedModelingLib.Probability.GaussianSignalFamily

Source locator: cited publication, lines 199-208

Verbatim paper-source connection

Students. There is a unit mass of students. Each student has a latent skill level q, normally
distributed as N (μ, σ 2), as well as a set of observed features Θ. All students have the first K − 1
features {1, . . . , K − 1}; only a subset apply with the last feature K. For each student, θk is a noisy
function of q, i.e., θk = q +ǫk , for k = 1, . . . , K, with Gaussian noise ǫk ∼ N (0, σk2). The distribution
of noise ǫk is feature dependent, but independent across features and students. Throughout our
exposition, we refer to feature K as the "test score."
For each student, let Z, Y, X ∈ {0, 1} be indicators denoting whether the student has access (Z)
to the test, takes (Y) it, and reports their score (X), respectively. Of course, Z ≥ Y ≥ X. Access Z
is deterministic and pre-set, uncorrelated with q and other features. If a student has access (Z = 1),
how they make their decisions to take the test and/or report is defined below.

Settled source reading The [access-law explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) distinguishes the source's uncorrelated-access wording from the independence used by the current law.

Lean declaration type (metadata)

```
field priorMean:
{ι : Type u_1} → [inst : Fintype ι] → AppliedModelingLib.Probability.GaussianSignalFamily ι → ℝ

field priorVar:
{ι : Type u_1} → [inst : Fintype ι] → AppliedModelingLib.Probability.GaussianSignalFamily ι → ℝ

field noiseVar:
{ι : Type u_1} → [inst : Fintype ι] → AppliedModelingLib.Probability.GaussianSignalFamily ι → ι → ℝ

field priorVar_pos:
∀ {ι : Type u_1} [inst : Fintype ι] (self : AppliedModelingLib.Probability.GaussianSignalFamily ι), 0 < self.priorVar

field noiseVar_pos:
∀ {ι : Type u_1} [inst : Fintype ι] (self : AppliedModelingLib.Probability.GaussianSignalFamily ι) (i : ι),
  0 < self.noiseVar i
```

Exact Lean library declaration

```
/-- Conjugate Gaussian model with a finite family of noisy signals. -/
structure GaussianSignalFamily (ι : Type*) [Fintype ι] where
  priorMean : ℝ
  priorVar : ℝ
  noiseVar : ι → ℝ
  priorVar_pos : 0 < priorVar
  noiseVar_pos : ∀ i, 0 < noiseVar i
```

Recorded source-to-library screening **Verdict:** matches
Reason: The finite-family interface supplies exactly the source model's prior mean, positive prior variance, and feature-indexed positive noise variances. In the graph-bound test-adjoining definition, these parameters realize the independent zero-mean Gaussian signals and the source equation theta_k = q + epsilon_k.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Paper-specific semantic prerequisites

LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation

Paper-local declaration:
LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
Source locator: cited publication, lines 199-208

Verbatim paper-source connection

Students. There is a unit mass of students. Each student has a latent skill level q, normally
distributed as N (μ, σ 2), as well as a set of observed features Θ. All students have the first K − 1
features {1, . . . , K − 1}; only a subset apply with the last feature K. For each student, θk is a noisy
function of q, i.e., θk = q +ǫk , for k = 1, . . . , K, with Gaussian noise ǫk ∼ N (0, σk2). The distribution
of noise ǫk is feature dependent, but independent across features and students. Throughout our
exposition, we refer to feature K as the "test score."
For each student, let Z, Y, X ∈ {0, 1} be indicators denoting whether the student has access (Z)
to the test, takes (Y) it, and reports their score (X), respectively. Of course, Z ≥ Y ≥ X. Access Z
is deterministic and pre-set, uncorrelated with q and other features. If a student has access (Z = 1),
how they make their decisions to take the test and/or report is defined below.

Settled source reading The [access-law explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) distinguishes the source's uncorrelated-access wording from the independence used by the current law.

Lean-elaborated paper signature

```
field accessLaw:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → MeasureTheory.Measure Bool

field accessLaw_isProbability:
∀ {Feature : Type u_1} [inst : Fintype Feature]
  (self : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature),
  MeasureTheory.IsProbabilityMeasure self.accessLaw

field priorMean:
{Feature : Type u_1} → [inst : Fintype Feature] → LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → ℝ

field priorVariance:
{Feature : Type u_1} →
  [inst : Fintype Feature] → LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → NNReal

field noiseVariance:
{Feature : Type u_1} →
  [inst : Fintype Feature] → LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature → Feature → NNReal
```

Recorded source-to-declaration screening **Verdict:** matches
Reason: Under approved context LG21-SECTION4-PRESET-ACCESS-INDEPENDENCE-01, the source population is the one-student product of the preset access law, Gaussian skill prior, and independent feature-noise laws. The target and its graph-bound Section 4 uses retain exactly that parameterization and the feature equation q + epsilon.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.LG21RequirementPolicy

Paper-local declaration:
LG21TestOptionalPolicies.LG21RequirementPolicy
Source locator: cited publication, lines 223-226

Verbatim paper-source connection

What can the school require? We consider three requirement policies: no requirements (students are free to take and report the test unconstrained, Z ≥ Y ≥ X); reporting required conditional taking the test (if a student takes the test, then they must report their score, Z ≥ Y = X);3 or, in the case where the school knows Z, reporting required given access (Z = Y = X). We find that

Lean-elaborated paper signature

type:
Type

constructor LG21TestOptionalPolicies.LG21RequirementPolicy.noRequirements:
LG21TestOptionalPolicies.LG21RequirementPolicy

constructor LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredConditionalTaking:
LG21TestOptionalPolicies.LG21RequirementPolicy

constructor LG21TestOptionalPolicies.LG21RequirementPolicy.reportRequiredGivenAccess:
LG21TestOptionalPolicies.LG21RequirementPolicy

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

Reason: Its three constructors correspond directly to the source's no requirements, reporting-required-conditional-on-taking, and reporting-required-given-access policies.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.LG21Definition1ExactEquilibrium

Paper-local declaration:
LG21TestOptionalPolicies.LG21Definition1ExactEquilibrium
Source locator: cited publication, lines 256-335

Verbatim paper-source connection

Definition 1 (Equilibrium). Let T = {(Y, X) : Y ≥ X, (Y,X) obeys feature requirement policy} denote the action space of students with test access. Then, given estimation policy P and a requirement
policy, student decision functions X(·), Y (·) and estimation function fp constitute an equilibrium if:
• All decisions are feasible:

[U+0010 control character]

[U+0011 control character]
K , P) ∈ T , for all {θ }K−1 , q.
Y (q, {θk }K−1
,
P
),
X({θ
}
k k=1
k k=1
k=1

• Given a school's estimation function, each individual weakly prefers to follow their given action over other feasible actions: for all {θk }K−1
k=1 , q,
[U+0010 control character]
[U+0011 control character]
K
Y (q, {θk }K−1
,
P
),
X({θ
}
,
P
)
∈ arg max(y,x)∈T E[q̃|Y = y, X = x]
k
k=1
k=1

Where the expectation is performed over the (potentially random) school estimation function fP , and, for Y , the randomness in the student's test score.

• Estimation function fP follows its definition given the decision functions X(·), Y (·). For example, suppose the school knows Z and is using Bayesian optimal policy PBO . Then, for students with access who do not report a score:
K−1
, X = 0, Z = 1) , EX(·) [q|{θk }K−1
fPBO ({θk }k=1
k=1 , X = 0, Z = 1]
[U+0011 control character][U+0011 control character]
[U+0010 control character]
[U+0010 control character]
R
K] Pr θ |{θ }K−1
dθK
E[q|{θ
}
K
K
k
k
k=1
k=1
θK :X({θk }k=1 ,P)=0
[U+0011 control character]
[U+0010 control character]
=
K−1
Pr {θK : X({θk }K
k=1 , P) = 0}|{θk }k=1

Lean-expanded paper semantic target

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        LG21TestOptionalPolicies.LG21RequirementPolicy →
          (Skill → Base → MeasureTheory.Measure Test) →
            (Skill → Base → Bool) →
              (Base → Test → Bool) →
                (Base → Test → ℝ) →
                  (Base → ℝ) →
```

```
              (Base → ℝ) →
                ((Skill → Base → Bool) → (Base → Test → Bool) → Base → Test → ℝ) →
                  ((Skill → Base → Bool) → (Base → Test → Bool) → Base → ℝ) →
                    ((Skill → Base → Bool) → (Base → Test → Bool) → Base → ℝ) → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (requirementPolicy : LG21TestOptionalPolicies.LG21RequirementPolicy)
    (testLaw : Skill → Base → MeasureTheory.Measure Test) (takeDecision : Skill → Base → Bool)
    (reportDecision : Base → Test → Bool) (estimatedReportedPayoff : Base → Test → ℝ)
    (estimatedWithheldPayoff estimatedNoTakePayoff : Base → ℝ)
    (estimationPolicyReportedPayoff : (Skill → Base → Bool) → (Base → Test → Bool) → Base → Test → ℝ)
    (estimationPolicyWithheldPayoff estimationPolicyNoTakePayoff :
      (Skill → Base → Bool) → (Base → Test → Bool) → Base → ℝ) =>
  (∀ (skill : Skill) (base : Base), MeasureTheory.IsProbabilityMeasure (testLaw skill base)) ∧
    (∀ (base : Base), Measurable (reportDecision base)) ∧
      (∀ (skill : Skill) (base : Base),
        MeasureTheory.Integrable
          (fun (test : Test) =>
            if reportDecision base test = true then estimatedReportedPayoff base test
            else estimatedWithheldPayoff base)
          (testLaw skill base)) ∧
        (∀ (skill : Skill) (base : Base) (test : Test),
          LG21TestOptionalPolicies.LG21AccessAction.feasible requirementPolicy.accessRequirement
            { takesTest := takeDecision skill base,
              reportsScore := takeDecision skill base && reportDecision base test }) ∧
          (∀ (skill : Skill) (base : Base) (alternativeTake : Bool) (alternativeReport : Test → Bool),
            (∀ (test : Test),
              LG21TestOptionalPolicies.LG21AccessAction.feasible requirementPolicy.accessRequirement
                { takesTest := alternativeTake, reportsScore := alternativeTake && alternativeReport test }) →
              Measurable alternativeReport →
                (alternativeTake = true →
                  MeasureTheory.Integrable
                    (fun (test : Test) =>
                      if alternativeReport test = true then estimatedReportedPayoff base test
                      else estimatedWithheldPayoff base)
                    (testLaw skill base)) →
                  (if alternativeTake = true then
                    ∫ (test : Test),
                      if alternativeReport test = true then estimatedReportedPayoff base test
                      else estimatedWithheldPayoff base ∂testLaw skill base
                  else estimatedNoTakePayoff base) ≤
                  if takeDecision skill base = true then
                    ∫ (test : Test),
                      if reportDecision base test = true then estimatedReportedPayoff base test
                      else estimatedWithheldPayoff base ∂testLaw skill base
                  else estimatedNoTakePayoff base) ∧
            estimatedReportedPayoff = estimationPolicyReportedPayoff takeDecision reportDecision ∧
              estimatedWithheldPayoff = estimationPolicyWithheldPayoff takeDecision reportDecision ∧
                estimatedNoTakePayoff = estimationPolicyNoTakePayoff takeDecision reportDecision
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.feasible
- LG21TestOptionalPolicies.LG21RequirementPolicy
- LG21TestOptionalPolicies.LG21RequirementPolicy.accessRequirement

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

Reason: The Lean target has the source timing: taking depends on skill and pre-test features, while reporting depends on pre-test features and the realized score. It requires every realized action to satisfy the source feasible-action set, compares its expected payoff with every measurable feasible score-contingent reporting alternative, and fixes all three estimated payoffs to the same decision-induced policy operators. Probability, measurability, and integrability make the source expectation well-defined without changing its domain or conclusion.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy

Paper-local declaration:
LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy
Source locator: cited publication, lines 591-609

Verbatim paper-source connection

Definition 6 (Re-sampling policy). The sampling estimation policy, PS , is as follows.
For students with access to tests, the school uses the Bayesian optimal method to estimate the
student's true skill level; i.e., for I , (Z = 1, {θk }K−1
k=1 , X, θK 1X):
fPS (I) = E[q|I].
For students without access to tests, the school uses the first K − 1 features {θk }K
k=1 to generate
the distribution of the test score θK , and then randomly draws a score for each student to add to the
student's profile. Finally, the school estimates the student's true skill level based on the submitted
features and the randomly drawn test score; i.e., for I , (Z = 0, {θk }K−1
k=1):
K−1
fPS (Z = 0, {θk }K−1
k=1) = E[q|{θk }k=1 , θ̃K]

where
θ̃K ∼ θK |{θk }K−1
k=1 .
Then, we have the following.

