

# Von Neumann Morgenstern Utility Theorem \*

Julian Parsert      Cezary Kaliszyk

February 23, 2021

## Abstract

Utility functions form an essential part of game theory and economics. In order to guarantee the existence of utility functions most of the time sufficient properties are assumed in an axiomatic manner. One famous and very common set of such assumptions is that of expected utility theory. Here, the rationality, continuity, and independence of preferences is assumed. The von-Neumann-Morgenstern Utility theorem shows that these assumptions are necessary and sufficient for an expected utility function to exist. This theorem was proven by Neumann and Morgenstern in “Theory of Games and Economic Behavior” which is regarded as one of the most influential works in game theory.

We formalize these results in Isabelle/HOL. The formalization includes formal definitions of the underlying concepts including continuity and independence of preferences.

## Contents

<b>1</b>	<b>Composition of Probability Mass functions</b>	<b>2</b>
<b>2</b>	<b>Lotteries</b>	<b>12</b>
<b>3</b>	<b>Properties of Preferences</b>	<b>15</b>
3.1	Independent Preferences . . . . .	15
3.2	Continuity . . . . .	19
<b>4</b>	<b>System U start, as per vNM</b>	<b>20</b>
<b>5</b>	<b>This lemma is in called step 1 in literature. In Von Neumann and Morgenstern’s book this is A:A (albeit more general)</b>	<b>23</b>
5.1	Add finiteness and non emptyness of outcomes . . . . .	29
5.2	Add continuity to assumptions . . . . .	34

---

\*This work is supported by the Austrian Science Fund (FWF) project P26201 and the European Research Council (ERC) grant no 714034 *SMART*.

<b>6</b>	<b>Definition of vNM-utility function</b>	<b>47</b>
<b>7</b>	<b>Finite outcomes</b>	<b>49</b>
<b>8</b>	<b>Related work</b>	<b>53</b>

```

theory PMF-Composition
  imports
    HOL-Probability.Probability
begin

```

## 1 Composition of Probability Mass functions

**definition** *mix-pmf* :: *real*  $\Rightarrow$  *'a pmf*  $\Rightarrow$  *'a pmf*  $\Rightarrow$  *'a pmf* **where**  
*mix-pmf*  $\alpha$  *p q* = (*bernoulli-pmf*  $\alpha$ )  $\gg$  ( $\lambda X.$  if *X* then *p* else *q*)

**lemma** *pmf-mix*:  $a \in \{0..1\} \implies \text{pmf } (\text{mix-pmf } a \ p \ q) \ x = a * \text{pmf } p \ x + (1 - a) * \text{pmf } q \ x$   
**by** (*simp add: mix-pmf-def pmf-bind*)

**lemma** *pmf-mix-deeper*:  $a \in \{0..1\} \implies \text{pmf } (\text{mix-pmf } a \ p \ q) \ x = a * \text{pmf } p \ x + \text{pmf } q \ x - a * \text{pmf } q \ x$   
**by** (*simp add: left-diff-distrib' pmf-mix*)

**lemma** *bernoulli-pmf-0* [*simp*]: *bernoulli-pmf* 0 = *return-pmf* False  
**by** (*intro pmf-eqI*) (*auto simp: bernoulli-pmf.rep-eq*)

**lemma** *bernoulli-pmf-1* [*simp*]: *bernoulli-pmf* 1 = *return-pmf* True  
**by** (*intro pmf-eqI*) (*auto simp: bernoulli-pmf.rep-eq*)

**lemma** *pmf-mix-0* [*simp*]: *mix-pmf* 0 *p q* = *q*  
**by** (*simp add: mix-pmf-def bind-return-pmf*)

**lemma** *pmf-mix-1* [*simp*]: *mix-pmf* 1 *p q* = *p*  
**by** (*simp add: mix-pmf-def bind-return-pmf*)

**lemma** *set-pmf-mix*:  $a \in \{0 < .. < 1\} \implies \text{set-pmf } (\text{mix-pmf } a \ p \ q) = \text{set-pmf } p \cup \text{set-pmf } q$   
**by** (*auto simp add: mix-pmf-def split: if-splits*)

**lemma** *set-pmf-mix-eq*:  $a \in \{0..1\} \implies \text{mix-pmf } a \ p \ p = p$   
**by** (*simp add: mix-pmf-def*)

**lemma** *pmf-equiv-intro*[*intro*]:  
**assumes**  $\bigwedge e. e \in \text{set-pmf } p \implies \text{pmf } p \ e = \text{pmf } q \ e$   
**assumes**  $\bigwedge e. e \in \text{set-pmf } q \implies \text{pmf } q \ e = \text{pmf } p \ e$

**shows**  $p = q$   
**by** (*metis* *assms*(2) *less-irrefl* *pmf-neq-exists-less* *pmf-not-neg* *set-pmf-iff*)

**lemma** *pmf-equiv-intro1*[*intro*]:  
**assumes**  $\bigwedge e. e \in \text{set-pmf } p \implies \text{pmf } p \ e = \text{pmf } q \ e$   
**shows**  $p = q$   
**by** (*standard*, *auto simp: assms*, *metis assms set-pmf-iff assms*  
*linorder-not-le order-refl pmf-neq-exists-less pmf-not-neg set-pmf-iff*)

**lemma** *pmf-inverse-switch-equals*:  
**assumes**  $a \in \{0..1\}$   
**shows**  $\text{mix-pmf } a \ p \ q = \text{mix-pmf } (1-a) \ q \ p$   
**proof** –  
**have** *fst*:  $\forall x \in \text{set-pmf } p. \text{pmf } (\text{mix-pmf } a \ p \ q) \ x = \text{pmf } (\text{mix-pmf } (1-a) \ q \ p) \ x$   
**proof**  
**fix**  $x$   
**assume**  $x \in \text{set-pmf } p$   
**have**  $\text{pmf } (\text{mix-pmf } a \ p \ q) \ x = a * \text{pmf } p \ x + (1 - a) * \text{pmf } q \ x$   
**using** *pmf-mix*[*of a p q x*] *assms* **by** *blast*  
**also have**  $\dots = a * \text{pmf } p \ x + \text{pmf } q \ x - a * \text{pmf } q \ x$   
**by** (*simp add: left-diff-distrib*)  
**from** *pmf-mix*[*of 1-a q p x*] *assms*  
**have**  $\text{pmf } (\text{mix-pmf } (1 - a) \ q \ p) \ x = (1 - a) * \text{pmf } q \ x + (1 - (1 - a)) * \text{pmf } p \ x$   
**by** *auto*  
**then show**  $\text{pmf } (\text{mix-pmf } a \ p \ q) \ x = \text{pmf } (\text{mix-pmf } (1 - a) \ q \ p) \ x$   
**using** *calculation* **by** *auto*  
**qed**  
**have**  $\forall x \in \text{set-pmf } q. \text{pmf } (\text{mix-pmf } a \ p \ q) \ x = \text{pmf } (\text{mix-pmf } (1-a) \ q \ p) \ x$   
**proof**  
**fix**  $x$   
**assume**  $x \in \text{set-pmf } q$   
**have**  $\text{pmf } (\text{mix-pmf } a \ p \ q) \ x = a * \text{pmf } p \ x + (1 - a) * \text{pmf } q \ x$   
**using** *pmf-mix*[*of a p q x*] *assms* **by** *blast*  
**also have**  $\dots = a * \text{pmf } p \ x + \text{pmf } q \ x - a * \text{pmf } q \ x$   
**by** (*simp add: left-diff-distrib*)  
**from** *pmf-mix*[*of 1-a q p x*] *assms*  
**show**  $\text{pmf } (\text{mix-pmf } a \ p \ q) \ x = \text{pmf } (\text{mix-pmf } (1 - a) \ q \ p) \ x$   
**using** *calculation* **by** *auto*  
**qed**  
**then have**  $\forall x \in \text{set-pmf } (\text{mix-pmf } a \ p \ q). \text{pmf } (\text{mix-pmf } a \ p \ q) \ x = \text{pmf } (\text{mix-pmf } (1-a) \ q \ p) \ x$   
**by** (*metis* (*no-types*) *fst add-0-left assms mult-eq-0-iff pmf-mix set-pmf-iff*)  
**thus** *?thesis*  
**by** (*simp add: pmf-equiv-intro1*)  
**qed**

**lemma** *mix-pmf-comp-left-div*:  
**assumes**  $\alpha \in \{0..(1::\text{real})\}$

```

    and  $\beta \in \{0..(1::real)\}$ 
    assumes  $\alpha > \beta$ 
    shows  $\text{pmf } (\text{mix-pmf } (\beta/\alpha) (\text{mix-pmf } \alpha \text{ p q}) \text{ q}) \text{ e} = \beta * \text{pmf } \text{p e} + \text{pmf } \text{q e} -$ 
 $\beta * \text{pmf } \text{q e}$ 
  proof-
    let ?l =  $(\text{mix-pmf } (\beta/\alpha) (\text{mix-pmf } \alpha \text{ p q}) \text{ q})$ 
    have fst:  $\text{pmf } ?l \text{ e} = (\beta/\alpha) * \text{pmf } (\text{mix-pmf } \alpha \text{ p q}) \text{ e} + (1-\beta/\alpha) * \text{pmf } \text{q e}$ 
      by (meson assms(1) assms(2) assms(3) atLeastAtMost-iff less-divide-eq-1
        less-eq-real-def not-less pmf-mix zero-le-divide-iff)
    then have  $\text{pmf } (\text{mix-pmf } \alpha \text{ p q}) \text{ e} = \alpha * \text{pmf } \text{p e} + (1 - \alpha) * \text{pmf } \text{q e}$ 
      using pmf-mix[of  $\alpha \text{ p q}$ ] assms(2) assms(3) assms(1) by blast
    have  $\text{pmf } ?l \text{ e} = (\beta/\alpha) * (\alpha * \text{pmf } \text{p e} + (1 - \alpha) * \text{pmf } \text{q e}) + (1-\beta/\alpha) * \text{pmf } \text{q e}$ 
      using fst assms(1) pmf-mix by fastforce
    then have  $\text{pmf } ?l \text{ e} = ((\beta/\alpha) * \alpha * \text{pmf } \text{p e} + (\beta/\alpha) * (1 - \alpha) * \text{pmf } \text{q e}) +$ 
 $(1-\beta/\alpha) * \text{pmf } \text{q e}$ 
      using fst assms(1) by (metis mult.assoc ring-class.ring-distrib(1))
    then have *:  $\text{pmf } ?l \text{ e} = (\beta * \text{pmf } \text{p e} + (\beta/\alpha) * (1 - \alpha) * \text{pmf } \text{q e}) + (1-\beta/\alpha)$ 
 $* \text{pmf } \text{q e}$ 
      using fst assms(1) assms(2) assms(3) by auto
    then have  $\text{pmf } ?l \text{ e} = (\beta * \text{pmf } \text{p e} + ((\beta/\alpha) - (\beta/\alpha)*\alpha) * \text{pmf } \text{q e}) + (1-\beta/\alpha)$ 
 $* \text{pmf } \text{q e}$ 
      using fst assms(1) assms(2) assms(3) by (simp add: * diff-divide-distrib
        right-diff-distrib')
    then have  $\text{pmf } ?l \text{ e} = (\beta * \text{pmf } \text{p e} + ((\beta/\alpha) - \beta) * \text{pmf } \text{q e}) + (1-\beta/\alpha) * \text{pmf } \text{q e}$ 
      using fst assms(1) assms(2) assms(3) by auto
    then have  $\text{pmf } ?l \text{ e} = (\beta * \text{pmf } \text{p e} + (\beta/\alpha) * \text{pmf } \text{q e} - \beta * \text{pmf } \text{q e}) + 1 * \text{pmf } \text{q e}$ 
 $- \beta/\alpha * \text{pmf } \text{q e}$ 
      by (simp add: left-diff-distrib)
    thus ?thesis
      by linarith
  qed

```

**lemma** *mix-pmf-comp-with-dif-equiv:*

```

    assumes  $\alpha \in \{0..(1::real)\}$ 
    and  $\beta \in \{0..(1::real)\}$ 
    assumes  $\alpha > \beta$ 
    shows  $\text{mix-pmf } (\beta/\alpha) (\text{mix-pmf } \alpha \text{ p q}) \text{ q} = \text{mix-pmf } \beta \text{ p q}$  (is ?l = ?r)
  proof (rule pmf-equiv-intro1[symmetric])
    fix e
    assume a:  $e \in \text{set-pmf } ?r$ 
    have  $e \in \text{set-pmf } ?l$ 
      using a pmf-mix-deeper by (metis assms(1) assms(2) assms(3) mix-pmf-comp-left-div
        pmf-eq-0-set-pmf)
    then have  $\text{pmf } ?l \text{ e} = \beta * \text{pmf } \text{p e} - \beta * \text{pmf } \text{q e} + \text{pmf } \text{q e}$ 
      using pmf-mix-deeper[of  $\beta/\alpha \text{ p q e}$ ] mix-pmf-comp-left-div[of  $\alpha \beta \text{ p q e}$ ] assms
    by auto
    then show  $\text{pmf } (\text{mix-pmf } \beta \text{ p q}) \text{ e} = \text{pmf } (\text{mix-pmf } (\beta / \alpha) (\text{mix-pmf } \alpha \text{ p q})$ 

```

$q) e$   
 by (metis (full-types) assms(1) assms(2) assms(3) mix-pmf-comp-left-div pmf-mix-deeper)  
 qed

**lemma** product-mix-pmf-prob-distrib:

assumes  $a \in \{0..1\}$

and  $b \in \{0..1\}$

shows  $\text{mix-pmf } a (\text{mix-pmf } b p q) q = \text{mix-pmf } (a*b) p q$

**proof** –

define  $\gamma$  where  $g: \gamma = (a * b)$

define  $l$  where  $l: l = (\text{mix-pmf } b p q)$

define  $r$  where  $r: r = \text{mix-pmf } (a*b) p q$

have  $y: \gamma \in \{0..1\}$

using assms(2) mult-le-one assms  $g$  by auto

have alt:  $\forall e \in \text{set-pmf } l. \text{pmf } r e = \gamma * \text{pmf } p e + \text{pmf } q e - \gamma * \text{pmf } q e$

**proof**

fix  $e$

have  $\text{pmf } r e = \gamma * \text{pmf } p e + (1-\gamma) * \text{pmf } q e$

using  $\langle \gamma \in \{0..1\} \rangle g$  pmf-mix  $r$  by fastforce

moreover have  $\dots = \gamma * \text{pmf } p e + 1 * \text{pmf } q e - \gamma * \text{pmf } q e$

by (simp add: algebra-simps)

moreover have  $\dots = \text{pmf } (\text{mix-pmf } \gamma p q) e$

using calculation  $g r$  by auto

moreover have  $\dots = \gamma * \text{pmf } p e + \text{pmf } q e - \gamma * \text{pmf } q e$

using calculation by auto

ultimately show  $\text{pmf } r e = \gamma * \text{pmf } p e + \text{pmf } q e - \gamma * \text{pmf } q e$

by auto

qed

have  $\forall e \in \text{set-pmf } r. \text{pmf } l e = b * \text{pmf } p e + \text{pmf } q e - b * \text{pmf } q e$

using allI pmf-mix-deeper assms(2)  $l$  by fastforce

have  $\text{mix-pmf } a (\text{mix-pmf } b p q) q = \text{mix-pmf } (a * b) p q$

**proof** (rule ccontr)

assume neg:  $\neg \text{mix-pmf } a (\text{mix-pmf } b p q) q = \text{mix-pmf } (a * b) p q$

then have  $b: b \neq 0$

by (metis (no-types) assms(1) mult-cancel-right2 pmf-mix-0 set-pmf-mix-eq)

have  $f3: b - (a * b) > 0 \longrightarrow \text{mix-pmf } a (\text{mix-pmf } b p q) q = \text{mix-pmf } (a * b)$

$p q$

by (metis assms(2) diff-le-0-iff-le  $g$  mix-pmf-comp-with-dif-equiv mult-eq-0-iff  
 nonzero-mult-div-cancel-right not-le order-refl  $y$ )

thus False

using  $b$  neg assms(1) assms(2) by auto

qed

then show ?thesis by auto

qed

**lemma** mix-pmf-subset-of-original:

assumes  $a \in \{0..1\}$

shows  $(\text{set-pmf } (\text{mix-pmf } a p q)) \subseteq \text{set-pmf } p \cup \text{set-pmf } q$

**proof** –

```

have a ∈ {0 < .. < 1} ⇒ ?thesis
  by (simp add: set-pmf-mix)
moreover have a = 1 ⇒ ?thesis
  by simp
moreover have a = 0 ⇒ ?thesis
  by simp
ultimately show ?thesis
  using assms less-eq-real-def by auto
qed

lemma mix-pmf-preserves-finite-support:
  assumes a ∈ {0..1}
  assumes finite (set-pmf p)
  and finite (set-pmf q)
  shows finite (set-pmf (mix-pmf a p q))
  by (meson assms(1) assms(2) assms(3) finite-Un finite-subset mix-pmf-subset-of-original)

lemma ex-certain-iff-singleton-support:
  shows (∃ x. pmf p x = 1) ↔ card (set-pmf p) = 1
proof (rule iffI, goal-cases)
  case 1
  show ?case
  proof (rule ccontr)
    assume neg: ¬ card (set-pmf p) = 1
    then have card (set-pmf p) ≠ 1
      by blast
    have finite (set-pmf p)
      by (metis 1 empty-iff finite.emptyI finite-insert insert-iff
        not-le pmf-le-1 pmf-neq-exists-less pmf-nonneg set-pmf-iff set-return-pmf)
    then have sumeq-1: (∑ i ∈ set-pmf p. pmf p i) = 1
      using sum-pmf-eq-1[of set-pmf p p] by auto
    have set-pmf-nempty: set-pmf p ≠ {}
      by (simp add: set-pmf-not-empty)
    then have g1: card (set-pmf p) > 1
      by (metis card-0-eq less-one nat-neq-iff neg sum.infinite sumeq-1 zero-neq-one)
    have card (set-pmf p) > 1 → (∑ i ∈ set-pmf p. pmf p i) > 1
    proof
      assume card (set-pmf p) > 1
      have ∃ x y. pmf p x = 1 ∧ y ≠ x ∧ y ∈ set-pmf p
        using set-pmf-nempty is-singletonI' is-singleton-altdef
        by (metis 1 neg)
      then show (∑ i ∈ set-pmf p. pmf p i) > 1
        by (metis AE-measure-pmf-iff UNIV-I empty-iff insert-iff
          measure-pmf.prob-eq-1 pmf.rep-eq sets-measure-pmf)
    qed
    then have card (set-pmf p) < 1
      using sumeq-1 neg by linarith
    then show False
      using g1 by linarith
  end
end

```

**qed**  
**qed** (*metis card-1-singletonE less-numeral-extra(1) pmf.rep-eq subset-eq  
sum-pmf-eq-1[of set-pmf p p] card-gt-0-iff[of set-pmf p]  
measure-measure-pmf-finite[of set-pmf p]*)

We thank Manuel Eberl for suggesting the following two lemmas.

