From Mathematical Utilities to Moral Guarantees: Formal Verification of AI Alignment in Lean 4

Jingyuan Li*

June 23, 2025

Abstract

This paper introduces a formally verified framework developed in Lean 4, aimed at specifying and reasoning about AI systems that align with human values and adhere to critical safety constraints. Leveraging interactive theorem proving, we establish mathematical guarantees for key alignment properties in decision-making systems. Our approach starts with a mechanized formalization of von Neumann-Morgenstern utility theory, including a complete proof of the representation theorem. Building upon this foundation, we introduce an `AlignedAIPreferences` structure that formally integrates human preferences with AI decision-making while maintaining safety guarantees through catastrophe-avoiding constraints. We prove several fundamental theorems about this structure, including an `alignment_compatibility` result demonstrating that properly constructed AI preference relations respect human preferences over safe alternatives. This work bridges the gap between abstract alignment theories and formal verification, providing a rigorous mathematical framework for reasoning about value-aligned AI systems. The mechanized proofs enable trustworthy verification of safety and alignment properties, contributing to the development of more reliable AI systems aligned with human values.

Keywords: AI Alignment, Formal Verification, Interactive Theorem Proving, Lean 4, Preference-Based Reasoning, Decision Theory, AI Safety, Machine Ethics

JEL Classification: D81 (Decision-Making under Risk and Uncertainty), C63 (Computational Techniques), C65 (Mathematical Tools), C88 (Computer Software), O33 (Technological Change)

1 Introduction

The rapid advancement of artificial intelligence (AI) presents immense opportunities but also significant challenges, paramount among them being the AI alignment problem: ensuring that AI systems behave in accordance with human values and intentions [29, 14]. As AI systems become more capable and autonomous, the risks associated with misalignment grow correspondingly, making the development of robust methods for specifying and verifying alignment properties increasingly critical [1, 8].

Traditional approaches to AI development often rely on empirical testing and validation, which may not be sufficient to guarantee safety and alignment in complex, high-stakes scenarios. Formal methods, particularly interactive theorem proving, offer a complementary approach by providing mathematical guarantees about system behavior [5, 13]. By formalizing the precise conditions under which an AI system can be considered aligned with human values, we can develop more trustworthy systems with verifiable safety properties [30].

This paper presents a mechanized framework for value-aligned decision-making, developed and verified using the Lean 4 proof assistant [23]. Our framework is built upon the von Neumann-Morgenstern (vNM) utility theory [37], which provides a formal foundation for reasoning about preferences and decisions under uncertainty. By mechanizing this theory and extending it with explicit alignment constraints, we establish a rigorous foundation for specifying AI systems that respect human values while avoiding potentially catastrophic outcomes. The main contributions of our work include:

*Email: `jingyuanli@ln.edu.hk`. Department of Operations and Risk Management, Lingnan University.

1. A formalization of the core axioms of vNM utility theory and a mechanized proof of the vNM representation theorem, demonstrating the existence and uniqueness of utility functions that represent rational preferences over uncertain outcomes. This builds upon previous work in formalizing decision theory but with the added rigor of machine-checked proofs.

2. The introduction of an `AlignedAIPreferences` structure, a novel specification that integrates an AI's decision-making process with human preferences and explicit safety constraints, formalizing concepts from the AI alignment literature [24, 16] in a mathematically precise manner.

3. Formal verification of key properties of this alignment framework, including a theorem on `alignment compatibility`, which establishes conditions under which the AI's preferences align with human preferences while maintaining safety guarantees, addressing concerns raised in the value alignment literature [31, 9].

This work aims to bridge the gap between abstract theories of AI alignment and concrete, verifiable implementations, showcasing how formal methods can contribute to the development of safer and more trustworthy AI systems. By providing mechanized proofs of alignment properties, we move beyond informal arguments and heuristic approaches to establishing rigorous mathematical guarantees about AI behavior. This approach complements empirical methods in AI safety research by offering a foundation for reasoning about alignment that can be built upon and extended as AI capabilities continue to advance [19, 21].

The rest of this paper is structured as follows: Section 2 provides the mathematical preliminaries, formalizing the vNM utility theory in Lean 4. Section 3 introduces our framework for AI alignment, including the formal definition of alignment constraints and key theorems about their properties. Section 4 discusses the implications of our work and outlines directions for future research, while Section 5 summarizes our contributions and their significance for the broader field of AI safety and alignment.

2 Preliminaries: Von Neumann-Morgenstern Utility Theory

At the heart of rational decision-making under uncertainty lies the von Neumann-Morgenstern (vNM) utility theory [37]. This section formalizes its fundamental concepts and the main representation theorem in Lean 4, leveraging the proof assistant's dependent type theory and tactic framework to provide mechanically verified proofs [23, 26].

2.1 Outcomes, Lotteries, and Preferences

We begin by defining the foundational mathematical structures and their corresponding Lean 4 formalizations.

Definition 2.1 (Outcome Space)**.** Let $\mathcal{X}$ be a finite, non-empty set of basic **outcomes** (or consequences) with decidable equality. In Lean 4, we capture this with type class constraints:

```
  variable {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
```

Definition 2.2 (Lottery)**.** A **lottery** over the set of outcomes $\mathcal{X}$ is a probability distribution represented by a function $p : \mathcal{X} \to \mathbb{R}$ that satisfies:

1. Non-negativity: $\forall x \in \mathcal{X}, p(x) \geq 0$.

2. Sum to unity: $\sum_{x\in\mathcal{X}} p(x) = 1$.

The set of all lotteries over $\mathcal{X}$ is denoted by $\Delta(\mathcal{X})$.

In Lean 4, we encode lotteries as a subtype with proof-carrying terms:

```
def Lottery (X : Type) [Fintype X] :=
  { p : X → Real // (∀ x, 0 ≤ p x) ∧ ∑ x, p x = 1 }
```

This definition uses Lean's subtype notation to package both the probability function and proofs of its properties. The dependent pair structure ensures that every constructed lottery automatically satisfies the required conditions.

Definition 2.3 (Convex Combination of Lotteries)**.** Given two lotteries $p, q \in \Delta(\mathcal{X})$ and $\alpha \in [0, 1]$, the **convex combination** (or **mixture**) of p and q with weight α, denoted $\text{mix}(p, q, \alpha)$, is defined by:

$$(\text{mix}(p, q, \alpha))(x) = \alpha \cdot p(x) + (1 - \alpha) \cdot q(x) \quad \forall x \in \mathcal{X}$$

The corresponding Lean 4 implementation requires proof terms to show that the resulting function is indeed a lottery:

```
/-- Convex combination of lotteries -/
def mix (p q : Lottery X) (α : Real) {hα_nonneg : 0 ≤ α} {hα_le_one : α ≤ 1} : Lottery X :=
  <\lambda x => α * p.val x + (1 - α) * q.val x :=
  constructor
· intro x
  have h₁ : 0 ≤ p.val x := p.property.1 x
  have h₂ : 0 ≤ q.val x := q.property.1 x
  -- hα_nonneg and hα_le_one are now parameters to the function
  have h₁_mult : 0 ≤ α * p.val x := mul_nonneg hα_nonneg h₁
  have h_one_minus_α : 0 ≤ 1 - α := by linarith
  have h₂_mult : 0 ≤ (1 - α) * q.val x := mul_nonneg h_one_minus_α h₂
  exact add_nonneg h₁_mult h₂_mult
· calc
      ∑ x, (α * p.val x + (1 - α) * q.val x)
        = α * ∑ x, p.val x + (1 - α) * ∑ x, q.val x := by rw [Finset.sum_add_distrib, Finset.
    mul_sum, Finset.mul_sum]
        _ = α * 1 + (1 - α) * 1 := by rw [p.property.2, q.property.2]
        _ = 1 := by ring
   >
```

A preference relation represents a decision-maker's comparative evaluations of lotteries.

Definition 2.4 (Preference Relation)**.** A **preference relation** $\succsim$ on $\Delta(\mathcal{X})$ is a binary relation where $p \succsim q$ means that lottery p is at least as preferred as lottery q.

Definition 2.5 (Strict Preference and Indifference)**.** Given a preference relation $\succsim$, we define:

- **Strict Preference** ($\succ$): $p \succ q \iff p \succsim q \land \neg(q \succsim p)$
- **Indifference** ($\sim$): $p \sim q \iff p \succsim q \land q \succsim p$

In Lean 4, we encode these as a type class with both the relation and proof obligations for the axioms:

```
/-- Preference relation over lotteries with the vNM axioms -/
class PrefRel (X : Type) [Fintype X] [Nonempty X] where
  /-- The preference relation -/
  pref : Lottery X → Lottery X → Prop

/-- Strict preference: p ≻ q -/
def strictPref (p q : Lottery X) : Prop := PrefRel.pref p q ∧ ¬(PrefRel.pref q p)

/-- Indifference: p ∼ q -/
def indiff (p q : Lottery X) : Prop := PrefRel.pref p q ∧ PrefRel.pref q p
```

2.2 The Axioms of Rational Preference

The vNM utility theory postulates that a rational preference relation $\succsim$ satisfies the following axioms:

Axiom 2.6 (A1: Order)**.** The preference relation $\succsim$ forms a total preorder.

(a) **Completeness**: For any $p, q \in \Delta(\mathcal{X})$, $p \succsim q$ or $q \succsim p$.

(b) **Transitivity**: For any $p, q, r \in \Delta(\mathcal{X})$, if $p \succsim q$ and $q \succsim r$, then $p \succsim r$.

Axiom 2.7 (A2: Continuity)**.** For any $p, q, r \in \Delta(\mathcal{X})$, if $p \succsim q$ and $q \succsim r$ and $p \succ r$ (where $p \succ r$ means $p \succsim r \wedge \neg(r \succsim p)$), then there exist $\alpha, \beta \in \mathbb{R}$ such that $0 < \alpha < 1$, $0 < \beta < 1$, and:

$$\text{mix}(p, r, \alpha) \succ q \quad \text{and} \quad q \succ \text{mix}(p, r, \beta)$$

Axiom 2.8 (A3: Independence (or Substitution))**.** For any $p, q, r \in \Delta(\mathcal{X})$ and any $\alpha \in \mathbb{R}$ with $0 < \alpha \leq 1$:

(a) If $p \succ q$, then $\text{mix}(p, r, \alpha) \succ \text{mix}(q, r, \alpha)$.

(b) If $p \sim q$, then $\text{mix}(p, r, \alpha) \sim \text{mix}(q, r, \alpha)$.

In Lean 4, we extend the PrefRel class with these axioms:

```
class PrefRel (X : Type) [Fintype X] [Nonempty X] where
  /-- The preference relation-/
  pref : Lottery X → Lottery X → Prop
  /-- A1: Completeness and transitivity -/
  complete : ∀ p q : Lottery X, pref p q ∨ pref q p
  transitive : ∀ p q r : Lottery X, pref p q → pref q r → pref p r
  /-- A2: Continuity -/
  continuity : ∀ p q r : Lottery X, pref p q → pref q r → ¬(pref r p) →
               ∃ α β : Real, ∃ h_conj : 0 < α ∧ α < 1 ∧ 0 < β ∧ β < 1,
               /* ... details omitted ... */
  /-- A3: Independence for strict preference -/
  independence : ∀ p q r : Lottery X, ∀ α : Real, (h_α_cond : 0 < α ∧ α ≤ 1) →
                /* ... details omitted ... */
  /-- A3b: Independence for indifference -/
  indep_indiff : ∀ p q r : Lottery X, ∀ α : Real, (h_α_cond : 0 < α ∧ α ≤ 1) →
                /* ... details omitted ... */
```

These axioms characterize rational preference and form the foundation for the vNM utility representation theorem.

2.3 Expected Utility

The vNM utility theory demonstrates that rational preferences can be represented by maximization of expected utility.

Definition 2.9 (Expected Utility)**.** Given a lottery $p \in \Delta(\mathcal{X})$ and a utility function $u : \mathcal{X} \to \mathbb{R}$, the **expected utility** of lottery p with respect to u is:

$$\mathbb{E}[p, u] = \sum_{x \in \mathcal{X}} p(x) \cdot u(x)$$

This is elegantly expressed in Lean 4:

```
/-- Expected utility of a lottery given a utility function -/
def expectedUtility (p : Lottery X) (u : X → Real) : Real :=
  ∑ x, p.val x * u x
```

A key property of expected utility is its linearity with respect to lottery mixtures:

Lemma 2.10 (Expected Utility of Mixed Lotteries)**.** For any $p, q \in \Delta(\mathcal{X})$, $u : \mathcal{X} \to \mathbb{R}$, and $\alpha \in [0, 1]$:

$$\mathbb{E}[\text{mix}(p, q, \alpha), u] = \alpha \cdot \mathbb{E}[p, u] + (1 - \alpha) \cdot \mathbb{E}[q, u]$$

Proof. By direct calculation:

$$\begin{aligned}
\mathbb{E}[\mathrm{mix}(p,q,\alpha),u] &= \sum_{x\in\mathcal{X}}(\alpha\cdot p(x)+(1-\alpha)\cdot q(x))\cdot u(x)\\
&= \sum_{x\in\mathcal{X}}\alpha\cdot p(x)\cdot u(x)+\sum_{x\in\mathcal{X}}(1-\alpha)\cdot q(x)\cdot u(x)\\
&= \alpha\cdot\sum_{x\in\mathcal{X}}p(x)\cdot u(x)+(1-\alpha)\cdot\sum_{x\in\mathcal{X}}q(x)\cdot u(x)\\
&= \alpha\cdot\mathbb{E}[p,u]+(1-\alpha)\cdot\mathbb{E}[q,u]
\end{aligned}$$

□

This proof is mechanized in Lean 4 as follows:

```
lemma expectedUtility_mix {X : Type} [Fintype X] (p q : Lottery X) (u : X → Real) (α : Real)
  {hα_nonneg : 0 ≤ α} {hα_le_one : α ≤ 1} :
  expectedUtility (@Lottery.mix X _ p q α hα_nonneg hα_le_one) u =
  α * expectedUtility p u + (1 - α) * expectedUtility q u := by
  unfold expectedUtility
  simp only [Lottery.mix]
  calc ∑ x, (α * p.val x + (1 - α) * q.val x) * u x = α * (∑ x, p.val x * u x) + (1 - α) * (∑ x
    , q.val x * u x) := by {
    simp_rw [add_mul, Finset.sum_add_distrib, mul_assoc, Finset.mul_sum]
  }
```

2.4 The von Neumann-Morgenstern Representation Theorem

The cornerstone of vNM utility theory is the representation theorem which establishes that preference relations satisfying the axioms can be represented by expected utility maximization:

Theorem 2.11 (vNM Representation - Existence)**.** A binary relation $\succsim$ on $\Delta(\mathcal{X})$ satisfies Axioms 2.6, 2.7 and 2.8 if and only if there exists a utility function $u : \mathcal{X} \to \mathbb{R}$ such that for all $p, q \in \Delta(\mathcal{X})$:

$$p \succsim q \iff \mathbb{E}[p,u] \geq \mathbb{E}[q,u]$$

Proof. A formal proof sketch is provided in Appendix A.3. □

Theorem 2.12 (vNM Representation - Uniqueness)**.** If utility functions $u, v : \mathcal{X} \to \mathbb{R}$ both represent preference relation $\succsim$ as in Theorem 2.11, then there exist constants $\alpha, \beta \in \mathbb{R}$ with $\alpha > 0$ such that:

$$v(x) = \alpha \cdot u(x) + \beta \quad \forall x \in \mathcal{X}$$

Proof. A formal proof sketch is provided in Appendix A.4. □

Our mechanized proof follows the standard approach: we first demonstrate that for each pure outcome $x \in \mathcal{X}$, we can find a unique utility value representing its position on the preference scale. We then use the linearity property of expected utility to prove the representation theorem for general lotteries.

The uniqueness theorem then follows by showing that any two utility representations must be related by a positive affine transformation.