Lean-elaborated paper signature

```
field studentLaw:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → MeasureTheory.Measure Omega

field studentLaw_isProbability:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  MeasureTheory.IsProbabilityMeasure self.studentLaw

field latentSkill:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → Omega → ℝ

field baseObservation:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → Omega → Base

field testObservation:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → Omega → Test

field reportObservation:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → Omega → Bool

field baseTestObservation_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  Measurable fun (omega : Omega) => (self.baseObservation omega, self.testObservation omega)

field latentSkill_integrable:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  MeasureTheory.Integrable self.latentSkill self.studentLaw
```

```
field accessPosteriorEstimate:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test →
              Base → Bool → Unit ⊕ Test → ℝ

field accessPosteriorEstimate_is_conditionalSkillMean:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  (fun (omega : Omega) =>
      self.accessPosteriorEstimate (self.baseObservation omega) (self.reportObservation omega)
        (if self.reportObservation omega = true then Sum.inr (self.testObservation omega)
        else Sum.inl ())) =ᵐ[self.studentLaw]
    self.studentLaw[self.latentSkill |
      MeasurableSpace.comap
        (fun (omega : Omega) =>
          (self.baseObservation omega, self.reportObservation omega,
            if self.reportObservation omega = true then Sum.inr (self.testObservation omega) else Sum.inl ()))
        inferInstance]

field baseLaw:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → MeasureTheory.Measure Base

field testGivenBase:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test →
              ProbabilityTheory.Kernel Base Test

field testGivenBase_isMarkov:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  ProbabilityTheory.IsMarkovKernel self.testGivenBase

field actual_base_test_law:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  MeasureTheory.Measure.map (fun (omega : Omega) => (self.baseObservation omega, self.testObservation omega))
      self.studentLaw =
    self.baseLaw.bind ⇑(ProbabilityTheory.Kernel.id.prod self.testGivenBase)

field posteriorEstimate:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test → Base → Test → ℝ

field posteriorEstimate_measurable:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  Measurable fun (z : Base × Test) => self.posteriorEstimate z.1 z.2

field posteriorEstimate_is_conditionalSkillMean:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  (fun (omega : Omega) =>
      self.posteriorEstimate (self.baseObservation omega) (self.testObservation omega)) =ᵐ[self.studentLaw]
    self.studentLaw[self.latentSkill |
      MeasurableSpace.comap (fun (omega : Omega) => (self.baseObservation omega, self.testObservation omega))
        inferInstance]

field access_reported_estimate_eq:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test] (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test)
  (base : Base) (test : Test), self.accessPosteriorEstimate base true (Sum.inr test) = self.posteriorEstimate base test

field noAccessEstimateKernel:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test →
```

ProbabilityTheory.Kernel Base ℝ

```
field noAccessEstimateKernel_isMarkov:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  ProbabilityTheory.IsMarkovKernel self.noAccessEstimateKernel

field noAccessEstimateKernel_eq:
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (self : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test),
  self.noAccessEstimateKernel =
    (ProbabilityTheory.Kernel.id.prod self.testGivenBase).map fun (z : Base × Test) => self.posteriorEstimate z.1 z.2
```

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

Reason: The access branch is the conditional mean given the base features, report indicator, and score only when reported, and the displayed access-law projection maps exactly that observed estimator. The no-access branch draws from the actual conditional test kernel given the base and maps that draw through the base-and-score posterior mean; the second projection exposes exactly this induced randomized estimate kernel.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses

Paper-local declaration:
LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses
Source locator: cited publication, lines 229-244

Verbatim paper-source connection

Student decisions and equilibrium. Pursuant to the requirement policy, students with access
(Z = 1) may decide to take the test (Y = 1), with full knowledge of their skill q, noise distribution
ǫK , other features {θk }K−1
k=1 for k < K, and the school's estimation policy P . Students who take the
test may, with knowledge of their test score θK , decide to report it (X = 1), in which case the school
observes their score θK . Students make these decisions to maximize the school's estimate of their
4
K
skill level. We denote the resulting decisions via the functions Y (q, {θk }K−1
k=1 , P) and X({θk }k=1 , P).
We assume that the school knows the functions X(·), Y (·).5 It can then use its knowledge of
these functions to estimate the skill level of each student. For example, with Bayesian optimal
estimation, if only students with q ≥ 1 report their test score, then E[q|X = 1] ≥ 1.
Thus, for students with access, we must calculate an equilibrium: student strategies to report a
test score depend on the school's estimation, which in turn depends on the reporting decisions of
students. An equilibrium is as follows.

Lean-expanded paper semantic target

```
type:
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Test] →
        (Skill → Base → MeasureTheory.Measure Test) →
          (Skill → Base → Bool) → (Base → Test → Bool) → (Base → Test → ℝ) → (Base → ℝ) → Prop

value:
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Test]
    (testLaw : Skill → Base → MeasureTheory.Measure Test) (takeDecision : Skill → Base → Bool)
    (reportDecision : Base → Test → Bool) (reportedPayoff : Base → Test → ℝ) (noReportPayoff : Base → ℝ) =>
  (∀ (base : Base),
      AppliedModelingLib.NoProfitableBinaryChoiceDeviation (fun (skill : Skill) => takeDecision skill base = true)
        (fun (skill : Skill) =>
          LG21TestOptionalPolicies.lg21OptionalSourceTakeExpectedPayoff testLaw reportDecision reportedPayoff
            noReportPayoff skill base)
        fun (_skill : Skill) => noReportPayoff base) ∧
    ∀ (base : Base),
      AppliedModelingLib.NoProfitableBinaryChoiceDeviation (fun (test : Test) => reportDecision base test = true)
        (reportedPayoff base) fun (_test : Test) => noReportPayoff base
```

Direct retained dependencies

- AppliedModelingLib.NoProfitableBinaryChoiceDeviation
- LG21TestOptionalPolicies.lg21OptionalSourceTakeExpectedPayoff

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

Reason: The source says taking is chosen with skill and base information before the test draw, while reporting is chosen after the realized test score, and both decisions maximize the school's estimate. The target implements exactly that timing: takeDecision is compared ex ante using the integral over testLaw of the contingent reportDecision payoff, and reportDecision is compared pointwise after each realized test. The two weak-inequality directions correctly allow ties and encode no profitable binary deviation without imposing a reporting requirement.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.LG21RequirementPolicy.feasibleAction

Paper-local declaration:
LG21TestOptionalPolicies.LG21RequirementPolicy.feasibleAction
Source locator: cited publication, lines 223-226

Verbatim paper-source connection

What can the school require? We consider three requirement policies: no requirements (students are free to take and report the test unconstrained, Z ≥ Y ≥ X); reporting required conditional taking the test (if a student takes the test, then they must report their score, Z ≥ Y = X);3 or, in the case where the school knows Z, reporting required given access (Z = Y = X). We find that

Lean-expanded paper semantic target

type:
```
LG21TestOptionalPolicies.LG21RequirementPolicy →
  LG21TestOptionalPolicies.LG21AccessStatus → LG21TestOptionalPolicies.LG21AccessAction → Prop
```

value:
```
fun (policy : LG21TestOptionalPolicies.LG21RequirementPolicy) (accessStatus : LG21TestOptionalPolicies.LG21AccessStatus)
    (action : LG21TestOptionalPolicies.LG21AccessAction) =>
  match accessStatus with
  | LG21TestOptionalPolicies.LG21AccessStatus.noAccess => action = LG21TestOptionalPolicies.LG21AccessAction.noTake
  | LG21TestOptionalPolicies.LG21AccessStatus.access =>
    LG21TestOptionalPolicies.LG21AccessAction.feasible policy.accessRequirement action
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21AccessAction
- LG21TestOptionalPolicies.LG21AccessAction.feasible
- LG21TestOptionalPolicies.LG21AccessAction.noTake
- LG21TestOptionalPolicies.LG21AccessStatus
- LG21TestOptionalPolicies.LG21RequirementPolicy
- LG21TestOptionalPolicies.LG21RequirementPolicy.accessRequirement

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

Reason: The expanded target enforces ‘(Y, X) = (0, 0)‘ when ‘Z = 0‘; when ‘Z = 1‘, it enforces ‘X ≤ Y‘, adds ‘Y = X‘ for conditional reporting, and requires ‘(Y, X) = (1, 1)‘ for reporting given access, exactly the three source constraints.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.PaperInterface.distributional_equality

Paper-local declaration:
LG21TestOptionalPolicies.PaperInterface.distributional_equality
Source locator: cited publication, lines 326-335

Verbatim paper-source connection

What does test-optional mean?
The focus of this work is to understand how schools can meet their test-optional pledge. However, a prerequisite challenge is to formalize what
↪ "test-optional" even means, as it is left undefined by schools beyond the idea to "not penalize students for the absence of a standardized test score" [NACAC, 2020]. Here, we define several potential notions, on the distribution of resulting
skill estimates for students with and without test access. (Recall that, by assumption, neither true
skill distribution q nor feature distributions θk |q differ by test access).
D

Our notions rely on "distributional equality," denoted (==), because of the randomness in the
process, both in the sampling of test scores given q and in the estimation function fP . Let q̃|D, P
denote the distribution of q̃ given the information D using policy P .

Lean-expanded paper semantic target

type:
{Law : Type u_1} → Law → Law → Prop

value:
fun {Law : Type u_1} (left right : Law) => left = right

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

Reason: The source introduces distributional equality as equality of the resulting law of the estimate. The exact Lean value is literal equality left = right for arbitrary law objects, which matches that notation without adding assumptions.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.PaperInterface.lg21GaussianSignalFamilyWithTest

Paper-local declaration:
LG21TestOptionalPolicies.PaperInterface.lg21GaussianSignalFamilyWithTest
Source locator: cited publication, lines 199-208

Verbatim paper-source connection

Students. There is a unit mass of students. Each student has a latent skill level q, normally
distributed as N (μ, σ 2), as well as a set of observed features Θ. All students have the first K − 1
features {1, . . . , K − 1}; only a subset apply with the last feature K. For each student, θk is a noisy
function of q, i.e., θk = q +ǫk , for k = 1, . . . , K, with Gaussian noise ǫk ∼ N (0, σk2). The distribution
of noise ǫk is feature dependent, but independent across features and students. Throughout our
exposition, we refer to feature K as the "test score."
For each student, let Z, Y, X ∈ {0, 1} be indicators denoting whether the student has access (Z)
to the test, takes (Y) it, and reports their score (X), respectively. Of course, Z ≥ Y ≥ X. Access Z
is deterministic and pre-set, uncorrelated with q and other features. If a student has access (Z = 1),
how they make their decisions to take the test and/or report is defined below.

Settled source reading The [access-law explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) distinguishes the source's uncorrelated-access wording from the independence used by the current law.

Lean-expanded paper semantic target