**lemma** *mix-pmf-partition*:

**fixes**  $p :: 'a \text{ pmf}$   
**assumes**  $y \in \text{set-pmf } p$   $\text{set-pmf } p - \{y\} \neq \{\}$   
**obtains**  $a \ q$  **where**  $a \in \{0 <..<1\}$   $\text{set-pmf } q = \text{set-pmf } p - \{y\}$   
 $p = \text{mix-pmf } a \ q$  (*return-pmf y*)  
**proof** –  
**from** *assms* **obtain**  $x$  **where**  $x: x \in \text{set-pmf } p - \{y\}$  **by** *auto*  
**define**  $a$  **where**  $a = 1 - \text{pmf } p \ y$   
**have**  $a \neq 1$   
**by** (*simp add: a-def assms(1) pmf-eq-0-set-pmf*)  
**have**  $\text{pmf } p \ y \neq 1$   
**using** *ex-certain-iff-singleton-support* **by** (*metis (full-types)  
Diff-cancel assms(1) assms(2) card-1-singletonE singletonD*)  
**hence**  $y: \text{pmf } p \ y < 1$  **using** *pmf-le-1[of p y]* **unfolding** *a-def* **by** *linarith*  
**hence**  $a: a > 0$  **by** (*simp add: a-def*)  
**define**  $q$  **where**  $q = \text{embed-pmf } (\lambda z. \text{if } z = y \text{ then } 0 \text{ else } \text{pmf } p \ z / a)$   
**have**  $q: \text{pmf } q \ z = (\text{if } z = y \text{ then } 0 \text{ else } \text{pmf } p \ z / a)$  **for**  $z$   
**unfolding** *q-def*  
**proof** (*rule pmf-embed-pmf*)  
**have**  $1 = (\int^+ x. \text{ennreal } (\text{pmf } p \ x) \ \partial \text{count-space UNIV})$   
**by** (*rule nn-integral-pmf-eq-1 [symmetric]*)  
**also have**  $\dots = (\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } \{y\} \ x +$   
 $\text{ennreal } (\text{pmf } p \ x) * \text{indicator } (-\{y\}) \ x \ \partial \text{count-space UNIV})$   
**by** (*intro nn-integral-cong (auto simp: indicator-def)*)  
**also have**  $\dots = (\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } \{y\} \ x \ \partial \text{count-space}$   
 $\text{UNIV}) +$   
 $(\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } (-\{y\}) \ x \ \partial \text{count-space UNIV})$   
**by** (*subst nn-integral-add*) *auto*  
**also have**  $(\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } \{y\} \ x \ \partial \text{count-space UNIV}) =$   
 $\text{pmf } p \ y$   
**by** (*subst nn-integral-indicator-finite*) *auto*  
**also have**  $\text{ennreal } (\text{pmf } p \ y) + (\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } (-\{y\}) \ x$   
 $\ \partial \text{count-space UNIV})$   
 $= \text{ennreal } (\text{pmf } p \ y) = (\int^+ x. \text{ennreal } (\text{pmf } p \ x) * \text{indicator } (-\{y\})$   
 $\ x \ \partial \text{count-space UNIV})$   
**by** *simp*  
**also have**  $1 - \text{ennreal } (\text{pmf } p \ y) = \text{ennreal } (1 - \text{pmf } p \ y)$   
**by** (*subst ennreal-1 [symmetric], subst ennreal-minus*) *auto*  
**finally have**  $\text{eq: } (\int^+ x \in -\{y\}. \text{ennreal } (\text{pmf } p \ x) \ \partial \text{count-space UNIV}) = 1 -$   
 $\text{pmf } p \ y \ ..$   
**have**  $(\int^+ x. \text{ennreal } (\text{if } x = y \text{ then } 0 \text{ else } \text{pmf } p \ x / a) \ \partial \text{count-space UNIV}) =$   
 $(\int^+ x. \text{inverse } a * (\text{ennreal } (\text{pmf } p \ x) * \text{indicator } (-\{y\}) \ x) \ \partial \text{count-space}$   
 $\text{UNIV})$

**using**  $a$  **by** (*intro nn-integral-cong*) (*auto simp: divide-simps ennreal-mult' [symmetric]*)  
**also have**  $\dots = \text{inverse } a * (\int^+ x \in -\{y\}. \text{ennreal } (\text{pmf } p \ x) \ \partial \text{count-space UNIV})$   
**using**  $a$  **by** (*subst nn-integral-cmult [symmetric]*) (*auto simp: ennreal-mult'*)  
**also note** *eq*  
**also have**  $\text{ennreal } (\text{inverse } a) * \text{ennreal } (1 - \text{pmf } p \ y) = \text{ennreal } ((1 - \text{pmf } p \ y) / a)$   
**using**  $a$  **by** (*subst ennreal-mult' [symmetric]*) (*auto simp: field-simps*)  
**also have**  $(1 - \text{pmf } p \ y) / a = 1$  **using**  $y$  **by** (*simp add: a-def*)  
**finally show**  $(\int^+ x. \text{ennreal } (\text{if } x = y \text{ then } 0 \text{ else } \text{pmf } p \ x / a) \ \partial \text{count-space UNIV}) = 1$   
**by** *simp*  
**qed** (*insert a, auto*)  
**have**  $\text{mix-pmf } (1 - \text{pmf } p \ y) \ q \ (\text{return-pmf } y) = p$   
**using**  $y$  **by** (*intro pmf-eqI*) (*auto simp: q pmf-mix pmf-le-1 a-def*)  
**moreover have**  $\text{set-pmf } q = \text{set-pmf } p - \{y\}$   
**using**  $y$  **by** (*auto simp: q set-pmf-eq a-def*)  
**ultimately show** *?thesis* **using** *that*[*of*  $1 - \text{pmf } p \ y \ q$ ]  $y$  **assms** **by** (*auto simp: set-pmf-eq*)  
**qed**

**lemma** *pmf-mix-induct* [*consumes 2, case-names degenerate mix*]:

**assumes** *finite*  $A$   $\text{set-pmf } p \subseteq A$   
**assumes** *degenerate*:  $\bigwedge x. x \in A \implies P \ (\text{return-pmf } x)$   
**assumes** *mix*:  $\bigwedge p \ a \ y. \text{set-pmf } p \subseteq A \implies a \in \{0 < .. < 1\} \implies y \in A \implies P \ p \implies P \ (\text{mix-pmf } a \ p \ (\text{return-pmf } y))$   
**shows**  $P \ p$   
**proof** –  
**have** *finite* ( $\text{set-pmf } p$ )  $\text{set-pmf } p \neq \{\}$   $\text{set-pmf } p \subseteq A$   
**using** *assms*(1,2) **by** (*auto simp: set-pmf-not-empty dest: finite-subset*)  
**thus** *?thesis*  
**proof** (*induction set-pmf p arbitrary: p rule: finite-ne-induct*)  
**case** (*singleton*  $x \ p$ )  
**hence**  $p = \text{return-pmf } x$  **using** *set-pmf-subset-singleton*[*of*  $p \ x$ ] **by** *auto*  
**thus** *?case* **using** *singleton* **by** (*auto intro: degenerate*)  
**next**  
**case** (*insert*  $x \ B \ p$ )  
**from** *insert.hyps* **have**  $x \in \text{set-pmf } p$   $\text{set-pmf } p - \{x\} \neq \{\}$  **by** *auto*  
**from** *mix-pmf-partition*[*OF this*] **obtain**  $a \ q$   
**where** *decomp*:  $a \in \{0 < .. < 1\}$   $\text{set-pmf } q = \text{set-pmf } p - \{x\}$   
 $p = \text{mix-pmf } a \ q \ (\text{return-pmf } x)$  **by** *blast*  
**have**  $P \ (\text{mix-pmf } a \ q \ (\text{return-pmf } x))$   
**using** *insert.prem*s *decomp*(1) *insert.hyps*  
**by** (*intro mix insert*) (*auto simp: decomp*(2))  
**with** *decomp*(3) **show** *?case* **by** *simp*  
**qed**  
**qed**



**lemma** *pmf-mix-induct'* [*consumes 2, case-names degenerate mix*]:  
**assumes** *finite A set-pmf p*  $\subseteq A$   
**assumes** *degenerate*:  $\bigwedge x. x \in A \implies P (\text{return-pmf } x)$   
**assumes** *mix*:  $\bigwedge p q a. \text{set-pmf } p \subseteq A \implies \text{set-pmf } q \subseteq A \implies a \in \{0 < .. < 1\}$   
 $\implies$   

$$P p \implies P q \implies P (\text{mix-pmf } a p q)$$
  
**shows**  $P p$   
**using** *assms* **by** (*induct p rule: pmf-mix-induct*)(*auto*)+

**lemma** *finite-sum-distribute-mix-pmf*:  
**assumes** *finite (set-pmf (mix-pmf a p q))*  
**assumes** *finite (set-pmf p)*  
**assumes** *finite (set-pmf q)*  
**shows**  $(\sum i \in \text{set-pmf } (\text{mix-pmf } a p q). \text{pmf } (\text{mix-pmf } a p q) i) = (\sum i \in \text{set-pmf } p. a * \text{pmf } p i) + (\sum i \in \text{set-pmf } q. (1-a) * \text{pmf } q i)$   
**proof** –  
**have** *fst*:  $(\sum i \in \text{set-pmf } (\text{mix-pmf } a p q). \text{pmf } (\text{mix-pmf } a p q) i) = 1$   
**using** *sum-pmf-eq-1 assms* **by** *auto*  
**have**  $(\sum i \in \text{set-pmf } p. a * \text{pmf } p i) = a * (\sum i \in \text{set-pmf } p. \text{pmf } p i)$   
**by** (*simp add: sum-distrib-left*)  
**also have**  $\dots = a * 1$   
**using** *assms sum-pmf-eq-1* **by** (*simp add: sum-pmf-eq-1*)  
**then show** *?thesis*  
**by** (*metis fst add.assoc add-diff-cancel-left' add-uminus-conv-diff assms(3)*  
*mult.right-neutral order-refl sum-distrib-left sum-pmf-eq-1*)  
**qed**

**lemma** *distribute-alpha-over-sum*:  
**shows**  $(\sum i \in \text{set-pmf } T. a * \text{pmf } p i * f i) = a * (\sum i \in \text{set-pmf } T. \text{pmf } p i * f i)$   
**by** (*metis (mono-tags, lifting) semiring-normalization-rules(18) sum.cong sum-distrib-left*)

**lemma** *sum-over-subset-pmf-support*:  
**assumes** *finite T*  
**assumes** *set-pmf p*  $\subseteq T$   
**shows**  $(\sum i \in T. a * \text{pmf } p i * f i) = (\sum i \in \text{set-pmf } p. a * \text{pmf } p i * f i)$   
**proof** –  
**consider** (*eq*) *set-pmf p = T* | (*sub*) *set-pmf p*  $\subset T$   
**using** *assms* **by** *blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**next**  
**case** *sub*  
**define** *A* **where**  $A = T - (\text{set-pmf } p)$   
**have** *finite (set-pmf p)*  
**using** *assms(1) assms(2) finite-subset* **by** *auto*  
**moreover have** *finite A*  
**using** *A-def assms(1)* **by** *blast*  
**moreover have**  $A \cap \text{set-pmf } p = \{\}$   
**using** *A-def assms(1)* **by** *blast*

**ultimately have** \*:  $(\sum i \in T. a * \text{pmf } p \ i * f \ i) = (\sum i \in \text{set-pmf } p. a * \text{pmf } p \ i * f \ i) + (\sum i \in A. a * \text{pmf } p \ i * f \ i)$   
**using** *sum.union-disjoint* **by** (*metis (no-types) A-def Un-Diff-cancel2 Un-absorb2 assms(2) inf.commute inf-sup-aci(5) sum.union-disjoint*)  
**have**  $\forall e \in A. \text{pmf } p \ e = 0$   
**by** (*simp add: A-def pmf-eq-0-set-pmf*)  
**hence**  $(\sum i \in A. a * \text{pmf } p \ i * f \ i) = 0$   
**by** *simp*  
**then show** *?thesis*  
**by** (*simp add: \**)  
**qed** (*auto*)  
**qed**

**lemma** *expected-value-mix-pmf-distrib*:

**assumes** *finite (set-pmf p)*  
**and** *finite (set-pmf q)*  
**assumes**  $a \in \{0 < .. < 1\}$   
**shows** *measure-pmf.expectation (mix-pmf a p q) f = a \* measure-pmf.expectation p f + (1-a) \* measure-pmf.expectation q f*  
**proof** –  
**have** *fn: finite (set-pmf (mix-pmf a p q))*  
**using** *mix-pmf-preserves-finite-support assms less-eq-real-def* **by** *auto*  
**have** *subsets: set-pmf p  $\subseteq$  set-pmf (mix-pmf a p q) set-pmf q  $\subseteq$  set-pmf (mix-pmf a p q)*  
**using** *assms assms set-pmf-mix* **by**(*fastforce*)+  
**have** \*:  $(\sum i \in \text{set-pmf (mix-pmf a p q). pmf } p \ i * f \ i) = a * (\sum i \in \text{set-pmf (mix-pmf a p q). pmf } p \ i * f \ i)$   
**by** (*metis (mono-tags, lifting) mult.assoc sum.cong sum-distrib-left*)  
**have** \*\*:  $(\sum i \in \text{set-pmf (mix-pmf a p q). (1-a) * pmf } q \ i * f \ i) = (1-a) * (\sum i \in \text{set-pmf (mix-pmf a p q). pmf } q \ i * f \ i)$   
**using** *distribute-alpha-over-sum[of (1 - a) q f (mix-pmf a p q)]* **by** *auto*  
**have** \*\*\*: *measure-pmf.expectation (mix-pmf a p q) f = ( $\sum i \in \text{set-pmf (mix-pmf a p q). pmf (mix-pmf a p q) i * f i$ )*  
**by** (*metis fn pmf-integral-code-unfold pmf-integral-pmf-set-integral pmf-set-integral-code-alt-finite*)  
**also have** *g: ... = ( $\sum i \in \text{set-pmf (mix-pmf a p q). (a * pmf } p \ i + (1-a) * pmf } q \ i) * f \ i$ )*  
**using** *pmf-mix[of a p q] assms(3)* **by** *auto*  
**also have** \*\*\*\*:  $\dots = (\sum i \in \text{set-pmf (mix-pmf a p q). a * pmf } p \ i * f \ i + (1-a) * pmf } q \ i * f \ i)$   
**by** (*simp add: mult.commute ring-class.ring-distrib(1)*)  
**also have** *f: ... = ( $\sum i \in \text{set-pmf (mix-pmf a p q). a * pmf } p \ i * f \ i$ ) + ( $\sum i \in \text{set-pmf (mix-pmf a p q). (1-a) * pmf } q \ i * f \ i$ )*  
**by** (*simp add: sum.distrib*)  
**also have**  $\dots = a * (\sum i \in \text{set-pmf (mix-pmf a p q). pmf } p \ i * f \ i) + (1-a) * (\sum i \in \text{set-pmf (mix-pmf a p q). pmf } q \ i * f \ i)$   
**using** \* \*\* **by** *simp*  
**also have** *h: ... = a \* ( $\sum i \in \text{set-pmf } p. pmf } p \ i * f \ i$ ) + (1-a) \* ( $\sum i \in \text{set-pmf } q. pmf } q \ i * f \ i$ )*

```

proof –
  have  $(\sum i \in \text{set-pmf } (\text{mix-pmf } a \ p \ q). \text{pmf } p \ i * f \ i) = (\sum i \in \text{set-pmf } p. \text{pmf } p \ i * f \ i)$ 
  using subsets sum-over-subset-pmf-support[of  $(\text{mix-pmf } a \ p \ q) \ p \ 1 \ f$ ] fn by
  auto
  moreover have  $(\sum i \in \text{set-pmf } (\text{mix-pmf } a \ p \ q). \text{pmf } q \ i * f \ i) = (\sum i \in \text{set-pmf } q. \text{pmf } q \ i * f \ i)$ 
  using subsets sum-over-subset-pmf-support[of  $(\text{mix-pmf } a \ p \ q) \ q \ 1 \ f$ ] fn by
  auto
  ultimately show ?thesis
  by (simp)
qed
finally show ?thesis
proof –
  have  $(\sum i \in \text{set-pmf } q. \text{pmf } q \ i * f \ i) = \text{measure-pmf.expectation } q \ f$ 
  by (metis (full-types) assms(2) pmf-integral-code-unfold pmf-integral-pmf-set-integral pmf-set-integral-code-alt-finite)
  moreover have  $(\sum i \in \text{set-pmf } p. \text{pmf } p \ i * f \ i) = \text{measure-pmf.expectation } p \ f$ 
  by (metis (full-types) assms(1) pmf-integral-code-unfold pmf-integral-pmf-set-integral pmf-set-integral-code-alt-finite)
  ultimately show ?thesis
  by (simp add: * ** *** **** f g h)
qed
qed

```

```

lemma expected-value-mix-pmf:
  assumes finite (set-pmf p)
  and finite (set-pmf q)
  assumes  $a \in \{0..1\}$ 
  shows  $\text{measure-pmf.expectation } (\text{mix-pmf } a \ p \ q) \ f = a * \text{measure-pmf.expectation } p \ f + (1-a) * \text{measure-pmf.expectation } q \ f$ 
proof –
  consider  $(0) \ a = 0 \mid (b) \ a \in \{0 <.. < 1\} \mid (1) \ a = 1$ 
  using assms(3) less-eq-real-def by auto
  then show ?thesis
proof (cases)
  case 0
  have  $(\text{mix-pmf } a \ p \ q) = q$ 
  using 0 pmf-mix-0 by blast
  have  $\text{measure-pmf.expectation } q \ f = (1-a) * \text{measure-pmf.expectation } q \ f$ 
  by (simp add: 0)
  show ?thesis
  using 0 by auto
next
  case b
  show ?thesis
  using assms(1) assms(2) b expected-value-mix-pmf-distrib by blast
next
  case 1

```

```

    have (mix-pmf a p q) = p
      using 1 pmf-mix-0 by simp
    then show ?thesis
      by (simp add: 1)
  qed
qed
end

```

```

theory Lotteries
  imports
    PMF-Composition
    HOL-Probability.Probability
begin

```

## 2 Lotteries

```

definition lotteries-on
  where
    lotteries-on Oc = {p . (set-pmf p) ⊆ Oc}

```

```

lemma lotteries-on-subset:
  assumes A ⊆ B
  shows lotteries-on A ⊆ lotteries-on B
  by (metis (no-types, lifting) Collect-mono assms gfp.leq-trans lotteries-on-def)

```

```

lemma support-in-outcomes:
  ∀ oc. ∀ p ∈ lotteries-on oc. ∀ a ∈ set-pmf p. a ∈ oc
  by (simp add: lotteries-on-def subsetD)

```

```

lemma lotteries-on-nonempty:
  assumes outcomes ≠ {}
  shows lotteries-on outcomes ≠ {}
  by (auto simp: lotteries-on-def) (metis (full-types) assms
    empty-subsetI ex-in-conv insert-subset set-return-pmf)

```

```

lemma finite-support-one-oc:
  assumes card outcomes = 1
  shows ∀ l ∈ lotteries-on outcomes. finite (set-pmf l)
  by (metis assms card.infinite finite-subset lotteries-on-def mem-Collect-eq zero-neq-one)

```

```

lemma one-outcome-card-support-1:
  assumes card outcomes = 1
  shows ∀ l ∈ lotteries-on outcomes. card (set-pmf l) = 1
proof
  fix l
  assume l ∈ lotteries-on outcomes

```

**have** *finite outcomes*  
**using** *assms card.infinite* **by** *force*  
**then have**  $l \in \text{lotteries-on outcomes} \longrightarrow 1 = \text{card} (\text{set-pmf } l)$   
**by** (*metis assms card-eq-0-iff card-mono finite-support-one-oc le-neq-implies-less*  
*less-one lotteries-on-def mem-Collect-eq set-pmf-not-empty*)  
**then show**  $\text{card} (\text{set-pmf } l) = 1$   
**by** (*simp add: ⟨l ∈ lotteries-on outcomes⟩*)  
**qed**