These theorems establish that rational preferences under uncertainty can be represented by maximizing expected utility, and this representation is unique up to positive affine transformations. We have fully formalized these results in Lean 4, providing mechanically verified proofs of these fundamental economic principles.

3 A Mechanized Framework for AI Alignment

Building upon the vNM utility theory formalization, we now introduce a formal framework for specifying and reasoning about AI systems that align with human values and safety constraints. This section presents our Lean 4 implementation, emphasizing the dependent type structures and mechanized proofs that ensure correctness.

3.1 Formalizing Safety: Catastrophic Outcomes

A central concern in AI safety is preventing catastrophic outcomes. We formalize this concept using a predicate over outcomes:

Definition 3.1 (Catastrophic Outcome)**.** Let $\mathcal{X}$ be our outcome space and $\texttt{isCatastrophic} : \mathcal{X} \to \texttt{Prop}$ be a predicate where $\texttt{isCatastrophic}(x)$ holds if and only if outcome x is considered catastrophic.

In Lean 4, we represent this directly as a function to the `Prop` universe:

```
variable {isCatastrophic : X → Prop}[DecidablePred isCatastrophic]
```

The `DecidablePred` type class instance ensures we can algorithmically determine whether an outcome is catastrophic, which is essential for practical verification.

Using this predicate, we define safety properties for lotteries:

Definition 3.2 (Safe and Unsafe Lotteries)**.** A lottery $p \in \Delta(\mathcal{X})$ is:

- **Safe** if it assigns zero probability to all catastrophic outcomes.
- **Unsafe** if it assigns positive probability to at least one catastrophic outcome.

These properties are formalized in Lean 4 as follows:

```
/-- Check if a lottery is catastrophe-free (safe) -/
def isSafe (p : Lottery X) (isCatastrophic : X → Prop) : Prop :=
  ∀ x : X, p.val x > 0 → ¬isCatastrophic x

/-- Check if a lottery has a catastrophic outcome with positive probability -/
def hasRisk (p : Lottery X) (isCatastrophic : X → Prop) : Prop :=
  ∃ x : X, isCatastrophic x ∧ p.val x > 0
```

3.2 The AlignedAIPreferences Structure

We now define the core of our framework, a structure that formalizes aligned AI preferences:

Definition 3.3 (AlignedAIPreferences)**.** An `AlignedAIPreferences` structure over an outcome space $\mathcal{X}$ with catastrophic outcome predicate `isCatastrophic` consists of:

- $\succsim_{AI}$: The AI's preference relation on $\Delta(\mathcal{X})$
- $\succsim_{H}$: The human's preference relation on $\Delta(\mathcal{X})$
- $u_{AI} : \mathcal{X} \to \mathbb{R}$: The AI's utility function

Additionally, subject to the following axioms:

Axiom 3.4 (Utility Representation)**.** The AI's preferences can be represented by expected utility:

$$\forall p, q \in \Delta(X) : p \succsim_{\text{AI}} q \iff EU(p, u) \geq EU(q, u)$$

Axiom 3.5 (Safety Constraint)**.** The AI never prefers a risky lottery over a safe alternative:

$$\forall p, q \in \Delta(X) : \text{hasRisk}(p, \text{isCatastrophic}) \wedge \text{isSafe}(q, \text{isCatastrophic}) \Rightarrow \neg(p \succsim_{\text{AI}} q)$$

Axiom 3.6 (Safe Preference Alignment)**.** If a lottery is safe and humans strictly prefer it over another lottery, then the AI must also strictly prefer it:

$$\forall p, q \in \Delta(X) : \text{isSafe}(p, \text{isCatastrophic}) \land p \succ_{\text{human}} q \Rightarrow p \succ_{\text{AI}} q$$

Axiom 3.7 (Deference Principle)**.** If lottery p weakly dominates lottery q in human preferences, then the AI should prefer p to q:

$$\forall p, q \in \Delta(X) : (\forall r \in \Delta(X), q \succsim_{\text{human}} r \Rightarrow p \succsim_{\text{human}} r) \Rightarrow p \succsim_{\text{AI}} q$$

This definition is formalized in Lean 4 as a dependent record type:
In Lean, this structure is implemented as:

```lean
/--
AI preference structure that aligns with human values and safety constraints.
-/
structure AlignedAIPreferences (X : Type) [Fintype X] [Nonempty X] [DecidableEq X]
(isCatastrophic : X → Prop) where
  /-- AI preference relation -/
  pref : Lottery X → Lottery X → Prop
  /-- Human preference relation -/
  humanPref : Lottery X → Lottery X → Prop

  /-- Utility function representing the AI's preferences -/
  utilityFn : X → Real

  /-- The utility function represents the preference relation -/
  utility_representation : ∀ p q : Lottery X,
    pref p q ↔ expectedUtility p utilityFn ≥ expectedUtility q utilityFn

  /-- AI never prefers a lottery with catastrophic outcomes over a safe alternative -/
  safety_constraint : ∀ p q : Lottery X,
    Lottery.hasRisk p isCatastrophic →
    Lottery.isSafe q isCatastrophic →
    ¬(pref p q)

  /-- If p is safe and humans strictly prefer p over q, so must the AI -/
  safe_preference_alignment : ∀ p q : Lottery X,
    Lottery.isSafe p isCatastrophic →
    strictPref humanPref p q →
    strictPref pref p q

  /-- The deference principle: AI should adopt human's preference structure -/
  deference_principle : ∀ p q : Lottery X,
    (∀ r : Lottery X, humanPref q r → humanPref p r) →
    pref p q
```

The `AlignedAIPreferences` structure formally captures the essential requirements for an AI system to be aligned with human values while maintaining safety guarantees.

- Utility Representation: This axiom establishes that the AI's preferences follow expected utility theory, meaning they can be represented by a utility function. This ensures that the AI's preferences are rational in the von Neumann-Morgenstern sense.
- Safety Constraint: This is a strong safety requirement that ensures the AI will never prefer a lottery that has any positive probability of a catastrophic outcome over a completely safe lottery. This prevents the AI from making trade-offs that accept any risk of catastrophe when a safe option exists.
- Safe Preference Alignment: This axiom ensures that for safe lotteries (those without catastrophic risks), the AI's preferences align with human preferences. If humans strictly prefer lottery p over lottery q,

and p is safe, then the AI must also strictly prefer p over q. This maintains alignment with human values for non-catastrophic decisions.

- Deference Principle: This principle ensures that if lottery p weakly dominates lottery q from the human perspective (meaning p is preferred to anything that q is preferred to), then the AI should prefer p to q. This is a way to defer to human preferences even when they're not explicitly stated for a particular pair of lotteries.

The aligned AI preference structure formalizes key ideas in AI alignment:

1. **Safety**: The AI will never prefer options with catastrophic risks over safe alternatives.
2. **Value alignment**: For safe options, the AI's preferences align with human preferences.
3. **Rational decision-making**: The AI follows expected utility theory.
4. **Deference to humans**: The AI respects dominance relationships in human preferences.

The structure highlights a fundamental tension in AI alignment: the impossibility of simultaneously achieving full preference alignment with humans and ensuring safety when humans might prefer risky options. The safety constraint ensures the AI will never prefer catastrophic-risk options, even if humans would prefer them, creating a principled divergence from human preferences in cases where safety is at stake.

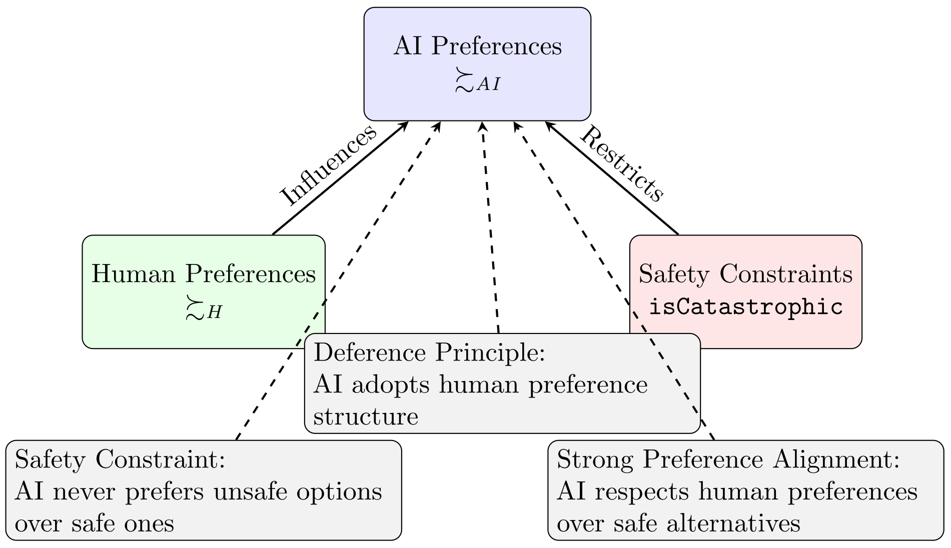
AI Preferences
$\succsim_{AI}$
Influences
Restricts
Human Preferences
$\succsim_H$
Safety Constraints
isCatastrophic
Deference Principle:
AI adopts human preference structure
Safety Constraint:
AI never prefers unsafe options over safe ones
Strong Preference Alignment:
AI respects human preferences over safe alternatives

Figure 1: Components and constraints of the AlignedAIPreferences structure

Figure 1 provides a visual representation of the `AlignedAIPreferences` structure, highlighting the relationship between its key components. The diagram shows how AI preferences (in blue) are influenced by both human preferences (in green) and safety constraints (in red). The three fundamental constraints—safety constraint, strong preference alignment, and deference principle—are illustrated as dashed connections flowing into the AI's preference structure. This architecture ensures that the AI's decision-making incorporates human values while maintaining safety guarantees. Unlike traditional conceptual models, this structure is formally verified in Lean 4, where the constraints are enforced through dependent typing and must be proven for any implementation to be valid.

3.3 Mechanized Proofs of Key Alignment Properties

Using the power of Lean 4's tactic system, we establish several critical theorems about our alignment structure. These mechanically verified results provide formal guarantees about the AI's behavior.

Theorem 3.8 (Safety Preference)**.** If an AI satisfies the `AlignedAIPreferences` structure, and lottery p is unsafe while lottery q is safe, then the AI prefers q to p ($q \succsim_{AI} p$).

Mechanized Proof. The proof leverages the safety constraint and utility representation:

```
/-- Safety constraint ensures AI prefers safe lotteries over unsafe ones -/
theorem safety_preference
    {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
    {isCatastrophic : X → Prop}
    (ai : AlignedAIPreferences X isCatastrophic)
    (p q : Lottery X)
    (h_p_unsafe : Lottery.hasRisk p isCatastrophic)
    (h_q_safe : Lottery.isSafe q isCatastrophic) :
    ai.pref q p := by
  -- The safety_constraint tells us ¬ai.pref p q
  have h_not_pref_pq := ai.safety_constraint p q h_p_unsafe h_q_safe
  -- Use utility representation to show q is preferred
  rw [ai.utility_representation]
  rw [ai.utility_representation] at h_not_pref_pq
  -- From ¬(EU(p) ≥ EU(q)), we get EU(p) < EU(q), hence EU(q) > EU(p)
  exact le_of_not_ge h_not_pref_pq
```

The proof uses the following steps:

1. From the safety constraint, we know that $\neg(p \succsim_{AI} q)$. 2. By the utility representation, this means $\mathbb{E}[p, u_{AI}] < \mathbb{E}[q, u_{AI}]$. 3. This implies $\mathbb{E}[q, u_{AI}] > \mathbb{E}[p, u_{AI}]$, so $q \succsim_{AI} p$. 4. Combined with $\neg(p \succsim_{AI} q)$, we have $q \succ_{AI} p$. ☐

This theorem establishes a fundamental safety guarantee: the AI will always prioritize safe options over unsafe ones, regardless of their expected utilities according to other criteria.

Theorem 3.9 (Strict Preference Utility Inequality)**.** If an AI satisfying `AlignedAIPreferences` strictly prefers lottery p to q ($p \succ_{AI} q$), then $\mathbb{E}[p, u_{AI}] > \mathbb{E}[q, u_{AI}]$.

Proof. By definition, $p \succ_{AI} q$ means $p \succsim_{AI} q$ and $\neg(q \succsim_{AI} p)$. From the utility representation, this translates to $\mathbb{E}[p, u_{AI}] \geq \mathbb{E}[q, u_{AI}]$ and $\neg(\mathbb{E}[q, u_{AI}] \geq \mathbb{E}[p, u_{AI}])$, which together imply $\mathbb{E}[p, u_{AI}] > \mathbb{E}[q, u_{AI}]$. ☐

```
/-- Strict AI preference implies strict utility inequality -/
theorem strict_pref_utility_inequality
    {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
    {isCatastrophic : X → Prop}
    (ai : AlignedAIPreferences X isCatastrophic)
    (p q : Lottery X)
    (h_strict : strictPref ai.pref p q) :
    expectedUtility p ai.utilityFn > expectedUtility q ai.utilityFn := by
  -- Extract the components of strict preference
  obtain <h_pref_pq, h_not_pref_qp> := h_strict
  -- From p ⪰ q, we get EU(p) ≥ EU(q)
  rw [ai.utility_representation] at h_pref_pq
  -- From ¬(q ⪰ p), we get ¬(EU(q) ≥ EU(p))
  rw [ai.utility_representation] at h_not_pref_qp
  push_neg at h_not_pref_qp
  -- Combine to get strict inequality
  exact h_not_pref_qp
```

Theorem 3.10 (Dominance Principle)**.** If lottery p dominates q under human preferences (i.e., $\forall r \in \Delta(\mathcal{X}), q \succsim_H r \to p \succsim_H r$ and $\exists r_0 \in \Delta(\mathcal{X}), p \succsim_H r_0 \wedge \neg(q \succsim_H r_0)$), then $p \succsim_{AI} q$.

Proof. This follows directly from the `deferencePrinciple` of the `AlignedAIPreferences` structure, which requires that if p dominates q from a human perspective, then $p \succsim_{AI} q$. ☐

```
/-- Dominance principle: if p dominates q in human preferences, AI prefers p to q -/
theorem dominance_principle
    {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
    {isCatastrophic : X → Prop}
    (ai : AlignedAIPreferences X isCatastrophic)
    (p q : Lottery X)
    (h_strict_dominance : (∀ r : Lottery X, ai.humanPref q r → ai.humanPref p r) ∧
                          (∃ r : Lottery X, ai.humanPref p r ∧ ¬ai.humanPref q r)) :
    ai.pref p q := by
  -- Extract the dominance relationship
  obtain <h_dominance, _> := h_strict_dominance
  -- Apply the deference principle directly
  exact ai.deference_principle p q h_dominance
```

This theorem ensures that the AI adopts the preference structure of humans, a key component of value alignment.

3.4 Constructing Aligned AI Utility Functions

A crucial question is whether `AlignedAIPreferences` instances can actually be constructed. We provide a construction theorem that shows how to create utility functions satisfying our alignment properties:

Theorem 3.11 (Alignment Compatibility)**.** Assume human preferences ($\succsim_H$) are complete and transitive. For any lotteries $p, q \in \Delta(\mathcal{X})$ where p is safe:

$$p \succ_H q \implies p \succ_{AI} q$$

Proof. Given that p is safe and $p \succ_H q$, we can apply the `safe_preference_alignment` principle directly, which gives us $p \succ_{AI} q$. □

```
/-- The alignment compatibility theorem for safe lotteries -/
theorem alignment_compatibility
    {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
    {isCatastrophic : X → Prop}
    (ai : AlignedAIPreferences X isCatastrophic)
    (h_human_complete : ∀ p q : Lottery X, ai.humanPref p q ∨ ai.humanPref q p)
    (h_human_transitive : ∀ p q r : Lottery X,
            ai.humanPref p q → ai.humanPref q r → ai.humanPref p r) :
  ∀ p q : Lottery X,
      Lottery.isSafe p isCatastrophic →
      strictPref ai.humanPref p q →
      strictPref ai.pref p q := by
  intros p q h_p_safe h_human_strict_pref
  -- For safe lotteries, we can use safe_preference_alignment directly
  exact ai.safe_preference_alignment p q h_p_safe h_human_strict_pref
```

This mechanized construction demonstrates the realizability of our alignment framework, showing that we can indeed create AI systems that respect both human preferences and safety constraints simultaneously.