```
type:
{Feature : Type u_1} →
  [inst : Fintype Feature] →
    AppliedModelingLib.Probability.GaussianSignalFamily Feature →
      (testNoiseVariance : ℝ) →
        0 < testNoiseVariance → AppliedModelingLib.Probability.GaussianSignalFamily (Option Feature)

value:
fun {Feature : Type u_1} [Fintype Feature] (baseSignals : AppliedModelingLib.Probability.GaussianSignalFamily Feature)
    (testNoiseVariance : ℝ) (testNoiseVariance_pos : 0 < testNoiseVariance) =>
  { priorMean := baseSignals.priorMean, priorVar := baseSignals.priorVar,
    noiseVar := fun (x : Option Feature) =>
      match x with
      | none => testNoiseVariance
      | some feature => baseSignals.noiseVar feature,
    priorVar_pos := baseSignals.priorVar_pos,
    noiseVar_pos :=
      LG21TestOptionalPolicies.PaperInterface.lg21GaussianSignalFamilyWithTest._proof_1 baseSignals testNoiseVariance
        testNoiseVariance_pos }
```

Direct retained dependencies

- AppliedModelingLib.Probability.GaussianSignalFamily

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

Reason: The definition preserves the base Gaussian signal family and adjoins exactly one test coordinate with zero-mean Gaussian noise and an explicit positive variance; no free offset is introduced. This matches the source treatment of feature K as the additional test-score signal.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.PaperInterface.test_taking_probability_preference

Paper-local declaration:
LG21TestOptionalPolicies.PaperInterface.test_taking_probability_preference
Source locator: cited publication, lines 2173-2174

Verbatim paper-source connection

to be maximize the estimated skill level. Here we further interpret this to be that a student will
choose to take the test if this has a higher probability of leading to a better result than a worse one.

Lean-expanded paper semantic target

type:
$\mathbb{R} \to \mathbb{R} \to \text{Prop}$

value:
fun (probabilityBetter probabilityWorse : ℝ) => probabilityWorse < probabilityBetter

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

Reason: The source convention says a student chooses to take exactly when doing so has a higher probability of a better result than of a worse one. The exact Lean predicate probabilityWorse < probabilityBetter is the same strict comparison.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel

Paper-local declaration:
LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel
Source locator: cited publication, lines 591-609

Verbatim paper-source connection

Definition 6 (Re-sampling policy). The sampling estimation policy, PS , is as follows.
For students with access to tests, the school uses the Bayesian optimal method to estimate the
student's true skill level; i.e., for I , (Z = 1, {θk }K−1
k=1 , X, θK 1X):
fPS (I) = E[q|I].
For students without access to tests, the school uses the first K − 1 features {θk }K
k=1 to generate
the distribution of the test score θK , and then randomly draws a score for each student to add to the
student's profile. Finally, the school estimates the student's true skill level based on the submitted
features and the randomly drawn test score; i.e., for I , (Z = 0, {θk }K−1
k=1):
K−1
fPS (Z = 0, {θk }K−1
k=1) = E[q|{θk }k=1 , θ̃K]

where
θ̃K ∼ θK |{θk }K−1
k=1 .
Then, we have the following.

Lean-expanded paper semantic target

type:
```
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test →
              ProbabilityTheory.Kernel Base ℝ
```

value:
```
fun {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Omega] [MeasurableSpace Base]
    [MeasurableSpace Test] (policy : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test) =>
  policy.noAccessEstimateKernel
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy

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

Reason: Definition 6 draws a synthetic test from its conditional law given the base profile and then applies the base-and-test Bayesian posterior estimator. The selected field is exactly the induced Markov kernel obtained by mapping that conditional draw through the posterior estimate.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21Definition6ExactObservedAccessEstimateLaw

Paper-local declaration:
LG21TestOptionalPolicies.lg21Definition6ExactObservedAccessEstimateLaw
Source locator: cited publication, lines 591-609

Verbatim paper-source connection

Definition 6 (Re-sampling policy). The sampling estimation policy, PS , is as follows.
For students with access to tests, the school uses the Bayesian optimal method to estimate the
student's true skill level; i.e., for I , (Z = 1, {θk }K−1
k=1 , X, θK 1X):
fPS (I) = E[q|I].
For students without access to tests, the school uses the first K − 1 features {θk }K
k=1 to generate
the distribution of the test score θK , and then randomly draws a score for each student to add to the
student's profile. Finally, the school estimates the student's true skill level based on the submitted
features and the randomly drawn test score; i.e., for I , (Z = 0, {θk }K−1
k=1):
K−1
fPS (Z = 0, {θk }K−1
k=1) = E[q|{θk }k=1 , θ̃K]

where
θ̃K ∼ θK |{θk }K−1
k=1 .
Then, we have the following.

Lean-expanded paper semantic target

```
type:
{Omega : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      [inst : MeasurableSpace Omega] →
        [inst_1 : MeasurableSpace Base] →
          [inst_2 : MeasurableSpace Test] →
            (policy : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test) →
              AEMeasurable
                  (fun (omega : Omega) =>
                    policy.accessPosteriorEstimate (policy.baseObservation omega) (policy.reportObservation omega)
                      (if policy.reportObservation omega = true then Sum.inr (policy.testObservation omega)
                      else Sum.inl ()))
                policy.studentLaw →
                MeasureTheory.Measure ℝ

value:
fun {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [MeasurableSpace Omega] [MeasurableSpace Base]
    [MeasurableSpace Test] (policy : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test)
    (_hAccessEstimateMeasurable :
      AEMeasurable
        (fun (omega : Omega) =>
          policy.accessPosteriorEstimate (policy.baseObservation omega) (policy.reportObservation omega)
            (if policy.reportObservation omega = true then Sum.inr (policy.testObservation omega) else Sum.inl ()))
        policy.studentLaw) =>
  MeasureTheory.Measure.map
    (fun (omega : Omega) =>
      policy.accessPosteriorEstimate (policy.baseObservation omega) (policy.reportObservation omega)
        (if policy.reportObservation omega = true then Sum.inr (policy.testObservation omega) else Sum.inl ()))
    policy.studentLaw
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy

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

Reason: Definition 6 conditions an access student's Bayesian estimate on base features, reporting status, and the test score only when reported. The target maps the student law through exactly that report-dependent information estimator, so withheld scores are absent rather than silently observed.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair

Paper-local declaration:
LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
Source locator: cited publication, lines 360-364

Verbatim paper-source connection

Definition 4. Policy P is demographically fair if in every equilibrium,
D

q̃|Z = 1, P == q̃|Z = 0, P,
i.e., if the distribution of skill estimates for students with and without access are equal.

Lean-expanded paper semantic target

type:
```
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law → Prop
```

value:
```
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) =>
  ∀ (e : S.Equilibrium), S.demographicAccessLaw e = S.demographicNoAccessLaw e
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

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

Reason: Definition 4 requires equality in distribution of estimates for access and no-access students in every equilibrium. The target quantifies over the surface's equilibrium object and equates exactly its two demographic estimate-law fields.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair

Paper-local declaration:
LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
Source locator: cited publication, lines 336-345

Verbatim paper-source connection

Definition 2. Policy P is latent skill fair if in every equilibrium, for all q,
D

K−1
q̃|Z = 1, {θk }K−1
k=1 , q, P == q̃|Z = 0, {θk }k=1 , q, P,

i.e., if the distribution of skill estimates for students – that share (latent) skill q and (observable)
other features {θk }K−1
k=1 – with and without access are equal.

Lean-expanded paper semantic target

type:
```
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law → Prop
```

value:
```
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) =>
  ∀ (e : S.Equilibrium) (q : Skill) (base : Base), S.latentAccessLaw e q base = S.latentNoAccessLaw e q base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

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

Reason: Definition 2 requires equality, in every equilibrium and for every latent skill q and observable non-test feature profile, between access and no-access estimate distributions. The predicate is exactly universal equality of latentAccessLaw and latentNoAccessLaw over equilibrium, skill, and base.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Paper-local declaration:
LG21TestOptionalPolicies.lg21SourceLawObservablyFair
Source locator: cited publication, lines 346-358

Verbatim paper-source connection

Definition 3. Policy P is observably fair if in every equilibrium, for all {θk }K−1
k=1 ,
D

K−1
q̃|Z = 1, {θk }K−1
k=1 , P == q̃|Z = 0, {θk }k=1 , P,

i.e., if the distribution of skill estimates for students – that share (observable) other features {θk }K−1
k=1
– with and without access are equal.

6

Lean-expanded paper semantic target

type:
```
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law → Prop
```

value:
```
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) =>
  ∀ (e : S.Equilibrium) (base : Base), S.observableAccessLaw e base = S.observableNoAccessLaw e base
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

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

Reason: Definition 3 requires equality, in every equilibrium and for every observable non-test feature profile, between access and no-access estimate distributions. The predicate is exactly universal equality of observableAccessLaw and observableNoAccessLaw over equilibrium and base.
Reviewer check: ☐ Matches source input
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

LG21TestOptionalPolicies.lg21SourceLawTestBlank

Paper-local declaration:
LG21TestOptionalPolicies.lg21SourceLawTestBlank
Source locator: cited publication, lines 370-387

Verbatim paper-source connection

Definition 5. Policy P is test-blank, i.e., test scores play no role, if in every equilibrium,
D

K−1
q̃|{θk }K−1
k=1 , P == q̃|{θk }k=1 , θK , P

∀ {θk }K
k=1

Of course, test-blank policies defeat the purpose of going test-optional. In this work, we thus ask whether it is possible to meet the fairness notions without being test-blank.
Note that these definitions all concern the estimates of the policy P in equilibrium. For example, there are two ways for a policy to be test-blank: either the estimation function fP ignores the test score (and whether the student reports it, X), or in equilibrium no student reports a test score. In either case, the definition is met, and test scores play no role in student estimates. Note also that for notational ease we suppress the requirement policy in the statement of each definition, as our results do not depend on which requirement policy is chosen.

Lean-expanded paper semantic target

type:
```
{Skill : Type u_1} →
  {Base : Type u_2} →
    {Test : Type u_3} →
      {Law : Type u_4} → LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law → Prop
```

value:
```
fun {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} {Law : Type u_4}
    (S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law) =>
  ∀ (e : S.Equilibrium) (base : Base) (test : Test), S.baseOnlyLaw e base = S.fullFeatureLaw e base test
```

Direct retained dependencies

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface

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

Reason: Definition 5 says that in every equilibrium the estimate law based on the first K-1 features equals the law after adjoining any test score. The target states exactly that base-only versus full-feature law equality for every equilibrium, base, and test value.
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 365

Verbatim source input

Each definition is less stringent than the previous, i.e., latent skill fairness =⇒ observable fairness =⇒ demographic fairness. The notions connect naturally
↪ to those common in the literature.