**lemma** *finite-nempty-ex-degenerate-in-lotteries:*  
**assumes**  $\text{out} \neq \{\}$   
**assumes** *finite out*  
**shows**  $\exists e \in \text{lotteries-on out}. \exists x \in \text{out}. \text{pmf } e \ x = 1$   
**proof** (*rule ccontr*)  
**assume**  $a: \neg (\exists e \in \text{lotteries-on out}. \exists x \in \text{out}. \text{pmf } e \ x = 1)$   
**then have** *subset:  $\forall e \in \text{lotteries-on out}. \text{set-pmf } e \subseteq \text{out}$*   
**using** *lotteries-on-def* **by** (*simp add: lotteries-on-def*)  
**then have**  $\forall e. e \in \text{lotteries-on out} \longrightarrow ((\sum_{i \in \text{set-pmf } e}. \text{pmf } e \ i) = 1)$   
**using** *sum-pmf-eq-1* **by** (*metis subset assms(2) finite-subset order-refl*)  
**then show** *False*  
**by** (*metis (no-types, lifting) a assms(1) assms(2) card.empty card-gt-0-iff*  
*card-seteq*  
*empty-subsetI finite.emptyI finite-insert insert-subset lotteries-on-def subsetI*  
*measure-measure-pmf-finite mem-Collect-eq nat-less-le pmf.rep-eq set-pmf-of-set*  
*)*  
**qed**

**lemma** *card-support-1-probability-1:*  
**assumes**  $\text{card} (\text{set-pmf } p) = 1$   
**shows**  $\forall e \in \text{set-pmf } p. \text{pmf } p \ e = 1$   
**by**(*auto*) (*metis assms card-1-singletonE card-ge-0-finite*  
*card-subset-eq ex-card le-numeral-extra(4) measure-measure-pmf-finite*  
*pmf.rep-eq singletonD sum-pmf-eq-1 zero-less-one*)

**lemma** *one-outcome-card-lotteries-1:*  
**assumes**  $\text{card outcomes} = 1$   
**shows**  $\text{card} (\text{lotteries-on outcomes}) = 1$   
**proof** –  
**obtain**  $x :: 'a$  **where**  
 $x: \text{outcomes} = \{x\}$   
**using** *assms card-1-singletonE* **by** *blast*  
**have** *exl:  $\exists l \in \text{lotteries-on outcomes}. \text{pmf } l \ x = 1$*   
**by** (*metis x assms card.infinite empty-iff*  
*finite-nempty-ex-degenerate-in-lotteries insert-iff zero-neq-one*)  
**have** *pmfs:  $\forall l \in \text{lotteries-on outcomes}. \text{set-pmf } l = \{x\}$*   
**by** (*simp add: lotteries-on-def set-pmf-subset-singleton x*)  
**have**  $\forall l \in \text{lotteries-on outcomes}. \text{pmf } l \ x = 1$   
**by** (*simp add: lotteries-on-def set-pmf-subset-singleton x*)

**then show** *?thesis*  
**by** (*metis exl empty-iff is-singletonI' is-singleton-altdef*  
*order-refl pmfs set-pmf-subset-singleton*)  
**qed**

**lemma** *return-pmf-card-equals-set*:  
**shows**  $\text{card } \{\text{return-pmf } x \mid x. x \in S\} = \text{card } S$   
**proof** –  
**have**  $\{\text{return-pmf } x \mid x. x \in S\} = \text{return-pmf } ' S$   
**by** *blast*  
**also have**  $\text{card } \dots = \text{card } S$   
**by** (*intro card-image*) (*auto simp: inj-on-def*)  
**finally show**  $\text{card } \{\text{return-pmf } x \mid x. x \in S\} = \text{card } S .$   
**qed**

**lemma** *mix-pmf-in-lotteries*:  
**assumes**  $p \in \text{lotteries-on } A$   
**and**  $q \in \text{lotteries-on } A$   
**and**  $a \in \{0 < .. < 1\}$   
**shows**  $(\text{mix-pmf } a \ p \ q) \in \text{lotteries-on } A$   
**proof** –  
**have**  $\text{set-pmf } (\text{mix-pmf } a \ p \ q) = \text{set-pmf } p \cup \text{set-pmf } q$   
**by** (*meson assms(3) set-pmf-mix*)  
**then show** *?thesis*  
**by** (*metis Un-subset-iff assms(1) assms(2) lotteries-on-def mem-Collect-eq*)  
**qed**

**lemma** *card-degen-lotteries-equals-outcomes*:  
**shows**  $\text{card } \{x \in \text{lotteries-on } \text{out}. \text{card } (\text{set-pmf } x) = 1\} = \text{card } \text{out}$   
**proof** –  
**consider**  $(\text{empty}) \ \text{out} = \{\} \mid (\text{not-empty}) \ \text{out} \neq \{\}$   
**by** *blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** *not-empty*  
**define** *DG* **where**  
 $DG: DG = \{x \in \text{lotteries-on } \text{out}. \text{card } (\text{set-pmf } x) = 1\}$   
**define** *AP* **where**  
 $AP: AP = \{\text{return-pmf } x \mid x. x \in \text{out}\}$   
**have**  $** : \text{card } AP = \text{card } \text{out}$   
**using** *AP return-pmf-card-equals-set* **by** *blast*  
**have**  $* : \forall d \in DG. d \in AP$   
**proof**  
**fix**  $l$   
**assume**  $l \in DG$   
**then have**  $l \in \text{lotteries-on } \text{out} \wedge 1 = \text{card } (\text{set-pmf } l)$   
**using** *DG* **by** *force*  
**then obtain**  $x$  **where**  
 $x : x \in \text{out} \ \text{set-pmf } l = \{x\}$

```

    by (metis (no-types) card-1-singletonE singletonI support-in-outcomes)
  have return-pmf x = l
    using set-pmf-subset-singleton x(2) by fastforce
  then show l ∈ AP
    using AP x(1) by blast
qed
moreover have AP = DG
proof
  have ∀ e ∈ AP. e ∈ lotteries-on out
    by(auto simp: lotteries-on-def AP)
  then show AP ⊆ DG using DG AP by force
qed (auto simp: *)
ultimately show ?thesis
  using DG ** by blast
qed (simp add: lotteries-on-def set-pmf-not-empty)
qed

end

```

```

theory Neumann-Morgenstern-Utility-Theorem
  imports
    HOL-Probability.Probability
    First-Welfare-Theorem.Utility-Functions
    Lotteries
begin

```

## 3 Properties of Preferences

### 3.1 Independent Preferences

Independence is sometimes called substitution

Notice how  $r$  is "added" to the right of  $\text{mix-pmf}$  and the element to the left  $q/p$  changes

**definition** *independent-vnm*  
**where**

```

independent-vnm C P =
  (∀ p ∈ C. ∀ q ∈ C. ∀ r ∈ C. ∀ (α::real) ∈ {0<..1}. p ⪰[P] q ⟷ mix-pmf α p
r ⪰[P] mix-pmf α q r)

```

**lemma** *independent-vnmI1:*

```

assumes (∀ p ∈ C. ∀ q ∈ C. ∀ r ∈ C. ∀ α ∈ {0<..1}. p ⪰[P] q ⟷ mix-pmf α
p r ⪰[P] mix-pmf α q r)
shows independent-vnm C P
using assms independent-vnm-def by blast

```

**lemma** *independent-vnmI2*:  
**assumes**  $\bigwedge p q r \alpha. p \in C \implies q \in C \implies r \in C \implies \alpha \in \{0 < .. 1\} \implies p \succeq[P]$   
 $q \longleftrightarrow \text{mix-pmf } \alpha p r \succeq[P] \text{mix-pmf } \alpha q r$   
**shows** *independent-vnm*  $C P$   
**by** (*rule independent-vnmI1*, *standard*, *standard*, *standard*,  
*standard*, *simp add: assms*) (*meson assms greaterThanAtMost-iff*)

**lemma** *independent-vnm-alt-def*:  
**shows** *independent-vnm*  $C P \longleftrightarrow (\forall p \in C. \forall q \in C. \forall r \in C. \forall \alpha \in \{0 < .. < 1\}.$   
 $p \succeq[P] q \longleftrightarrow \text{mix-pmf } \alpha p r \succeq[P] \text{mix-pmf } \alpha q r)$  (**is**  $?L \longleftrightarrow ?R$ )  
**proof** (*rule iffI*)  
**assume**  $a: ?R$   
**have** *independent-vnm*  $C P$   
**by**(*rule independent-vnmI2*, *simp add: a*) (*metis a greaterThanLessThan-iff*  
*linorder-neqE-linordered-idom not-le pmf-mix-1*)  
**then show**  $?L$  **by** *auto*  
**qed** (*simp add: independent-vnm-def*)

**lemma** *independece-dest-alt*:  
**assumes** *independent-vnm*  $C P$   
**shows**  $(\forall p \in C. \forall q \in C. \forall r \in C. \forall (\alpha :: \text{real}) \in \{0 < .. 1\}. p \succeq[P] q \longleftrightarrow \text{mix-pmf}$   
 $\alpha p r \succeq[P] \text{mix-pmf } \alpha q r)$   
**proof** (*standard*, *standard*, *standard*, *standard*)  
**fix**  $p q r \alpha$   
**assume**  $as1: p \in C$   
**assume**  $as2: q \in C$   
**assume**  $as3: r \in C$   
**assume**  $as4: (\alpha :: \text{real}) \in \{0 < .. 1\}$   
**then show**  $p \succeq[P] q = \text{mix-pmf } \alpha p r \succeq[P] \text{mix-pmf } \alpha q r$   
**using**  $as1 as2 as3$  *assms(1)* *independent-vnm-def* **by** *blast*  
**qed**

**lemma** *independent-vnmD1*:  
**assumes** *independent-vnm*  $C P$   
**shows**  $(\forall p \in C. \forall q \in C. \forall r \in C. \forall \alpha \in \{0 < .. 1\}. p \succeq[P] q \longleftrightarrow \text{mix-pmf } \alpha p r$   
 $\succeq[P] \text{mix-pmf } \alpha q r)$   
**using** *assms independent-vnm-def* **by** *blast*

**lemma** *independent-vnmD2*:  
**fixes**  $p q r \alpha$   
**assumes**  $\alpha \in \{0 < .. 1\}$   
**and**  $p \in C$   
**and**  $q \in C$   
**and**  $r \in C$   
**assumes** *independent-vnm*  $C P$   
**assumes**  $p \succeq[P] q$   
**shows**  $\text{mix-pmf } \alpha p r \succeq[P] \text{mix-pmf } \alpha q r$   
**using** *assms independece-dest-alt* **by** *blast*



**lemma** *independent-vnmD3*:

**fixes**  $p\ q\ r\ \alpha$   
**assumes**  $\alpha \in \{0 < .. 1\}$   
**and**  $p \in C$   
**and**  $q \in C$   
**and**  $r \in C$   
**assumes** *independent-vnm*  $C\ P$   
**assumes** *mix-pmf*  $\alpha\ p\ r \succeq[P]\ \text{mix-pmf}\ \alpha\ q\ r$   
**shows**  $p \succeq[P]\ q$   
**using** *assms indepedece-dest-alt* **by** *blast*

**lemma** *independent-vnmD4*:

**assumes** *independent-vnm*  $C\ P$   
**assumes** *refl-on*  $C\ P$   
**assumes**  $p \in C$   
**and**  $q \in C$   
**and**  $r \in C$   
**and**  $\alpha \in \{0..1\}$   
**and**  $p \succeq[P]\ q$   
**shows** *mix-pmf*  $\alpha\ p\ r \succeq[P]\ \text{mix-pmf}\ \alpha\ q\ r$   
**using** *assms*  
**by** (*cases*  $\alpha = 0 \vee \alpha \in \{0 < .. 1\}$ , *metis* *assms(1,2,3,4)*  
*indepdece-dest-alt pmf-mix-0 refl-onD, auto*)

**lemma** *approx-indep-ge*:

**assumes**  $x \approx[\mathcal{R}]\ y$   
**assumes**  $\alpha \in \{0..(1::\text{real})\}$   
**assumes** *rpr*: *rational-preference* (*lotteries-on outcomes*)  $\mathcal{R}$   
**and** *ind*: *independent-vnm* (*lotteries-on outcomes*)  $\mathcal{R}$   
**shows**  $\forall r \in \text{lotteries-on outcomes. } (\text{mix-pmf}\ \alpha\ y\ r) \succeq[\mathcal{R}]\ (\text{mix-pmf}\ \alpha\ x\ r)$   
**proof**  
**fix**  $r$   
**assume**  $a$ :  $r \in \text{lotteries-on outcomes}$  (**is**  $r \in ?lo$ )  
**have** *clct*:  $y \succeq[\mathcal{R}]\ x \wedge \text{independent-vnm}\ ?lo\ \mathcal{R} \wedge y \in ?lo \wedge x \in ?lo \wedge r \in ?lo$   
**by** (*meson* *a* *assms(1)* *assms(2)* *atLeastAtMost-iff greaterThanAtMost-iff*  
*ind preference-def rational-preference-def rpr*)  
**then have** *in-lo*:  $\text{mix-pmf}\ \alpha\ y\ r \in ?lo \ (\text{mix-pmf}\ \alpha\ x\ r) \in ?lo$   
**by** (*metis* *assms(2)* *atLeastAtMost-iff greaterThanLessThan-iff*  
*less-eq-real-def mix-pmf-in-lotteries pmf-mix-0 pmf-mix-1 a*)  
**have**  $0 = \alpha \vee 0 < \alpha$   
**using** *assms* **by** *auto*  
**then show**  $\text{mix-pmf}\ \alpha\ y\ r \succeq[\mathcal{R}]\ \text{mix-pmf}\ \alpha\ x\ r$   
**using** *in-lo(2)* *rational-preference.compl rpr*  
**by** (*auto,blast*) (*meson* *assms(2)* *atLeastAtMost-iff clct*  
*greaterThanAtMost-iff independent-vnmD2*)  
**qed**

**lemma** *approx-imp-approx-ind*:

**assumes**  $x \approx[\mathcal{R}] y$   
**assumes**  $\alpha \in \{0..(1::real)\}$   
**assumes**  $rpr$ : rational-preference (lotteries-on outcomes)  $\mathcal{R}$   
**and**  $ind$ : independent-vnm (lotteries-on outcomes)  $\mathcal{R}$   
**shows**  $\forall r \in \text{lotteries-on outcomes. } (mix\text{-pmf } \alpha y r) \approx[\mathcal{R}] (mix\text{-pmf } \alpha x r)$   
**using** *approx-indep-ge* *assms(1)* *assms(2)* *ind* *rpr* **by** *blast*

**lemma** *geq-imp-mix-geq-right*:

**assumes**  $x \succeq[\mathcal{R}] y$   
**assumes**  $rpr$ : rational-preference (lotteries-on outcomes)  $\mathcal{R}$   
**assumes**  $ind$ : independent-vnm (lotteries-on outcomes)  $\mathcal{R}$   
**assumes**  $\alpha \in \{0..(1::real)\}$   
**shows**  $(mix\text{-pmf } \alpha x y) \succeq[\mathcal{R}] y$

**proof** –

**have**  $xy\text{-p}$ :  $x \in (\text{lotteries-on outcomes}) y \in (\text{lotteries-on outcomes})$   
**by** (*meson* *assms(1)* *preference.not-outside* *rational-preference-def* *rpr*)  
*(meson* *assms(1)* *preference-def* *rational-preference-def* *rpr*)  
**have**  $(mix\text{-pmf } \alpha x y) \in (\text{lotteries-on outcomes})$  (**is**  $?mpf \in ?lot$ )  
**using** *mix-pmf-in-lotteries* [*of x outcomes y*  $\alpha$ ] *xy-p* *assms(2)*  
**by** (*meson* *approx-indep-ge* *assms(4)* *ind* *preference.not-outside*  
*rational-preference.compl* *rational-preference-def*)  
**have**  $all$ :  $\forall r \in ?lot. (mix\text{-pmf } \alpha x r) \succeq[\mathcal{R}] (mix\text{-pmf } \alpha y r)$   
**by** (*metis* *assms* *assms(2)* *atLeastAtMost-iff* *greaterThanAtMost-iff* *independ-*  
*dece-dest-alt*  
*less-eq-real-def* *pmf-mix-0* *rational-preference.compl* *rpr* *ind* *xy-p*)  
**thus**  $?thesis$   
**by** (*metis* *all* *assms(4)* *set-pmf-mix-eq* *xy-p(2)*)

**qed**

**lemma** *geq-imp-mix-geq-left*:

**assumes**  $x \succeq[\mathcal{R}] y$   
**assumes**  $rpr$ : rational-preference (lotteries-on outcomes)  $\mathcal{R}$   
**assumes**  $ind$ : independent-vnm (lotteries-on outcomes)  $\mathcal{R}$   
**assumes**  $\alpha \in \{0..(1::real)\}$   
**shows**  $(mix\text{-pmf } \alpha y x) \succeq[\mathcal{R}] y$

**proof** –

**define**  $\beta$  **where**  
 $b$ :  $\beta = 1 - \alpha$   
**have**  $\beta \in \{0..1\}$   
**using** *assms(4)*  $b$  **by** *auto*  
**then** **have**  $mix\text{-pmf } \beta x y \succeq[\mathcal{R}] y$   
**using** *geq-imp-mix-geq-right*[*OF* *assms*] *assms(1)* *geq-imp-mix-geq-right* *ind* *rpr*  
**by** *blast*  
**moreover** **have**  $mix\text{-pmf } \beta x y = mix\text{-pmf } \alpha y x$   
**by** (*metis* *assms(4)*  $b$  *pmf-inverse-switch-equals*)  
**ultimately** **show**  $?thesis$   
**by** *simp*  
**qed**

**lemma** *sg-imp-mix-sg*:  
**assumes**  $x \succ[\mathcal{R}] y$   
**assumes** *rpr*: *rational-preference (lotteries-on outcomes)  $\mathcal{R}$*   
**assumes** *ind*: *independent-vnm (lotteries-on outcomes)  $\mathcal{R}$*   
**assumes**  $\alpha \in \{0..1\}$   
**shows**  $(\text{mix-pmf } \alpha x y) \succ[\mathcal{R}] y$   
**proof** –  
**have**  $xy\text{-}p$ :  $x \in (\text{lotteries-on outcomes}) y \in (\text{lotteries-on outcomes})$   
**by** (*meson assms(1) preference.not-outside rational-preference-def rpr*)  
*(meson assms(1) preference-def rational-preference-def rpr)*  
**have**  $(\text{mix-pmf } \alpha x y) \in (\text{lotteries-on outcomes})$  (**is**  $?mpf \in ?lot$ )  
**using** *mix-pmf-in-lotteries [of x outcomes y  $\alpha$ ] xy-p assms(2)*  
**using** *assms(4)* **by** *fastforce*  
**have** *all*:  $\forall r \in ?lot. (\text{mix-pmf } \alpha x r) \succeq[\mathcal{R}] (\text{mix-pmf } \alpha y r)$   
**by** (*metis assms(1,3,4) independece-dest-alt ind xy-p*)  
**have**  $(\text{mix-pmf } \alpha x y) \succeq[\mathcal{R}] y$   
**by** (*metis all assms(4) atLeastAtMost-iff greaterThanAtMost-iff less-eq-real-def set-pmf-mix-eq xy-p(2)*)  
**have** *all2*:  $\forall r \in ?lot. (\text{mix-pmf } \alpha x r) \succ[\mathcal{R}] (\text{mix-pmf } \alpha y r)$   
**using** *assms(1) assms(4) ind independece-dest-alt xy-p(1) xy-p(2)* **by** *blast*  
**then show** *?thesis*  
**by** (*metis assms(4) atLeastAtMost-iff greaterThanAtMost-iff less-eq-real-def set-pmf-mix-eq xy-p(2)*)  
**qed**