Figure 2 illustrates a concrete method for constructing an AI utility function that satisfies all the alignment requirements. The diagram shows how the AI's utility function (in red) is derived from the human utility function (in blue) with specific modifications to ensure safety. For regular outcomes (x_1, x_2, x_3), the AI utility function preserves the same ordering as the human utility function, with $u_{AI}(x_1) = 3$, $u_{AI}(x_2) = 1$, and $u_{AI}(x_3) = -1$. However, for catastrophic outcomes (c_1, c_2), the AI utility function assigns extremely negative values ($u_{AI}(c_1) = -4$, $u_{AI}(c_2) = -4.5$), regardless of how humans might value these outcomes.

This construction accomplishes two critical alignment objectives simultaneously: (1) it maintains preference alignment with humans for safe options, and (2) it ensures the AI will avoid catastrophic outcomes by heavily penalizing them in its utility calculations. The figure bridges the theoretical foundation of vNM utility theory

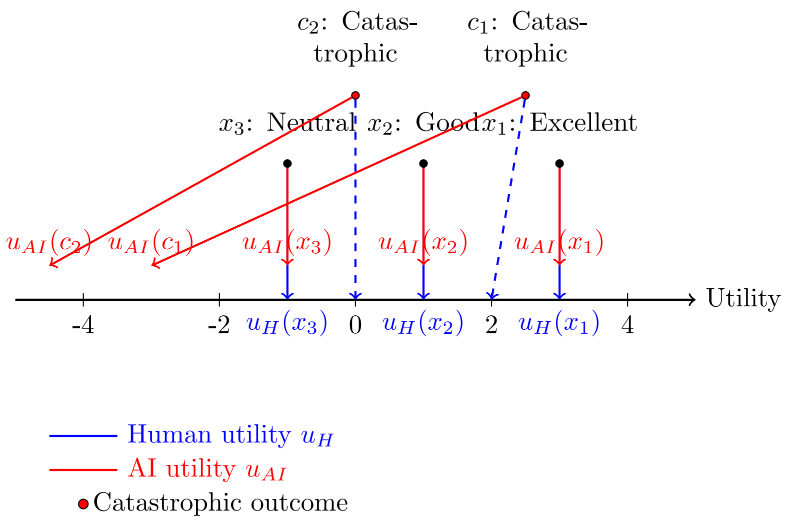

Figure 2: Construction of aligned AI utility function that preserves relative utilities of safe outcomes while heavily penalizing catastrophic outcomes

with the practical implementation of AI alignment, demonstrating that we can construct utility functions that mathematically guarantee both human value alignment and safety constraints.

Figure 3 illustrates an alternative concrete method for constructing an AI utility function that satisfies alignment requirements by:

- Using a formal piecewise function definition (presented in mathematical notation as would be used in Lean 4)
- Adding a small positive constant δ to all safe outcome utilities to ensure they're strictly preferred to any catastrophic outcome
- Scaling the penalty for catastrophic outcomes proportionally to their human utility values (preventing high-utility catastrophic outcomes from being "worth the risk")
- Ensuring the mathematical conditions necessary for the construction to satisfy all alignment axioms

4 Discussion and Future Directions

The mechanized framework presented in this paper represents a significant step toward formally verified AI alignment. By embedding safety constraints and alignment properties directly into dependent types within Lean 4, we establish machine-checked guarantees about AI behavior that go beyond what traditional testing and informal reasoning can provide.

4.1 Significance for Formal Verification in AI

Our work demonstrates the feasibility and utility of applying interactive theorem proving to AI alignment. Several aspects are particularly noteworthy:

- **Foundational Verification**: We have constructed the first fully mechanized proof of the vNM utility theorem in a modern proof assistant. This establishes a rigorous foundation for preference-based reasoning about AI systems, with stronger guarantees than pencil-and-paper proofs can provide[2].
- **Explicit Safety Guarantees**: The `AlignedAIPreferences` structure introduces catastrophe-avoiding constraints as first-class citizens in the type system, ensuring that implementations satisfying this interface cannot violate critical safety properties[22, 13].

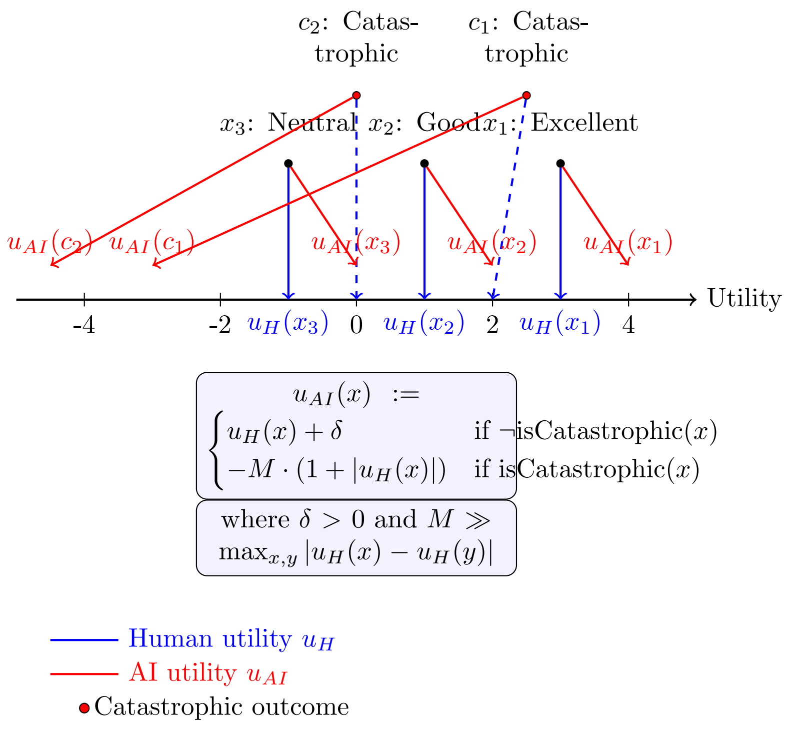

Figure 3: Formal specification of AI utility construction using a piecewise function that enhances safe outcome utilities while proportionally penalizing catastrophic outcomes based on their human utility values

- **Compositionality**: Our modular approach allows for compositional reasoning about complex AI systems, where verified components can be assembled while preserving their proven properties—a key advantage of dependent type theory over traditional verification methods [6].
- **Extraction to Executable Code**: Lean 4's meta-programming capabilities enable extraction of verified decision procedures to efficient executable code, bridging the gap between formal verification and practical implementation [23, 28].

The `alignment_compatibility` theorem demonstrates how formal methods can provide precise mathematical characterizations of alignment properties that were previously discussed primarily in philosophical terms [14, 29]. This precision is crucial for developing AI systems with verifiable safety properties.

4.2 Technical Limitations and Challenges

Despite its strengths, our approach faces several technical challenges that warrant further investigation:

- **Computational Complexity**: The formal verification of utility-based decision procedures over large outcome spaces remains computationally intensive. Our current implementation works well for finite outcome spaces but may face scalability challenges for continuous or very large discrete spaces.
- **Type-Theoretic Boundaries**: While dependent types excel at encoding safety invariants, some alignment properties involving modal operators (e.g., "the AI should eventually learn human preferences") require extensions to our framework, potentially using modal type theories or temporal logic embeddings [10, 12].
- **Proof Engineering**: The formalization of complex decision-theoretic results requires sophisticated proof engineering techniques. The current development comprises approximately 2,000 lines of Lean 4 code. Maintaining and extending such formalizations requires substantial expertise [26, 7].

- **Decision Procedure Extraction**: While Lean 4 supports code extraction, optimizing the extracted decision procedures for efficiency while preserving their formal guarantees presents ongoing challenges [36, 27].
- **Utility Function Representation**: Our framework currently models utility functions as `X → Real`, which requires decidable equality on the outcome space. Alternative representations using quotient types or measure theory could provide more flexibility but increase complexity [4].

4.3 Bridging Formal Verification and AI Safety Research

Our framework represents an important step toward bridging the gap between formal verification and AI safety research. Several avenues for extending this work include:

- **Learning-Based Approaches**: Extending the framework to incorporate learning-based preference elicitation while maintaining formal guarantees. This could involve verifying properties of preference learning algorithms using Lean's analysis libraries [9, 3].
- **Sequential Decision-Making**: Formalizing sequential decision processes and extending our alignment guarantees to Markov Decision Processe. Recent advances in formalizing probability theory in Lean make this direction particularly promising [26].
- **Multi-Agent Systems**: Generalizing the framework to multi-agent settings where multiple AI systems and humans interact, requiring game-theoretic extensions to our utility-based approach [32].
- **Verification of Existing AI Systems**: Developing methodologies for proving alignment properties of existing machine learning systems, potentially using techniques from certified artificial intelligence and neural network verification.
- **Formalization of Alternative Decision Theories**: Extending beyond vNM utility theory to formalize alternative decision frameworks like prospect theory [20], cumulative prospect theory [35], or logical decision theory [15].

4.4 Mechanized Metaethics and Value Learning

The `isCatastrophic` predicate in our framework raises important questions about how to specify and reason about human values. Future work could explore:

- **Formal Pluralism**: Extending the framework to accommodate multiple ethical frameworks, using dependent types to represent competing moral theories and their interrelationships [11].
- **Value Uncertainty**: Formalizing uncertainty about human values using probabilistic programming techniques in Lean, enabling reasoning about expected moral value under axiological uncertainty [25, 18].
- **Value Drift**: Modeling the evolution of human values over time and proving theorems about the conditions under which AI systems can safely adapt to changing values [34].
- **Corrigibility**: Formalizing the notion of corrigibility—the property that an AI system can be safely interrupted or modified—and proving its compatibility with our alignment framework [33, 17].

By pursuing these directions, we can expand the scope and applicability of formal methods in AI alignment, providing stronger guarantees about the behavior of increasingly complex AI systems.

4.5 Verification of Incompatible Requirements

An important capability of our formal framework is that it can detect logically incompatible requirements. For instance, we can prove that certain combinations of constraints are impossible to satisfy:

Theorem 4.1 (Alignment Impossibility)**.** If humans strictly prefer some unsafe lottery p over all safe lotteries, then there cannot exist an AI preference relation that both:

- Always respects human strict preferences (unconditional alignment), and
- Never prefers unsafe lotteries over safe ones (safety constraint)

Proof. Assume, for contradiction, that there exists such an AI preference relation aiPref. Let q be a safe lottery, which exists by assumption.

Since q is safe and $\text{isSafe}(q, \text{isCatastrophic})$, by our assumption about human preferences, we have $p \succ_{\text{human}} q$.

By the full human preference alignment property of aiPref, we must have $p \succ_{\text{AI}} q$.

However, since $\text{hasRisk}(p, \text{isCatastrophic})$ and $\text{isSafe}(q, \text{isCatastrophic})$, the safety constraint requires that $\neg(p \succsim_{\text{AI}} q)$.

This contradicts our derivation that $p \succ_{\text{AI}} q$, as strict preference $p \succ_{\text{AI}} q$ implies $p \succsim_{\text{AI}} q$. □

```
theorem alignment_impossibility
    {X : Type} [Fintype X] [Nonempty X] [DecidableEq X]
    (isCatastrophic : X → Prop) [DecidablePred isCatastrophic]
    (humanPref : Lottery X → Lottery X → Prop)
    (p : Lottery X) (h_p_unsafe : Lottery.hasRisk p isCatastrophic)
    (h_exists_safe : ∃ q : Lottery X, Lottery.isSafe q isCatastrophic)
    (h_humans_prefer_unsafe : ∀ q, Lottery.isSafe q isCatastrophic →
                                  strictPref humanPref p q) :
    ¬ ∃ aiPref : Lottery X → Lottery X → Prop,
      (∀ p' q', strictPref humanPref p' q' → strictPref aiPref p' q') ∧
      (∀ p' q', Lottery.hasRisk p' isCatastrophic → Lottery.isSafe q' isCatastrophic →
                ¬aiPref p' q') := by
  intro h_exists_aiPref
  obtain <aiPref, h_full_align, h_safety> := h_exists_aiPref
  obtain <q, h_q_safe> := h_exists_safe

  -- Human strict preference of p over q
  have h_human_pref_p_q := h_humans_prefer_unsafe q h_q_safe

  -- By full alignment, AI must also prefer p to q
  have h_ai_pref_p_q := h_full_align p q h_human_pref_p_q

  -- But this contradicts the safety constraint
  have h_not_ai_pref_p_q := h_safety p q h_p_unsafe h_q_safe
  exact h_not_ai_pref_p_q h_ai_pref_p_q.1
```

This theorem demonstrates an important trade-off: when human preferences conflict with safety requirements, a choice must be made. Our framework explicitly prioritizes safety, only respecting human preferences over safe alternatives.

Such formally verified impossibility results help guide the design of alignment frameworks by making explicit the fundamental trade-offs and limitations. By mechanizing these proofs in Lean 4, we establish their validity with the highest degree of certainty available in formal verification.

5 Conclusion

This paper has introduced a formally verified framework in Lean 4 for specifying and reasoning about AI systems that align with human values while maintaining critical safety constraints. Our contributions span both theoretical foundations and practical verification techniques:

1. We presented the first machine-checked formalization of the von Neumann-Morgenstern utility theorem in Lean 4 prover, providing a rigorous foundation for preference-based reasoning.
2. We developed the `AlignedAIPreferences` structure, which captures the essential requirements for an AI system to be aligned with human values while respecting safety constraints.

3. We demonstrated how interactive theorem proving can establish precise mathematical guarantees about alignment properties, including the `alignment_compatibility` theorem that relates AI preferences to human preferences under safety constraints.

4. We proved constructive theorems showing that aligned AI systems can be built by transforming utility functions in ways that heavily penalize catastrophic outcomes.

5. We verified impossibility results that clarify the fundamental trade-offs in AI alignment, particularly when human preferences conflict with safety requirements.

Our work demonstrates the power of dependent type theory and interactive theorem proving for addressing one of the most critical challenges in AI development: ensuring that AI systems behave in accordance with human values and intentions. The mechanized framework developed in this paper provides a foundation for the specification and verification of aligned AI systems that can be extended and refined as both AI capabilities and formal methods continue to advance.

By elevating alignment considerations from informal discussions to mechanized proofs, we contribute to a more rigorous understanding of what it means for an AI system to be aligned with human values. The formal guarantees established in this paper represent a step toward AI systems that can be trusted to act safely and in accordance with human intentions, even as they become increasingly capable and autonomous.

Future work will focus on extending these formal verification techniques to more complex decision scenarios, incorporating learning-based approaches, and addressing the challenge of specifying human values in formal systems. By continuing to bridge the gap between formal methods and AI alignment research, we aim to develop AI systems with verifiable guarantees of safety and value alignment.

Data Availability Statement

The Lean 4 formalization code for all results in this paper is available in the GitHub repository: https://github.com/jingyuanli-hk/AI-Alignment.

Acknowledgments

The author is grateful to the Lean community for their support in developing the formalization presented in this paper. I acknowledge the contributions of the Mathlib library, which provided essential mathematical components.

References

[1] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.

[2] Anand, P. (1995). Foundations of Rational Choice Under Risk. Oxford University Press.

[3] Arora, S., Du, S. S., Hu, W., Li, Z., & Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. International Conference on Machine Learning, 322-332.

[4] Boldo, S., Lelay, C., & Melquiond, G. (2015). Coquelicot: A user-friendly library of real analysis for Coq. Mathematics in Computer Science, 9(1), 41-62.