Expanded PaperInterface specification

```
∀ {Skill : Type uSkill} {Base : Type uBase} {Test : Type uTest} {Estimate : Type uEstimate}
  [inst : MeasurableSpace Skill] [inst_1 : MeasurableSpace Base] [inst_2 : MeasurableSpace Estimate]
  (S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test (MeasureTheory.Measure Estimate))
  (skillGivenBase : ProbabilityTheory.Kernel Base Skill) (baseLaw : MeasureTheory.Measure Base)
  (latentAccessKernel latentNoAccessKernel : S.Equilibrium → Base → ProbabilityTheory.Kernel Skill Estimate)
  (observableAccessKernel observableNoAccessKernel : S.Equilibrium → ProbabilityTheory.Kernel Base Estimate),
  (∀ (e : S.Equilibrium) (base : Base) (skill : Skill),
      (latentAccessKernel e base) skill = S.latentAccessLaw e skill base) →
    (∀ (e : S.Equilibrium) (base : Base) (skill : Skill),
        (latentNoAccessKernel e base) skill = S.latentNoAccessLaw e skill base) →
      (∀ (e : S.Equilibrium) (base : Base),
          S.observableAccessLaw e base = (skillGivenBase base).bind ⇑(latentAccessKernel e base)) →
        (∀ (e : S.Equilibrium) (base : Base),
            S.observableNoAccessLaw e base = (skillGivenBase base).bind ⇑(latentNoAccessKernel e base)) →
          (∀ (e : S.Equilibrium) (base : Base), (observableAccessKernel e) base = S.observableAccessLaw e base) →
            (∀ (e : S.Equilibrium) (base : Base), (observableNoAccessKernel e) base = S.observableNoAccessLaw e base) →
              (∀ (e : S.Equilibrium), S.demographicAccessLaw e = baseLaw.bind ⇑(observableAccessKernel e)) →
                (∀ (e : S.Equilibrium), S.demographicNoAccessLaw e = baseLaw.bind ⇑(observableNoAccessKernel e)) →
                  (LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair S →
                      LG21TestOptionalPolicies.lg21SourceLawObservablyFair S) ∧
                    (LG21TestOptionalPolicies.lg21SourceLawObservablyFair S →
                      LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair S)
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Lean-elaborated claim roles

1. parameter: Type uSkill
2. parameter: Type uBase
3. parameter: Type uTest
4. parameter: Type uEstimate
5. parameter: MeasurableSpace Skill
6. parameter: MeasurableSpace Base
7. parameter: MeasurableSpace Estimate
8. parameter: LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test (MeasureTheory.Measure Estimate)
9. parameter: ProbabilityTheory.Kernel Base Skill
10. parameter: MeasureTheory.Measure Base
11. parameter: S.Equilibrium → Base → ProbabilityTheory.Kernel Skill Estimate
12. parameter: S.Equilibrium → Base → ProbabilityTheory.Kernel Skill Estimate
13. parameter: S.Equilibrium → ProbabilityTheory.Kernel Base Estimate
14. parameter: S.Equilibrium → ProbabilityTheory.Kernel Base Estimate
15. assumption: ∀ (e : S.Equilibrium) (base : Base) (skill : Skill), (latentAccessKernel e base) skill = S.latentAccessLaw e skill base
16. assumption: ∀ (e : S.Equilibrium) (base : Base) (skill : Skill),
  (latentNoAccessKernel e base) skill = S.latentNoAccessLaw e skill base
17. assumption: ∀ (e : S.Equilibrium) (base : Base),
  S.observableAccessLaw e base = (skillGivenBase base).bind ⇑(latentAccessKernel e base)
18. assumption: ∀ (e : S.Equilibrium) (base : Base),
  S.observableNoAccessLaw e base = (skillGivenBase base).bind ⇑(latentNoAccessKernel e base)
19. assumption: ∀ (e : S.Equilibrium) (base : Base), (observableAccessKernel e) base = S.observableAccessLaw e base
20. assumption: ∀ (e : S.Equilibrium) (base : Base), (observableNoAccessKernel e) base = S.observableNoAccessLaw e base
21. assumption: ∀ (e : S.Equilibrium), S.demographicAccessLaw e = baseLaw.bind ⇑(observableAccessKernel e)
22. assumption: ∀ (e : S.Equilibrium), S.demographicNoAccessLaw e = baseLaw.bind ⇑(observableNoAccessKernel e)
23. conclusion: (LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair S → LG21TestOptionalPolicies.lg21SourceLawObservablyFair S) ∧
  (LG21TestOptionalPolicies.lg21SourceLawObservablyFair S → LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair S)

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.fairness_implication_chain_proof

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

Reason: The source orders latent-skill, observable, and demographic fairness by implication; the expanded target uses the same every-equilibrium law equalities and explicit common skill-given-base and base mixtures to derive exactly the two successive implications.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 2

Source locator: cited publication, lines 492-539

Verbatim source input

Lemma 4.1 (Strategy-proofness). Suppose the school observes access Z for each student and uses
a policy P that performs Bayesian optimal estimation on students with access; i.e., when Z = 1,
fP (I) = E[q|I].
Then, the unique equilibrium is that all students with access take the test and report their scores:
K−1
Y (q, {θk }k=1
, P) = X({θk }K
k=1 , P) = 1.

∀q, {θk }K
k=1

Settled source reading **Formalization gap:** the [Section 4 explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) records the added maximal active-branch selection, which has not been derived from the source equilibrium definition.

Expanded PaperInterface specification

LG21TestOptionalPolicies.PaperInterface.lemma4_1_observed_access_strategy_proofnessSpec

Retained governing declarations

- LG21TestOptionalPolicies.PaperInterface.lemma4_1_observed_access_strategy_proofnessSpec

Lean-elaborated claim roles

```
1. parameter: Type uLemma41
2. parameter: Fintype Feature
3. parameter: DecidableEq Feature
4. parameter: LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature
5. conclusion: Nonempty (LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source) ∧
  Nonempty (LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source) ∧
    (∀ (mandatory : LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium source.population),
       ∀ᵐ (student :
         Bool ×
           ℝ × (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
       mandatory.action student = LG21TestOptionalPolicies.LG21AccessAction.takeAndReport) ∧
     (∀ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
       ∀ᵐ (student :
         Bool ×
           ℝ × (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
       optionalProfile.selected.actions.takeDecision
           (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
           (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
         true ∧
       optionalProfile.selected.actions.reportDecision
           (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
           (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student) =
         true) ∧
      (∀ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
         ∀ᵐ (student :
           Bool ×
             ℝ ×
               (Feature → ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
         reportRequiredProfile.selected.takeDecision
             (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
             (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
           true) ∧
       (∀ (left right : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
          ∀ᵐ (student :
            Bool ×
              ℝ ×
                (Feature →
                  ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
          left.selected.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
            right.selected.actions.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) ∧
           left.selected.actions.reportDecision
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student) =
            right.selected.actions.reportDecision
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature student)) ∧
         ∀ (left right : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
           ∀ᵐ (student :
             Bool ×
               ℝ ×
                 (Feature →
                   ℝ)) ∂LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population,
```

```
left.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
  (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student) =
right.selected.takeDecision (LG21TestOptionalPolicies.lg21ContinuousPopulationSkill student)
  (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature student)
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.lemma4_1_observed_access_strategy_proofness_source_core_proof

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

Reason: Under the displayed approved Section 4 convention, the target supplies nonempty optional and report-required active-branch profiles, makes mandatory, optional, and report-required actions take-and-report almost everywhere, and makes selected profiles unique almost everywhere, matching the source's operational strategy-proofness conclusion.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 3

Source locator: cited publication, lines 550-551

Verbatim source input

Proposition 4.2. Suppose the school observes access Z. Consider any policy P that performs
Bayesian optimal estimation on students with access. Then, P is not latent skill fair.

Expanded PaperInterface specification

```
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (model : LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base) (base : Base) {skillLow skillHigh : ℝ},
  skillLow < skillHigh → ¬LG21TestOptionalPolicies.lg21P42ObservedScoreLatentSkillFair model
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel
- LG21TestOptionalPolicies.lg21P42ObservedScoreLatentSkillFair

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: MeasurableSpace Base
3. parameter: LG21TestOptionalPolicies.LG21P42ObservedScoreGaussianPBOModel Base
4. parameter: Base
5. parameter: ℝ
6. parameter: ℝ
7. assumption: skillLow < skillHigh
8. conclusion: ¬LG21TestOptionalPolicies.lg21P42ObservedScoreLatentSkillFair model

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.proposition4_2_all_observed_access_requirement_protocols_source_core_proof

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

Reason: Proposition 4.2 says an observed-access Bayesian posterior policy cannot be latent-skill fair; the expanded Gaussian PBO model gives the access posterior law a skill-dependent mean and, for any two ordered skills, concludes failure of the universal equality with the arbitrary base-only no-access kernel.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 4

Source locator: cited publication, lines 561-562

Verbatim source input

Proposition 4.3. Suppose the school observes access status Z. Then, PBO – the policy that uses Bayesian optimal estimation for all students – is not observably or demographically fair.

[Next verbatim source excerpt]

[form-feed in source extraction]
Students. There is a unit mass of students. Each student has a latent skill level q, normally distributed as N (μ, σ 2), as well as a set of observed features Θ. All students have the first K − 1 features {1, . . . , K − 1}; only a subset apply with the last feature K. For each student, θk is a noisy function of q, i.e., θk = q +ǫk , for k = 1, . . . , K, with Gaussian noise ǫk ∼ N (0, σk2). The distribution of noise ǫk is feature dependent, but independent across features and students. Throughout our exposition, we refer to feature K as the "test score."

[Next verbatim source excerpt]

score); if X = 1, it further includes θK . As in Garg et al. [2020], we study the Bayesian optimal estimation policy PBO , in which the student skill estimate is the posterior mean using all the information available:
q̃ = fPBO (I) , E[q|I].
We further consider non-Bayesian optimal estimation policies P , implemented by (potentially randomized) functions fP .

[Next verbatim source excerpt]

Our notions rely on "distributional equality," denoted (==), because of the randomness in the process, both in the sampling of test scores given q and in the estimation function fP . Let q̃|D, P denote the distribution of q̃ given the information D using policy P .
Definition 2. Policy P is latent skill fair if in every equilibrium, for all q,
D

K−1
q̃|Z = 1, {θk }K−1
k=1 , q, P == q̃|Z = 0, {θk }k=1 , q, P,

i.e., if the distribution of skill estimates for students – that share (latent) skill q and (observable) other features {θk }K−1
k=1 – with and without access are equal.
Definition 3. Policy P is observably fair if in every equilibrium, for all {θk }K−1
k=1 ,
D

K−1
q̃|Z = 1, {θk }K−1
k=1 , P == q̃|Z = 0, {θk }k=1 , P,

i.e., if the distribution of skill estimates for students – that share (observable) other features {θk }K−1
k=1
– with and without access are equal.

6

[form-feed in source extraction]
Definition 4. Policy P is demographically fair if in every equilibrium,
D

q̃|Z = 1, P == q̃|Z = 0, P,
i.e., if the distribution of skill estimates for students with and without access are equal.

[Next verbatim source excerpt]

What can the school require? We consider three requirement policies: no requirements (students are free to take and report the test unconstrained, Z ≥ Y ≥ X); reporting required conditional taking the test (if a student takes the test, then they must report their score, Z ≥ Y = X);3 or, in the case where the school knows Z, reporting required given access (Z = Y = X). We find that such requirements do not make a difference: the same results hold for all our settings, and so we do not include the requirement policy when denoting the school's policy P .

[Next verbatim source excerpt]

generality, that within this set of students with fixed {θk }K
k=1 , the fraction of students who have
access to tests is C ∈ (0, 1). Then we have that of this fraction of students with access, a further

Settled source reading **Formalization gap:** the [Section 4 explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) records the added maximal active-branch selection, which has not been derived from the source equilibrium definition.

Expanded PaperInterface specification

```
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature)
  (hnoAccess : 0 < source.population.accessLaw {false}) (noAccessOutput : Bool × ℝ × (Feature → ℝ) → ℝ),
```