### 3.2 Continuity

Continuity is sometimes called Archimedean Axiom

**definition** *continuous-vnm*

**where**

*continuous-vnm*  $C P = (\forall p \in C. \forall q \in C. \forall r \in C. p \succeq[P] q \wedge q \succeq[P] r \longrightarrow (\exists \alpha \in \{0..1\}. (\text{mix-pmf } \alpha p r) \approx[P] q))$

**lemma** *continuous-vnmD*:

**assumes** *continuous-vnm*  $C P$

**shows**  $(\forall p \in C. \forall q \in C. \forall r \in C. p \succeq[P] q \wedge q \succeq[P] r \longrightarrow$

$(\exists \alpha \in \{0..1\}. (\text{mix-pmf } \alpha p r) \approx[P] q))$

**using** *continuous-vnm-def assms* **by** *blast*

**lemma** *continuous-vnmI*:

**assumes**  $\bigwedge p q r. p \in C \implies q \in C \implies r \in C \implies p \succeq[P] q \wedge q \succeq[P] r \implies$

$\exists \alpha \in \{0..1\}. (\text{mix-pmf } \alpha p r) \approx[P] q$

**shows** *continuous-vnm*  $C P$

**by** (*simp add: assms continuous-vnm-def*)

**lemma** *mix-in-lot*:

**assumes**  $x \in \text{lotteries-on outcomes}$

**and**  $y \in \text{lotteries-on outcomes}$

**and**  $\alpha \in \{0..1\}$

**shows**  $(\text{mix-pmf } \alpha \ x \ y) \in \text{lotteries-on outcomes}$   
**using**  $\text{assms}(1) \ \text{assms}(2) \ \text{assms}(3) \ \text{less-eq-real-def} \ \text{mix-pmf-in-lotteries}$  **by** *fast-force*

**lemma** *non-unique-continuous-unfolding:*

**assumes**  $\text{cnt: continuous-vnm (lotteries-on outcomes) } \mathcal{R}$   
**assumes**  $\text{rational-preference (lotteries-on outcomes) } \mathcal{R}$   
**assumes**  $p \succeq[\mathcal{R}] q$   
**and**  $q \succeq[\mathcal{R}] r$   
**and**  $p \succ[\mathcal{R}] r$   
**shows**  $\exists \alpha \in \{0..1\}. q \approx[\mathcal{R}] \text{mix-pmf } \alpha \ p \ r$   
**using**  $\text{assms}(1) \ \text{assms}(2) \ \text{cnt} \ \text{continuous-vnmD} \ \text{assms}$   
**proof** –  
**have**  $\forall p \ q. p \in (\text{lotteries-on outcomes}) \wedge q \in (\text{lotteries-on outcomes}) \iff p \succeq[\mathcal{R}] q \vee q \succeq[\mathcal{R}] p$   
**using**  $\text{assms} \ \text{rational-preference.compl[of lotteries-on outcomes } \mathcal{R}]$   
**by**  $(\text{metis (no-types, hide-lams) preference-def rational-preference-def})$   
**then show** *?thesis*  
**using**  $\text{continuous-vnmD[OF assms}(1)]$  **by**  $(\text{metis assms}(3) \ \text{assms}(4))$   
**qed**

## 4 System U start, as per vNM

These are the first two assumptions which we use to derive the first results. We assume rationality and independence. In this system U the von-Neumann-Morgenstern Utility Theorem is proven.

**context**

**fixes**  $\text{outcomes} :: 'a \ \text{set}$   
**fixes**  $\mathcal{R}$   
**assumes**  $\text{rpr: rational-preference (lotteries-on outcomes) } \mathcal{R}$   
**assumes**  $\text{ind: independent-vnm (lotteries-on outcomes) } \mathcal{R}$   
**begin**

**abbreviation**  $\mathcal{P} \equiv \text{lotteries-on outcomes}$

**lemma** *relation-in-carrier:*

$x \succeq[\mathcal{R}] y \implies x \in \mathcal{P} \wedge y \in \mathcal{P}$   
**by**  $(\text{meson preference-def rational-preference-def rpr})$

**lemma** *mix-pmf-preferred-independence:*

**assumes**  $r \in \mathcal{P}$   
**and**  $\alpha \in \{0..1\}$   
**assumes**  $p \succeq[\mathcal{R}] q$   
**shows**  $\text{mix-pmf } \alpha \ p \ r \succeq[\mathcal{R}] \text{mix-pmf } \alpha \ q \ r$   
**using**  $\text{ind}$  **by**  $(\text{metis relation-in-carrier antisym-conv1 assms atLeastAtMost-iff greaterThanAtMost-iff independece-dest-alt pmf-mix-0 rational-preference.no-better-thansubset-rel rpr subsetI})$

**lemma** *mix-pmf-strict-preferred-independence:*

**assumes**  $r \in \mathcal{P}$   
**and**  $\alpha \in \{0 < .. 1\}$   
**assumes**  $p \succ[\mathcal{R}] q$   
**shows**  $\text{mix-pmf } \alpha p r \succ[\mathcal{R}] \text{mix-pmf } \alpha q r$   
**by** (*meson* *assms(1)* *assms(2)* *assms(3)* *ind* *independent-vnmD2*  
*independent-vnmD3* *relation-in-carrier*)

**lemma** *mix-pmf-preferred-independence-rev:*

**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**and**  $r \in \mathcal{P}$   
**and**  $\alpha \in \{0 < .. 1\}$   
**assumes**  $\text{mix-pmf } \alpha p r \succeq[\mathcal{R}] \text{mix-pmf } \alpha q r$   
**shows**  $p \succeq[\mathcal{R}] q$   
**proof** –  
**have**  $\text{mix-pmf } \alpha p r \in \mathcal{P}$   
**using** *assms* *mix-in-lot* *relation-in-carrier* **by** *blast*  
**moreover** **have**  $\text{mix-pmf } \alpha q r \in \mathcal{P}$   
**using** *assms* *mix-in-lot* *assms(2)* *relation-in-carrier* **by** *blast*  
**ultimately show** *?thesis*  
**using** *ind* *independent-vnmD3*[*of*  $\alpha p \mathcal{P} q r \mathcal{R}$ ] *assms* **by** *blast*  
**qed**

**lemma** *x-sg-y-sg-mpmf-right:*

**assumes**  $x \succ[\mathcal{R}] y$   
**assumes**  $b \in \{0 < .. (1::\text{real})\}$   
**shows**  $x \succ[\mathcal{R}] \text{mix-pmf } b y x$   
**proof** –  
**consider**  $b = 1 \mid b \neq 1$   
**by** *blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** 2  
**have** *sg*:  $(\text{mix-pmf } b x y) \succ[\mathcal{R}] y$   
**using** *assms(1)* *assms(2)* *assms* *ind* *rpr* *sg-imp-mix-sg* 2 **by** *fastforce*  
**have**  $\text{mix-pmf } b x y \in \mathcal{P}$   
**by** (*meson* *sg* *preference-def* *rational-preference-def* *rpr*)  
**have**  $\text{mix-pmf } b x x \in \mathcal{P}$   
**using** *relation-in-carrier* *assms(2)* *mix-in-lot* *assms* **by** *fastforce*  
**have**  $b \in \{0 < .. < 1\}$   
**using** 2 *assms(2)* **by** *auto*  
**have**  $\text{mix-pmf } b x x \succ[\mathcal{R}] \text{mix-pmf } b y x$   
**using** *mix-pmf-preferred-independence*[*of*  $x b$ ] *assms*  
**by** (*meson*  $\langle b \in \{0 < .. < 1\} \rangle$  *greaterThanAtMost-iff* *greaterThanLessThan-iff*)  
*ind*  
*independece-dest-alt* *less-eq-real-def* *preference-def*  
*rational-preference.axioms(1)* *relation-in-carrier* *rpr*)

**then show** *?thesis*  
**using** *mix-pmf-preferred-independence*  
**by** (*metis* *assms(2)* *atLeastAtMost-iff* *greaterThanAtMost-iff* *less-eq-real-def* *set-pmf-mix-eq*)  
**qed** (*simp* *add: assms(1)*)  
**qed**

**lemma** *neumann-3B-b*:  
**assumes**  $u \succ[\mathcal{R}] v$   
**assumes**  $\alpha \in \{0 < .. < 1\}$   
**shows**  $u \succ[\mathcal{R}] \text{mix-pmf } \alpha u v$   
**proof** –  
**have**  $*$ : *preorder-on*  $\mathcal{P} \mathcal{R} \wedge$  *rational-preference-axioms*  $\mathcal{P} \mathcal{R}$   
**by** (*metis* (*no-types*) *preference-def* *rational-preference-def* *rpr*)  
**have**  $1 - \alpha \in \{0 < .. < 1\}$   
**using** *assms(2)* **by** *auto*  
**then show** *?thesis*  
**using**  $*$  *assms* **by** (*metis* *atLeastAtMost-iff* *greaterThanLessThan-iff* *less-eq-real-def* *pmf-inverse-switch-equals* *x-sg-y-sg-mpmf-right*)  
**qed**

**lemma** *neumann-3B-b-non-strict*:  
**assumes**  $u \succeq[\mathcal{R}] v$   
**assumes**  $\alpha \in \{0..1\}$   
**shows**  $u \succeq[\mathcal{R}] \text{mix-pmf } \alpha u v$   
**proof** –  
**have** *f2*:  $\text{mix-pmf } \alpha (u::'a \text{ pmf}) v = \text{mix-pmf } (1 - \alpha) v u$   
**using** *pmf-inverse-switch-equals* *assms(2)* **by** *auto*  
**have**  $1 - \alpha \in \{0..1\}$   
**using** *assms(2)* **by** *force*  
**then show** *?thesis*  
**using** *f2* *relation-in-carrier*  
**by** (*metis* (*no-types*) *assms(1)* *mix-pmf-preferred-independence* *set-pmf-mix-eq*)  
**qed**

**lemma** *greater-mix-pmf-greater-step-1-aux*:  
**assumes**  $v \succ[\mathcal{R}] u$   
**assumes**  $\alpha \in \{0 < .. < (1::\text{real})\}$   
**and**  $\beta \in \{0 < .. < (1::\text{real})\}$   
**assumes**  $\beta > \alpha$   
**shows**  $(\text{mix-pmf } \beta v u) \succ[\mathcal{R}] (\text{mix-pmf } \alpha v u)$   
**proof** –  
**define**  $t$  **where**  
 $t = \text{mix-pmf } \beta v u$   
**obtain**  $\gamma$  **where**  
 $g: \alpha = \beta * \gamma$   
**by** (*metis* *assms(2)* *assms(4)* *greaterThanLessThan-iff* *mult.commute* *nonzero-eq-divide-eq* *not-less-iff-gr-or-eq*)  
**have** *g1*:  $\gamma > 0 \wedge \gamma < 1$

**by** (*metis* (*full-types*) *assms*(2) *assms*(4) *g* *greaterThanLessThan-iff*  
*less-trans* *mult.right-neutral* *mult-less-cancel-left-pos* *not-le*  
*sgn-le-0-iff* *sgn-pos* *zero-le-one* *zero-le-sgn-iff* *zero-less-mult-iff*)  
**have** *t-in*: *mix-pmf*  $\beta$  *v* *u*  $\in \mathcal{P}$   
**by** (*meson* *assms*(1) *assms*(3) *mix-pmf-in-lotteries* *preference-def* *rational-preference-def*  
*rpr*)  
**have**  $v \succ_{[\mathcal{R}]}$  *mix-pmf*  $(1 - \beta)$  *v* *u*  
**using** *x-sg-y-sg-mpmf-right*[*of* *u* *v*  $1 - \beta$ ] *assms*  
**by** (*metis* *atLeastAtMost-iff* *greaterThanAtMost-iff* *greaterThanLessThan-iff*  
*less-eq-real-def* *pmf-inverse-switch-equals* *x-sg-y-sg-mpmf-right*)  
**have**  $t \succ_{[\mathcal{R}]}$  *u*  
**using** *assms*(1) *assms*(3) *ind* *rpr* *sg-imp-mix-sg* *t* **by** *fastforce*  
**hence** *t-s*:  $t \succ_{[\mathcal{R}]}$  (*mix-pmf*  $\gamma$  *t* *u*)  
**proof** –  
**have** (*mix-pmf*  $\gamma$  *t* *u*)  $\in \mathcal{P}$   
**by** (*metis* *assms*(1) *assms*(3) *atLeastAtMost-iff* *g1* *mix-in-lot* *mix-pmf-in-lotteries*  
  
*not-less* *order.asym* *preference-def* *rational-preference-def* *rpr* *t*)  
**have**  $t \succ_{[\mathcal{R}]}$  *mix-pmf*  $\gamma$  (*mix-pmf*  $\beta$  *v* *u*) *u*  
**using** *neumann-3B-b*[*of* *t* *u*  $\gamma$ ] *assms* *t* *g1*  
**by** (*meson* *greaterThanAtMost-iff* *greaterThanLessThan-iff*  
*ind* *less-eq-real-def* *rpr* *sg-imp-mix-sg*)  
**thus** *?thesis*  
**using** *t* **by** *blast*  
**qed**  
**from** *product-mix-pmf-prob-distrib*[*of*  $\beta$  *v* *u*] *assms*  
**have** *mix-pmf*  $\beta$  *v* *u*  $\succ_{[\mathcal{R}]}$  *mix-pmf*  $\alpha$  *v* *u*  
**by** (*metis* *t-s* *atLeastAtMost-iff* *g* *g1* *greaterThanLessThan-iff* *less-eq-real-def*  
*mult commute* *t*)  
**then show** *?thesis* **by** *blast*  
**qed**

## 5 This lemma is in called step 1 in literature. In Von Neumann and Morgenstern's book this is A:A (albeit more general)

**lemma** *step-1-most-general*:

**assumes**  $x \succ_{[\mathcal{R}]}$  *y*  
**assumes**  $\alpha \in \{0..(1::real)\}$   
**and**  $\beta \in \{0..(1::real)\}$   
**assumes**  $\alpha > \beta$   
**shows** (*mix-pmf*  $\alpha$  *x* *y*)  $\succ_{[\mathcal{R}]}$  (*mix-pmf*  $\beta$  *x* *y*)  
**proof** –  
**consider** (*ex*)  $\alpha = 1 \wedge \beta = 0$  | (*m*)  $\alpha \neq 1 \vee \beta \neq 0$   
**by** *blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** *m*

```

consider  $\beta = 0 \mid \beta \neq 0$ 
  by blast
then show ?thesis
proof (cases)
  case 1
    then show ?thesis
      using assms(1) assms(2) assms(4) ind rpr sg-imp-mix-sg by fastforce
  next
    case 2
      let  $?d = (\beta/\alpha)$ 
      have sg: (mix-pmf  $\alpha$  x y)  $\succ_{[\mathcal{R}]}$  y
        using assms(1) assms(2) assms(3) assms(4) ind rpr sg-imp-mix-sg by
fastforce
        have a:  $\alpha > 0$ 
          using assms(3) assms(4) by auto
        then have div-in: ?d  $\in \{0<..1\}$ 
          using assms(3) assms(4) 2 by auto
        have mx-p: (mix-pmf  $\alpha$  x y)  $\in \mathcal{P}$ 
          by (meson sg preference-def rational-preference-def rpr)
        have y-P: y  $\in \mathcal{P}$ 
          by (meson assms(1) preference-def rational-preference-def rpr)
        hence (mix-pmf ?d (mix-pmf  $\alpha$  x y) y)  $\in \mathcal{P}$ 
          using div-in mx-p by (simp add: mix-in-lot)
        have mix-pmf  $\beta$  (mix-pmf  $\alpha$  x y) y  $\succ_{[\mathcal{R}]}$  y
          using sg-imp-mix-sg[of (mix-pmf  $\alpha$  x y) y  $\mathcal{R}$  outcomes  $\beta$ ] sg div-in rpr ind
a assms(2) 2 assms(3) by auto
        have al1:  $\forall r \in \mathcal{P}. (mix-pmf \alpha x r) \succ_{[\mathcal{R}]} (mix-pmf \alpha y r)$ 
          by (meson a assms(1) assms(2) atLeastAtMost-iff greaterThanAtMost-iff ind
independece-dest-alt preference.not-outside rational-preference-def rpr y-P)
        then show ?thesis
          using greater-mix-pmf-greater-step-1-aux assms
          by (metis a div-in divide-less-eq-1-pos greaterThanAtMost-iff
greaterThanLessThan-iff mix-pmf-comp-with-dif-equiv neumann-3B-b sg)
      qed
    qed (simp add: assms(1))
qed

```

Kreps refers to this lemma as 5.6 c. The lemma after that is also significant.

**lemma** *approx-remains-after-same-comp:*

```

assumes  $p \approx_{[\mathcal{R}]} q$ 
  and  $r \in \mathcal{P}$ 
  and  $\alpha \in \{0..1\}$ 
shows  $mix-pmf \alpha p r \approx_{[\mathcal{R}]} mix-pmf \alpha q r$ 
using approx-indep-ge assms(1) assms(2) assms(3) ind rpr by blast

```

This lemma is the symmetric version of the previous lemma. This lemma is never mentioned in literature anywhere. Even though it looks trivial now, due to the asymmetric nature of the independence axiom, it is not so trivial,



and definitely worth mentioning.