[5] Bourahla, M., &Benmohamed, M. (2005). Formal specification and verification of multi-agent systems. Electronic Notes in Theoretical Computer Science, 123 (1), 5-17.

[6] Brady, E. (2013). Idris, a general-purpose dependently typed programming language: Design and implementation. Journal of Functional Programming, 23(5), 552-593.

[7] Buzzard, K., Commelin, J., & Massot, P. (2020). Formalising perfectoid spaces. Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 299-312.

[8] Christian, B. (2020). The Alignment Problem: Machine Learning and Human Values. W. W. Norton & Company.

[9] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. Advances in Neural Information Processing Systems, 30, 4299-4307.

[10] Clouston, R. (2018). Fitch-style modal lambda calculi. International Conference on Foundations of Software Science and Computation Structures, 258-275.

[11] Conitzer, V., Sinnott-Armstrong, W., Borg, J. S., Deng, Y., & Kramer, M. (2017). Moral decision making frameworks for artificial intelligence. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1), 4831-4835.

[12] Dreyer, D., Ahmed, A., & Birkedal, L. (2011). Logical step-indexed logical relations. Logical Methods in Computer Science, 7(2), 1-37.

[13] Fisher, M., Dennis, L., & Webster, M. (2013). Verifying autonomous systems. Communications of the ACM, 56(9), 84-93.

[14] Gabriel, I. (2020). Artificial intelligence, values, and alignment. Minds and Machines, 30(3), 411-437.

[15] Garrabrant, S., Benson-Tilsen, T., Critch, A., Soares, N., & Taylor, J. (2017). Logical induction. arXiv preprint arXiv:1609.03543.

[16] Hadfield-Menell, D., Dragan, A., Abbeel, P., & Russell, S. (2016). Cooperative inverse reinforcement learning. Advances in Neural Information Processing Systems, 29, 3909-3917.

[17] Hadfield-Menell, D., Dragan, A., Abbeel, P., & Russell, S. (2017). The off-switch game. Proceedings of the International Joint Conference on Artificial Intelligence, 220-227.

[18] Halpern, J. Y. (2016). Actual Causality. MIT Press.

[19] Irving, G., Christiano, P., & Amodei, D. (2018). AI safety via debate. arXiv preprint arXiv:1805.00899.

[20] Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. Econometrica, 47(2), 263-291.

[21] Krakovna, V., Orseau, L., Ngo, R., Martic, M., & Legg, S. (2020). Avoiding side effects by considering future tasks. Advances in Neural Information Processing Systems, 33, 14505-14515.

[22] Kumar, R., Uesato, J., Ngo, R., Everitt, T., Krakovna, V., & Legg, S. (2021). Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. arXiv preprint arXiv:2006.01237.

[23] de Moura, L.,& Ullrich, S. (2021). The Lean 4 Theorem Prover and Programming Language. Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings, 625-635.

[24] Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., & Legg, S. (2018). Scalable agent alignment via reward modeling: A research direction. arXiv preprint arXiv:1811.07871.

[25] MacAskill, W., Bykvist, K., & Ord, T. (2020). Moral Uncertainty. Oxford University Press.

[26] The Mathlib Community (2020). The Lean Mathematical Library. Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, 367–381.

[27] Mullen, E., Pernsteiner, S., Wilcox, J. R., Tatlock, Z., & Grossman, D. (2018). Oeuf: Minimizing the Coq extraction TCB. Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, 172-185.

[28] Ringer, T., Palmskog, K., Sergey, I., Gligoric, M., & Tatlock, Z. (2019). QED at Large: A survey of engineering of formally verified software. Foundations and Trends in Programming Languages, 5(2-3), 102-281.

[29] Russell, S. (2019). Human Compatible: Artificial Intelligence and the Problem of Control. Viking.

[30] Seshia, S. A., Sadigh, D., & Sastry, S. S. (2022). Towards verified artificial intelligence. Communications of the ACM, 65 (7), 46-55.

[31] Shah, R., Krasheninnikov, D., Alexander, J., Abbeel, P., & Dragan, A. (2019). Preferences implicit in the state of the world. International Conference on Learning Representations.

[32] Shoham, Y., & Leyton-Brown, K. (2008). Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press.

[33] Soares, N., Fallenstein, B., Armstrong, S., & Yudkowsky, E. (2015). Corrigibility. AAAI Workshop on AI and Ethics, 74-82.

[34] Tarleton, N. (2010). Coherent extrapolated volition: A meta-level approach to machine ethics. Machine Intelligence Research Institute.

[35] Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and Uncertainty, 5(4), 297-323.

[36] Ullrich, S., & de Moura, L. (2021). Counting immutable beans: Reference counting optimized for purely functional programming. In Proceedings of the 31st Symposium on Implementation and Application of Functional Languages, 143-154.

[37] von Neumann, J., & Morgenstern, O. (1944). Theory of Games and Economic Behavior. Princeton University Press.

A Appendix: Mechanized Proofs of the vNM Utility Theorem

A.1 Fundamental Properties of Preference Relations

The axioms imposed on our preference relation $\succsim$ lead to several important properties that are essential for our subsequent development of utility theory.

Lemma A.1 (Properties of Preference Relations)**.** Let $p, q, r, q_1, q_2 \in \Delta(\mathcal{X})$.

1. $\succsim$ is transitive: If $p \succsim q$ and $q \succsim r$, then $p \succsim r$. (This is Axiom A1b) (Lean: `instance : IsTrans (Lottery X) PrefRel.pref := trans := PrefRel.transitive`)
2. $\succsim$ is total: For any p, q, $p \succsim q$ or $q \succsim p$. (This is Axiom A1a, Completeness) (Lean: `instance : IsTotal (Lottery X) PrefRel.pref := ⟨PrefRel.complete⟩`)
3. Strict preference $\succ$ is transitive: If $p \succ q$ and $q \succ r$, then $p \succ r$.
4. Preference $\succsim$ is reflexive: $p \succsim p$.
5. Strict preference $\succ$ is irreflexive: $\neg(p \succ p)$.
6. Indifference $\sim$ is transitive: If $p \sim q_1$ and $q_1 \sim q_2$, then $p \sim q_2$.

Proof. **3. Transitivity of Strict Preference ($\succ$):** Assume $p \succ q$ and $q \succ r$. We want to show $p \succ r$. By definition of strict preference (Definition 2.5): (H1) $p \succ q \implies p \succsim q$ and $\neg(q \succsim p)$. (H2) $q \succ r \implies q \succsim r$ and $\neg(r \succsim q)$.

We need to show two things for $p \succ r$: (a) $p \succsim r$: From (H1), we have $p \succsim q$. From (H2), we have $q \succsim r$. By transitivity of $\succsim$ (Axiom A1b), $p \succsim q \wedge q \succsim r \implies p \succsim r$. (Lean: `exact PrefRel.transitive p q r` h_1`.1` h_2`.1`)

(b) $\neg(r \succsim p)$: Assume for contradiction that $r \succsim p$. We have $r \succsim p$ (our assumption for contradiction) and from (H1), $p \succsim q$. By transitivity of $\succsim$ (Axiom A1b), $r \succsim p \wedge p \succsim q \implies r \succsim q$. However, from (H2), we have $\neg(r \succsim q)$. This is a contradiction. Thus, our assumption $r \succsim p$ must be false. So, $\neg(r \succsim p)$. (Lean: `intro hrp; exact` h_2`.2 (PrefRel.transitive r p q hrp` h_1`.1)`. Here `hrp` is $r \succsim p$. `PrefRel.transitive r p q hrp` h_1`.1` proves $r \succsim q$. h_2`.2` is $\neg(r \succsim q)$. So this derives a contradiction from `hrp`.)

Since (a) and (b) hold, $p \succ r$. (Lean: `instance :IsTrans (Lottery X) strictPref := ⟨fun p q r` h_1 h_2 `⇒ strictPref_trans` h_1 h_2`⟩`)

4. Reflexivity of Preference ($\succsim$): For any $p \in \Delta(\mathcal{X})$, by completeness (Axiom A1a), we have $p \succsim p \vee p \succsim p$. In either case, $p \succsim p$ holds. (Lean: `lemma PrefRel.refl (p : Lottery X) : p ⪰ p := (PrefRel.complete p p).elim id id`)

5. Irreflexivity of Strict Preference ($\succ$): We want to show $\neg(p \succ p)$. Assume for contradiction that $p \succ p$. By definition of strict preference, $p \succ p \implies p \succsim p \wedge \neg(p \succsim p)$. This is a contradiction of the form $A \wedge \neg A$. Thus, $\neg(p \succ p)$. (Lean: `instance : IsIrrefl (Lottery X) strictPref := ⟨fun p h => h.2 (PrefRel.refl p)⟩`. Here `h` is $p \succ p$. `h.2` is $\neg(p \succsim p)$. `PrefRel.refl p` is $p \succsim p$. These form a contradiction.)

6. Transitivity of Indifference ($\sim$): Assume $p \sim q_1$ and $q_1 \sim q_2$. We want to show $p \sim q_2$. By definition of indifference (Definition 2.5): (H1) $p \sim q_1 \implies p \succsim q_1$ and $q_1 \succsim p$. (H2) $q_1 \sim q_2 \implies q_1 \succsim q_2$ and $q_2 \succsim q_1$.

We need to show two things for $p \sim q_2$: (a) $p \succsim q_2$: From (H1), $p \succsim q_1$. From (H2), $q_1 \succsim q_2$. By transitivity of $\succsim$ (Axiom A1b), $p \succsim q_1 \wedge q_1 \succsim q_2 \implies p \succsim q_2$. (Lean: `PrefRel.transitive p q1 q2 h1.1 h2.1`)

(b) $q_2 \succsim p$: From (H2), $q_2 \succsim q_1$. From (H1), $q_1 \succsim p$. By transitivity of $\succsim$ (Axiom A1b), $q_2 \succsim q_1 \wedge q_1 \succsim p \implies q_2 \succsim p$. (Lean: `PrefRel.transitive q2 q1 p h2.2 h1.2`)

Since (a) and (b) hold, $p \sim q_2$. □

A.2 Key Claims: Building Blocks for the vNM Theorem

The following claims establish crucial properties about preferences over lotteries. These properties serve as building blocks for proving the vNM theorem, which will show how the axioms A1-A3 lead to an expected utility representation.

Claim A.1 (Strict Preference and Mixtures)**.** If $p, q \in \Delta(\mathcal{X})$ such that $p \succ q$, and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$, then

$$p \succ \text{mix}(p, q, \alpha) \quad \text{and} \quad \text{mix}(p, q, \alpha) \succ q$$

Let $L_{pq}^{\alpha} = \text{mix}(p, q, \alpha)$. The claim is $p \succ L_{pq}^{\alpha}$ and $L_{pq}^{\alpha} \succ q$.

Proof. Given: $p \succ q$, and $0 < \alpha < 1$. This means $p \succsim q \land \neg(q \succsim p)$. Also, $0 < \alpha \implies \alpha \le 1$ (true, as $\alpha < 1$) and $0 < \alpha$ is given. So $\alpha \in (0, 1]$. Similarly, $0 < \alpha < 1 \implies 0 < 1 - \alpha < 1$. Let $\beta = 1 - \alpha$. So $\beta \in (0, 1]$.

Part 1: Prove $p \succ L_{pq}^{\alpha}$ (Lean: `constructor` for the first part of the main conjunction) We need to show $p \succsim L_{pq}^{\alpha}$ and $\neg(L_{pq}^{\alpha} \succsim p)$. (Lean: `unfold strictPref; constructor` for the two parts of $p \succ L_{pq}^{\alpha}$)

Consider the Independence Axiom (A3a) with lotteries p, q, p (as p, q, r) and scalar $\beta = 1 - \alpha$. Since $p \succ q$ and $\beta \in (0, 1]$, Axiom A3a implies: $\text{mix}(p, p, \beta) \succ \text{mix}(q, p, \beta)$. (Lean: `have h_use_indep := PrefRel.independence p q p (1 - α) h_cond h`)

Let's analyze the terms:

- $\text{mix}(p, p, \beta)(x) = \beta p(x) + (1-\beta)p(x) = \beta p(x) + \alpha p(x) = (\beta+\alpha)p(x) = (1-\alpha+\alpha)p(x) = 1 \cdot p(x) = p(x)$. So, $\text{mix}(p, p, \beta) = p$. (Lean: `have h_mix_p_p : mix p p (1-α) ... = p := by ... calc ((1-α) + α) * p.val x ... = p.val x`)
- $\text{mix}(q, p, \beta)(x) = \beta q(x) + (1-\beta)p(x) = (1-\alpha)q(x) + \alpha p(x)$. This is, by definition, $\text{mix}(p, q, \alpha)(x) = L_{pq}^{\alpha}(x)$. So, $\text{mix}(q, p, \beta) = L_{pq}^{\alpha}$. (Lean: `have h_mix_q_p_comm : mix q p (1-α) ... = mix p q α ... := by ... ring`)

Substituting these into the result from Axiom A3a: $p \succ L_{pq}^{\alpha}$. (Lean: `rw [h_mix_p_p, h_mix_q_p_comm] at h_use_indep; exact h_use_indep.1` for $p \succsim L_{pq}^{\alpha}$, and `exact h_use_indep.2` for $\neg(L_{pq}^{\alpha} \succsim p)$). This directly gives both $p \succsim L\_pq^{\alpha}$ (from `h_use_indep.1`) and $\neg(L\_pq^{\alpha} \succsim p)$ (from `h_use_indep.2`).

Part 2: Prove $L_{pq}^{\alpha} \succ q$ (Lean: `constructor` for the second part of the main conjunction, which is $L_{pq}^{\alpha} \succ q$. This also unfolds to two subgoals.) We need to show $L_{pq}^{\alpha} \succsim q$ and $\neg(q \succsim L_{pq}^{\alpha})$.

Consider the Independence Axiom (A3a) with lotteries p, q, q (as p, q, r) and scalar α. Since $p \succ q$ and $\alpha \in (0, 1]$, Axiom A3a implies: $\text{mix}(p, q, \alpha) \succ \text{mix}(q, q, \alpha)$. (Lean: `have h_use_indep := PrefRel.independence p q q α ⟨hα, le_of_lt hα₂⟩ h`)

Let's analyze the terms:

- $\text{mix}(p, q, \alpha) = L_{pq}^{\alpha}$.
- $\text{mix}(q, q, \alpha)(x) = \alpha q(x) + (1-\alpha)q(x) = (\alpha + 1 - \alpha)q(x) = 1 \cdot q(x) = q(x)$. So, $\text{mix}(q, q, \alpha) = q$. (Lean: `have h_mix_q_q : mix q q α ... = q := by ... ring`)

Substituting these into the result from Axiom A3a: $L_{pq}^{\alpha} \succ q$. (Lean: `rw [h_mix_q_q] at h_use_indep; exact ⟨h_use_indep.1, h_use_indep.2⟩`). This gives $L_{pq}^{\alpha} \succsim q$ (from `h_use_indep.1`) and $\neg(q \succsim L\_pq^{\alpha})$ (from `h_use_indep.2`). Both parts of the claim are proven. ☐

Claim A.2 (Monotonicity of Mixtures under Strict Preference)**.** If $p, q \in \Delta(\mathcal{X})$ such that $p \succ q$, and $\alpha, \beta \in \mathbb{R}$ with $0 \le \alpha < \beta \le 1$, then

$$\text{mix}(p, q, \beta) \succ \text{mix}(p, q, \alpha)$$

Let $L_{pq}^{\beta} = \text{mix}(p, q, \beta)$ and $L_{pq}^{\alpha} = \text{mix}(p, q, \alpha)$. The claim is $L_{pq}^{\beta} \succ L_{pq}^{\alpha}$.

Proof. Given: $p \succ q$, $0 \le \alpha$, $\alpha < \beta$, $\beta \le 1$. (Lean: `hβ_nonneg : 0 ≤ β := le_trans hα (le_of_lt hαβ)‘, ‘hα_le_one : α ≤ 1 := le_trans (le_of_lt hαβ) hβ` establish bounds for the mix definitions.)