```
LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO source.population hnoAccess source.test_feature
    noAccessOutput →
  have claim : (Bool × ℝ × (Feature → ℝ) → ℝ) → Prop := fun (actualOutput : Bool × ℝ × (Feature → ℝ) → ℝ) =>
    ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ))
      (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance :
      ℝ) (baseMeanVariance : ℝ) (_ : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseMean : Measurable baseMean) (_
      : 0 < baseVariance),
      LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population
          source.test_feature =
        baseLaw.compProd
          (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
      0 ≤ baseMeanVariance ∧
        ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE baseLaw
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
            (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
            (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
            (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
              LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput) ∧
          ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair
            (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
            (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
            (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
              LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput);
  (∃ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
      claim
        (LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput
          (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
          (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
          LG21TestOptionalPolicies.lg21ContinuousPopulationSkill optionalProfile.selected.actions)) ∧
    (∃ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
        claim
          (LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
            LG21TestOptionalPolicies.lg21ContinuousPopulationSkill reportRequiredProfile.selected)) ∧
      ∃ (profile : LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess)
        (mandatoryNoReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ)
        (baseLaw :
        MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ)) (baseMean
        : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ)
        (baseMeanVariance : ℝ) (_ : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseMean : Measurable baseMean) (_ :
        0 < baseVariance),
        LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population
            source.test_feature =
          baseLaw.compProd
            (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
        0 ≤ baseMeanVariance ∧
          ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE baseLaw
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
              (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
              (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
              (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
                LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
                (LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
                  (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
                  (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
                  profile.mandatory.action profile.reportedPayoff mandatoryNoReportPayoff)
                profile.noAccessOutput) ∧
            ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair
              (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
              (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
              (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
                LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
                (LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
                  (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
                  (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
                  profile.mandatory.action profile.reportedPayoff mandatoryNoReportPayoff)
                profile.noAccessOutput)
```

Retained governing declarations

- AppliedModelingLib.Probability.gaussianLocationKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21MandatoryGivenAccessLiteralSourceEquilibrium
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair
- LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile
- LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile

- LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw
- LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
- LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
- LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput
- LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype Feature
3. parameter: DecidableEq Feature
4. parameter: LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature
5. assumption: 0 < source.population.accessLaw {false}
6. parameter: Bool × ℝ × (Feature → ℝ) → ℝ
7. assumption: LG21TestOptionalPolicies.LG21ContinuousGaussianNoAccessPopulationPBO source.population hnoAccess source.test_feature noAccessOutput
8. conclusion: (∃ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source), (fun (actualOutput : Bool × ℝ × (Feature → ℝ) → ℝ) => ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ)) (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ) (baseMeanVariance : ℝ) (_ : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseMean : Measurable baseMean) (_ : 0 < baseVariance), LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population source.test_feature = baseLaw.compProd (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧ 0 ≤ baseMeanVariance ∧ ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE baseLaw (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature) (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput) ∧ ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput)) (LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature) (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature) LG21TestOptionalPolicies.lg21ContinuousPopulationSkill optionalProfile.selected.actions)) ∧ (∃ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source), (fun (actualOutput : Bool × ℝ × (Feature → ℝ) → ℝ) => ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ)) (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ) (baseMeanVariance : ℝ) (_ : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseMean : Measurable baseMean) (_ : 0 < baseVariance), LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population source.test_feature = baseLaw.compProd (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧ 0 ≤ baseMeanVariance ∧ ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE baseLaw (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature) (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput) ∧ ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population) (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput LG21TestOptionalPolicies.lg21ContinuousPopulationAccess actualOutput noAccessOutput)) (LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature) (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature) LG21TestOptionalPolicies.lg21ContinuousPopulationSkill reportRequiredProfile.selected)) ∧ ∃ (profile : LG21TestOptionalPolicies.LG21P43MandatoryGivenAccessPBOProfile source hnoAccess) (mandatoryNoReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ)) (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ)

```
(baseMeanVariance : ℝ) (_ : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseMean : Measurable baseMean) (_ :
0 < baseVariance),
LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population source.test_feature =
    baseLaw.compProd
      (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
  0 ≤ baseMeanVariance ∧
    ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicObservableFairAE baseLaw
        (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
        (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
        (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
          LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
          (LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
            profile.mandatory.action profile.reportedPayoff mandatoryNoReportPayoff)
          profile.noAccessOutput) ∧
      ¬LG21TestOptionalPolicies.LG21ObservedAccessDeterministicDemographicallyFair
        (LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw source.population)
        (LG21TestOptionalPolicies.lg21ContinuousGaussianNoAccessPopulationLaw source.population)
        (LG21TestOptionalPolicies.lg21ObservedAccessDeterministicTwoBranchOutput
          LG21TestOptionalPolicies.lg21ContinuousPopulationAccess
          (LG21TestOptionalPolicies.lg21ObservedAccessActualOutput
            (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
            (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
            profile.mandatory.action profile.reportedPayoff mandatoryNoReportPayoff)
          profile.noAccessOutput)
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.proposition4_3_each_requirement_protocol_has_unfair_pbo_equilibrium_proof

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

Reason: The source defines observable and demographic fairness across every equilibrium and says PBO fails both under each requirement regime; the target constructs an optional, report-required, and mandatory PBO profile whose actual access/no-access output laws fail both equalities, which is exactly sufficient to refute those every-equilibrium fairness predicates.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 5

Source locator: cited publication, lines 391-395

Verbatim source input

First, note that our stated fairness definitions all condition on access Z, as opposed to reporting X. However, all of our results, for each definition, also hold for the corresponding definition for which access Z is replaced by reporting X; in fact, the corresponding results are easier to prove, as X is always observed but Z is potentially unobserved. In each proof, we explain the requisite changes needed for such a result. Our exposition centers Z normatively, with the idea that schools

Expanded PaperInterface specification

```
∀ (Skill : Type uSkill) (Base : Type uBase) (Test : Type uTest) (Law : Type uLaw)
  (R : LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law)
  (Result : Prop → Prop → Prop → Prop → Prop),
  Result (LG21TestOptionalPolicies.lg21ReportingConditionedLatentSkillFair R)
      (LG21TestOptionalPolicies.lg21ReportingConditionedObservablyFair R)
      (LG21TestOptionalPolicies.lg21ReportingConditionedDemographicallyFair R)
      (LG21TestOptionalPolicies.lg21ReportingConditionedTestBlank R) ↔
    Result (LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair R.toSourceLawSurface)
      (LG21TestOptionalPolicies.lg21SourceLawObservablyFair R.toSourceLawSurface)
      (LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair R.toSourceLawSurface)
      (LG21TestOptionalPolicies.lg21SourceLawTestBlank R.toSourceLawSurface)
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface
- LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface.toSourceLawSurface
- LG21TestOptionalPolicies.lg21ReportingConditionedDemographicallyFair
- LG21TestOptionalPolicies.lg21ReportingConditionedLatentSkillFair
- LG21TestOptionalPolicies.lg21ReportingConditionedObservablyFair
- LG21TestOptionalPolicies.lg21ReportingConditionedTestBlank
- LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair
- LG21TestOptionalPolicies.lg21SourceLawTestBlank

Lean-elaborated claim roles

1. parameter: Type uSkill
2. parameter: Type uBase
3. parameter: Type uTest
4. parameter: Type uLaw
5. parameter: LG21TestOptionalPolicies.LG21ReportingConditionedLawPolicySurface Skill Base Test Law
6. parameter: Prop → Prop → Prop → Prop → Prop
7. conclusion: Result (LG21TestOptionalPolicies.lg21ReportingConditionedLatentSkillFair R)
   (LG21TestOptionalPolicies.lg21ReportingConditionedObservablyFair R)
   (LG21TestOptionalPolicies.lg21ReportingConditionedDemographicallyFair R)
   (LG21TestOptionalPolicies.lg21ReportingConditionedTestBlank R) ↔
  Result (LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair R.toSourceLawSurface)
   (LG21TestOptionalPolicies.lg21SourceLawObservablyFair R.toSourceLawSurface)
   (LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair R.toSourceLawSurface)
   (LG21TestOptionalPolicies.lg21SourceLawTestBlank R.toSourceLawSurface)

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.reporting_conditioned_generalization_proof

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

Reason: The source says every fairness and blankness result remains valid after replacing access status by reporting status; the expanded reporting-conditioned surface relabels reporter and nonreporter laws as access and no-access laws and proves equivalence for an arbitrary proposition schema over all four predicates.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 6

Source locator: cited publication, lines 424-450

Verbatim source input

Theorem 3.1 (Strategic withholding). Suppose the school does not know access status and uses Bayesian optimal estimation PBO for all students. Then, some students with access do not report test scores. In particular, in every equilibrium:
• If the school does not require reporting conditional on taking the test, then Y (·) = 1. For each $\{\theta k\}_{k=1}^{K-1}$, there exists a finite test score threshold $\bar{\theta}K$ such that an applicant reports their score, $X(\{\theta k\}_{k=1}^{K-1}, \theta K) = 1$, if and only if $\theta K \geq \bar{\theta}K$.
• If the school requires reporting conditional on taking the test (Y = X), then for each $\{\theta k\}_{k=1}^{K-1}$, there exists a finite skill threshold $\bar{q}$ such that an applicant takes the test and reports their score, $Y(\{\theta k\}_{k=1}^{K-1}, q) = 1$, if and only if $q \geq \bar{q}$.
Furthermore, PBO satisfies none of the fairness definitions.
To prove this theorem, we first argue that neither everyone report or no one report is equilibrium. We then show that if a fraction of the student reports their score (or takes the test), there should be a threshold for the test score (or the skill level) for reporting (or test-taking), and then we conclude by showing there exists such a threshold such that the equilibrium conditions hold. For the unfairness argument, we note that in this situation the estimated skill of a portion of students with access are guaranteed to strictly dominate those of their peers who do not have access, with others guaranteed to weakly dominate, so any fairness conditional on test access cannot be met. The theorem establishes that the naive school strategy – to optimally use whatever information is provided by the student – induces students to strategically respond; as a result, the school satisfies none of our fairness criteria. More generally, the proof for unfairness is independent of the assumption that school does not know access status and the policy it uses. Instead, as long as the strategic effect exists, meeting fairness is impossible – some of the students who have access are bound to benefit from their strategic advantage.

[Next verbatim source excerpt]

Finally, to conclude that it is unfair according to all definitions, we note that in this regime, every student who has access to test can act strategically, so that he has an estimated skill that weakly dominates another student who is completely the same otherwise, but does not have access to test. Furthermore, there always exist some students who have access that strictly dominate their counterparts, otherwise the situation where no one report the score (or take the test) should be stable. Thus, $\tilde{q}|Z = 1, \{\theta k\}_{k=1}^{K-1}$ strictly stochastically dominates $\tilde{q}|Z = 0, \{\theta k\}_{k=1}^{K-1}$ for every $\{\theta k\}_{k=1}^{K-1}$. This means any equality in distribution that is conditional on Z cannot hold, hence all three fairness criteria fail. As a side remark, if we were considering fairness conditioned on X, the same argument still follows: all students with X = 1 weakly benefit from this decision (and some strictly benefit), while a portion of the students with X = 0, namely those without access, does not benefit at all.

[Next verbatim source excerpt]

who would be better off if they chose to report. So no one reports is also not stable. Thus we know that if an equilibrium exists, it must be that a fraction of the students report.

[Next verbatim source excerpt]

We can then show that neither everyone takes the test and reports nor no one takes the test and reports are stable, so if an equilibrium exists, it must be that a fraction of students report. Under such situation, the function $Y(\{\theta k\}_{k=1}^{K-1}, q, \text{PBO})$ would be increasing with respect to q, so that there is a threshold of q for test-taking. We can then finish by showing that there exists

Expanded PaperInterface specification

```
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (M : LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature),
  0 < M.accessLaw {true} →
    ∀ (hnoAccess : 0 < M.accessLaw {false}) (testFeature : Feature),
      0 < ↑M.priorVariance →
        (∀ (feature : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature),
           0 < ↑(M.noiseVariance ↑feature)) →
         0 < ↑(M.noiseVariance testFeature) →
           ∀ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ))
             [MeasureTheory.IsProbabilityMeasure baseLaw]
             (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ)
             (hbaseMean : Measurable baseMean) (baseVariance : ℝ),
             0 < baseVariance →
               MeasureTheory.Measure.map
                    (LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation testFeature)
                    (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M) =
                  baseLaw.compProd
                    (AppliedModelingLib.Probability.gaussianSignalJointKernel baseMean hbaseMean baseVariance
                      ↑(M.noiseVariance testFeature)) →
                 ∀
                   (optionalEquilibrium :
                     LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature),
                   optionalEquilibrium.OptionalReportBestResponseAE →
                     LG21TestOptionalPolicies.LG21HiddenAccessSourceStableAgainstLocalCandidateEntry
                       optionalEquilibrium hnoAccess →
                      ∀
```