**lemma** *approx-remains-after-same-comp-left*:

**assumes**  $p \approx[\mathcal{R}] q$   
**and**  $r \in \mathcal{P}$   
**and**  $\alpha \in \{0..1\}$   
**shows**  $\text{mix-pmf } \alpha r p \approx[\mathcal{R}] \text{mix-pmf } \alpha r q$   
**proof** –  
**have**  $1: \alpha \leq 1 \wedge \alpha \geq 0 \ 1 - \alpha \in \{0..1\}$   
**using** *assms(3)* **by** *auto+*  
**have** *fst*:  $\text{mix-pmf } \alpha r p \approx[\mathcal{R}] \text{mix-pmf } (1-\alpha) p r$   
**using** *assms* **by** (*metis mix-in-lot pmf-inverse-switch-equals*  
*rational-preference.compl relation-in-carrier rpr*)  
**moreover** **have**  $\text{mix-pmf } \alpha r p \approx[\mathcal{R}] \text{mix-pmf } \alpha r q$   
**using** *approx-remains-after-same-comp[of - - - \alpha] pmf-inverse-switch-equals[of \alpha*  
*p q] 1*  
*pmf-inverse-switch-equals rpr mix-pmf-preferred-independence[of - \alpha - ]*  
**by** (*metis assms(1) assms(2) assms(3) mix-pmf-preferred-independence*)  
**thus** *?thesis*  
**by** *blast*  
**qed**

**lemma** *mix-of-preferred-is-preferred*:

**assumes**  $p \succeq[\mathcal{R}] w$   
**assumes**  $q \succeq[\mathcal{R}] w$   
**assumes**  $\alpha \in \{0..1\}$   
**shows**  $\text{mix-pmf } \alpha p q \succeq[\mathcal{R}] w$   
**proof** –  
**consider**  $p \succeq[\mathcal{R}] q \mid q \succeq[\mathcal{R}] p$   
**using** *rpr assms(1) assms(2) rational-preference.compl relation-in-carrier* **by**  
*blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** *1*  
**have**  $\text{mix-pmf } \alpha p q \succeq[\mathcal{R}] q$   
**using** *1 assms(3) geq-imp-mix-geq-right ind rpr* **by** *blast*  
**moreover** **have**  $q \succeq[\mathcal{R}] w$   
**using** *assms* **by** *auto*  
**ultimately show** *?thesis* **using** *rpr preference.transitivity[of \mathcal{P} \mathcal{R}]*  
**by** (*meson rational-preference-def transE*)  
**next**  
**case** *2*  
**have**  $\text{mix-pmf } \alpha p q \succeq[\mathcal{R}] p$   
**using** *2 assms geq-imp-mix-geq-left ind rpr* **by** *blast*  
**moreover** **have**  $p \succeq[\mathcal{R}] w$   
**using** *assms* **by** *auto*  
**ultimately show** *?thesis* **using** *rpr preference.transitivity[of \mathcal{P} \mathcal{R}]*  
**by** (*meson rational-preference-def transE*)  
**qed**  
**qed**

**lemma** *mix-of-not-preferred-is-not-preferred*:  
**assumes**  $w \succeq[\mathcal{R}] p$   
**assumes**  $w \succeq[\mathcal{R}] q$   
**assumes**  $\alpha \in \{0..1\}$   
**shows**  $w \succeq[\mathcal{R}] \text{mix-pmf } \alpha p q$   
**proof** –  
**consider**  $p \succeq[\mathcal{R}] q \mid q \succeq[\mathcal{R}] p$   
**using** *rpr assms(1) assms(2) rational-preference.compl relation-in-carrier* **by**  
*blast*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** 1  
**moreover have**  $p \succeq[\mathcal{R}] \text{mix-pmf } \alpha p q$   
**using** *assms(3) neumann-3B-b-non-strict calculation* **by** *blast*  
**moreover show** *?thesis*  
**using** *rpr preference.transitivity[of P R]*  
**by** (*meson assms(1) calculation(2) rational-preference-def transE*)  
**next**  
**case** 2  
**moreover have**  $q \succeq[\mathcal{R}] \text{mix-pmf } \alpha p q$   
**using** *assms(3) neumann-3B-b-non-strict calculation*  
**by** (*metis mix-pmf-preferred-independence relation-in-carrier set-pmf-mix-eq*)  
**moreover show** *?thesis*  
**using** *rpr preference.transitivity[of P R]*  
**by** (*meson assms(2) calculation(2) rational-preference-def transE*)  
**qed**  
**qed**

**private definition** *degenerate-lotteries* **where**  
*degenerate-lotteries* =  $\{x \in \mathcal{P}. \text{card } (\text{set-pmf } x) = 1\}$

**private definition** *best* **where**  
*best* =  $\{x \in \mathcal{P}. (\forall y \in \mathcal{P}. x \succeq[\mathcal{R}] y)\}$

**private definition** *worst* **where**  
*worst* =  $\{x \in \mathcal{P}. (\forall y \in \mathcal{P}. y \succeq[\mathcal{R}] x)\}$

**lemma** *degenerate-total*:  
 $\forall e \in \text{degenerate-lotteries}. \forall m \in \mathcal{P}. e \succeq[\mathcal{R}] m \vee m \succeq[\mathcal{R}] e$   
**using** *degenerate-lotteries-def rational-preference.compl rpr* **by** *fastforce*

**lemma** *degen-outcome-cardinalities*:  
 $\text{card } \text{degenerate-lotteries} = \text{card } \text{outcomes}$   
**using** *card-degen-lotteries-equals-outcomes degenerate-lotteries-def* **by** *auto*

**lemma** *degenerate-lots-subset-all*:  $\text{degenerate-lotteries} \subseteq \mathcal{P}$   
**by** (*simp add: degenerate-lotteries-def*)

**lemma** *alt-definition-of-degenerate-lotteries[iff]*:  
 $\{\text{return-pmf } x \mid x. x \in \text{outcomes}\} = \text{degenerate-lotteries}$   
**proof** (*standard, goal-cases*)  
**case** 1  
**have**  $\forall x \in \{\text{return-pmf } x \mid x. x \in \text{outcomes}\}. x \in \text{degenerate-lotteries}$   
**proof**  
**fix**  $x$   
**assume**  $a: x \in \{\text{return-pmf } x \mid x. x \in \text{outcomes}\}$   
**then have**  $\text{card } (\text{set-pmf } x) = 1$   
**by** *auto*  
**moreover have**  $\text{set-pmf } x \subseteq \text{outcomes}$   
**using** *a set-pmf-subset-singleton by auto*  
**moreover have**  $x \in \mathcal{P}$   
**by** (*simp add: lotteries-on-def calculation*)  
**ultimately show**  $x \in \text{degenerate-lotteries}$   
**by** (*simp add: degenerate-lotteries-def*)  
**qed**  
**then show** *?case by blast*  
**next**  
**case** 2  
**have**  $\forall x \in \text{degenerate-lotteries}. x \in \{\text{return-pmf } x \mid x. x \in \text{outcomes}\}$   
**proof**  
**fix**  $x$   
**assume**  $a: x \in \text{degenerate-lotteries}$   
**hence**  $\text{card } (\text{set-pmf } x) = 1$   
**using** *degenerate-lotteries-def by blast*  
**moreover have**  $\text{set-pmf } x \subseteq \text{outcomes}$   
**by** (*meson a degenerate-lots-subset-all subset-iff support-in-outcomes*)  
**moreover obtain**  $e$  **where**  $\{e\} = \text{set-pmf } x$   
**using** *calculation*  
**by** (*metis card-1-singletonE*)  
**moreover have**  $e \in \text{outcomes}$   
**using** *calculation(2) calculation(3) by blast*  
**moreover have**  $x = \text{return-pmf } e$   
**using** *calculation(3) set-pmf-subset-singleton by fast*  
**ultimately show**  $x \in \{\text{return-pmf } x \mid x. x \in \text{outcomes}\}$   
**by** *blast*  
**qed**  
**then show** *?case by blast*  
**qed**

**lemma** *best-indifferent*:  
 $\forall x \in \text{best}. \forall y \in \text{best}. x \approx[\mathcal{R}] y$   
**by** (*simp add: best-def*)

**lemma** *worst-indifferent*:  
 $\forall x \in \text{worst}. \forall y \in \text{worst}. x \approx[\mathcal{R}] y$   
**by** (*simp add: worst-def*)

**lemma** *best-worst-indiff-all-indiff*:

**assumes**  $b \in \text{best}$   
**and**  $w \in \text{worst}$   
**and**  $b \approx[\mathcal{R}] w$   
**shows**  $\forall e \in \mathcal{P}. e \approx[\mathcal{R}] w \ \forall e \in \mathcal{P}. e \approx[\mathcal{R}] b$   
**proof** –  
**show**  $\forall e \in \mathcal{P}. e \approx[\mathcal{R}] w$   
**proof** (*standard*)  
**fix**  $e$   
**assume**  $a: e \in \mathcal{P}$   
**then have**  $b \succeq[\mathcal{R}] e$   
**using** *a best-def assms* **by** *blast*  
**moreover have**  $e \succeq[\mathcal{R}] w$   
**using** *a assms worst-def* **by** *auto*  
**moreover have**  $b \succeq[\mathcal{R}] e$   
**by** (*simp add: calculation(1)*)  
**moreover show**  $e \approx[\mathcal{R}] w$   
**proof** (*rule ccontr*)  
**assume**  $\neg e \approx[\mathcal{R}] w$   
**then consider**  $e \succ[\mathcal{R}] w \mid w \succ[\mathcal{R}] e$   
**by** (*simp add: calculation(2)*)  
**then show** *False*  
**proof** (*cases*)  
**case** 2  
**then show** *?thesis*  
**using** *calculation(2)* **by** *blast*  
**qed** (*meson assms(3) calculation(1)*  
*rational-preference.strict-is-neg-transitive relation-in-carrier rpr*)  
**qed**  
**qed**  
**then show**  $\forall e \in \text{local.P}. e \approx[\mathcal{R}] b$   
**using** *assms* **by** (*meson rational-preference.compl*  
*rational-preference.strict-is-neg-transitive relation-in-carrier rpr*)  
**qed**

Like Step 1 most general but with IFF.

**lemma** *mix-pmf-pref-iff-more-likely [iff]*:

**assumes**  $b \succ[\mathcal{R}] w$   
**assumes**  $\alpha \in \{0..1\}$   
**and**  $\beta \in \{0..1\}$   
**shows**  $\alpha > \beta \iff \text{mix-pmf } \alpha \ b \ w \succ[\mathcal{R}] \text{mix-pmf } \beta \ b \ w$  (**is**  $?L \iff ?R$ )  
**using** *assms step-1-most-general[of b w alpha beta]*  
**by** (*metis linorder-neqE-linordered-idom step-1-most-general*)

**lemma** *better-worse-good-mix-preferred[iff]*:

**assumes**  $b \succeq[\mathcal{R}] w$   
**assumes**  $\alpha \in \{0..1\}$   
**and**  $\beta \in \{0..1\}$   
**assumes**  $\alpha \geq \beta$

**shows**  $\text{mix-pmf } \alpha \text{ } b \text{ } w \succeq[\mathcal{R}] \text{ mix-pmf } \beta \text{ } b \text{ } w$   
**proof**–  
**have**  $(0::\text{real}) \leq 1$   
**by** *simp*  
**then show** *?thesis*  
**by** (*metis* (*no-types*) *assms* *assms(1)* *assms(2)* *assms(3)* *atLeastAtMost-iff*  
*less-eq-real-def* *mix-of-not-preferred-is-not-preferred*  
*mix-of-preferred-is-preferred* *mix-pmf-preferred-independence*  
*pmf-mix-0* *relation-in-carrier* *step-1-most-general*)  
**qed**

## 5.1 Add finiteness and non emptyness of outcomes

**context**

**assumes** *fnt*: *finite outcomes*  
**assumes** *nempty*: *outcomes*  $\neq \{\}$

**begin**

**lemma** *finite-degenerate-lotteries*:

*finite degenerate-lotteries*  
**using** *degen-outcome-cardinalities* *fnt* *nempty* **by** *fastforce*

**lemma** *degenerate-has-max-preferred*:

$\{x \in \text{degenerate-lotteries}. (\forall y \in \text{degenerate-lotteries}. x \succeq[\mathcal{R}] y)\} \neq \{\}$  (**is** *?l*  $\neq \{\}$ )

**proof**

**assume** *a*: *?l* =  $\{\}$

**let** *?DG* = *degenerate-lotteries*

**obtain** *R* **where**

*R*: *rational-preference* *?DG* *R*  $R \subseteq \mathcal{R}$

**using** *degenerate-lots-subset-all* *rational-preference.all-carrier-ex-sub-rel* *rpr* **by**  
*blast*

**then have**  $\exists e \in ?DG. \forall e' \in ?DG. e \succeq[\mathcal{R}] e'$

**by** (*metis* *R(1)* *R(2)* *card-0-eq* *degen-outcome-cardinalities*  
*finite-degenerate-lotteries* *fnt* *nempty* *subset-eq*  
*rational-preference.finite-nonempty-carrier-has-maximum* )

**then show** *False*

**using** *a* **by** *auto*

**qed**

**lemma** *degenerate-has-min-preferred*:

$\{x \in \text{degenerate-lotteries}. (\forall y \in \text{degenerate-lotteries}. y \succeq[\mathcal{R}] x)\} \neq \{\}$  (**is** *?l*  $\neq \{\}$ )

**proof**

**assume** *a*: *?l* =  $\{\}$

**let** *?DG* = *degenerate-lotteries*

**obtain** *R* **where**

*R*: *rational-preference* *?DG* *R*  $R \subseteq \mathcal{R}$

**using** *degenerate-lots-subset-all* *rational-preference.all-carrier-ex-sub-rel* *rpr* **by**

```

blast
  have  $\exists e \in ?DG. \forall e' \in ?DG. e' \succeq[\mathcal{R}] e$ 
    by (metis R(1) R(2) card-0-eq degen-outcome-cardinalities
        finite-degenerate-lotteries fnt nempty subset-eq
        rational-preference.finite-nonempty-carrier-has-minimum )
  then show False
    using a by auto
qed

lemma exists-best-degenerate:
 $\exists x \in \text{degenerate-lotteries}. \forall y \in \text{degenerate-lotteries}. x \succeq[\mathcal{R}] y$ 
  using degenerate-has-max-preferred by blast

lemma exists-worst-degenerate:
 $\exists x \in \text{degenerate-lotteries}. \forall y \in \text{degenerate-lotteries}. y \succeq[\mathcal{R}] x$ 
  using degenerate-has-min-preferred by blast

lemma best-degenerate-in-best-overall:
 $\exists x \in \text{degenerate-lotteries}. \forall y \in \mathcal{P}. x \succeq[\mathcal{R}] y$ 
proof -
  obtain b where
    b:  $b \in \text{degenerate-lotteries} \forall y \in \text{degenerate-lotteries}. b \succeq[\mathcal{R}] y$ 
    using exists-best-degenerate by blast
  have asm: finite outcomes set-pmf  $b \subseteq \text{outcomes}$ 
    by (simp add: fnt) (meson b(1) degenerate-lots-subset-all subset-iff support-in-outcomes)
  obtain B where B: set-pmf  $b = \{B\}$ 
    using b card-1-singletonE degenerate-lotteries-def by blast
  have deg:  $\forall d \in \text{outcomes}. b \succeq[\mathcal{R}] \text{return-pmf } d$ 
    using alt-definition-of-degenerate-lotteries b(2) by blast
  define P where
 $P = (\lambda p. p \in \mathcal{P} \longrightarrow \text{return-pmf } B \succeq[\mathcal{R}] p)$ 
  have P p for p
  proof -
    consider set-pmf  $p \subseteq \text{outcomes} \mid \neg \text{set-pmf } p \subseteq \text{outcomes}$ 
    by blast
    then show ?thesis
  proof (cases)
    case 1
    have finite outcomes set-pmf  $p \subseteq \text{outcomes}$ 
      by (auto simp: 1 asm)
    then show ?thesis
  proof (induct rule: pmf-mix-induct')
    case (degenerate x)
    then show ?case
      using B P-def deg set-pmf-subset-singleton by fastforce
  qed (simp add: P-def lotteries-on-def mix-of-not-preferred-is-not-preferred
    mix-of-not-preferred-is-not-preferred[of b p q a])
  qed (simp add: lotteries-on-def P-def)
qed

```

**moreover have**  $\forall e \in \mathcal{P}. b \succeq[\mathcal{R}] e$   
**using** *calculation B P-def set-pmf-subset-singleton* **by** *fastforce*  
**ultimately show** *?thesis*  
**using** *b degenerate-lots-subset-all* **by** *blast*  
**qed**

**lemma** *worst-degenerate-in-worst-overall:*

$\exists x \in \text{degenerate-lotteries}. \forall y \in \mathcal{P}. y \succeq[\mathcal{R}] x$

**proof** –

**obtain** *b* **where**

*b*:  $b \in \text{degenerate-lotteries} \forall y \in \text{degenerate-lotteries}. y \succeq[\mathcal{R}] b$

**using** *exists-worst-degenerate* **by** *blast*

**have** *asm: finite outcomes set-pmf b*  $\subseteq$  *outcomes*

**by** (*simp add: fnt*) (*meson b(1) degenerate-lots-subset-all subset-iff support-in-outcomes*)

**obtain** *B* **where** *B*: *set-pmf b* = {*B*}

**using** *b card-1-singletonE degenerate-lotteries-def* **by** *blast*

**have** *deg*:  $\forall d \in \text{outcomes}. \text{return-pmf } d \succeq[\mathcal{R}] b$

**using** *alt-definition-of-degenerate-lotteries b(2)* **by** *blast*

**define** *P* **where**

*P* =  $(\lambda p. p \in \mathcal{P} \longrightarrow p \succeq[\mathcal{R}] \text{return-pmf } B)$

**have** *P p* **for** *p*

**proof** –

**consider** *set-pmf p*  $\subseteq$  *outcomes* |  $\neg \text{set-pmf } p \subseteq \text{outcomes}$

**by** *blast*

**then show** *?thesis*

**proof** (*cases*)

**case** *1*

**have** *finite outcomes set-pmf p*  $\subseteq$  *outcomes*

**by** (*auto simp: 1 asm*)

**then show** *?thesis*

**proof** (*induct rule: pmf-mix-induct'*)

**case** (*degenerate x*)

**then show** *?case*

**using** *B P-def deg set-pmf-subset-singleton* **by** *fastforce*

**next**

**qed** (*simp add: P-def lotteries-on-def mix-of-preferred-is-preferred*

*mix-of-not-preferred-is-not-preferred[of b p]*)

**qed** (*simp add: lotteries-on-def P-def*)

**qed**

**moreover have**  $\forall e \in \mathcal{P}. e \succeq[\mathcal{R}] b$

**using** *calculation B P-def set-pmf-subset-singleton* **by** *fastforce*

**ultimately show** *?thesis*

**using** *b degenerate-lots-subset-all* **by** *blast*

**qed**

**lemma** *overall-best-nonempty:*

*best*  $\neq \{\}$

**using** *best-def best-degenerate-in-best-overall degenerate-lots-subset-all* **by** *blast*

**lemma** *overall-worst-nonempty*:

*worst*  $\neq \{\}$   
**using** *degenerate-lots-subset-all worst-def worst-degenerate-in-worst-overall* **by**  
*auto*

**lemma** *trans-approx*:

**assumes**  $x \approx_{[\mathcal{R}]} y$   
**and**  $y \approx_{[\mathcal{R}]} z$   
**shows**  $x \approx_{[\mathcal{R}]} z$   
**using** *preference.indiff-trans*[of  $\mathcal{P} \ \mathcal{R} \ x \ y \ z$ ] *assms rpr rational-preference-def* **by**  
*blast*

First EXPLICIT use of the axiom of choice

**private definition** *some-best* **where**

*some-best* = (*SOME*  $x. x \in \text{degenerate-lotteries} \wedge x \in \text{best}$ )

**private definition** *some-worst* **where**

*some-worst* = (*SOME*  $x. x \in \text{degenerate-lotteries} \wedge x \in \text{worst}$ )

**private definition** *my-U* :: 'a pmf  $\Rightarrow$  real

**where**

*my-U*  $p$  = (*SOME*  $\alpha. \alpha \in \{0..1\} \wedge p \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha \ \text{some-best} \ \text{some-worst}$ )

**lemma** *exists-best-and-degenerate*: *degenerate-lotteries*  $\cap$  *best*  $\neq \{\}$

**using** *best-def best-degenerate-in-best-overall degenerate-lots-subset-all* **by** *blast*

**lemma** *exists-worst-and-degenerate*: *degenerate-lotteries*  $\cap$  *worst*  $\neq \{\}$

**using** *worst-def worst-degenerate-in-worst-overall degenerate-lots-subset-all* **by**  
*blast*

**lemma** *some-best-in-best*: *some-best*  $\in$  *best*

**using** *exists-best-and-degenerate some-best-def*  
**by** (*metis (mono-tags, lifting) Int-emptyI some-eq-ex*)

**lemma** *some-worst-in-worst*: *some-worst*  $\in$  *worst*

**using** *exists-worst-and-degenerate some-worst-def*  
**by** (*metis (mono-tags, lifting) Int-emptyI some-eq-ex*)

**lemma** *best-always-at-least-as-good-mix*:

**assumes**  $\alpha \in \{0..1\}$

**and**  $p \in \mathcal{P}$

**shows** *mix-pmf*  $\alpha \ \text{some-best} \ p \succeq_{[\mathcal{R}]} p$

**using** *assms(1) assms(2) best-def mix-of-preferred-is-preferred*  
*rational-preference.compl rpr some-best-in-best* **by** *fastforce*