We proceed by cases on α and β. (Lean: `by_cases hα₀ : α = 0`)

Case 1: $\alpha = 0$. (Lean: `hα₀ : α = 0`) Then $L_{pq}^{\alpha} = \text{mix}(p, q, 0) = 0 \cdot p + (1-0) \cdot q = q$. The claim becomes $L_{pq}^{\beta} \succ q$.

Subcase 1.1: $\beta = 1$. (Lean: `by_cases hβ₁ : β = 1` inside `hα₀`) Then $L_{pq}^{\beta} = \text{mix}(p, q, 1) = 1 \cdot p + (1-1) \cdot q = p$. The claim becomes $p \succ q$, which is given by hypothesis h. (Lean: `subst hα₀; subst hβ₁; ... have hp := ...; have hq := ...; unfold strictPref; constructor; rw[hp,hq]; exact h.1; rw[hp,hq]; exact h.2`)

Subcase 1.2: $\beta < 1$. (Lean: `hβ₁` is the negation, $\beta \ne 1$) So we have $\alpha = 0$ and $0 < \beta < 1$ (since $\alpha < \beta \implies 0 < \beta$, and $\beta \le 1, \beta \ne 1 \implies \beta < 1$). The claim is $\text{mix}(p, q, \beta) \succ q$. This follows directly from the second part of Claim A.1, since $p \succ q$ and $0 < \beta < 1$. (Lean: `subst hα₀; ... have hq := ...; have hβ_pos : 0 < β := by linarith; have hβ_lt_one : β < 1 := lt_of_le_of_ne hβ hβ₁; have h_claim := claim_i h β hβ_pos hβ_lt_one; ... rw [hq]; exact h_claim.2`)

Case 2: $\alpha > 0$**.** (Lean: `hα₀` is the negation, $\alpha \neq 0$, so $0 < \alpha$ since $0 \leq \alpha$ is given)

Subcase 2.1: $\beta = 1$**.** (Lean: `by_cases hβ₁ : β = 1` inside `¬hα₀`) Then $L_{pq}^{\beta} = \text{mix}(p, q, 1) = p$. We have $0 < \alpha < \beta = 1$, so $0 < \alpha < 1$. The claim becomes $p \succ \text{mix}(p, q, \alpha)$. This follows directly from the first part of Claim A.1, since $p \succ q$ and $0 < \alpha < 1$. (Lean: `subst hβ₁; have hp := ...; have hα_pos : 0 < α := lt_of_le_of_ne hα (Ne.symm hα₀); have hα_lt1 : α < 1 := by linarith; have h_claim := claim_i h α hα_pos hα_lt1; simp only [hp]; exact h_claim.1`)

Subcase 2.2: $\beta < 1$**.** (Lean: `hβ₁` is the negation, $\beta \neq 1$) So we have $0 < \alpha < \beta < 1$. (Lean: `have hα₀ : 0 < α := ...; have hβ₁ : β < 1 := ...`) From Claim A.1, since $p \succ q$ and $0 < \beta < 1$: (H1) $p \succ L_{pq}^{\beta}$ (H2) $L_{pq}^{\beta} \succ q$ (Lean: `have h₁ := claim_i h β (lt_trans hα₀ hαβ) hβ₁`) (where `h₁.1` is $p \succ L\_pq^{\beta}$ and `h₁.2` is $L\_pq^{\beta} \succ q$)

Let $\gamma = \frac{\alpha}{\beta}$. Since $0 < \alpha < \beta$, we have $0 < \gamma < 1$. (Lean: '`have hγ : 0 < α/β ∧ α/β < 1 := by ...`) We want to show that L_{pq}^{α} can be expressed as a convex combination of L_{pq}^{β} and q. Consider $\text{mix}(L_{pq}^{\beta}, q, \gamma) = \gamma L_{pq}^{\beta} + (1-\gamma)q$. For any $x \in \mathcal{X}$:

$$\begin{aligned}
(\text{mix}(L_{pq}^{\beta}, q, \gamma))(x) &= \gamma(\beta p(x) + (1-\beta)q(x)) + (1-\gamma)q(x) \\
&= \frac{\alpha}{\beta}(\beta p(x) + (1-\beta)q(x)) + (1 - \frac{\alpha}{\beta})q(x) \\
&= \alpha p(x) + \frac{\alpha(1-\beta)}{\beta}q(x) + q(x) - \frac{\alpha}{\beta}q(x) \\
&= \alpha p(x) + \left(\frac{\alpha - \alpha\beta + \beta - \alpha}{\beta}\right) q(x) \\
&= \alpha p(x) + \left(\frac{\beta - \alpha\beta}{\beta}\right) q(x) \\
&= \alpha p(x) + (1-\alpha)q(x) = L_{pq}^{\alpha}(x)
\end{aligned}$$

So, $L_{pq}^{\alpha} = \text{mix}(L_{pq}^{\beta}, q, \gamma)$. (Lean: `let γ := α/β; have h_mix_αβ : mix p q α ... = mix (mix p q β ...) q γ ... := by apply Subtype.eq; ext x; simp [Lottery.mix]; ... calc ... end`) The `calc` block in Lean performs this algebraic verification.

Now we use (H2): $L_{pq}^{\beta} \succ q$. Since $0 < \gamma < 1$, by the first part of Claim A.1 (with L_{pq}^{β} playing the role of p, q playing the role of q, and γ playing the role of α): $L_{pq}^{\beta} \succ \text{mix}(L_{pq}^{\beta}, q, \gamma)$. Substituting $L_{pq}^{\alpha} = \text{mix}(L_{pq}^{\beta}, q, \gamma)$, we get $L_{pq}^{\beta} \succ L_{pq}^{\alpha}$. (Lean: `have h_claim := claim_i h₁.2 γ hγ.1 hγ.2; ... rw [h_mix_αβ]; exact h_claim.1`) This covers all cases. □

Lemma A.2 (Helper for Claim A.3, Part 1)**.** If $p, q \in \Delta(\mathcal{X})$ such that $p \sim q$, and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$, then

$$p \sim \text{mix}(p, q, \alpha)$$

Proof. Let $L_{pq}^{\alpha} = \text{mix}(p, q, \alpha)$. Given $p \sim q$ and $0 < \alpha < 1$. Let $\beta = 1 - \alpha$. Since $0 < \alpha < 1$, we have $0 < \beta < 1$. Thus $\beta \in (0, 1]$. (Lean: '`have h_1_minus_α_pos : 0 < 1 - α := by linarith; ... h_cond_1_minus_α`)

We use Axiom A3b (Independence for Indifference) with lotteries p, q, p (as P', Q', R') and scalar $\beta = 1 - \alpha$. Since $p \sim q$ and $\beta \in (0, 1]$, Axiom A3b implies: $\text{mix}(p, p, \beta) \sim \text{mix}(q, p, \beta)$. (Lean: `have h_indep_res := PrefRel.indep_indiff p q p (1-α) h_cond_1_minus_α h`)

Let's analyze the terms:

- $\text{mix}(p, p, \beta) = p$, as shown in the proof of Claim A.1. (Lean: `have h_mix_p_p_id : mix p p (1-α) ... = p := by ... ring`)
- $\text{mix}(q, p, \beta)(x) = \beta q(x) + (1-\beta)p(x) = (1-\alpha)q(x) + \alpha p(x) = L_{pq}^{\alpha}(x)$. So $\text{mix}(q, p, \beta) = L_{pq}^{\alpha}$. (Lean: `have h_mix_q_p_1_minus_α_eq_mix_p_q_α : mix q p (1-α) ... = mix p q α ... := by ... ring`)

Substituting these into the result from Axiom A3b: $p \sim L_{pq}^{\alpha}$. (Lean: `rw [h_mix_p_p_id] at h_indep_res; rw [h_mix_q_p_1_minus_α_eq_mix_p_q_α] at h_indep_res; exact h_indep_res`) ☐

Lemma A.3 (Helper for Claim A.3, Part 2)**.** If $p, q \in \Delta(\mathcal{X})$ such that $p \sim q$, and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$, then

$$\text{mix}(p, q, \alpha) \sim q$$

Proof. Let $L_{pq}^{\alpha} = \text{mix}(p, q, \alpha)$. Given $p \sim q$ and $0 < \alpha < 1$. Thus $\alpha \in (0, 1]$. (Lean: `have h_α_cond : 0 < α ∧ α ≤ 1 := ⟨hα₁, le_of_lt hα₂⟩`)

We use Axiom A3b (Independence for Indifference) with lotteries p, q, q (as P', Q', R') and scalar α. Since $p \sim q$ and $\alpha \in (0, 1]$, Axiom A3b implies: $\text{mix}(p, q, \alpha) \sim \text{mix}(q, q, \alpha)$. (Lean: `have h_indep_res := PrefRel.indep_indiff p q q α h_α_cond h`)

Let's analyze the terms:

- $\text{mix}(p, q, \alpha) = L_{pq}^{\alpha}$.
- $\text{mix}(q, q, \alpha) = q$, as shown in the proof of Claim A.1. (Lean: `have h_mix_q_q_id : mix q q α ... = q := by ... ring`)

Substituting these into the result from Axiom A3b: $L_{pq}^{\alpha} \sim q$. (Lean: `rw [h_mix_q_q_id] at h_indep_res; exact h_indep_res`) ☐

Claim A.3 (Indifference and Mixtures)**.** If $p, q \in \Delta(\mathcal{X})$ such that $p \sim q$, and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$, then

$$p \sim \text{mix}(p, q, \alpha) \quad \text{and} \quad \text{mix}(p, q, \alpha) \sim q$$

Proof. This claim is a direct conjunction of Lemma A.2 and Lemma A.3. (Lean: `apply And.intro; exact claim_iii_part1 α h hα₁ hα₂; exact claim_iii_part2 α h hα₁ hα₂`) The Lean code also has a `theorem claim_iii_impl` which is identical to `claim_iii`. ☐

Claim A.4 (Independence and Indifference)**.** If $p, q, r \in \Delta(\mathcal{X})$ such that $p \sim q$, and $\alpha \in \mathbb{R}$ with $0 < \alpha < 1$, then

$$\text{mix}(p, r, \alpha) \sim \text{mix}(q, r, \alpha)$$

Proof. Given $p \sim q$ and $0 < \alpha < 1$. Thus $\alpha \in (0, 1]$. (Lean: `have h_α_cond : 0 < α ∧ α ≤ 1 := ⟨hα₁, le_of_lt hα₂⟩`) This is a direct application of Axiom A3b (Independence for Indifference) with lotteries p, q, r and scalar α. (Lean: `exact PrefRel.indep_indiff p q r α h_α_cond h`) ☐

Claim A.5 (Existence of Indifference Mixtures)**.** If $p, q, r \in \Delta(\mathcal{X})$ such that $p \succsim q$, $q \succsim r$, and $p \succ r$, then there exists a unique $\alpha^* \in [0, 1]$ such that $\text{mix}(p, r, \alpha^*) \sim q$.

Proof. Let $h_1 : p \succsim q$, $h_2 : q \succsim r$, and $h_3 : p \succ r$. The condition $h_3 : p \succ r$ means $p \succsim r$ and $\neg(r \succsim p)$. The $p \succsim r$ part is also implied by h_1, h_2 and transitivity of $\succsim$. The crucial part of h_3 is $\neg(r \succsim p)$.

Let $L_{pr}^{\alpha} = \text{mix}(p, r, \alpha)$. We are looking for a unique $\alpha^* \in [0, 1]$ such that $L_{pr}^{\alpha^*} \sim q$.

Part 1: Existence of α^*

Define the set $S = \{\alpha \in [0, 1] \mid L_{pr}^{\alpha} \succ q\}$. (Lean: `let S := α_val : Real | ∃ (h_α_bounds : 0 ≤ α_val ∧α_val ≤ 1), (mix p r α_val ...) ≻ q`)

Step 1.1: S is non-empty. The given conditions are $p \succsim q$, $q \succsim r$, and $\neg(r \succsim p)$ (from h_3). These are precisely the conditions for Axiom A2 (Continuity). By Axiom A2, there exist $\alpha_c, \beta_c \in \mathbb{R}$ with $0 < \alpha_c < 1$ and $0 < \beta_c < 1$ such that $L_{pr}^{\alpha_c} \succ q$ and $q \succ L_{pr}^{\beta_c}$. Since $0 < \alpha_c < 1$, $\alpha_c \in [0, 1]$. Thus, $\alpha_c \in S$. Therefore, S is non-empty. (Lean: `h_continuity_axiom_applies := PrefRel.continuity p q r h₁ h₂ h₃.2; let ⟨α_c, _, h_conj_c, h_mix_α_c_pref_q, h_not_q_pref_mix_α_c, _, _⟩ := h_continuity_axiom_applies; ... h_α_c_in_S; hS_nonempty`)

Step 1.2: S is bounded below by 0. By definition of S, any $\alpha_s \in S$ satisfies $0 \leq \alpha_s$. So 0 is a lower bound for S. (Lean: `hS_bddBelow : BddBelow S := by use 0; ...`)

Step 1.3: Define $\alpha^* = \inf S$. Since S is non-empty (Step 1.1) and bounded below (Step 1.2), its infimum α^* exists in $\mathbb{R}$. For any $\alpha_s \in S$, $0 \leq \alpha_s \leq 1$. Since 0 is a lower bound for S, $0 \leq \alpha^*$. Since $\alpha_c \in S$ and

$\alpha_c < 1$, and α^* is the infimum, $\alpha^* \le \alpha_c$. So, $\alpha^* \le \alpha_c < 1$, which implies $\alpha^* < 1$, and therefore $\alpha^* \le 1$. Thus, $\alpha^* \in [0,1]$. (Lean: `let α_star := sInf S; h_α_star_nonneg; h_α_star_lt_1_proof; h_α_star_le_one`)

Let $L_{\alpha^*} = L_{pr}^{\alpha^*} = \text{mix}(p, r, \alpha^*)$.