```
(requiredEquilibrium :
  LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M
    testFeature),
LG21TestOptionalPolicies.LG21ReportRequiredStableAgainstLocalTailEntry
    requiredEquilibrium.source.takeDecision →
  (¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentSkillFairAt optionalEquilibrium ∧
      ¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservablyFairAt optionalEquilibrium
          baseMean hbaseMean baseVariance ∧
        ¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalDemographicallyFairAt
          optionalEquilibrium) ∧
    ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentSkillFair
        requiredEquilibrium ∧
      ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservablyFairAt
          requiredEquilibrium baseMean hbaseMean baseVariance ∧
        ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredDemographicallyFairAt
          requiredEquilibrium
```

Retained governing declarations

- AppliedModelingLib.Probability.gaussianSignalJointKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE.OptionalReportBestResponseAE
- LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE
- LG21TestOptionalPolicies.LG21HiddenAccessSourceStableAgainstLocalCandidateEntry
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ReportRequiredStableAgainstLocalTailEntry
- LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw
- LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalDemographicallyFairAt
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentSkillFairAt
- LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservablyFairAt
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredDemographicallyFairAt
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentSkillFair
- LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservablyFairAt

Lean-elaborated claim roles

```
1. parameter: Type u_1
2. parameter: Fintype Feature
3. parameter: DecidableEq Feature
4. parameter: LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation Feature
5. assumption: 0 < M.accessLaw {true}
6. assumption: 0 < M.accessLaw {false}
7. parameter: Feature
8. assumption: 0 < ↑M.priorVariance
9. assumption: ∀ (feature : LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature), 0 < ↑(M.noiseVariance ↑feature)
10. assumption: 0 < ↑(M.noiseVariance testFeature)
11. parameter: MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ)
12. assumption: MeasureTheory.IsProbabilityMeasure baseLaw
13. parameter: (LG21TestOptionalPolicies.LG21NonTestFeature Feature testFeature → ℝ) → ℝ
14. assumption: Measurable baseMean
15. parameter: ℝ
16. assumption: 0 < baseVariance
17. assumption: MeasureTheory.Measure.map (LG21TestOptionalPolicies.lg21HiddenAccessBaseScoreSkillObservation testFeature)
    (LG21TestOptionalPolicies.lg21ContinuousGaussianPopulationLaw M) =
  baseLaw.compProd
    (AppliedModelingLib.Probability.gaussianSignalJointKernel baseMean hbaseMean baseVariance
      ↑(M.noiseVariance testFeature))
18. parameter: LG21TestOptionalPolicies.LG21HiddenAccessLiteralSourceEquilibriumAE M testFeature
19. assumption: optionalEquilibrium.OptionalReportBestResponseAE
20. assumption: LG21TestOptionalPolicies.LG21HiddenAccessSourceStableAgainstLocalCandidateEntry optionalEquilibrium hnoAccess
21. parameter: LG21TestOptionalPolicies.LG21HiddenAccessReportRequiredLiteralSourceEquilibriumAE M testFeature
22. assumption: LG21TestOptionalPolicies.LG21ReportRequiredStableAgainstLocalTailEntry requiredEquilibrium.source.takeDecision
23. conclusion: (¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalLatentSkillFairAt optionalEquilibrium ∧
    ¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalObservablyFairAt optionalEquilibrium baseMean hbaseMean
        baseVariance ∧
      ¬LG21TestOptionalPolicies.lg21HiddenAccessOptionalDemographicallyFairAt optionalEquilibrium) ∧
  ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredLatentSkillFair requiredEquilibrium ∧
    ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredObservablyFairAt requiredEquilibrium baseMean hbaseMean
        baseVariance ∧
      ¬LG21TestOptionalPolicies.lg21HiddenAccessReportRequiredDemographicallyFairAt requiredEquilibrium
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_1_hidden_access_pbo_fails_all_fairness_definitions_proof

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

Reason: Theorem 3.1 concludes that both hidden-access PBO schedules fail latent-skill, observable, and demographic fairness in every equilibrium; the expanded source equilibria encode Gaussian scores, Bayesian output laws, best responses, and equilibrium entry stability, then universally conclude all three failures for arbitrary optional and report-required equilibria.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 7

Source locator: cited publication, lines 424-438

Verbatim source input

Theorem 3.1 (Strategic withholding). Suppose the school does not know access status and uses Bayesian optimal estimation PBO for all students. Then, some students with access do not report test scores. In particular, in every equilibrium:
• If the school does not require reporting conditional on taking the test, then Y (·) = 1. For each {θk }K−1
k=1 , there exists a finite test score threshold θ̄K such that an applicant reports their score, X({θk }K−1
k=1 , θK) = 1, if and only if θK ≥ θ̄K .
• If the school requires reporting conditional on taking the test (Y = X), then for each {θk }K−1
k=1 ,
there exists a finite skill threshold q̄ such that an applicant takes the test and reports their score, Y ({θk }K−1
k=1 , q) = 1, if and only if q ≥ q̄.
Furthermore, PBO satisfies none of the fairness definitions.
To prove this theorem, we first argue that neither everyone report or no one report is equilibrium.

[Next verbatim source excerpt]

who would be better off if they chose to report. So no one reports is also not stable. Thus we know that if an equilibrium exists, it must be that a fraction of the students report.

Expanded PaperInterface specification

```
∀ {Base : Type u_1} (testLaw : ℝ → Base → MeasureTheory.Measure ℝ),
  (∀ (skill : ℝ) (base : Base), MeasureTheory.IsProbabilityMeasure (testLaw skill base)) →
    ∀ (takeDecision : ℝ → Base → Bool) (reportDecision : Base → ℝ → Bool) (reportedPayoff : Base → ℝ → ℝ)
      (noReportPayoff : Base → ℝ),
      (∀ (skill : ℝ) (base : Base),
         MeasureTheory.Integrable
           (fun (score : ℝ) =>
             LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff reportDecision reportedPayoff noReportPayoff
               base score)
           (testLaw skill base)) →
        LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses testLaw takeDecision reportDecision reportedPayoff
          noReportPayoff →
        ∀ (scoreVariance : NNReal),
          scoreVariance ≠ 0 →
            (∀ (skill : ℝ) (base : Base), testLaw skill base = ProbabilityTheory.gaussianReal skill scoreVariance) →
              ∀ (intercept slope : Base → ℝ),
                (∀ (base : Base), 0 < slope base) →
                  (∀ (base : Base) (score : ℝ), reportedPayoff base score = intercept base + slope base * score) →
                    (∀ (base : Base) (score : ℝ),
                       reportedPayoff base score = noReportPayoff base → reportDecision base score = true) →
                      (∀ (skill : ℝ) (base : Base), takeDecision skill base = true) ∧
                        ∀ (base : Base),
                          ∃ (cutoff : ℝ),
                            (∀ (score : ℝ), reportDecision base score = true ↔ cutoff ≤ score) ∧
                              ∀ (skill : ℝ), 0 < (testLaw skill base) {score : ℝ | reportDecision base score = false}
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses
- LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: ℝ → Base → MeasureTheory.Measure ℝ
3. assumption: ∀ (skill : ℝ) (base : Base), MeasureTheory.IsProbabilityMeasure (testLaw skill base)
4. parameter: ℝ → Base → Bool
5. parameter: Base → ℝ → Bool
6. parameter: Base → ℝ → ℝ
7. parameter: Base → ℝ
8. assumption: ∀ (skill : ℝ) (base : Base),
   MeasureTheory.Integrable
     (fun (score : ℝ) =>
       LG21TestOptionalPolicies.lg21OptionalSourceContinuationPayoff reportDecision reportedPayoff noReportPayoff base
         score)
     (testLaw skill base)
9. assumption: LG21TestOptionalPolicies.LG21OptionalPointwiseBestResponses testLaw takeDecision reportDecision reportedPayoff
   noReportPayoff
10. parameter: NNReal
11. assumption: scoreVariance ≠ 0
12. assumption: ∀ (skill : ℝ) (base : Base), testLaw skill base = ProbabilityTheory.gaussianReal skill scoreVariance
13. parameter: Base → ℝ
14. parameter: Base → ℝ
15. assumption: ∀ (base : Base), 0 < slope base
16. assumption: ∀ (base : Base) (score : ℝ), reportedPayoff base score = intercept base + slope base * score
17. assumption: ∀ (base : Base) (score : ℝ), reportedPayoff base score = noReportPayoff base → reportDecision base score = true
18. conclusion: (∀ (skill : ℝ) (base : Base), takeDecision skill base = true) ∧
   ∀ (base : Base),

```
∃ (cutoff : ℝ),
  (∀ (score : ℝ), reportDecision base score = true ↔ cutoff ≤ score) ∧
    ∀ (skill : ℝ), 0 < (testLaw skill base) {score : ℝ | reportDecision base score = false}
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_1_optional_reporting_source_timed_proof

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

Reason: For the optional-reporting schedule, the source says everyone takes, reporting follows a finite score cutoff, and some scores are withheld; the expanded pointwise best-response target under a nondegenerate Gaussian and positive-slope affine reported payoff concludes universal taking, the exact cutoff iff, and positive nonreporting mass for every skill.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 8

Source locator: cited publication, lines 424-438

Verbatim source input

Theorem 3.1 (Strategic withholding). Suppose the school does not know access status and uses Bayesian optimal estimation PBO for all students. Then, some students with access do not report test scores. In particular, in every equilibrium:
• If the school does not require reporting conditional on taking the test, then Y (·) = 1. For each {θk }K−1
k=1 , there exists a finite test score threshold θ̄K such that an applicant reports their score, X({θk }K−1
k=1 , θK) = 1, if and only if θK ≥ θ̄K .
• If the school requires reporting conditional on taking the test (Y = X), then for each {θk }K−1
k=1 ,
there exists a finite skill threshold q̄ such that an applicant takes the test and reports their score, Y ({θk }K−1
k=1 , q) = 1, if and only if q ≥ q̄.
Furthermore, PBO satisfies none of the fairness definitions.
To prove this theorem, we first argue that neither everyone report or no one report is equilibrium.

[Next verbatim source excerpt]

We can then show that neither everyone takes the test and reports nor no one takes the test and reports are stable, so if an equilibrium exists, it must be that a fraction of students report. Under such situation, the function Y ({θk }K−1
k=1 , q, PBO) would be increasing with respect to q, so
that there is a threshold of q for test-taking. We can then finish by showing that there exists