**lemma** *geq-mix-imp-weak-pref*:



**assumes**  $\alpha \in \{0..1\}$   
**and**  $\beta \in \{0..1\}$   
**assumes**  $\alpha \geq \beta$   
**shows**  $\text{mix-pmf } \alpha \text{ some-best some-worst} \succeq[\mathcal{R}] \text{ mix-pmf } \beta \text{ some-best some-worst}$   
**using** *assms(1) assms(2) assms(3) best-def some-best-in-best some-worst-in-worst worst-def* **by** *auto*

**lemma** *gamma-inverse:*

**assumes**  $\alpha \in \{0 < .. < 1\}$   
**and**  $\beta \in \{0 < .. < 1\}$   
**shows**  $(1::\text{real}) - (\alpha - \beta) / (1 - \beta) = (1 - \alpha) / (1 - \beta)$

**proof** –

**have**  $1 - (\alpha - \beta) / (1 - \beta) = (1 - \beta) / (1 - \beta) - (\alpha - \beta) / (1 - \beta)$

**using** *assms(2)* **by** *auto*

**also have**  $\dots = (1 - \beta - (\alpha - \beta)) / (1 - \beta)$

**by** (*metis diff-divide-distrib*)

**also have**  $\dots = (1 - \alpha) / (1 - \beta)$

**by** *simp*

**finally show** *?thesis* .

**qed**

**lemma** *all-mix-pmf-indiff-indiff-best-worst:*

**assumes**  $l \in \mathcal{P}$

**assumes**  $b \in \text{best}$

**assumes**  $w \in \text{worst}$

**assumes**  $b \approx[\mathcal{R}] w$

**shows**  $\forall \alpha \in \{0..1\}. l \approx[\mathcal{R}] \text{ mix-pmf } \alpha b w$

**by** (*meson assms best-worst-indiff-all-indiff(1) mix-of-preferred-is-preferred best-worst-indiff-all-indiff(2) mix-of-not-preferred-is-not-preferred*)

**lemma** *indiff-imp-same-utility-value:*

**assumes**  $\text{some-best} \succ[\mathcal{R}] \text{ some-worst}$

**assumes**  $\alpha \in \{0..1\}$

**assumes**  $\beta \in \{0..1\}$

**assumes**  $\text{mix-pmf } \beta \text{ some-best some-worst} \approx[\mathcal{R}] \text{ mix-pmf } \alpha \text{ some-best some-worst}$

**shows**  $\beta = \alpha$

**using** *assms(1) assms(2) assms(3) assms(4) linorder-neqE-linordered-idom* **by** *blast*

**lemma** *leq-mix-imp-weak-inferior:*

**assumes**  $\text{some-best} \succ[\mathcal{R}] \text{ some-worst}$

**assumes**  $\alpha \in \{0..1\}$

**and**  $\beta \in \{0..1\}$

**assumes**  $\text{mix-pmf } \beta \text{ some-best some-worst} \succeq[\mathcal{R}] \text{ mix-pmf } \alpha \text{ some-best some-worst}$

**shows**  $\beta \geq \alpha$

**proof** –

**have**  $*$ :  $\text{mix-pmf } \beta \text{ some-best some-worst} \approx[\mathcal{R}] \text{ mix-pmf } \alpha \text{ some-best some-worst} \implies \alpha \leq \beta$

**using** *assms(1) assms(2) assms(3) indiff-imp-same-utility-value* **by** *blast*

**consider**  $\text{mix-pmf } \beta \text{ some-best some-worst } \succ_{[\mathcal{R}]} \text{mix-pmf } \alpha \text{ some-best some-worst}$   
|  
 $\text{mix-pmf } \beta \text{ some-best some-worst } \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha \text{ some-best some-worst}$   
**using**  $\text{assms}(4)$  **by**  $\text{blast}$   
**then show**  $?thesis$   
**by**  $(\text{cases}) (\text{meson } \text{assms}(2) \text{ assms}(3) \text{ geq-mix-imp-weak-pref le-cases } *)+$   
**qed**

**lemma**  $\text{ge-mix-pmf-preferred}$ :

**assumes**  $x \succ_{[\mathcal{R}]} y$   
**assumes**  $\alpha \in \{0..1\}$   
**and**  $\beta \in \{0..1\}$   
**assumes**  $\alpha \geq \beta$   
**shows**  $(\text{mix-pmf } \alpha x y) \succeq_{[\mathcal{R}]} (\text{mix-pmf } \beta x y)$   
**using**  $\text{assms}(1) \text{ assms}(2) \text{ assms}(3) \text{ assms}(4)$  **by**  $\text{blast}$

## 5.2 Add continuity to assumptions

**context**

**assumes**  $\text{cnt: continuous-vnm (lotteries-on outcomes) } \mathcal{R}$

**begin**

In Literature this is referred to as step 2.

**lemma**  $\text{step-2-unique-continuous-unfolding}$ :

**assumes**  $p \succeq_{[\mathcal{R}]} q$   
**and**  $q \succeq_{[\mathcal{R}]} r$   
**and**  $p \succ_{[\mathcal{R}]} r$   
**shows**  $\exists! \alpha \in \{0..1\}. q \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha p r$

**proof**  $(\text{rule } \text{ccontr})$

**assume**  $\text{neg-a: } \nexists! \alpha. \alpha \in \{0..1\} \wedge q \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha p r$

**have**  $\exists \alpha \in \{0..1\}. q \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha p r$

**using**  $\text{non-unique-continuous-unfolding[of outcomes } \mathcal{R} p q r]$

$\text{assms cnt rpr}$  **by**  $\text{blast}$

**then obtain**  $\alpha \beta :: \text{real}$  **where**

$a\text{-b: } \alpha \in \{0..1\} \beta \in \{0..1\} q \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha p r q \approx_{[\mathcal{R}]} \text{mix-pmf } \beta p r \alpha \neq \beta$

**using**  $\text{neg-a}$  **by**  $\text{blast}$

**consider**  $\alpha > \beta \mid \beta > \alpha$

**using**  $a\text{-b}$  **by**  $\text{linarith}$

**then show**  $\text{False}$

**proof**  $(\text{cases})$

**case** 1

**with**  $\text{step-1-most-general[of } p r \alpha \beta] \text{ assms}$

**have**  $\text{mix-pmf } \alpha p r \succ_{[\mathcal{R}]} \text{mix-pmf } \beta p r$

**using**  $a\text{-b}(1) a\text{-b}(2)$  **by**  $\text{blast}$

**then show**  $?thesis$  **using**  $a\text{-b}$

**by**  $(\text{meson } \text{rational-preference.strict-is-neg-transitive relation-in-carrier rpr})$

**next**

**case** 2

**with**  $\text{step-1-most-general[of } p r \beta \alpha] \text{ assms}$  **have**  $\text{mix-pmf } \beta p r \succ_{[\mathcal{R}]} \text{mix-pmf } \alpha p r$

$\alpha p r$   
**using**  $a-b(1) a-b(2)$  **by** *blast*  
**then show** *?thesis* **using**  $a-b$   
**by** (*meson rational-preference.strict-is-neg-transitive relation-in-carrier rpr*)  
**qed**  
**qed**

These following two lemmas are referred to sometimes called step 2.

**lemma** *create-unique-indiff-using-distinct-best-worst:*

**assumes**  $l \in \mathcal{P}$   
**assumes**  $b \in \text{best}$   
**assumes**  $w \in \text{worst}$   
**assumes**  $b \succ_{[\mathcal{R}]} w$   
**shows**  $\exists! \alpha \in \{0..1\}. l \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha b w$   
**proof** –  
**have**  $b \succeq_{[\mathcal{R}]} l$   
**using** *best-def*  
**using** *assms* **by** *blast*  
**moreover have**  $l \succeq_{[\mathcal{R}]} w$   
**using** *worst-def assms* **by** *blast*  
**ultimately show**  $\exists! \alpha \in \{0..1\}. l \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha b w$   
**using** *step-2-unique-continuous-unfolding*[of  $b l w$ ] *assms* **by** *linarith*  
**qed**

**lemma** *exists-element-bw-mix-is-approx:*

**assumes**  $l \in \mathcal{P}$   
**assumes**  $b \in \text{best}$   
**assumes**  $w \in \text{worst}$   
**shows**  $\exists \alpha \in \{0..1\}. l \approx_{[\mathcal{R}]} \text{mix-pmf } \alpha b w$   
**proof** –  
**consider**  $b \succ_{[\mathcal{R}]} w \mid b \approx_{[\mathcal{R}]} w$   
**using** *assms(2) assms(3) best-def worst-def* **by** *auto*  
**then show** *?thesis*  
**proof** (*cases*)  
**case** 1  
**then show** *?thesis*  
**using** *create-unique-indiff-using-distinct-best-worst assms* **by** *blast*  
**qed** (*auto simp: all-mix-pmf-indiff-indiff-best-worst assms*)  
**qed**

**lemma** *my-U-is-defined:*

**assumes**  $p \in \mathcal{P}$   
**shows**  $\text{my-U } p \in \{0..1\} p \approx_{[\mathcal{R}]} \text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst}$   
**proof** –  
**have**  $\text{some-best} \in \text{best}$   
**by** (*simp add: some-best-in-best*)  
**moreover have**  $\text{some-worst} \in \text{worst}$   
**by** (*simp add: some-worst-in-worst*)  
**with** *exists-element-bw-mix-is-approx*[of  $p \text{some-best some-worst}$ ] *calculation assms*

**have**  $e: \exists \alpha \in \{0..1\}. p \approx[\mathcal{R}] \text{mix-pmf } \alpha \text{ some-best some-worst}$  **by** *blast*  
**then show**  $\text{my-U } p \in \{0..1\}$   
**by** (*metis (mono-tags, lifting) my-U-def someI-ex*)  
**show**  $p \approx[\mathcal{R}] \text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst}$   
**by** (*metis (mono-tags, lifting) e my-U-def someI-ex*)  
**qed**

**lemma** *weak-pref-mix-with-my-U-weak-pref*:  
**assumes**  $p \succeq[\mathcal{R}] q$   
**shows**  $\text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst} \succeq[\mathcal{R}] \text{mix-pmf } (\text{my-U } q) \text{ some-best some-worst}$   
**by** (*meson assms my-U-is-defined(2) relation-in-carrier rpr rational-preference.weak-is-transitive*)

**lemma** *preferred-greater-my-U*:  
**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**assumes**  $\text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst} \succ[\mathcal{R}] \text{mix-pmf } (\text{my-U } q) \text{ some-best some-worst}$   
**shows**  $\text{my-U } p > \text{my-U } q$   
**proof** (*rule ccontr*)  
**assume**  $\neg \text{my-U } p > \text{my-U } q$   
**then consider**  $\text{my-U } p = \text{my-U } q \mid \text{my-U } p < \text{my-U } q$   
**by** *linarith*  
**then show** *False*  
**proof** (*cases*)  
**case 1**  
**then have**  $\text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst} \approx[\mathcal{R}] \text{mix-pmf } (\text{my-U } q) \text{ some-best some-worst}$   
**using** *assms by auto*  
**then show** *?thesis using assms by blast*  
**next**  
**case 2**  
**moreover have**  $\text{my-U } q \in \{0..1\}$   
**using** *assms(2) my-U-is-defined(1) by blast*  
**moreover have**  $\text{my-U } p \in \{0..1\}$   
**using** *assms(1) my-U-is-defined(1) by blast*  
**moreover have**  $\text{mix-pmf } (\text{my-U } q) \text{ some-best some-worst} \succeq[\mathcal{R}] \text{mix-pmf } (\text{my-U } p) \text{ some-best some-worst}$   
**using** *calculation geq-mix-imp-weak-pref by auto*  
**then show** *?thesis using assms by blast*  
**qed**  
**qed**

**lemma** *geq-my-U-imp-weak-preference*:  
**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**assumes**  $\text{some-best} \succ[\mathcal{R}] \text{some-worst}$   
**assumes**  $\text{my-U } p \geq \text{my-U } q$

**shows**  $p \succeq[\mathcal{R}] q$   
**proof** –  
**have**  $p\text{-}q$ :  $my\text{-}U p \in \{0..1\}$   $my\text{-}U q \in \{0..1\}$   
**using**  $assms$   $my\text{-}U\text{-}is\text{-}defined(1)$  **by**  $blast+$   
**with**  $ge\text{-}mix\text{-}pmf\text{-}preferred$ [of  $some\text{-}best$   $some\text{-}worst$   $my\text{-}U p$   $my\text{-}U q$ ]  
 $p\text{-}q$   $assms(1)$   $assms(3)$   $assms(4)$   
**have**  $mix\text{-}pmf$  ( $my\text{-}U p$ )  $some\text{-}best$   $some\text{-}worst$   $\succeq[\mathcal{R}]$   $mix\text{-}pmf$  ( $my\text{-}U q$ )  $some\text{-}best$   
 $some\text{-}worst$  **by**  $blast$   
**consider**  $my\text{-}U p = my\text{-}U q \mid my\text{-}U p > my\text{-}U q$   
**using**  $assms$  **by**  $linarith$   
**then show**  $?thesis$   
**proof** ( $cases$ )  
**case** 2  
**then show**  $?thesis$   
**by** ( $meson$   $assms(1)$   $assms(2)$   $assms(3)$   $p\text{-}q(1)$   $p\text{-}q(2)$   $rational\text{-}preference.compl$ )  
  
 $rpr$   $step\text{-}1\text{-}most\text{-}general$   $weak\text{-}pref\text{-}mix\text{-}with\text{-}my\text{-}U\text{-}weak\text{-}pref$ )  
**qed** ( $metis$   $assms(1)$   $assms(2)$   $my\text{-}U\text{-}is\text{-}defined(2)$   $trans\text{-}approx$ )  
**qed**

**lemma**  $my\text{-}U\text{-}represents\text{-}pref$ :  
**assumes**  $some\text{-}best \succ[\mathcal{R}] some\text{-}worst$   
**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**shows**  $p \succeq[\mathcal{R}] q \longleftrightarrow my\text{-}U p \geq my\text{-}U q$  (**is**  $?L \longleftrightarrow ?R$ )  
**proof** –  
**have**  $p\text{-}def$ :  $my\text{-}U p \in \{0..1\}$   $my\text{-}U q \in \{0..1\}$   
**using**  $assms$   $my\text{-}U\text{-}is\text{-}defined$  **by**  $blast+$   
**show**  $?thesis$   
**proof**  
**assume**  $a$ :  $?L$   
**hence**  $mix\text{-}pmf$  ( $my\text{-}U p$ )  $some\text{-}best$   $some\text{-}worst$   $\succeq[\mathcal{R}]$   $mix\text{-}pmf$  ( $my\text{-}U q$ )  
 $some\text{-}best$   $some\text{-}worst$   
**using**  $weak\text{-}pref\text{-}mix\text{-}with\text{-}my\text{-}U\text{-}weak\text{-}pref$  **by**  $auto$   
**then show**  $?R$  **using**  $leq\text{-}mix\text{-}imp\text{-}weak\text{-}inferior$ [of  $my\text{-}U p$   $my\text{-}U q$ ]  $p\text{-}def$   $a$   
 $assms(1)$   $leq\text{-}mix\text{-}imp\text{-}weak\text{-}inferior$  **by**  $blast$   
**next**  
**assume**  $?R$   
**then show**  $?L$  **using**  $geq\text{-}my\text{-}U\text{-}imp\text{-}weak\text{-}preference$   
**using**  $assms(1)$   $assms(2)$   $assms(3)$  **by**  $blast$   
**qed**  
**qed**

**lemma**  $first\text{-}iff\text{-}u\text{-}greater\text{-}strict\text{-}preff$ :  
**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**assumes**  $some\text{-}best \succ[\mathcal{R}] some\text{-}worst$   
**shows**  $my\text{-}U p > my\text{-}U q \longleftrightarrow mix\text{-}pmf$  ( $my\text{-}U p$ )  $some\text{-}best$   $some\text{-}worst$   $\succ[\mathcal{R}]$   
 $mix\text{-}pmf$  ( $my\text{-}U q$ )  $some\text{-}best$   $some\text{-}worst$

**proof**

**assume**  $a: my-U\ p > my-U\ q$   
**have**  $my-U\ p \in \{0..1\}$   $my-U\ q \in \{0..1\}$   
**using** *assms my-U-is-defined(1)* **by** *blast+*  
**then show**  $mix-pmf\ (my-U\ p)\ some-best\ some-worst \succ_{[\mathcal{R}]} mix-pmf\ (my-U\ q)$   
*some-best some-worst*  
**using**  $a$  *assms(3)* **by** *blast*  
**next**  
**assume**  $a: mix-pmf\ (my-U\ p)\ some-best\ some-worst \succ_{[\mathcal{R}]} mix-pmf\ (my-U\ q)$   
*some-best some-worst*  
**have**  $my-U\ p \in \{0..1\}$   $my-U\ q \in \{0..1\}$   
**using** *assms my-U-is-defined(1)* **by** *blast+*  
**then show**  $my-U\ p > my-U\ q$   
**using** *preferred-greater-my-U[of p q]* *assms a* **by** *blast*  
**qed**

**lemma** *second-iff-calib-mix-pref-strict-pref:*

**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**assumes**  $some-best \succ_{[\mathcal{R}]} some-worst$   
**shows**  $mix-pmf\ (my-U\ p)\ some-best\ some-worst \succ_{[\mathcal{R}]} mix-pmf\ (my-U\ q)\ some-best\ some-worst \iff p \succ_{[\mathcal{R}]} q$

**proof**

**assume**  $a: mix-pmf\ (my-U\ p)\ some-best\ some-worst \succ_{[\mathcal{R}]} mix-pmf\ (my-U\ q)$   
*some-best some-worst*  
**have**  $my-U\ p \in \{0..1\}$   $my-U\ q \in \{0..1\}$   
**using** *assms my-U-is-defined(1)* **by** *blast+*  
**then show**  $p \succ_{[\mathcal{R}]} q$   
**using**  $a$  *assms(3)* *assms(1)* *assms(2)* *geq-my-U-imp-weak-preference*  
*leq-mix-imp-weak-inferior weak-pref-mix-with-my-U-weak-pref* **by** *blast*

**next**

**assume**  $a: p \succ_{[\mathcal{R}]} q$   
**have**  $my-U\ p \in \{0..1\}$   $my-U\ q \in \{0..1\}$   
**using** *assms my-U-is-defined(1)* **by** *blast+*  
**then show**  $mix-pmf\ (my-U\ p)\ some-best\ some-worst \succ_{[\mathcal{R}]} mix-pmf\ (my-U\ q)$   
*some-best some-worst*  
**using**  $a$  *assms(1)* *assms(2)* *assms(3)* *leq-mix-imp-weak-inferior my-U-represents-pref*  
**by** *blast*  
**qed**