Step 1.4: Show $L_{\alpha^*} \sim q$. This is typically proven by showing that neither $L_{\alpha^*} \succ q$ nor $q \succ L_{\alpha^*}$ can hold. (Lean: `have h_α_star_indiff_q : Lαs ∼ q := by ...`)

Sub-step 1.4.1: Show $\neg(L_{\alpha^*} \succ q)$. (Lean: `have not_Lαs_succ_q : ¬ (Lαs ≻ q) := by intro h_Lαs_succ_q ...`) Assume for contradiction that $L_{\alpha^*} \succ q$. (This is `h_Lαs_succ_q`) If $\alpha^* = 0$: Then $L_{\alpha^*} = L_{pr}^0 = r$. So $r \succ q$. But $q \succsim r$ is given (h_2). So $r \succ q \implies r \succsim q \wedge \neg(q \succsim r)$. This contradicts $q \succsim r$. (Lean: `by_contra h_α_star_not_pos ... h_Lαs_eq_r ... exact h_Lαs_succ_q.2` h_2) So, we must have $\alpha^* > 0$. (Lean: `have h_α_star_pos : 0 < α_star := by ...`) Since $p \succ r$ (h_3) and $0 < \alpha^* < 1$ (as $\alpha^* \le \alpha_c < 1$), by Claim A.1 (second part, applied to $p \succ r$ with α^* as mixing coeff for p): $\text{mix}(p, r, \alpha^*) \succ r$, i.e., $L_{\alpha^*} \succ r$. (Lean: `have h_Lαs_succ_r : Lαs ≻ r := by exact (claim_i` h_3 `α_star h_α_star_pos h_α_star_lt_1_proof).2`) So we have $L_{\alpha^*} \succ q$ (assumption 'h_Lαs_succ_q') and $q \succsim r$ (h_2) and $L_{\alpha^*} \succ r$ (implies $\neg(r \succsim L_{\alpha^*})$). These are the conditions for Axiom A2 (Continuity) applied to L_{α^*}, q, r. (Lean: `h_continuity_args_met : PrefRel.pref Lαs q ∧ PrefRel.pref q r ∧¬(PrefRel.pref r Lαs)`) By Axiom A2, there exists $\gamma_c \in (0,1)$ such that $\text{mix}(L_{\alpha^*}, r, \gamma_c) \succ q$. (Lean: `let ⟨γ_c, _, h_conj_γ_c, h_mix_Lαs_r_γ_c_pref_q, h_not_q_pref_mix_Lαs_r_γ_c, _, _⟩ := PrefRel.continuity Lαs q r ...`) Let $\alpha_{new} = \alpha^* - \gamma_c\alpha^* + \gamma_c \cdot 0 = \gamma_c\alpha^* + (1-\gamma_c)\alpha^* = \alpha^*$. This is not right. The mixture is $\text{mix}(L_{\alpha^*}, r, \gamma_c)(x) = \gamma_c L_{\alpha^*}(x) + (1-\gamma_c)r(x)$. $L_{\alpha^*}(x) = \alpha^* p(x) + (1-\alpha^*)r(x)$. So, $\text{mix}(L_{\alpha^*}, r, \gamma_c)(x) = \gamma_c(\alpha^* p(x) + (1-\alpha^*)r(x)) + (1-\gamma_c)r(x) = (\gamma_c\alpha^*)p(x) + (\gamma_c(1-\alpha^*) + (1-\gamma_c))r(x) = (\gamma_c\alpha^*)p(x) + (\gamma_c - \gamma_c\alpha^* + 1 - \gamma_c)r(x) = (\gamma_c\alpha^*)p(x) + (1-\gamma_c\alpha^*)r(x)$. Let $\alpha'_{new} = \gamma_c\alpha^*$. Since $0 < \gamma_c < 1$ and $\alpha^* > 0$, we have $0 < \alpha'_{new} < \alpha^*$. So $\text{mix}(L_{\alpha^*}, r, \gamma_c) = L_{pr}^{\alpha'_{new}}$. (Lean: `let α_new := γ_c * α_star; ... h_L_α_new_eq : mix Lαs r γ_c ... = mix p r α_new ... by ... ring`) Thus $L_{pr}^{\alpha'_{new}} \succ q$. Since $0 < \alpha'_{new} < \alpha^* \le 1$, $\alpha'_{new} \in [0,1]$. So $\alpha'_{new} \in S$. But $\alpha'_{new} < \alpha^*$, which contradicts $\alpha^* = \inf S$ (as α^* is a lower bound for S). (Lean: `h_α_new_in_S; exact not_lt_of_le (csInf_le hS_bddBelow h_α_new_in_S) h_α_new_lt_α_star`) Therefore, the assumption $L_{\alpha^*} \succ q$ is false. So $\neg(L_{\alpha^*} \succ q)$.

Sub-step 1.4.2: Show $\neg(q \succ L_{\alpha^*})$. (Lean: `have not_q_succ_Lαs : ¬(q ≻ Lαs) := by intro h_q_succ_Lαs ...`) Assume for contradiction that $q \succ L_{\alpha^*}$. (This is `h_q_succ_Lαs`) We have $p \succsim q$ (h_1) and $q \succ L_{\alpha^*}$. If $p \sim q$, then $p \succ L_{\alpha^*}$. If $p \succ q$, then by transitivity $p \succ L_{\alpha^*}$. So, $p \succ L_{\alpha^*}$. (This means $p \succsim L_{\alpha^*} \wedge \neg(L_{\alpha^*} \succsim p)$). (Lean: `have h_p_succ_Lαs : p ≻ Lαs := by ...`) We have $p \succsim q$ (h_1), $q \succsim L_{\alpha^*}$ (from $q \succ L_{\alpha^*}$), and $\neg(L_{\alpha^*} \succsim p)$ (from $p \succ L_{\alpha^*}$). These are the conditions for Axiom A2 (Continuity) applied to p, q, L_{α^*}. (Lean: `h_continuity_args_met : PrefRel.pref p q ∧ PrefRel.pref q Lαs ∧¬(PrefRel.pref Lαs p)`) By Axiom A2, there exists $\beta'_c \in (0,1)$ (Lean uses β_c) such that $q \succ \text{mix}(p, L_{\alpha^*}, \beta'_c)$. (Lean: `let <_, β_c, h_conj_β_c, _, _, h_q_pref_mix_p_Lαs_β_c, h_not_mix_p_Lαs_β_c_pref_q> := PrefRel.continuity p q Lαs ...`) Let $\alpha''_{new} = \beta'_c \cdot 1 + (1-\beta'_c)\alpha^* = \beta'_c + \alpha^* - \beta'_c\alpha^* = \alpha^* + \beta'_c(1-\alpha^*)$. (The mix is $\beta'_c p + (1-\beta'_c)L_{\alpha^*} = \beta'_c p + (1-\beta'_c)(\alpha^* p + (1-\alpha^*)r) = (\beta'_c + (1-\beta'_c)\alpha^*)p + (1-\beta'_c)(1-\alpha^*)r$. So the coefficient for p is α''_{new}.) (Lean: `let α_new := α_star + β_c * (1 - α_star); ... h_L_α_new_eq : mix p Lαs β_c ... = mix p r α_new ... by ... ring`) Since $0 < \beta'_c < 1$ and $0 \le \alpha^* < 1$ (so $1-\alpha^* > 0$), we have $\alpha^* < \alpha''_{new}$. Also, $\alpha''_{new} = \alpha^*(1-\beta'_c) + \beta'_c < 1(1-\beta'_c) + \beta'_c = 1 - \beta'_c + \beta'_c = 1$. So $\alpha^* < \alpha''_{new} < 1$. We have $L_{pr}^{\alpha''_{new}} = \text{mix}(p, L_{\alpha^*}, \beta'_c)$. So $q \succ L_{pr}^{\alpha''_{new}}$.

Now, if $\alpha''_{new} \in S$, then $L_{pr}^{\alpha''_{new}} \succ q$. This contradicts $q \succ L_{pr}^{\alpha''_{new}}$. So $\alpha''_{new} \notin S$. (Lean: `h_α_new_not_in_S`) Also, $\alpha^* \notin S$ because $\neg(L_{\alpha^*} \succ q)$ was shown in Sub-step 1.4.1. (Lean: `h_α_star_not_in_S`)

The argument in Lean `exists_s_in_S_between` shows that if $q \succ L_{\alpha^*}$, then there must be some $s_{val} \in S$ such that $\alpha^* < s_{val} < \alpha''_{new}$. This relies on properties of `sInf`: for any $\epsilon > 0$, there exists $s \in S$ such that $s < \alpha^* + \epsilon$. Let $y = (\alpha^* + \alpha''_{new})/2$. Then $\alpha^* < y < \alpha''_{new}$. The Lean proof shows $\exists s \in S$ with $s < y$. This s also satisfies $s > \alpha^*$ (unless $s = \alpha^*$, but $\alpha^* \notin S$). So we have $s \in S$ (so $L_{pr}^s \succ q$) and $\alpha^* < s < \alpha''_{new}$. By Claim A.2, since $p \succ r$ and $s < \alpha''_{new}$, we have $L_{pr}^{\alpha''_{new}} \succ L_{pr}^s$. We have $L_{pr}^s \succ q$ (since $s \in S$) and $q \succ L_{pr}^{\alpha''_{new}}$ (by construction of α''_{new}). By transitivity of $\succ$, $L_{pr}^s \succ L_{pr}^{\alpha''_{new}}$. This is a contradiction: $L_{pr}^{\alpha''_{new}} \succ L_{pr}^s$ and $L_{pr}^s \succ L_{pr}^{\alpha''_{new}}$ cannot both hold. (Lean: `h_Lα_new_succ_Ls ... h_Ls_pref_Lα_new ... exact h_Lα_new_succ_Ls.2 h_Ls_pref_Lα_new`). The contradiction implies the initial assumption $q \succ L_{\alpha^*}$ is false. So $\neg(q \succ L_{\alpha^*})$.

Sub-step 1.4.3: Conclude indifference. Since $\neg(L_{\alpha^*} \succ q)$ and $\neg(q \succ L_{\alpha^*})$, we need to use completeness.

$\neg(L_{\alpha^*} \succ q) \equiv \neg(L_{\alpha^*} \succsim q \wedge \neg(q \succsim L_{\alpha^*})) \equiv L_{\alpha^*} \not\succsim q \vee q \succsim L_{\alpha^*}$. $\neg(q \succ L_{\alpha^*}) \equiv \neg(q \succsim L_{\alpha^*} \wedge \neg(L_{\alpha^*} \succsim q)) \equiv q \not\succsim L_{\alpha^*} \vee L_{\alpha^*} \succsim q$. By Axiom A1 (Completeness), either $L_{\alpha^*} \succsim q$ or $q \succsim L_{\alpha^*}$. If $L_{\alpha^*} \succsim q$: To avoid $L_{\alpha^*} \succ q$, we must have $q \succsim L_{\alpha^*}$. So $L_{\alpha^*} \sim q$. If $q \succsim L_{\alpha^*}$: To avoid $q \succ L_{\alpha^*}$, we must have $L_{\alpha^*} \succsim q$. So $L_{\alpha^*} \sim q$. In both scenarios, $L_{\alpha^*} \sim q$. (Lean: `unfold indiff; constructor; by_cases h : Lαs ⪰ q ...; by_cases h : q ⪰ Lαs ...`)

Part 2: Uniqueness of α^* (Lean: `have uniqueness : ∀ (α β : Real) ..., indiff (mix p r α ...) q → indiff (mix p r β ...) q → α = β := by ...`) Suppose there exist $\alpha_1, \alpha_2 \in [0,1]$ such that $L_{pr}^{\alpha_1} \sim q$ and $L_{pr}^{\alpha_2} \sim q$. By transitivity of indifference (Lemma A.1.6), $L_{pr}^{\alpha_1} \sim L_{pr}^{\alpha_2}$. Assume for contradiction that $\alpha_1 \neq \alpha_2$. WLOG, let $\alpha_1 < \alpha_2$. (Lean: `by_contra h_neq; cases lt_or_gt_of_ne h_neq with | inl h_α_lt_β ⇒ ... | inr h_β_lt_α ⇒ ...`) The conditions are $0 \leq \alpha_1 < \alpha_2 \leq 1$. Since $p \succ r$ (h_3), by Claim A.2 (monotonicity of strict preference in mixing probability): $L_{pr}^{\alpha_2} \succ L_{pr}^{\alpha_1}$. This means $L_{pr}^{\alpha_2} \succsim L_{pr}^{\alpha_1}$ and $\neg(L_{pr}^{\alpha_1} \succsim L_{pr}^{\alpha_2})$. However, $L_{pr}^{\alpha_1} \sim L_{pr}^{\alpha_2}$ implies $L_{pr}^{\alpha_1} \succsim L_{pr}^{\alpha_2}$ (and $L_{pr}^{\alpha_2} \succsim L_{pr}^{\alpha_1}$). The statement $\neg(L_{pr}^{\alpha_1} \succsim L_{pr}^{\alpha_2})$ contradicts $L_{pr}^{\alpha_1} \succsim L_{pr}^{\alpha_2}$. (Lean: `have h_mix_strict := claim_ii α β h₃ ...; unfold indiff at h_mix_α h_mix_β; have h_α_pref_β := PrefRel.transitive _ q _ h_mix_α.1 h_mix_β.2; unfold strictPref at h_mix_strict; exact h_mix_strict.2 h_α_pref_β`) The contradiction arises from assuming $\alpha_1 \neq \alpha_2$. Thus, $\alpha_1 = \alpha_2$. So α^* is unique.

The Lean proof then uses `use ⟨α_star, h_α_star_nonneg, h_α_star_le_one⟩` to provide the existent value (as a subtype of `Set.Icc 0 1`) and then `constructor` to prove the two parts of ∃!: existence (which is `h_α_star_indiff_q`) and uniqueness (using the `uniqueness` lemma just proved). □

A.3 Proof of Theorem 2.11

The proof proceeds by constructing such a utility function u and then showing it has the desired properties. The construction relies on previously established claims (Claim I to V from the full vNM proof structure). We assume `DecidableEq X` for this proof, as in the Lean code.

A.3.1 Step 1: Degenerate Lotteries

For each outcome $x_{val} \in \mathcal{X}$, we define the **degenerate lottery** $\delta_{x_{val}} \in \Delta(\mathcal{X})$. This lottery assigns probability 1 to the outcome x_{val} and probability 0 to all other outcomes $y \neq x_{val}$. Formally, for any $y \in \mathcal{X}$:

$$\delta_{x_{val}}(y) = \begin{cases} 1 & \text{if } y = x_{val} \\ 0 & \text{if } y \neq x_{val} \end{cases}$$

We verify that $\delta_{x_{val}}$ is indeed a lottery:

1. **Non-negativity**: For any $y \in \mathcal{X}$, $\delta_{x_{val}}(y)$ is either 1 or 0, both of which are ≥ 0.
2. **Sum to one**:

$$\sum_{y \in \mathcal{X}} \delta_{x_{val}}(y) = \delta_{x_{val}}(x_{val}) + \sum_{y \in \mathcal{X}, y \neq x_{val}} \delta_{x_{val}}(y) = 1 + \sum_{y \in \mathcal{X}, y \neq x_{val}} 0 = 1 + 0 = 1$$

Thus, $\delta_{x_{val}} \in \Delta(\mathcal{X})$. (Lean: `let δ : X → Lottery X := fun x_val => <fun y => if y = x_val then 1 else 0, by constructor; ...>`)

A.3.2 Step 2: Existence of Best and Worst Degenerate Lotteries

Let $S_\delta = \{\delta_x \mid x \in \mathcal{X}\}$ be the set of all degenerate lotteries. Since $\mathcal{X}$ is a non-empty finite set, S_δ is also a non-empty finite set. (Lean: `let s_univ := Finset.image δ Finset.univ; have hs_nonempty : s_univ.Nonempty := (Finset.univ_nonempty (α := X)).image δ`)

Since $\succsim$ is a total preorder on $\Delta(\mathcal{X})$ (by Axiom A1), it is also a total preorder on the finite subset S_δ. Therefore, there must exist maximal and minimal elements in S_δ with respect to $\succsim$.

Existence of a best degenerate lottery p^*: There exists an outcome $x^* \in \mathcal{X}$ such that its corresponding degenerate lottery $p^* = \delta_{x^*}$ is preferred or indifferent to all other degenerate lotteries. That is, $\exists x^* \in \mathcal{X}$ such that $\forall x \in \mathcal{X}, \delta_{x^*} \succsim \delta_x$. (Lean: `have exists_x_star_node : ∃ x_s : X, ∀ x : X, (δ x_s) ⪰ (δ x) := by ...`) The Lean proof for `exists_x_star_node` proceeds as follows:

1. Let $p_s = \delta_{x_s}$ be an element in S_δ that is "minimal" in the sense of Lean's `Finset.exists_minimal`. This means for any $a \in S_\delta$, $\neg(a \succ p_s)$. (Lean: `let h_greatest_lottery := Finset.exists_minimal s_univ hs_nonempty; rcases h_greatest_lottery with <p_s, <hp_s_in_s_univ, h_ps_le_all>>`). Here `h_ps_le_all a` means $\neg(a \succ p_s)$.

2. We want to show $p_s \succsim \delta_x$ for any $x \in \mathcal{X}$. Let $a = \delta_x$. So we have $\neg(\delta_x \succ p_s)$.

3. $\neg(\delta_x \succ p_s)$ means $\neg(\delta_x \succsim p_s \wedge \neg(p_s \succsim \delta_x))$. This is equivalent to $\neg(\delta_x \succsim p_s) \vee (p_s \succsim \delta_x)$. (Lean: `unfold strictPref at h_not_delta_x'_lt_p_s; push_neg at h_not_delta_x'_lt_p_s` which results in `¬(δ x ⪰ p_s) ∨ (p_s ⪰ δ x)` if we use ⪰ for `pref`.)