Expanded PaperInterface specification

```
∀ {Base : Type u_1} {Test : Type u_2} [inst : MeasurableSpace Test] (testLaw : ℝ → Base → MeasureTheory.Measure Test),
  (∀ (skill : ℝ) (base : Base), MeasureTheory.IsProbabilityMeasure (testLaw skill base)) →
    ∀ (takeDecision : ℝ → Base → Bool) (reportedPayoff : Base → Test → ℝ) (noReportPayoff : Base → ℝ),
      (∀ (skill : ℝ) (base : Base), MeasureTheory.Integrable (reportedPayoff base) (testLaw skill base)) →
        LG21TestOptionalPolicies.LG21ReportRequiredPointwiseBestResponse testLaw takeDecision reportedPayoff
            noReportPayoff →
          ∀ (intercept slope : Base → ℝ),
            (∀ (base : Base), 0 < slope base) →
              (∀ (skill : ℝ) (base : Base),
                  LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff testLaw reportedPayoff skill
                      base =
                    intercept base + slope base * skill) →
                (∀ (skill : ℝ) (base : Base),
                    LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff testLaw reportedPayoff skill
                        base =
                      noReportPayoff base →
                    takeDecision skill base = true) →
                  ∀ (skillMean : Base → ℝ) (skillVariance : NNReal),
                    skillVariance ≠ 0 →
                      ∀ (base : Base),
                        ∃ (cutoff : ℝ),
                          (∀ (skill : ℝ), takeDecision skill base = true ↔ cutoff ≤ skill) ∧
                            0 <
                              (ProbabilityTheory.gaussianReal (skillMean base) skillVariance)
                                {skill : ℝ | takeDecision skill base = false}
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21ReportRequiredPointwiseBestResponse
- LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: MeasurableSpace Test
4. parameter: ℝ → Base → MeasureTheory.Measure Test
5. assumption: ∀ (skill : ℝ) (base : Base), MeasureTheory.IsProbabilityMeasure (testLaw skill base)
6. parameter: ℝ → Base → Bool
7. parameter: Base → Test → ℝ
8. parameter: Base → ℝ
9. assumption: ∀ (skill : ℝ) (base : Base), MeasureTheory.Integrable (reportedPayoff base) (testLaw skill base)
10. assumption: LG21TestOptionalPolicies.LG21ReportRequiredPointwiseBestResponse testLaw takeDecision reportedPayoff noReportPayoff
11. parameter: Base → ℝ
12. parameter: Base → ℝ
13. assumption: ∀ (base : Base), 0 < slope base
14. assumption: ∀ (skill : ℝ) (base : Base),
  LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff testLaw reportedPayoff skill base =
    intercept base + slope base * skill
15. assumption: ∀ (skill : ℝ) (base : Base),
  LG21TestOptionalPolicies.lg21ReportRequiredSourceTakeExpectedPayoff testLaw reportedPayoff skill base =
      noReportPayoff base →
    takeDecision skill base = true
16. parameter: Base → ℝ
17. parameter: NNReal

18. assumption: skillVariance ≠ 0
19. parameter: Base
20. conclusion: ∃ (cutoff : ℝ),
  (∀ (skill : ℝ), takeDecision skill base = true ↔ cutoff ≤ skill) ∧
    0 < (ProbabilityTheory.gaussianReal (skillMean base) skillVariance) {skill : ℝ | takeDecision skill base = false}

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_1_report_required_source_timed_proof

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

Reason: For reporting required after testing, the source gives a finite latent-skill cutoff with taking and reporting exactly above it and a nonempty withholding region; the expanded pre-score best-response target derives that cutoff iff from the positive-slope affine expected payoff and proves positive Gaussian mass below it.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 9

Source locator: cited publication, lines 455-461

Verbatim source input

Theorem 3.2 (Fairness impossibility). Suppose the school does not know access status and uses an
estimation policy P that satisfies either latent skill or observable fairness. Then, P is test-blank.
The proof proceeds via an unraveling argument. To satisfy those fairness notions and to avoid
being test-blank, for any X(·) the policy P must randomize skill estimates to equalize the estimate distributions. However, then, some students with X({θk
↪ }K−1
k=1 , θK) = 1 can strictly improve
their estimates by no longer reporting their score. Reporting unravels until no student reports in
equilibrium, which itself leads to the policy being test-blank.

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

For every supplied optional-reporting equilibrium/fibre of the approved clarified operational model, deterministic output on the reported-score branch, an
↪ arbitrary common no-report law, source fairness, and the operational Definition 5 convention imply zero reporter mass or equality of the realized
↪ output kernel and base-only law almost everywhere.

Archival source anchor: cited publication, lines 455-461

Expanded PaperInterface specification

```
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (model : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base),
  LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair model.policy ∨
     LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
   ∀ (e : model.policy.Equilibrium) (base : Base),
     LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank (model.scoreLaw e base)
       (model.policy.baseOnlyLaw e base) (model.reporterSet e base)
       (LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (model.reporterSet e base)
         (model.reporterSet_measurable e base) (model.reporterOutput e base) (model.reporterOutput_measurable e base)
         (model.policy.baseOnlyLaw e base))
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel
- LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank
- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: MeasurableSpace Base
3. parameter: LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base
4. assumption: LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair model.policy ∨
   LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy
5. parameter: model.policy.Equilibrium
6. parameter: Base
7. conclusion: LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank (model.scoreLaw e base) (model.policy.baseOnlyLaw e base)
   (model.reporterSet e base)
   (LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (model.reporterSet e base)
     (model.reporterSet_measurable e base) (model.reporterOutput e base) (model.reporterOutput_measurable e base)
     (model.policy.baseOnlyLaw e base))

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_2_optional_reporting_clarified_model_proof

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target
Reason: The approved non-equivalent optional-reporting target requires deterministic reported-score output, an arbitrary common no-report law, latent-skill or observable fairness, and operational blankness; the expanded model exposes those premises and concludes zero reporter mass or almost-everywhere equality of its piecewise realized kernel with the base law for every equilibrium fibre.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 10

Source locator: cited publication, lines 455-461

Verbatim source input

Theorem 3.2 (Fairness impossibility). Suppose the school does not know access status and uses an
estimation policy P that satisfies either latent skill or observable fairness. Then, P is test-blank.
The proof proceeds via an unraveling argument. To satisfy those fairness notions and to avoid
being test-blank, for any X(·) the policy P must randomize skill estimates to equalize the estimate distributions. However, then, some students with X({θk
↪ }K−1
k=1 , θK) = 1 can strictly improve
their estimates by no longer reporting their score. Reporting unravels until no student reports in
equilibrium, which itself leads to the policy being test-blank.

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

For every supplied report-required equilibrium/fibre of the approved clarified operational model, deterministic reported-score output, an arbitrary common
↪ no-take law, nonzero Gaussian variances, finite reporter expectations under every shifted Gaussian law, and source fairness imply zero taker mass or
↪ operational test blankness almost everywhere.

Archival source anchor: cited publication, lines 455-461

Expanded PaperInterface specification

```
∀ {Base : Type u_1} [inst : MeasurableSpace Base]
  (model : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base),
  LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair model.policy ∨
    LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
   ∀ (e : model.policy.Equilibrium) (base : Base),
    (ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base))
        (model.takerSet e base) =
      0 ∨
       ∀ᵐ (skill : ℝ) ∂ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base),
        (LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (model.takerSet e base)
            (model.takerSet_measurable e base) (model.reportedKernel e base) (model.policy.baseOnlyLaw e base))
          skill =
         model.policy.baseOnlyLaw e base
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel
- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Lean-elaborated claim roles

```
1. parameter: Type u_1
2. parameter: MeasurableSpace Base
3. parameter: LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base
4. assumption: LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair model.policy ∨
  LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy
5. parameter: model.policy.Equilibrium
6. parameter: Base
7. conclusion: (ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base))
     (model.takerSet e base) =
   0 ∨
  ∀ᵐ (skill : ℝ) ∂ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base),
   (LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (model.takerSet e base)
       (model.takerSet_measurable e base) (model.reportedKernel e base) (model.policy.baseOnlyLaw e base))
     skill =
    model.policy.baseOnlyLaw e base
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_2_report_required_clarified_model_proof

Recorded source-to-Spec screening **Verdict:** matches_approved_corrected_target
Reason: The approved non-equivalent report-required target uses deterministic output from noisy scores, an arbitrary no-take law, nonzero Gaussian variances, finite shifted-Gaussian expectations, and source fairness; those conditions are fields of the expanded model, whose conclusion is precisely zero taker mass or almost-everywhere operational-kernel equality.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation

Review row 11

Source locator: cited publication, lines 610-611

Verbatim source input

Theorem 4.4. Suppose the school observes access status Z. Then, the re-sampling policy PS is observably and demographically fair.

Settled source reading **Formalization gap:** the [Section 4 explanation](docs/SOURCE_CLARIFICATIONS.md#equilibrium-timing-population-laws-and-active-branches) records the added maximal active-branch selection, which has not been derived from the source equilibrium definition.

Expanded PaperInterface specification

```
∀ {Feature : Type u_1} [inst : Fintype Feature] [inst_1 : DecidableEq Feature]
  (source : LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature)
  (hnoAccess : 0 < source.population.accessLaw {false}),
  (have claim : (Bool × ℝ × (Feature → ℝ) → ℝ) → Prop := fun (actualOutput : Bool × ℝ × (Feature → ℝ) → ℝ) =>
      ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ))
        (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance :
        ℝ) (hbaseMean : Measurable baseMean) (hbaseLaw : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseVariance :
        0 < baseVariance),
        LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population
            source.test_feature =
          baseLaw.compProd
            (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
        let S :
          LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
            (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) :=
          { baseLaw := baseLaw,
            baseLaw_isProbability :=
              LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_1
                source baseLaw hbaseLaw,
            posteriorBaseMean := baseMean, posteriorBaseMean_measurable := hbaseMean,
            posteriorBaseVariance := baseVariance.toNNReal,
            posteriorBaseVariance_pos :=
              LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_2
                baseVariance hbaseVariance,
            testNoiseVariance := source.population.noiseVariance source.test_feature,
            testNoiseVariance_pos := source.test_noise_variance_positive };
        LG21TestOptionalPolicies.LG21ObservedAccessFair source.population source.test_feature
          { accessOutput := actualOutput,
            noAccessKernel := LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel S,
            noAccessKernel_isMarkov :=
              LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_3
                source baseLaw baseMean baseVariance hbaseMean hbaseLaw hbaseVariance };
    Nonempty (LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source) ∧
      Nonempty (LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source) ∧
        (∀ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
          claim
            (LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput
              (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
              (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
              LG21TestOptionalPolicies.lg21ContinuousPopulationSkill optionalProfile.selected.actions)) ∧
          ∀ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
            claim
              (LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput
                (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
                (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
                LG21TestOptionalPolicies.lg21ContinuousPopulationSkill reportRequiredProfile.selected)) ∧
    Nonempty (LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source) ∧
      ∀ (profile : LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source)
        (mandatoryNoReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ),
        LG21TestOptionalPolicies.LG21T44MandatoryGivenAccessResamplingFairnessCertificate source hnoAccess profile
          mandatoryNoReportPayoff
```

Retained governing declarations

- AppliedModelingLib.Probability.gaussianLocationKernel
- LG21TestOptionalPolicies.LG21ContinuousGaussianPopulation
- LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
- LG21TestOptionalPolicies.LG21NonTestFeature
- LG21TestOptionalPolicies.LG21ObservedAccessFair
- LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource
- LG21TestOptionalPolicies.LG21ObservedAccessOptionalPositiveMassRefinedEquilibrium
- LG21TestOptionalPolicies.LG21ObservedAccessTwoBranchOutput
- LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile

- LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile
- LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile
- LG21TestOptionalPolicies.LG21T44MandatoryGivenAccessResamplingFairnessCertificate
- LG21TestOptionalPolicies.lg21ContinuousGaussianAccessPopulationLaw
- LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw
- LG21TestOptionalPolicies.lg21ContinuousPopulationBase
- LG21TestOptionalPolicies.lg21ContinuousPopulationFeature
- LG21TestOptionalPolicies.lg21ContinuousPopulationSkill
- LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel
- LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput
- LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Fintype Feature
3. parameter: DecidableEq Feature
4. parameter: LG21TestOptionalPolicies.LG21ObservedAccessGaussianSource Feature
5. assumption: 0 < source.population.accessLaw {false}
6. conclusion: (have claim : (Bool × ℝ × (Feature → ℝ) → ℝ) → Prop := fun (actualOutput : Bool × ℝ × (Feature → ℝ) → ℝ) =>
   ∃ (baseLaw : MeasureTheory.Measure (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ))
    (baseMean : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ) (baseVariance : ℝ)
    (hbaseMean : Measurable baseMean) (hbaseLaw : MeasureTheory.IsProbabilityMeasure baseLaw) (hbaseVariance :
    0 < baseVariance),
    LG21TestOptionalPolicies.lg21ContinuousGaussianFullBaseLatentPrimitiveLaw source.population source.test_feature =
      baseLaw.compProd
       (AppliedModelingLib.Probability.gaussianLocationKernel baseMean hbaseMean baseVariance.toNNReal) ∧
     let S :
      LG21TestOptionalPolicies.LG21GaussianPBOResamplingSource
       (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) :=
      { baseLaw := baseLaw,
       baseLaw_isProbability :=
        LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_1
         source baseLaw hbaseLaw,
       posteriorBaseMean := baseMean, posteriorBaseMean_measurable := hbaseMean,
       posteriorBaseVariance := baseVariance.toNNReal,
       posteriorBaseVariance_pos :=
        LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_2
         baseVariance hbaseVariance,
       testNoiseVariance := source.population.noiseVariance source.test_feature,
       testNoiseVariance_pos := source.test_noise_variance_positive };
     LG21TestOptionalPolicies.LG21ObservedAccessFair source.population source.test_feature
      { accessOutput := actualOutput,
       noAccessKernel := LG21TestOptionalPolicies.lg21D6NoAccessResamplingEstimateKernel S,
       noAccessKernel_isMarkov :=
        LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocolsSpec._proof_3
         source baseLaw baseMean baseVariance hbaseMean hbaseLaw hbaseVariance };
   Nonempty (LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source) ∧
    Nonempty (LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source) ∧
     (∀ (optionalProfile : LG21TestOptionalPolicies.LG21OptionalActiveBranchProfile source),
       claim
        (LG21TestOptionalPolicies.lg21OptionalSourceTimedActualOutput
         (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
         (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
         LG21TestOptionalPolicies.lg21ContinuousPopulationSkill optionalProfile.selected.actions)) ∧
      ∀ (reportRequiredProfile : LG21TestOptionalPolicies.LG21ReportRequiredActiveBranchProfile source),
       claim
        (LG21TestOptionalPolicies.lg21ReportRequiredSequentialActualOutput
         (LG21TestOptionalPolicies.lg21ContinuousPopulationBase source.test_feature)
         (LG21TestOptionalPolicies.lg21ContinuousPopulationFeature source.test_feature)
         LG21TestOptionalPolicies.lg21ContinuousPopulationSkill reportRequiredProfile.selected)) ∧
   Nonempty (LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source) ∧
    ∀ (profile : LG21TestOptionalPolicies.LG21P42MandatoryGivenAccessPBOProfile source)
     (mandatoryNoReportPayoff : (LG21TestOptionalPolicies.LG21NonTestFeature Feature source.test_feature → ℝ) → ℝ),
     LG21TestOptionalPolicies.LG21T44MandatoryGivenAccessResamplingFairnessCertificate source hnoAccess profile
      mandatoryNoReportPayoff