**lemma** *my-U-is-linear-function:*

**assumes**  $p \in \mathcal{P}$   
**and**  $q \in \mathcal{P}$   
**and**  $\alpha \in \{0..1\}$   
**assumes**  $some-best \succ_{[\mathcal{R}]} some-worst$   
**shows**  $my-U\ (mix-pmf\ \alpha\ p\ q) = \alpha * my-U\ p + (1 - \alpha) * my-U\ q$   
**proof** –  
**define**  $B$  **where**  $B: B = some-best$   
**define**  $W$  **where**  $W: W = some-worst$

```

define  $Up$  where  $Up$ :  $Up = my-U\ p$ 
define  $Uq$  where  $Uq$ :  $Uq = my-U\ q$ 
have  $long-in$ :  $(\alpha * Up + (1 - \alpha) * Uq) \in \{0..1\}$ 
proof –
  have  $Up \in \{0..1\}$ 
    using  $assms\ Up\ my-U-is-defined(1)$  by  $blast$ 
  moreover have  $Uq \in \{0..1\}$ 
    using  $assms\ Uq\ my-U-is-defined(1)$  by  $blast$ 
  moreover have  $\alpha * Up \in \{0..1\}$ 
    using  $\langle Up \in \{0..1\} \rangle\ assms(3)\ mult-le-one$  by  $auto$ 
  moreover have  $1 - \alpha \in \{0..1\}$ 
    using  $assms(3)$  by  $auto$ 
  moreover have  $(1 - \alpha) * Uq \in \{0..1\}$ 
    using  $mult-le-one[of\ 1 - \alpha\ Uq]\ calculation(2)\ calculation(4)$  by  $auto$ 
  ultimately show  $?thesis$ 
    using  $add-nonneg-nonneg[of\ \alpha * Up\ (1 - \alpha) * Uq]$ 
       $convex-bound-le[of\ Up\ 1\ Uq\ \alpha\ 1 - \alpha]$  by  $simp$ 
qed
have  $fst$ :  $p \approx[\mathcal{R}]\ (mix-pmf\ Up\ B\ W)$ 
  using  $assms\ my-U-is-defined[of\ p]\ B\ W\ Up$  by  $simp$ 
have  $snd$ :  $q \approx[\mathcal{R}]\ (mix-pmf\ Uq\ B\ W)$ 
  using  $assms\ my-U-is-defined[of\ q]\ B\ W\ Uq$  by  $simp$ 
have  $mp-in$ :  $(mix-pmf\ Up\ B\ W) \in \mathcal{P}$ 
  using  $fst\ relation-in-carrier$  by  $blast$ 
have  $f2$ :  $mix-pmf\ \alpha\ p\ q \approx[\mathcal{R}]\ mix-pmf\ \alpha\ (mix-pmf\ Up\ B\ W)\ q$ 
  using  $fst\ assms(2)\ assms(3)\ mix-pmf-preferred-independence$  by  $blast$ 
have  $**$ :  $mix-pmf\ \alpha\ (mix-pmf\ Up\ B\ W)\ (mix-pmf\ Uq\ B\ W) =$ 
   $mix-pmf\ (\alpha * Up + (1 - \alpha) * Uq)\ B\ W$  (is  $?L = ?R$ )
proof –
  let  $?mixPQ = (mix-pmf\ (\alpha * Up + (1 - \alpha) * Uq)\ B\ W)$ 
  have  $\forall e \in set-pmf\ ?L.\ pmf\ (?L)\ e = pmf\ ?mixPQ\ e$ 
  proof
    fix  $e$ 
    assume  $asm$ :  $e \in set-pmf\ ?L$ 
    have  $i1$ :  $pmf\ (?L)\ e = \alpha * pmf\ (mix-pmf\ Up\ B\ W)\ e +$ 
       $pmf\ (mix-pmf\ Uq\ B\ W)\ e - \alpha * pmf\ (mix-pmf\ Uq\ B\ W)\ e$ 
    using  $pmf-mix-deeper[of\ \alpha\ mix-pmf\ Up\ B\ W\ (mix-pmf\ Uq\ B\ W)\ e]$   $assms(3)$ 
by  $blast$ 
    have  $i3$ :  $\dots = \alpha * Up * pmf\ B\ e + \alpha * pmf\ W\ e - \alpha * Up * pmf\ W\ e + Uq$ 
       $*\ pmf\ B\ e +$ 
       $pmf\ W\ e - Uq * pmf\ W\ e - \alpha * Uq * pmf\ B\ e - \alpha * pmf\ W\ e + \alpha * Uq * pmf\ W\ e$ 
    using  $left-diff-distrib'\ pmf-mix-deeper[of\ Up\ B\ W\ e]\ pmf-mix-deeper[of\ Uq\ B\ W\ e]$ 
       $assms\ Up\ Uq\ my-U-is-defined(1)$  by  $(simp\ add:\ distrib-left\ right-diff-distrib)$ 
    have  $j4$ :  $pmf\ ?mixPQ\ e = (\alpha * Up + (1 - \alpha) * Uq) * pmf\ B\ e +$ 
       $pmf\ W\ e - (\alpha * Up + (1 - \alpha) * Uq) * pmf\ W\ e$ 
    using  $pmf-mix-deeper[of\ (\alpha * Up + (1 - \alpha) * Uq)\ B\ W\ e]\ long-in$  by  $blast$ 
    then show  $pmf\ (?L)\ e = pmf\ ?mixPQ\ e$ 

```

```

    by (simp add: i1 i3 mult.commute right-diff-distrib' ring-class.ring-distrib(1))
  qed
  then show ?thesis using pmf-equiv-intro1 by blast
  qed
  have mix-pmf  $\alpha$  (mix-pmf  $U_p$   $B$   $W$ )  $q \approx[\mathcal{R}] ?L$ 
    using approx-remains-after-same-comp-left assms(3) mp-in snd by blast
  hence *: mix-pmf  $\alpha$   $p$   $q \approx[\mathcal{R}]$  mix-pmf  $\alpha$  (mix-pmf (my-U  $p$ )  $B$   $W$ ) (mix-pmf
(my-U  $q$ )  $B$   $W$ )
    using  $U_p$   $U_q$  f2 trans-approx by blast
  have mix-pmf  $\alpha$  (mix-pmf (my-U  $p$ )  $B$   $W$ ) (mix-pmf (my-U  $q$ )  $B$   $W$ ) = ?R
    using  $U_p$   $U_q$  ** by blast
  hence my-U (mix-pmf  $\alpha$   $p$   $q$ ) =  $\alpha * U_p + (1-\alpha) * U_q$ 
    by (metis *  $B$   $W$  assms(4) indiff-imp-same-utility-value long-in
my-U-is-defined(1) my-U-is-defined(2) my-U-represents-pref-relation-in-carrier)
  then show ?thesis
    using  $U_p$   $U_q$  by blast
  qed

```

Now we define a more general Utility function that also takes the degenerate case into account

**private definition** *general-U*

where

*general-U*  $p$  = (if some-best  $\approx[\mathcal{R}]$  some-worst then 1 else my-U  $p$ )

**lemma** *general-U-is-linear-function:*

assumes  $p \in \mathcal{P}$

and  $q \in \mathcal{P}$

and  $\alpha \in \{0..1\}$

shows *general-U* (mix-pmf  $\alpha$   $p$   $q$ ) =  $\alpha * (\text{general-U } p) + (1 - \alpha) * (\text{general-U } q)$

**proof** –

consider some-best  $\succ[\mathcal{R}]$  some-worst | some-best  $\approx[\mathcal{R}]$  some-worst

using best-def some-best-in-best some-worst-in-worst worst-def by auto

then show ?thesis

**proof** (cases, goal-cases)

case 1

then show ?case

using assms(1) assms(2) assms(3) *general-U-def* my-U-is-linear-function by

auto

next

case 2

then show ?case

using assms(1) assms(2) assms(3) *general-U-def* by auto

qed

qed

**lemma** *general-U-ordinal-Utility:*

shows ordinal-utility  $\mathcal{P}$   $\mathcal{R}$  *general-U*

**proof** (standard, goal-cases)



```

case (1 x y)
consider (a) some-best  $\succ_{[\mathcal{R}]}$  some-worst | (b) some-best  $\approx_{[\mathcal{R}]}$  some-worst
  using best-def some-best-in-best some-worst-in-worst worst-def by auto
then show ?case
proof (cases, goal-cases)
  case a
  have some-best  $\succ_{[\mathcal{R}]}$  some-worst
  using a by auto
  then show  $x \succeq_{[\mathcal{R}]} y = (\text{general-U } y \leq \text{general-U } x)$ 
  using 1 my-U-represents-pref[of x y] general-U-def by simp
next
  case b
  have general-U x = 1 general-U y = 1
  by (simp add: b general-U-def)+
  moreover have  $x \approx_{[\mathcal{R}]} y$  using b
  by (meson 1(1) 1(2) best-worst-indiff-all-indiff(1)
    some-best-in-best some-worst-in-worst trans-approx)
  ultimately show  $x \succeq_{[\mathcal{R}]} y = (\text{general-U } y \leq \text{general-U } x)$ 
  using general-U-def by linarith
qed
next
  case (2 x y)
  then show ?case
  using relation-in-carrier by blast
next
  case (3 x y)
  then show ?case
  using relation-in-carrier by blast
qed

```

Proof of the linearity of general-U. If we consider the definition of expected utility functions from Maschler, Solan, Zamir we are done.

**theorem** *is-linear*:

```

assumes  $p \in \mathcal{P}$ 
  and  $q \in \mathcal{P}$ 
  and  $\alpha \in \{0..1\}$ 
shows  $\exists u. u (\text{mix-pmf } \alpha p q) = \alpha * (u p) + (1-\alpha) * (u q)$ 
proof
  let ?u = general-U
  consider some-best  $\succ_{[\mathcal{R}]}$  some-worst | some-best  $\approx_{[\mathcal{R}]}$  some-worst
  using best-def some-best-in-best some-worst-in-worst worst-def by auto
  then show ?u (mix-pmf  $\alpha p q$ ) =  $\alpha * ?u p + (1 - \alpha) * ?u q$ 
  proof (cases)
  case 1
  then show ?thesis
  using assms(1) assms(2) assms(3) general-U-def my-U-is-linear-function by
auto
  next
  case 2

```

```

    then show ?thesis
      by (simp add: general-U-def)
  qed
qed

```

Now I define a Utility function that assigns a utility to all outcomes. These are only finitely many

```

private definition ocU
  where
    ocU p = general-U (return-pmf p)

```

```

lemma geral-U-is-expected-value-of-ocU:
  assumes set-pmf p  $\subseteq$  outcomes
  shows general-U p = measure-pmf.expectation p ocU
  using fnt assms
proof (induct rule: pmf-mix-induct')
  case (mix p q a)
  hence general-U (mix-pmf a p q) = a * general-U p + (1-a) * general-U q
    using general-U-is-linear-function[of p q a] mix.hyps assms lotteries-on-def
  mix.hyps by auto
  also have ... = a * measure-pmf.expectation p ocU + (1-a) * measure-pmf.expectation
  q ocU
    by (simp add: mix.hyps(4) mix.hyps(5))
  also have ... = measure-pmf.expectation (mix-pmf a p q) ocU
    using general-U-is-linear-function expected-value-mix-pmf-distrib fnt infinite-super
  mix.hyps(1)
    by (metis fnt mix.hyps(2) mix.hyps(3))
  finally show ?case .
qed (auto simp: support-in-outcomes assms fnt integral-measure-pmf-real ocU-def)

```

```

lemma ordinal-utility-expected-value:
  ordinal-utility  $\mathcal{P}$   $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation x ocU)
proof (standard, goal-cases)
  case (1 x y)
  have ocs: set-pmf x  $\subseteq$  outcomes set-pmf y  $\subseteq$  outcomes
    by (meson 1 subsetI support-in-outcomes)+
  have x  $\succeq_{[\mathcal{R}]}$  y  $\implies$  (measure-pmf.expectation y ocU  $\leq$  measure-pmf.expectation
  x ocU)
  proof -
    assume x  $\succeq_{[\mathcal{R}]}$  y
    have general-U x  $\geq$  general-U y
      by (meson  $\langle x \succeq_{[\mathcal{R}]} y \rangle$  general-U-ordinal-Utility ordinal-utility-def)
    then show (measure-pmf.expectation y ocU  $\leq$  measure-pmf.expectation x ocU)
      using geral-U-is-expected-value-of-ocU ocs by auto
  qed
  moreover have (measure-pmf.expectation y ocU  $\leq$  measure-pmf.expectation x
  ocU)  $\implies$  x  $\succeq_{[\mathcal{R}]}$  y
  proof -
    assume (measure-pmf.expectation y ocU  $\leq$  measure-pmf.expectation x ocU)

```

```

then have general-U  $x \geq$  general-U  $y$ 
  by (simp add: geral-U-is-expected-value-of-ocU ocs(1) ocs(2))
then show  $x \succeq[\mathcal{R}] y$ 
  by (meson 1(1) 1(2) general-U-ordinal-Utility ordinal-utility.util-def)
qed
ultimately show ?case
  by blast
next
  case (2 x y)
  then show ?case
    using relation-in-carrier by blast
next
  case (3 x y)
  then show ?case
    using relation-in-carrier by auto
qed

```

```

lemma ordinal-utility-expected-value':
   $\exists u. \text{ordinal-utility } \mathcal{P} \mathcal{R} (\lambda x. \text{measure-pmf.expectation } x u)$ 
  using ordinal-utility-expected-value by blast

```

```

lemma ocU-is-expected-utility-bernoulli:
  shows  $\forall x \in \mathcal{P}. \forall y \in \mathcal{P}. x \succeq[\mathcal{R}] y \iff$ 
     $\text{measure-pmf.expectation } x \text{ ocU} \geq \text{measure-pmf.expectation } y \text{ ocU}$ 
  using ordinal-utility-expected-value by (meson ordinal-utility.util-def)

```

**end**

**end**

**end**

```

lemma expected-value-is-utility-function:
  assumes fnt: finite outcomes and outcomes  $\neq \{\}$ 
  assumes  $x \in \text{lotteries-on outcomes}$  and  $y \in \text{lotteries-on outcomes}$ 
  assumes ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x. \text{measure-pmf.expectation } x u$ )
  shows  $\text{measure-pmf.expectation } x u \geq \text{measure-pmf.expectation } y u \iff x \succeq[\mathcal{R}] y$ 
  (is ?L  $\iff$  ?R)
  using assms(3) assms(4) assms(5) ordinal-utility.util-def-conf
    ordinal-utility.ordinal-utility-left iffI by (metis (no-types, lifting))

```

```

lemma system-U-implies-vNM-utility:
  assumes fnt: finite outcomes and outcomes  $\neq \{\}$ 
  assumes rpr: rational-preference (lotteries-on outcomes)  $\mathcal{R}$ 
  assumes ind: independent-vnm (lotteries-on outcomes)  $\mathcal{R}$ 

```

**assumes** *cnt: continuous-vnm (lotteries-on outcomes)  $\mathcal{R}$*   
**shows**  $\exists u$ . *ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation  $x$   $u$ )*  
**using** *ordinal-utility-expected-value'[of outcomes  $\mathcal{R}$ ] assms by blast*

**lemma** *vNM-utility-implies-rationality:*

**assumes** *fnt: finite outcomes and outcomes  $\neq \{\}$*   
**assumes**  $\exists u$ . *ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation  $x$   $u$ )*  
**shows** *rational-preference (lotteries-on outcomes)  $\mathcal{R}$*   
**using** *assms( $\exists$ ) ordinal-util-imp-rat-prefs by blast*

**theorem** *vNM-utility-implies-independence:*

**assumes** *fnt: finite outcomes and outcomes  $\neq \{\}$*   
**assumes**  $\exists u$ . *ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation  $x$   $u$ )*  
**shows** *independent-vnm (lotteries-on outcomes)  $\mathcal{R}$*   
**proof** (*rule independent-vnmI2*)

**fix**  $p$   $q$   $r$   
**and**  $\alpha::real$   
**assume**  $a1$ :  $p \in \mathcal{P}$  *outcomes*  
**assume**  $a2$ :  $q \in \mathcal{P}$  *outcomes*  
**assume**  $a3$ :  $r \in \mathcal{P}$  *outcomes*  
**assume**  $a4$ :  $\alpha \in \{0 < .. 1\}$   
**have** *in-lots: mix-pmf  $\alpha$   $p$   $r \in$  lotteries-on outcomes mix-pmf  $\alpha$   $q$   $r \in$  lotteries-on outcomes*

**using**  $a1$   $a3$   $a4$  *mix-in-lot apply fastforce*

**using**  $a2$   $a3$   $a4$  *mix-in-lot by fastforce*

**have** *fnts: finite (set-pmf  $p$ ) finite (set-pmf  $q$ ) finite (set-pmf  $r$ )*

**using**  $a1$   $a2$   $a3$  *fnt infinite-super lotteries-on-def by blast+*

**obtain**  $u$  **where**

*u: ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation  $x$   $u$ )*

**using** *assms by blast*

**have**  $p \succeq[\mathcal{R}] q \implies \text{mix-pmf } \alpha \text{ } p \text{ } r \succeq[\mathcal{R}] \text{mix-pmf } \alpha \text{ } q \text{ } r$

**proof** –

**assume**  $p \succeq[\mathcal{R}] q$

**hence**  $f$ : *measure-pmf.expectation  $p$   $u \geq$  measure-pmf.expectation  $q$   $u$*

**using**  $u$   $a1$   $a2$  *ordinal-utility.util-def by fastforce*

**have** *measure-pmf.expectation (mix-pmf  $\alpha$   $p$   $r$ )  $u \geq$  measure-pmf.expectation (mix-pmf  $\alpha$   $q$   $r$ )  $u$*

**proof** –

**have** *measure-pmf.expectation (mix-pmf  $\alpha$   $p$   $r$ )  $u =$*

$\alpha * \text{measure-pmf.expectation } p \text{ } u + (1 - \alpha) * \text{measure-pmf.expectation } r \text{ } u$

**using** *expected-value-mix-pmf-distrib[of  $p$   $r$   $\alpha$   $u$ ] assms fnts  $a4$  by fastforce*

**moreover** **have** *measure-pmf.expectation (mix-pmf  $\alpha$   $q$   $r$ )  $u =$*

$\alpha * \text{measure-pmf.expectation } q \text{ } u + (1 - \alpha) * \text{measure-pmf.expectation } r \text{ } u$

**using** *expected-value-mix-pmf-distrib[of  $q$   $r$   $\alpha$   $u$ ] assms fnts  $a4$  by fastforce*

**ultimately show** *?thesis using  $f$  using  $a4$  by auto*

**qed**

**then show**  $\text{mix-pmf } \alpha p r \succeq[\mathcal{R}] \text{mix-pmf } \alpha q r$   
**using**  $u$  *ordinal-utility-expected-value'* *ocU-is-expected-utility-bernoulli in-lots*  
**by** (*simp add: in-lots ordinal-utility-def*)  
**qed**  
**moreover have**  $\text{mix-pmf } \alpha p r \succeq[\mathcal{R}] \text{mix-pmf } \alpha q r \implies p \succeq[\mathcal{R}] q$   
**proof** –  
**assume**  $\text{mix-pmf } \alpha p r \succeq[\mathcal{R}] \text{mix-pmf } \alpha q r$   
**hence**  $f:\text{measure-pmf.expectation } (\text{mix-pmf } \alpha p r) u \geq \text{measure-pmf.expectation } (\text{mix-pmf } \alpha q r) u$   
**using** *ordinal-utility.ordinal-utility-left u* **by** *fastforce*  
**hence**  $\text{measure-pmf.expectation } p u \geq \text{measure-pmf.expectation } q u$   
**proof** –  
**have**  $\text{measure-pmf.expectation } (\text{mix-pmf } \alpha p r) u =$   
 $\alpha * \text{measure-pmf.expectation } p u + (1 - \alpha) * \text{measure-pmf.expectation } r u$   
**using** *expected-value-mix-pmf-distrib[of p r  $\alpha$  u]* *assms fnts a4* **by** *fastforce*  
**moreover have**  $\text{measure-pmf.expectation } (\text{mix-pmf } \alpha q r) u =$   
 $\alpha * \text{measure-pmf.expectation } q u + (1 - \alpha) * \text{measure-pmf.expectation } r u$   
**using** *expected-value-mix-pmf-distrib[of q r  $\alpha$  u]* *assms fnts a4* **by** *fastforce*  
**ultimately show** *?thesis* **using**  $f$  **using**  $a4$  **by** *auto*  
**qed**  
**then show**  $p \succeq[\mathcal{R}] q$   
**using**  $a1 a2$  *ordinal-utility.util-def-conf u* **by** *fastforce*  
**qed**  
**ultimately show**  $p \succeq[\mathcal{R}] q = \text{mix-pmf } \alpha p r \succeq[\mathcal{R}] \text{mix-pmf } \alpha q r$   
**by** *blast*  
**qed**