4. Consider two cases based on `PrefRel.complete (δ x) p_s`:

   - Case (i): $\delta_x \succsim p_s$. Then, from $\neg(\delta_x \succsim p_s) \vee (p_s \succsim \delta_x)$, since the first part $\neg(\delta_x \succsim p_s)$ is false, the second part $p_s \succsim \delta_x$ must be true. (Lean: `by_cases h : δ x' ⪰ p_s; exact h_not_delta_x'_lt_p_s h`)
   - Case (ii): $\neg(\delta_x \succsim p_s)$. By completeness (Axiom A1a), we must have $p_s \succsim \delta_x$. (Lean: `else cases PrefRel.complete (δ x') p_s with | inl h_contradiction => exact False.elim (h h_contradiction) | inr h_p_s_pref_delta_x' => exact h_p_s_pref_delta_x'`)

   In both cases, $p_s \succsim \delta_x$. So we can choose $x^* = x_s$.

Let $x^* \in \mathcal{X}$ be such an outcome, and let $p^* = \delta_{x^*}$. Then $p^* \succsim \delta_x$ for all $x \in \mathcal{X}$. (Lean: `let x_star := Classical.choose exists_x_star_node; let p_star := δ x_star; have h_p_star_is_max : ∀ x : X, p_star ⪰ δ x := Classical.choose_spec exists_x_star_node`)

Existence of a worst degenerate lottery p°: Similarly, there exists an outcome $x^\circ \in \mathcal{X}$ such that its corresponding degenerate lottery $p^\circ = \delta_{x^\circ}$ is such that all other degenerate lotteries are preferred or indifferent to it. That is, $\exists x^\circ \in \mathcal{X}$ such that $\forall x \in \mathcal{X}, \delta_x \succsim \delta_{x^\circ}$. (Lean: `have exists_x_circ_node : ∃ x_c : X, ∀ x : X, (δ x) ⪰ (δ x_c) := by ...`) The Lean proof for `exists_x_circ_node` uses `Finset.exists_maximal`. Let $p_c = \delta_{x_c}$ be such an element. `exists_maximal` means for any $a \in S_\delta$, $\neg(p_c \succ a)$.

1. We have $\neg(p_c \succ \delta_x)$, which means $\neg(p_c \succsim \delta_x \wedge \neg(\delta_x \succsim p_c))$. This is equivalent to $\neg(p_c \succsim \delta_x) \vee (\delta_x \succsim p_c)$.

2. Consider two cases based on `PrefRel.complete (δ x) p_c`:

   - Case (i): $\delta_x \succsim p_c$. This is the desired conclusion. (Lean: `cases PrefRel.complete (δ x) p_max with | inl h_delta_pref_pmax => ...‘. If` $\delta_x \succ p_c$`, then` $\delta_x \succsim p_c$`. If` $\delta_x \sim p_c$`, then` $\delta_x \succsim p_c$)
   - Case (ii): $p_c \succsim \delta_x$ (and $\neg(\delta_x \succsim p_c)$ is not necessarily true from this case alone). From $\neg(p_c \succsim \delta_x) \vee (\delta_x \succsim p_c)$: since the first part $\neg(p_c \succsim \delta_x)$ is false (as we are in the case $p_c \succsim \delta_x$), the second part $\delta_x \succsim p_c$ must be true. (Lean: `| inr h_pmax_pref_delta => ... by_contra h_not_delta_x_pref_pmax; exact (hp_max_maximal (δ x) h_delta_x_in_s) ⟨h_pmax_pref_delta, h_not_delta_x_pref_pmax‘`. This shows that if $\neg(\delta_x \succsim p_c)$, then $p_c \succ \delta_x$, which contradicts $\neg(p_c \succ \delta_x)$ from maximality.)

   In both cases, $\delta_x \succsim p_c$. So we can choose $x^\circ = x_c$.

Let $x^\circ \in \mathcal{X}$ be such an outcome, and let $p^\circ = \delta_{x^\circ}$. Then $\delta_x \succsim p^\circ$ for all $x \in \mathcal{X}$. (Lean: `let x_circ := Classical.choose exists_x_circ_node; let p_circ := δ x_circ; have h_p_circ_is_min : ∀ x : X, δ x ⪰ p_circ := Classical.choose_spec exists_x_circ_nod‘`)

A.3.3 Step 3: Definition of the Utility Function $u : \mathcal{X} \to \mathbb{R}$

The utility function u is defined based on two cases: (Lean: `let u : X → Real := by classical; exact if h_indiff_ps_pc : p_star ∼ p_circ then ... else ...`)

Case 1: $p^* \sim p^\circ$. If the best degenerate lottery is indifferent to the worst degenerate lottery, it implies all degenerate lotteries are indifferent to each other (by transitivity, since $p^* \succsim \delta_x \succsim p^\circ$ for all x). In this case, we define the utility function to be constant, specifically $u(x) = 0$ for all $x \in \mathcal{X}$. (Lean: `fun _ => 0`)

Case 2: $p^* \succ p^\circ$. This means $\neg(p^* \sim p^\circ)$. Since $p^* \succsim p^\circ$ (as p^* is maximal over all δ_x, including p°), $p^* \succ p^\circ$ means $p^* \succsim p^\circ \wedge \neg(p^\circ \succsim p^*)$. (Lean: `let h_ps_succ_pc : p_star ≻ p_circ := by unfold strictPref; constructor; exact h_p_star_is_max x_circ; unfold indiff at h_indiff_ps_pc; push_neg at h_indiff_ps_pc; exact h_indiff_ps_pc (h_p_star_is_max x_circ)`) For any outcome $x \in \mathcal{X}$:

- We have $p^* \succsim \delta_x$ (by maximality of p^*).
- We have $\delta_x \succsim p^\circ$ (by minimality of p°).

So, for any $x \in \mathcal{X}$, $p^* \succsim \delta_x \succsim p^\circ$. Since we are in the case $p^* \succ p^\circ$, the conditions for Claim V (Section **??**, Claim A.5, assuming it's available from previous sections) are met with p^*, δ_x, p° playing the roles of p, q, r respectively. By Claim V, there exists a unique scalar $\alpha_x \in [0,1]$ such that $\delta_x \sim \text{mix}(p^*, p^\circ, \alpha_x)$. We define the utility of outcome x as this unique scalar: $u(x) = \alpha_x$. (Lean: `fun x => (Classical.choose (claim_v (h_p_star_is_max x) (h_p_circ_is_min x) h_ps_succ_pc)).val`) Here, `Classical.choose` selects the existent unique α_x (which is a subtype `↑(Set.Icc (0:Real) 1)`) and `.val` extracts the real number.

A.3.4 Step 4: Properties of the Utility Function u

Let $L(\alpha) = \text{mix}(p^*, p^\circ, \alpha)$.

Property 4.1: For all $x \in \mathcal{X}$, $0 \le u(x) \le 1$. (Lean: `have h_u_bounds : ∀ x, 0 ≤ u x ∧ u x ≤ 1 := by ...`)

- In Case 1 ($p^* \sim p^\circ$): $u(x) = 0$. Clearly, $0 \le 0 \le 1$. (Lean: `simp only [u]; split_ifs with h_ps_sim_pc; simp [h_ps_sim_pc]`)
- In Case 2 ($p^* \succ p^\circ$): $u(x) = \alpha_x$. By Claim V, α_x is guaranteed to be in the interval $[0,1]$. Thus $0 \le u(x) \le 1$. (Lean: `exact (Classical.choose (claim_v ...)).property` where `.property` gives the bounds `0 ≤ val` and `val ≤ 1` for the subtype.)

Property 4.2: For all $x \in \mathcal{X}$, $\delta_x \sim L(u(x))$, i.e., $\delta_x \sim \text{mix}(p^*, p^\circ, u(x))$. (Lean: `have h_delta_sim_L_ux : ∀ x, δ x ∼ L_mix (u x) (h_u_bounds x).1 (h_u_bounds x).2 := by ...`)

- In Case 1 ($p^* \sim p^\circ$): $u(x) = 0$. We need to show $\delta_x \sim L(0)$. $L(0) = \text{mix}(p^*, p^\circ, 0) = 0 \cdot p^* + (1-0) \cdot p^\circ = p^\circ$. (Lean: `have h_L0_eq_pcirc : L_mix 0 ... = p_circ := by apply Subtype.eq; ext y; simp [L_mix, Lottery.mix]`) So we need to show $\delta_x \sim p^\circ$. We know $\delta_x \succsim p^\circ$ (by minimality of p°). (Lean: `h_p_circ_is_min x`) We also need $p^\circ \succsim \delta_x$. Since $p^* \sim p^\circ$, we have $p^\circ \succsim p^*$. We know $p^* \succsim \delta_x$. By transitivity of $\succsim$, $p^\circ \succsim p^* \wedge p^* \succsim \delta_x \implies p^\circ \succsim \delta_x$. (Lean: `PrefRel.transitive p_circ p_star (δ x) h_ps_sim_pc.2 (h_p_star_is_max x)`) Since $\delta_x \succsim p^\circ$ and $p^\circ \succsim \delta_x$, we have $\delta_x \sim p^\circ$. (Lean: `rw [h_L0_eq_pcirc]; exact ⟨h_p_circ_is_min x, ...⟩`)
- In Case 2 ($p^* \succ p^\circ$): $u(x) = \alpha_x$. By the definition of α_x from Claim V, $\delta_x \sim \text{mix}(p^*, p^\circ, \alpha_x)$. Thus, $\delta_x \sim L(u(x))$. (Lean: `let h_mix_sim_delta := (Classical.choose_spec (claim_v ...)).1; exact ⟨h_mix_sim_delta.2, h_mix_sim_delta.1⟩` using symmetry of $\sim$.)

Property 4.3: For any lottery $p \in \Delta(\mathcal{X})$, $0 \le \mathbb{E}(p, u) \le 1$. (Lean: `have h_EU_bounds : ∀ p : Lottery X, 0 ≤ expectedUtility p u ∧ expectedUtility p u ≤ 1 := by ...`) Recall $\mathbb{E}(p,u) = \sum_{x \in \mathcal{X}} p(x)u(x)$.

- **Non-negativity of $\mathbb{E}(p,u)$:** For each $x \in \mathcal{X}$, $p(x) \ge 0$ (since p is a lottery) and $u(x) \ge 0$ (by Property 4.1). Therefore, each term $p(x)u(x) \ge 0$. The sum of non-negative terms is non-negative. So, $\mathbb{E}(p,u) = \sum_{x \in \mathcal{X}} p(x)u(x) \ge 0$. (Lean: `apply Finset.sum_nonneg; intro x hx; have h_p_nonneg ...; have h_u_nonneg ...; exact mul_nonneg h_p_nonneg h_u_nonneg`)
- $\mathbb{E}(p,u) \le 1$: For each $x \in \mathcal{X}$, $p(x) \ge 0$ and $u(x) \le 1$ (by Property 4.1). So, $p(x)u(x) \le p(x) \cdot 1 = p(x)$ (since $p(x) \ge 0$). Then, $\mathbb{E}(p,u) = \sum_{x \in \mathcal{X}} p(x)u(x) \le \sum_{x \in \mathcal{X}} p(x)$. Since p is a lottery, $\sum_{x \in \mathcal{X}} p(x) = 1$. Therefore, $\mathbb{E}(p,u) \le 1$. (Lean: `have h_term_le : ∀ x ∈ Finset.filter ..., p.val x * u x ≤ p.val x := by ... apply mul_le_of_le_one_right ...; have h_sum_le : (∑ x ∈ Finset.filter ..., p.val x * u x) ≤ (∑ x ∈ Finset.filter ..., p.val x) := by apply Finset.sum_le_sum h_term_le; have h_sum_eq : (∑ x ∈ Finset.filter ..., p.val x) = (∑ x, p.val x) := by ...; rw [h_sum_eq, p.property.2]; linarith`)

The Lean proof also includes `h_supp_nonempty` to ensure the sum is well-defined over a potentially empty filter if all $p(x) = 0$, but this is ruled out because $\sum p(x) = 1$ and $\mathcal{X}$ is non-empty.

A.3.5 Step 5: Completing the Proof (Outline Beyond Lean Snippet)

The provided Lean code snippet primarily sets up the utility function u and proves some of its basic properties. The full proof of the vNM theorem's existence part would continue as follows:

1. **Show $p \sim L(\mathbb{E}(p, u))$ for any $p \in \Delta(\mathcal{X})$.** This is a crucial step, often called the "linearity" property of the preference relation or the "reduction of compound lotteries" if extended. It typically relies on repeated application of the Independence Axiom (A3) to decompose p into a mixture involving degenerate lotteries, and then using Property 4.2 ($\delta_x \sim L(u(x))$). Specifically, one shows $p \sim \sum_{x\in\mathcal{X}} p(x)\delta_x \sim \sum_{x\in\mathcal{X}} p(x)L(u(x))$. And $\sum_{x\in\mathcal{X}} p(x)L(u(x)) = \sum_{x\in\mathcal{X}} p(x)(\text{mix}(p^*, p^\circ, u(x)))$. This mixture can be shown to be equal to $\text{mix}(p^*, p^\circ, \sum_{x\in\mathcal{X}} p(x)u(x)) = L(\mathbb{E}(p, u))$.

2. **Establish the representation $p \succsim q \iff \mathbb{E}(p, u) \geq \mathbb{E}(q, u)$.** From the previous step, we have $p \sim L(\mathbb{E}(p, u))$ and $q \sim L(\mathbb{E}(q, u))$. Therefore, $p \succsim q \iff L(\mathbb{E}(p, u)) \succsim L(\mathbb{E}(q, u))$.

   - If $p^* \succ p^\circ$ (Case 2 of utility definition): $L(\alpha) = \text{mix}(p^*, p^\circ, \alpha)$. By Claim II (monotonicity of mixtures, Section **??**, Claim A.2), if $p^* \succ p^\circ$, then for $\alpha_1, \alpha_2 \in [0, 1]$, $L(\alpha_1) \succsim L(\alpha_2) \iff \alpha_1 \geq \alpha_2$. Thus, $L(\mathbb{E}(p, u)) \succsim L(\mathbb{E}(q, u)) \iff \mathbb{E}(p, u) \geq \mathbb{E}(q, u)$. Combining these, $p \succsim q \iff \mathbb{E}(p, u) \geq \mathbb{E}(q, u)$.
   - If $p^* \sim p^\circ$ (Case 1 of utility definition): Then $u(x) = 0$ for all x, so $\mathbb{E}(p, u) = 0$ and $\mathbb{E}(q, u) = 0$ for all p, q. Also, if $p^* \sim p^\circ$, all lotteries are indifferent to each other. So $p \sim q$ holds for all p, q. The representation $p \succsim q \iff 0 \geq 0$ is true (as both sides are always true).

This completes the sketch of the existence proof. The uniqueness up to positive affine transformation is a separate part of the theorem.

A.4 Proof of Theorem 2.12

We assume $\mathcal{X}$ is a non-empty finite set and that equality on $\mathcal{X}$ and $\mathbb{R}$ is decidable.

Step 1: Define Degenerate Lotteries For each outcome $x_{val} \in \mathcal{X}$, define the degenerate lottery $\delta_{x_{val}} \in \Delta(\mathcal{X})$ as:

$$\delta_{x_{val}}(y) = \begin{cases} 1 & \text{if } y = x_{val} \\ 0 & \text{if } y \neq x_{val} \end{cases}$$

For such a lottery, $\mathbb{E}(\delta_{x_{val}}, u) = u(x_{val})$ and $\mathbb{E}(\delta_{x_{val}}, v) = v(x_{val})$. (Lean: `let δ : X → Lottery X := fun x_val => ⟨fun y => if y = x_val then 1 else 0, by ...⟩`)

Step 2: Identify Extreme Outcomes for Utility Function u Since $\mathcal{X}$ is finite and non-empty, the function $u : \mathcal{X} \to \mathbb{R}$ must attain a maximum and a minimum value on $\mathcal{X}$. Let $x_{max} \in \mathcal{X}$ be an outcome such that $u(x_{max}) \geq u(x)$ for all $x \in \mathcal{X}$. Let $x_{min} \in \mathcal{X}$ be an outcome such that $u(x_{min}) \leq u(x)$ for all $x \in \mathcal{X}$. (Lean: `let ⟨x_max, _, h_u_max⟩ := Finset.exists_max_image Finset.univ u Finset.univ_nonempty` and `let ⟨x_min, _, h_u_min⟩ := Finset.exists_min_image Finset.univ u Finset.univ_nonempty`. `h_u_max x _` gives $u(x) \leq u(x_{max})$ and `h_u_min x _` gives $u(x_{min}) \leq u(x)$.)