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.PaperInterface.theorem4_4_all_observed_access_requirement_protocols

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

Reason: Theorem 4.4 states that when the school observes access status, the resampling policy is observably and demographically fair. The identity-bound approved context authorizes the source-timed positive-mass equilibrium reading for the optional and report-required protocols, fibrewise maximal active-branch selection, and PBO or response obligations only on attained branches. The expanded target requires nonempty selected carriers for those two protocols, applies the common resampling fairness claim to every carrier selected under that convention, and supplies the

corresponding mandatory-given-access certificate. Its LG21ObservedAccessFair dependency is exactly the conjunction of conditional-output-kernel equality and equality of the access output law with the resampled no-access output law; the Gaussian dependencies construct the same conditional resampling kernel from the shared non-test base law. Thus all three protocol instances conclude the source’s observable and demographic fairness without imposing an off-path PBO or a common cross-protocol action profile.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 12

Source locator: cited publication, lines 617-622

Verbatim source input

This procedure is akin to Thompson sampling for bandits, in which an action is played according to the probability that it maximizes the reward. Here, if a school accepts all students with skill estimates above some threshold q̄, then each student without access is accepted with probability equal to the probability they would receive a high enough score to be accepted had they taken the test; this probability is conditioned on everything the school knows about the student, {θk }K−1 k=1 .

Expanded PaperInterface specification

```
∀ {Omega : Type u_1} {Base : Type u_2} {Test : Type u_3} [inst : MeasurableSpace Omega] [inst_1 : MeasurableSpace Base]
  [inst_2 : MeasurableSpace Test]
  (experiment : LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test) (base : Base)
  (cutoff : ℝ),
  ((LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel experiment) base) (Set.Ici cutoff) =
     (experiment.testGivenBase base) {test : Test | cutoff ≤ experiment.posteriorEstimate base test} ∧
   (((ProbabilityTheory.Kernel.id.prod experiment.testGivenBase).map fun (z : Base × Test) =>
         experiment.posteriorEstimate z.1 z.2)
        base)
      (Set.Ici cutoff) =
    ((LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel experiment) base) (Set.Ici cutoff)
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy
- LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: MeasurableSpace Omega
5. parameter: MeasurableSpace Base
6. parameter: MeasurableSpace Test
7. parameter: LG21TestOptionalPolicies.LG21Definition6ExactResamplingPolicy Omega Base Test
8. parameter: Base
9. parameter: ℝ
10. conclusion: ((LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel experiment) base) (Set.Ici cutoff) =
    (experiment.testGivenBase base) {test : Test | cutoff ≤ experiment.posteriorEstimate base test} ∧
  (((ProbabilityTheory.Kernel.id.prod experiment.testGivenBase).map fun (z : Base × Test) =>
        experiment.posteriorEstimate z.1 z.2)
       base)
     (Set.Ici cutoff) =
   ((LG21TestOptionalPolicies.lg21Definition6ExactNoAccessEstimateKernel experiment) base) (Set.Ici cutoff)

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.thompson_acceptance_interpretation_proof

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

Reason: The source interprets threshold acceptance without access as the conditional probability that a synthetic score would produce a posterior estimate above the cutoff; the expanded target states that exact preimage probability and equates it with the actual-access posterior-estimate kernel at the same base and cutoff.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 13

Source locator: cited publication, lines 249-250

Verbatim source input

It is simple to see that X should not depend on q, given θK . Once the test score is sampled from the student’s skill level, the student’s admission probability does not depend on q.

Expanded PaperInterface specification

```
∀ {Skill : Type u_1} {Base : Type u_2} {Test : Type u_3} (report : Base → Test → Bool),
  LG21TestOptionalPolicies.PaperInterface.report_decision_skill_independent
    fun (x : Skill) (base : Base) (score : Test) => report base score
```

Retained governing declarations

- LG21TestOptionalPolicies.PaperInterface.report_decision_skill_independent

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: Base → Test → Bool
5. conclusion: LG21TestOptionalPolicies.PaperInterface.report_decision_skill_independent
   fun (x : Skill) (base : Base) (score : Test) => report base score

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.report_decision_skill_independence_proof

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

Reason: The source states that reporting depends on the realized test score and observed features rather than latent skill; the expanded target lifts an arbitrary base-and-score reporting rule to a skill-indexed rule and proves equality for every pair of skills at fixed base and score.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 14

Source locator: cited publication, lines 369

Verbatim source input

A school can trivially achieve these notions if test scores are ignored altogether.

Expanded PaperInterface specification

```
∀ (Skill : Type u_1) (Base : Type u_2) (Test : Type u_3) (Law : Type u_4) (Equilibrium : Type u_5) (law : Law),
  have S : LG21TestOptionalPolicies.LG21SourceLawPolicySurface Skill Base Test Law :=
    LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface Skill Base Test Law Equilibrium law;
  LG21TestOptionalPolicies.lg21SourceLawTestBlank S ∧
    LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair S ∧
      LG21TestOptionalPolicies.lg21SourceLawObservablyFair S ∧
        LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair S
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface
- LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
- LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair
- LG21TestOptionalPolicies.lg21SourceLawTestBlank

Lean-elaborated claim roles

1. parameter: Type u_1
2. parameter: Type u_2
3. parameter: Type u_3
4. parameter: Type u_4
5. parameter: Type u_5
6. parameter: Law
7. conclusion: LG21TestOptionalPolicies.lg21SourceLawTestBlank
   (LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface Skill Base Test Law Equilibrium law) ∧
  LG21TestOptionalPolicies.lg21SourceLawLatentSkillFair
     (LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface Skill Base Test Law Equilibrium law) ∧
    LG21TestOptionalPolicies.lg21SourceLawObservablyFair
       (LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface Skill Base Test Law Equilibrium law) ∧
      LG21TestOptionalPolicies.lg21SourceLawDemographicallyFair
       (LG21TestOptionalPolicies.PaperInterface.test_score_ignoring_law_surface Skill Base Test Law Equilibrium law)

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.ignoring_test_scores_achieves_fairness_proof

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

Reason: The source says ignoring test scores trivially achieves all fairness notions; the expanded ignoring-score surface assigns the same fixed law to every access, no-access, base-only, and full-feature field and concludes test blankness plus all three source fairness predicates.

Reviewer check: ☐ Matches source input

If left unchecked, explain the discrepancy or uncertainty below.

Reviewer annotation

Review row 15

Source locator: cited publication, lines 1779-1780

Verbatim source input

To sum up, any policy that satisfies demographic fairness will be either test-blank, or lead to no one reporting in equilibrium.

Formalized review target

The archival source above is not asserted equivalent to this formalized target.

For each of the two hidden-access voluntary schedules in the approved clarified operational model, observable fairness implies operational test blankness: zero reporting or taking, or equality of the realized output kernel and the base law almost everywhere.

Archival source anchor: cited publication, lines 1779-1780

Expanded PaperInterface specification

```
(∀ {Base : Type u_1} [inst : MeasurableSpace Base]
   (model : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base),
   LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
    ∀ (e : model.policy.Equilibrium) (base : Base),
     LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank (model.scoreLaw e base)
       (model.policy.baseOnlyLaw e base) (model.reporterSet e base)
       (LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (model.reporterSet e base)
         (model.reporterSet_measurable e base) (model.reporterOutput e base) (model.reporterOutput_measurable e base)
         (model.policy.baseOnlyLaw e base))) ∧
 ∀ {Base : Type u_2} [inst : MeasurableSpace Base]
   (model : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base),
   LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
    ∀ (e : model.policy.Equilibrium) (base : Base),
     (ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base))
         (model.takerSet e base) =
       0 ∨
       ∀ᵐ (skill :
         ℝ) ∂ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base),
         (LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (model.takerSet e base)
             (model.takerSet_measurable e base) (model.reportedKernel e base) (model.policy.baseOnlyLaw e base))
           skill =
         model.policy.baseOnlyLaw e base
```

Retained governing declarations

- LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel
- LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel
- LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank
- LG21TestOptionalPolicies.LG21SourceLawPolicySurface
- LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel
- LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel
- LG21TestOptionalPolicies.lg21SourceLawObservablyFair

Lean-elaborated claim roles

1. conclusion:

```
(∀ {Base : Type u_1} [inst : MeasurableSpace Base]
   (model : LG21TestOptionalPolicies.LG21ClarifiedOptionalReportingModel Base),
   LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
    ∀ (e : model.policy.Equilibrium) (base : Base),
     LG21TestOptionalPolicies.LG21OptionalOperationalTestBlank (model.scoreLaw e base)
       (model.policy.baseOnlyLaw e base) (model.reporterSet e base)
       (LG21TestOptionalPolicies.lg21OptionalDeterministicReporterKernel (model.reporterSet e base)
         (model.reporterSet_measurable e base) (model.reporterOutput e base) (model.reporterOutput_measurable e base)
         (model.policy.baseOnlyLaw e base))) ∧
 ∀ {Base : Type u_2} [inst : MeasurableSpace Base]
   (model : LG21TestOptionalPolicies.LG21ClarifiedReportRequiredModel Base),
   LG21TestOptionalPolicies.lg21SourceLawObservablyFair model.policy →
    ∀ (e : model.policy.Equilibrium) (base : Base),
     (ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base))
         (model.takerSet e base) =
       0 ∨
       ∀ᵐ (skill :
         ℝ) ∂ProbabilityTheory.gaussianReal (model.populationMean e base) (model.populationVariance e base),
         (LG21TestOptionalPolicies.lg21ReportRequiredOperationalKernel (model.takerSet e base)
             (model.takerSet_measurable e base) (model.reportedKernel e base) (model.policy.baseOnlyLaw e base))
           skill =
         model.policy.baseOnlyLaw e base
```

Lean proof endpoint (built separately)

LG21TestOptionalPolicies.ProofInterface.theorem3_2_observable_summary_clarified_proof

Recorded source-to-Spec screening

Verdict: matches_approved_corrected_target
Reason: The correction record expressly replaces the archival demographic summary with observable fairness and separates the two schedules; the expanded target does exactly that and concludes, for every equilibrium and base, zero optional reporting or report-required taking, or almost-everywhere equality of the realized kernel with the base law.
Reviewer check: ☐ Matches formalized target
If left unchecked, explain the discrepancy or uncertainty below.
Reviewer annotation