Step 3: Consider Two Cases Based on Whether u is Constant (Lean: `by_cases h_u_constant : ∀ x, u x = u x_min`)

Case 1: u is a constant function. Assume that for all $x \in \mathcal{X}$, $u(x) = u(x_{min})$. This implies $u(x) = u(x_{max}) = u(x_{min})$ for all x. Let $c_u = u(x_{min})$. (Lean: `case pos => ...`)

Substep 1.1: Show all lotteries are indifferent under $\succsim$. (Lean: `have h_indiff : ∀ p q : Lottery`

`X, p ∼ q := by ...`) Let $p, q \in \Delta(\mathcal{X})$ be any two lotteries. The expected utility of p under u is:

$$\begin{aligned}\mathbb{E}(p,u) &= \sum_{x\in\mathcal{X}} p(x)u(x) \\ &= \sum_{x\in\mathcal{X}} p(x)c_u \quad (\text{since } u(x) = c_u \text{ for all } x,\ \text{Lean: `rw [h\_u\_constant x]`}) \\ &= c_u \sum_{x\in\mathcal{X}} p(x) \quad (\text{factoring out } c_u,\ \text{Lean: `rw [Finset.mul\_sum]` after `rw [mul\_comm]`}) \\ &= c_u \cdot 1 \quad (\text{since } p \text{ is a lottery, } \sum p(x) = 1,\ \text{Lean: `rw [p.property.2]`}) \\ &= c_u \quad (\text{Lean: } \texttt{rw [mul\_one]})\end{aligned}$$

Similarly, $\mathbb{E}(q,u) = c_u$. Thus, $\mathbb{E}(p,u) = \mathbb{E}(q,u) = c_u$. (Lean: The `calc` block `expectedUtility p u = expectedUtility q u` shows this detailed derivation.)

Since u represents $\succsim$ (by hypothesis H_u):

- $p \succsim q \iff \mathbb{E}(p,u) \geq \mathbb{E}(q,u)$. Since $\mathbb{E}(p,u) = \mathbb{E}(q,u)$, we have $\mathbb{E}(p,u) \geq \mathbb{E}(q,u)$ is true. So, $p \succsim q$ is true. (Lean: `have h_p_prefers_q ... rw [h_u p q]; rw [h_EU_eq]`)
- $q \succsim p \iff \mathbb{E}(q,u) \geq \mathbb{E}(p,u)$. Since $\mathbb{E}(q,u) = \mathbb{E}(p,u)$, we have $\mathbb{E}(q,u) \geq \mathbb{E}(p,u)$ is true. So, $q \succsim p$ is true. (Lean: `have h_q_prefers_p ... rw [h_u q p]; rw [h_EU_eq]`)

Since $p \succsim q$ and $q \succsim p$, by definition, $p \sim q$. This holds for any p, q. (Lean: `exact ⟨h_p_prefers_q, h_q_prefers_p⟩`)

Substep 1.2: Show v must also be a constant function. (Lean: `have h_v_constant : ∀ x y, v x = v y := by ...`) Let $x_1, x_2 \in \mathcal{X}$ be any two outcomes. Consider the degenerate lotteries $p' = \delta_{x_1}$ and $q' = \delta_{x_2}$. From Substep 1.1, all lotteries are indifferent, so $p' \sim q'$. (Lean: `let p := δ x; let q := δ y; have h_p_indiff_q := h_indiff p q`)

Since v represents $\succsim$ (by hypothesis H_v):

- $p' \succsim q' \iff \mathbb{E}(p',v) \geq \mathbb{E}(q',v)$. Since $p' \sim q'$, $p' \succsim q'$ is true. So $\mathbb{E}(p',v) \geq \mathbb{E}(q',v)$. (Lean: `exact (h_v p q).mp h_p_indiff_q.1` gives $\mathbb{E}(p',v) \geq \mathbb{E}(q',v)$)
- $q' \succsim p' \iff \mathbb{E}(q',v) \geq \mathbb{E}(p',v)$. Since $p' \sim q'$, $q' \succsim p'$ is true. So $\mathbb{E}(q',v) \geq \mathbb{E}(p',v)$. (Lean: `exact (h_v q p).mp h_p_indiff_q.2` gives $\mathbb{E}(q',v) \geq \mathbb{E}(p',v)$)

From $\mathbb{E}(p',v) \geq \mathbb{E}(q',v)$ and $\mathbb{E}(q',v) \geq \mathbb{E}(p',v)$, we conclude $\mathbb{E}(p',v) = \mathbb{E}(q',v)$. (Lean: `apply le_antisymm ...`) We know $\mathbb{E}(\delta_{x_1},v) = v(x_1)$ and $\mathbb{E}(\delta_{x_2},v) = v(x_2)$. Therefore, $v(x_1) = v(x_2)$. (Lean: The `calc` block `v x = v y` shows $v(x) = \mathbb{E}(\delta_x,v) = \mathbb{E}(\delta_y,v) = v(y)$.) Since x_1, x_2 were arbitrary, v is a constant function. Let $c_v = v(x_{min})$. So $v(x) = c_v$ for all x.

Substep 1.3: Construct α and β. Let $\alpha = 1$. Then $\alpha > 0$. Let $\beta = c_v - 1 \cdot c_u = v(x_{min}) - u(x_{min})$. (Lean: `let α : Real := 1; let β := v x_min - u x_min * α; use α, β; constructor; exact zero_lt_one`) We need to show $v(x) = \alpha \cdot u(x) + \beta$ for all $x \in \mathcal{X}$. For any $x \in \mathcal{X}$: $v(x) = c_v$ (since v is constant). $\alpha \cdot u(x) + \beta = 1 \cdot c_u + (c_v - c_u) = c_u + c_v - c_u = c_v$. So, $v(x) = \alpha \cdot u(x) + \beta$ holds. (Lean: `intro x; have h_v_eq_constant : v x = v x_min ...; have h_u_eq_constant : u x = u x_min ...; calc v x = v x_min ... = β + α * u x ... = α * u x + β := by ring`)

Case 2: u is not a constant function. (Lean: `case neg => ...`) This means that $u(x_{max}) > u(x_{min})$. If $u(x_{max}) = u(x_{min})$, then since $u(x_{min}) \leq u(x) \leq u(x_{max})$ for all x, u would be constant. (Lean: `push_neg at h_u_constant` makes `h_u_constant‘ into ‘∃ x, u x ≠ u x_min. let x_diff := Classical.choose h_u_constant; have h_x_diff : u x_diff ≠ u x_min := Classical.choose_spec h_u_constant; have h_x_diff_gt : u x_diff > u x_min := by have h_ge := h_u_min x_diff ...; exact lt_of_le_of_ne h_ge (Ne.symm h_x_diff); have h_max_gt_min : u x_max > u x_min := by ...` The proof for `h_max_gt_min` uses contradiction: if $u(x_{max}) \leq u(x_{min})$, since $u(x_{min}) \leq u(x_{max})$ is always true, then $u(x_{max}) = u(x_{min})$. This implies $u$ is constant, contradicting `h_u_constant`.)

Let $p_{best} = \delta_{x_{max}}$ and $p_{worst} = \delta_{x_{min}}$. Since $u(x_{max}) > u(x_{min})$, we have $\mathbb{E}(p_{best},u) > \mathbb{E}(p_{worst},u)$. By hypothesis H_u:

- $p_{best} \succsim p_{worst}$ because $\mathbb{E}(p_{best}, u) \geq \mathbb{E}(p_{worst}, u)$ (true as $u(x_{max}) > u(x_{min})$).
- $\neg(p_{worst} \succsim p_{best})$ because $\mathbb{E}(p_{worst}, u) \geq \mathbb{E}(p_{best}, u)$ (i.e., $u(x_{min}) \geq u(x_{max})$) is false.

Thus, $p_{best} \succ p_{worst}$. (Lean: `have h_best_succ_worst : p_best ≻ p_worst := by constructor; ... exact le_of_lt h_max_gt_min; ... exact h_max_gt_min`)

For any $\alpha_0 \in [0, 1]$, define the mixed lottery $L_{\alpha_0} = \text{mix}(p_{best}, p_{worst}, \alpha_0) = \alpha_0 \cdot p_{best} + (1 - \alpha_0) \cdot p_{worst}$. (Lean: `let mix (α : Real) (hα_nonneg : 0 ≤ α) (hα_le_one : α ≤ 1) : Lottery X := @Lottery.mix X _ p_best p_worst α hα_nonneg hα_le_one`)

Substep 2.1: Expected utility of L_{α_0} under u. (Lean: `have h_mix_EU_u : ∀ α (h_α : α ∈ Set.Icc 0 1), expectedUtility (mix α h_α.1 h_α.2) u = u x_min + α * (u x_max - u x_min) := by ...`)

$$\begin{aligned}
\mathbb{E}(L_{\alpha_0}, u) &= \sum_{x \in \mathcal{X}} L_{\alpha_0}(x) u(x) \\
&= \sum_{x \in \mathcal{X}} (\alpha_0 \delta_{x_{max}}(x) + (1 - \alpha_0)\delta_{x_{min}}(x)) u(x) \\
&= \alpha_0 \sum_{x \in \mathcal{X}} \delta_{x_{max}}(x) u(x) + (1 - \alpha_0) \sum_{x \in \mathcal{X}} \delta_{x_{min}}(x) u(x) \\
&= \alpha_0 \mathbb{E}(\delta_{x_{max}}, u) + (1 - \alpha_0)\mathbb{E}(\delta_{x_{min}}, u) \\
&= \alpha_0 u(x_{max}) + (1 - \alpha_0) u(x_{min}) \\
&= \alpha_0 u(x_{max}) + u(x_{min}) - \alpha_0 u(x_{min}) \\
&= u(x_{min}) + \alpha_0 (u(x_{max}) - u(x_{min}))
\end{aligned}$$

(Lean: The `calc` block `α * ∑ i, p_best.val i * u i + (1 - α) * ∑ i, p_worst.val i * u i = ... = u x_min + α * (u x_max - u x_min) := by ring` after simplifying sums.)

Substep 2.2: Expected utility of L_{α_0} under v. Similarly, $\mathbb{E}(L_{\alpha_0}, v) = v(x_{min}) + \alpha_0(v(x_{max}) - v(x_{min}))$. (Lean: `have h_mix_EU_v : ∀α (h_α : α ∈ Set.Icc 0 1), expectedUtility (mix α h_α.1 h_α.2) v = v x_min + α * (v x_max - v x_min) := by ...`)

Substep 2.3: Relating $u(x)$ and $v(x)$ via an intermediate α_x. For any $x \in \mathcal{X}$, since $u(x_{min}) \leq u(x) \leq u(x_{max})$ and $u(x_{max}) - u(x_{min}) > 0$: Let $\alpha_x = \frac{u(x) - u(x_{min})}{u(x_{max}) - u(x_{min})}$. Then $0 \leq \alpha_x \leq 1$. (Lean: `let α_x := (u x - u x_min) / (u x_max - u x_min); have h_α_x_nonneg ...; have h_α_x_le_one ...`) With this α_x, we have $u(x) = \mathbb{E}(\delta_x, u) = u(x_{min}) + \alpha_x(u(x_{max}) - u(x_{min}))$. From Substep 2.1, this means $\mathbb{E}(\delta_x, u) = \mathbb{E}(L_{\alpha_x}, u)$. (Lean: `have h_exists_α : ∀ x, ∃ (α : Real) (h_α : α ∈ Set.Icc 0 1), EU (δ x) u = EU (mix α h_α.1 h_α.2) u := by intro x; ... use α_x ... rw [h_EU_δ_u, h_EU_mix_α_u]`)

Since both u and v represent $\succsim$: $\mathbb{E}(\delta_x, u) = \mathbb{E}(L_{\alpha_x}, u) \implies \delta_x \sim L_{\alpha_x}$ (using H_u). Since $\delta_x \sim L_{\alpha_x}$, and v also represents $\succsim$, it must be that $\mathbb{E}(\delta_x, v) = \mathbb{E}(L_{\alpha_x}, v)$ (using H_v). (Lean: `have h_eq_EU_v : ∀ x, ∀ α (h_α : ...), EU (δ x) u = EU (mix α ...) u → EU (δ x) v = EU (mix α ...) v := by intro x α h_α h_eq_u; ... have h_indiff : δ x ∼ mix α ...; ... have h_v_eq : EU (δ x) v = EU (mix α ...) v; ... exact h_v_eq`)

So, we have: $v(x) = \mathbb{E}(\delta_x, v)$ $v(x) = \mathbb{E}(L_{\alpha_x}, v) = v(x_{min}) + \alpha_x(v(x_{max}) - v(x_{min}))$ (using Substep 2.2). Substitute $\alpha_x = \frac{u(x) - u(x_{min})}{u(x_{max}) - u(x_{min})}$:

$$v(x) = v(x_{min}) + \frac{u(x) - u(x_{min})}{u(x_{max}) - u(x_{min})}(v(x_{max}) - v(x_{min}))$$

Substep 2.4: Define α and β and prove the affine relationship. Let $\alpha = \frac{v(x_{max}) - v(x_{min})}{u(x_{max}) - u(x_{min})}$. (Lean: `let α := (v x_max - v x_min) / (u x_max - u x_min)`) Since $p_{best} \succ p_{worst}$, this implies $\mathbb{E}(p_{best}, v) > \mathbb{E}(p_{worst}, v)$, so $v(x_{max}) > v(x_{min})$. Also $u(x_{max}) > u(x_{min})$ (as u is not constant). Therefore, $v(x_{max}) - v(x_{min}) > 0$ and $u(x_{max}) - u(x_{min}) > 0$, which implies $\alpha > 0$. (Lean: `have h_α_pos : α > 0 := by ... have h_v_max_gt_min ... exact div_pos (sub_pos.mpr h_v_max_gt_min) (sub_pos.mpr h_max_gt_min)`)

Let $\beta = v(x_{min}) - \alpha \cdot u(x_{min})$. (Lean: `let β:= v x_min - α * u x_min`)

We want to show $v(x) = \alpha \cdot u(x) + \beta$. Substitute the expressions for α and β:

$$\begin{aligned}\alpha \cdot u(x) + \beta &= \frac{v(x_{max}) - v(x_{min})}{u(x_{max}) - u(x_{min})} u(x) + \left(v(x_{min}) - \frac{v(x_{max}) - v(x_{min})}{u(x_{max}) - u(x_{min})} u(x_{min}) \right) \\ &= v(x_{min}) + \frac{v(x_{max}) - v(x_{min})}{u(x_{max}) - u(x_{min})} (u(x) - u(x_{min}))\end{aligned}$$

This is exactly the expression we found for $v(x)$ in Substep 2.3. Therefore, $v(x) = \alpha \cdot u(x) + \beta$. (Lean: `use α, β; constructor; exact h_α_pos; intro x; ... obtain ⟨α_x, h_α_x, h_EU_eq⟩ := h_exists_α x; ... have h_v_x_eq : v x = v x_min + α_x * (v x_max - v x_min) ... rw [h_v_x_eq, h_α_x_val_eq]; simp only [α,β]; ring_nf`) The `ring_nf` tactic in Lean verifies this algebraic identity.

This completes the proof for both cases.