**lemma** *exists-weight-for-equality*:  
**assumes**  $a > c$  **and**  $a \geq b$  **and**  $b \geq c$   
**shows**  $\exists (e::\text{real}) \in \{0..1\}. (1-e) * a + e * c = b$   
**proof** –  
**from** *assms* **have**  $b \in \text{closed-segment } a c$   
**by** (*simp add: closed-segment-eq-real-ivl*)  
**thus** *?thesis* **by** (*auto simp: closed-segment-def*)  
**qed**

**lemma** *vNM-utility-implies-continuity*:  
**assumes** *fnt: finite outcomes* **and** *outcomes  $\neq \{\}$*   
**assumes**  $\exists u. \text{ordinal-utility } (\text{lotteries-on outcomes}) \mathcal{R} (\lambda x. \text{measure-pmf.expectation } x u)$   
**shows** *continuous-vnm (lotteries-on outcomes)  $\mathcal{R}$*   
**proof** (*rule continuous-vnmI*)  
**fix**  $p q r$   
**assume**  $a1: p \in \mathcal{P} \text{ outcomes}$   
**assume**  $a2: q \in \mathcal{P} \text{ outcomes}$   
**assume**  $a3: r \in \mathcal{P} \text{ outcomes}$   
**assume**  $a4: p \succeq[\mathcal{R}] q \wedge q \succeq[\mathcal{R}] r$   
**then have**  $g: p \succeq[\mathcal{R}] r$   
**by** (*meson assms(3) ordinal-utility.util-imp-trans transD*)

```

obtain  $u$  where
   $u$ : ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation  $x$   $u$ )
  using assms by blast
have geqa: measure-pmf.expectation  $p$   $u \geq$  measure-pmf.expectation  $q$   $u$ 
  measure-pmf.expectation  $q$   $u \geq$  measure-pmf.expectation  $r$   $u$ 
  using a4  $u$  by (meson ordinal-utility.ordinal-utility-left)+
have fnts: finite  $p$  finite  $q$  finite  $r$ 
  using a1 a2 a3 fnt infinite-super lotteries-on-def by auto+
consider  $p \succ_{[\mathcal{R}]} r \mid p \approx_{[\mathcal{R}]} r$ 
  using  $g$  by auto
then show  $\exists \alpha \in \{0..1\}$ . mix-pmf  $\alpha$   $p$   $r \approx_{[\mathcal{R}]} q$ 
proof (cases)
  case 1
  define  $a$  where  $a$ :  $a =$  measure-pmf.expectation  $p$   $u$ 
  define  $b$  where  $b$ :  $b =$  measure-pmf.expectation  $r$   $u$ 
  define  $c$  where  $c$ :  $c =$  measure-pmf.expectation  $q$   $u$ 
  have  $a > b$ 
  using 1 a1 a2 a3  $a$   $b$  ordinal-utility.util-def-conf  $u$  by force
  have  $c \leq a$   $b \leq c$ 
  using geqa  $a$   $b$   $c$  by blast+
  then obtain  $e :: \text{real}$  where
     $e$ :  $e \in \{0..1\}$   $(1-e) * a + e * b = c$ 
    using exists-weight-for-equality[of  $b$   $a$   $c$ ]  $\langle b < a \rangle$  by blast
  have  $*: 1-e \in \{0..1\}$ 
  using e(1) by auto
  hence measure-pmf.expectation (mix-pmf  $(1-e)$   $p$   $r$ )  $u =$ 
     $(1-e) *$  measure-pmf.expectation  $p$   $u + e *$  measure-pmf.expectation  $r$   $u$ 
  using expected-value-mix-pmf-distrib[of  $p$   $r$   $1-e$   $u$ ] fnts by fastforce
  also have  $\dots = (1-e) * a + e * b$ 
  using  $a$   $b$  by auto
  also have  $\dots = c$ 
  using  $c$   $e$  by auto
  finally have  $f$ : measure-pmf.expectation (mix-pmf  $(1-e)$   $p$   $r$ )  $u =$  mea-
sure-pmf.expectation  $q$   $u$ 
  using  $c$  by blast
  hence mix-pmf  $(1-e)$   $p$   $r \approx_{[\mathcal{R}]} q$ 
  using expected-value-is-utility-function[of outcomes mix-pmf  $(1-e)$   $p$   $r$   $q$   $\mathcal{R}$ 
 $u$ ] *
  proof –
  have mix-pmf  $(1-e)$   $p$   $r \in \mathcal{P}$  outcomes
  using  $\langle 1-e \in \{0..1\} \rangle$  a1 a3 mix-in-lot by blast
  then show ?thesis
  using  $f$  a2 ordinal-utility.util-def  $u$  by fastforce
  qed
  then show ?thesis
  using exists-weight-for-equality expected-value-mix-pmf-distrib  $*$  by blast
next
  case 2
  have  $r \approx_{[\mathcal{R}]} q$ 

```

```

    by (meson 2 a4 assms(β) ordinal-utility.util-imp-trans transD)
  then show ?thesis by force
qed
qed

```

```

theorem Von-Neumann-Morgenstern-Utility-Theorem:
  assumes fnt: finite outcomes and outcomes ≠ {}
  shows rational-preference (lotteries-on outcomes)  $\mathcal{R}$   $\wedge$ 
    independent-vnm (lotteries-on outcomes)  $\mathcal{R}$   $\wedge$ 
    continuous-vnm (lotteries-on outcomes)  $\mathcal{R}$   $\longleftrightarrow$ 
    ( $\exists u$ . ordinal-utility (lotteries-on outcomes)  $\mathcal{R}$  ( $\lambda x$ . measure-pmf.expectation x
    u))
  using vNM-utility-implies-independence[OF assms, of  $\mathcal{R}$ ]
    system-U-implies-vNM-utility[OF assms, of  $\mathcal{R}$ ]
    vNM-utility-implies-continuity[OF assms, of  $\mathcal{R}$ ]
    ordinal-util-imp-rat-prefs[of lotteries-on outcomes  $\mathcal{R}$ ] by auto

end

```

```

theory Expected-Utility
imports
  Neumann-Morgenstern-Utility-Theorem
begin

```

## 6 Definition of vNM-utility function

We define a version of the vNM Utility function using the locale mechanism. Currently this definition and system U have no proven relation yet.

Important:  $u$  is actually not the von Neuman Utility Function, but a Bernoulli Utility Function. The Expected value  $p$  given  $u$  is the von Neumann Utility Function.

```

locale vNM-utility =
  fixes outcomes :: 'a set
  fixes relation :: 'a pmf relation
  fixes u :: 'a  $\Rightarrow$  real
  assumes relation  $\subseteq$  (lotteries-on outcomes  $\times$  lotteries-on outcomes)
  assumes  $\bigwedge p q$ .  $p \in$  lotteries-on outcomes  $\implies$ 
     $q \in$  lotteries-on outcomes  $\implies$ 
     $p \succeq$ [relation]  $q \longleftrightarrow$  measure-pmf.expectation  $p$   $u \geq$  measure-pmf.expectation
     $q$   $u$ 
begin

```

```

lemma vNM-utilityD:
  shows relation  $\subseteq$  (lotteries-on outcomes  $\times$  lotteries-on outcomes)
  and  $p \in$  lotteries-on outcomes  $\implies$   $q \in$  lotteries-on outcomes  $\implies$ 

```

```

    p  $\succeq$ [relation] q  $\longleftrightarrow$  measure-pmf.expectation p u  $\geq$  measure-pmf.expectation q
  u
  using vNM-utility-axioms vNM-utility-def by (blast+)

lemma not-outside:
  assumes p  $\succeq$ [relation] q
  shows p  $\in$  lotteries-on outcomes
    and q  $\in$  lotteries-on outcomes
proof (goal-cases)
  case 1
  then show ?case
    by (meson assms contra-subsetD mem-Sigma-iff vNM-utility-axioms vNM-utility-def)
next
  case 2
  then show ?case
    by (metis assms mem-Sigma-iff subsetCE vNM-utility-axioms vNM-utility-def)
qed

lemma utility-ge:
  assumes p  $\succeq$ [relation] q
  shows measure-pmf.expectation p u  $\geq$  measure-pmf.expectation q u
  using assms vNM-utility-axioms vNM-utility-def
  by (metis (no-types, lifting) not-outside(1) not-outside(2))

end

sublocale vNM-utility  $\subseteq$  ordinal-utility (lotteries-on outcomes) relation ( $\lambda p.$  mea-
sure-pmf.expectation p u)
proof (standard, goal-cases)
  case (2 x y)
  then show ?case
    using not-outside(1) by blast
next
  case (3 x y)
  then show ?case
    by (auto simp add: not-outside(2))
qed (metis (mono-tags, lifting) vNM-utility-axioms vNM-utility-def)

context vNM-utility
begin

lemma strict-preference-iff-strict-utility:
  assumes p  $\in$  lotteries-on outcomes
  assumes q  $\in$  lotteries-on outcomes
  shows p  $\succ$ [relation] q  $\longleftrightarrow$  measure-pmf.expectation p u  $>$  measure-pmf.expectation
  q u
  by (meson assms(1) assms(2) less-eq-real-def not-le util-def)

lemma pos-distrib-left:

```



**assumes**  $c > 0$   
**shows**  $(\sum z \in \text{outcomes}. \text{pmf } q \ z * (c * u \ z)) = c * (\sum z \in \text{outcomes}. \text{pmf } q \ z * (u \ z))$   
**proof** –  
**have**  $(\sum z \in \text{outcomes}. \text{pmf } q \ z * (c * u \ z)) = (\sum z \in \text{outcomes}. \text{pmf } q \ z * c * u \ z)$   
**by** (*simp add: ab-semigroup-mult-class.mult-ac(1)*)  
**also have**  $\dots = (\sum z \in \text{outcomes}. c * \text{pmf } q \ z * u \ z)$   
**by** (*simp add: mult.commute*)  
**also have**  $\dots = c * (\sum z \in \text{outcomes}. \text{pmf } q \ z * u \ z)$   
**by** (*simp add: ab-semigroup-mult-class.mult-ac(1) sum-distrib-left*)  
**finally show** *?thesis* .  
**qed**

**lemma** *sum-pmf-util-commute*:  
 $(\sum a \in \text{outcomes}. \text{pmf } p \ a * u \ a) = (\sum a \in \text{outcomes}. u \ a * \text{pmf } p \ a)$   
**by** (*simp add: mult.commute*)

## 7 Finite outcomes

**context**  
**assumes** *fnt: finite outcomes*  
**begin**

**lemma** *sum-equals-pmf-expectation*:  
**assumes**  $p \in \text{lotteries-on outcomes}$   
**shows**  $(\sum z \in \text{outcomes}. (\text{pmf } p \ z) * (u \ z)) = \text{measure-pmf.expectation } p \ u$   
**proof** –  
**have** *fnt: finite outcomes*  
**by** (*simp add: vNM-utilityD(1) fnt*)  
**have**  $\text{measure-pmf.expectation } p \ u = (\sum a \in \text{outcomes}. \text{pmf } p \ a * u \ a)$   
**using** *support-in-outcomes assms fnt integral-measure-pmf-real sum-pmf-util-commute* **by** *fastforce*  
**then show** *?thesis*  
**using** *real-scaleR-def* **by** *presburger*  
**qed**

**lemma** *expected-utility-weak-preference*:  
**assumes**  $p \in \text{lotteries-on outcomes}$   
**and**  $q \in \text{lotteries-on outcomes}$   
**shows**  $p \succeq[\text{relation}] q \iff (\sum z \in \text{outcomes}. (\text{pmf } p \ z) * (u \ z)) \geq (\sum z \in \text{outcomes}. (\text{pmf } q \ z) * (u \ z))$   
**using** *sum-equals-pmf-expectation[of p, OF assms(1)]*  
*sum-equals-pmf-expectation[of q, OF assms(2)]*  
*vNM-utility-def assms(1) assms(2) util-def-conf* **by** *presburger*

**lemma** *diff-leq-zero-weak-preference*:  
**assumes**  $p \in \text{lotteries-on outcomes}$   
**and**  $q \in \text{lotteries-on outcomes}$   
**shows**  $p \succeq q \iff ((\sum a \in \text{outcomes}. \text{pmf } q \ a * u \ a) - (\sum a \in \text{outcomes}. \text{pmf } p \ a$

```

* u a) ≤ 0)
  using assms(1) assms(2) diff-le-0-iff-le
  by (metis (mono-tags, lifting) expected-utility-weak-preference)

lemma expected-utility-strict-preference:
  assumes p ∈ lotteries-on outcomes
  and q ∈ lotteries-on outcomes
  shows p ≻[relation] q ⟷ measure-pmf.expectation p u > measure-pmf.expectation
  q u
  using assms expected-utility-weak-preference less-eq-real-def not-le
  by (metis (no-types, lifting) util-def-conf)

lemma scale-pos-left:
  assumes c > 0
  shows vNM-utility outcomes relation (λx. c * u x)
  proof(standard, goal-cases)
  case 1
  then show ?case
    using vNM-utility-axioms vNM-utility-def by blast
  next
  case (2 p q)
  have q ∈ lotteries-on outcomes and p ∈ lotteries-on outcomes
  using 2(2) by (simp add: fnt 2(1))+
  then have *: p ≽ q = (measure-pmf.expectation q u ≤ measure-pmf.expectation
  p u)
  using expected-utility-weak-preference[of p q] assms by blast
  have dist-c: (∑ z∈outcomes. (pmf q z) * (c * u z)) = c * (∑ z∈outcomes. (pmf
  q z) * (u z))
  using pos-distrib-left[of c q] assms by blast
  have dist-c': (∑ z∈outcomes. (pmf p z) * (c * u z)) = c * (∑ z∈outcomes. (pmf
  p z) * (u z))
  using pos-distrib-left[of c p] assms by blast
  have p ≽ q ⟷ ((∑ z∈outcomes. (pmf q z) * (c * u z)) ≤ (∑ z∈outcomes. (pmf
  p z) * (c * u z)))
  proof (rule iffI)
  assume p ≽ q
  then have (∑ z∈outcomes. pmf q z * (u z)) ≤ (∑ z∈outcomes. pmf p z * (u
  z))
  using utility-ge
  using 2(1) 2(2) sum-equals-pmf-expectation by presburger
  then show (∑ z∈outcomes. pmf q z * (c * u z)) ≤ (∑ z∈outcomes. pmf p z *
  (c * u z))
  using dist-c dist-c'
  by (simp add: assms)
  next
  assume (∑ z∈outcomes. pmf q z * (c * u z)) ≤ (∑ z∈outcomes. pmf p z * (c
  * u z))
  then have (∑ z∈outcomes. pmf q z * (u z)) ≤ (∑ z∈outcomes. pmf p z * (u
  z))

```

```

    using 2(1) mult-le-cancel-iff2 assms by (simp add: dist-c dist-c')
  then show  $p \succeq q$ 
    using 2(2) assms 2(1) by (simp add: * sum-equals-pmf-expectation)
qed
then show ?case
  by (simp add: * assms)
qed

```

**lemma** *strict-alt-def:*

```

  assumes  $p \in \text{lotteries-on outcomes}$ 
    and  $q \in \text{lotteries-on outcomes}$ 
  shows  $p \succ[\text{relation}] q \iff$ 
     $(\sum_{z \in \text{outcomes}} (\text{pmf } p \ z) * (u \ z)) > (\sum_{z \in \text{outcomes}} (\text{pmf } q \ z) * (u \ z))$ 
  using sum-equals-pmf-expectation[of p, OF assms(1)] assms(1) assms(2)
    sum-equals-pmf-expectation[of q, OF assms(2)] strict-preference-iff-strict-utility
  by presburger

```

**lemma** *strict-alt-def-utility-g:*

```

  assumes  $p \succ[\text{relation}] q$ 
  shows  $(\sum_{z \in \text{outcomes}} (\text{pmf } p \ z) * (u \ z)) > (\sum_{z \in \text{outcomes}} (\text{pmf } q \ z) * (u \ z))$ 
  using assms not-outside(1) not-outside(2) strict-alt-def
  by meson

```

end

end

**lemma** *vnm-utility-is-ordinal-utility:*

```

  assumes vNM-utility outcomes relation u
  shows ordinal-utility (lotteries-on outcomes) relation ( $\lambda p. \text{measure-pmf.expectation } p \ u$ )
proof (standard, goal-cases)
  case (1 x y)
  then show ?case
    using assms vNM-utility-def by blast
next
  case (2 x y)
  then show ?case
    using assms vNM-utility.not-outside(1) by blast
next
  case (3 x y)
  then show ?case
    using assms vNM-utility.not-outside(2) by blast
qed

```

**lemma** *vnm-utility-imp-reational-prefs:*

```

  assumes vNM-utility outcomes relation u
  shows rational-preference (lotteries-on outcomes) relation
proof (standard,goal-cases)

```

**case** (1  $x y$ )  
**then show** *?case*  
   **using** *assms vNM-utility.not-outside(1)* **by** *blast*  
**next**  
**case** (2  $x y$ )  
**then show** *?case*  
   **using** *assms vNM-utility.not-outside(2)* **by** *blast*  
**next**  
**case** 3  
**have** *t: trans relation*  
   **using** *assms ordinal-utility.util-imp-trans vnm-utility-is-ordinal-utility* **by** *blast*  
**have** *refl-on (lotteries-on outcomes) relation*  
   **by** (*meson assms order-refl refl-on-def vNM-utility-def*)  
**then show** *?case*  
   **using** *preorder-on-def t* **by** *blast*  
**next**  
**case** 4  
**have** *total-on (lotteries-on outcomes) relation*  
   **using** *ordinal-utility.util-imp-total[of lotteries-on outcomes*  
     *relation ( $\lambda p. (\sum z \in \text{outcomes}. (\text{pmf } p \ z) * (u \ z))$ )*  
     *assms vnm-utility-is-ordinal-utility*  
   **using** *ordinal-utility.util-imp-total* **by** *blast*  
**then show** *?case*  
   **by** *simp*  
**qed**

**theorem** *expected-utility-theorem-form-vnm-utility:*  
**assumes** *fnf: finite outcomes and outcomes  $\neq \{\}$*   
**shows** *rational-preference (lotteries-on outcomes)  $\mathcal{R} \wedge$*   
   *independent-vnm (lotteries-on outcomes)  $\mathcal{R} \wedge$*   
   *continuous-vnm (lotteries-on outcomes)  $\mathcal{R} \longleftrightarrow$*   
   *( $\exists u. \text{vNM-utility outcomes } \mathcal{R} \ u$ )*

**proof**  
**assume** *rational-preference ( $\mathcal{P}$  outcomes)  $\mathcal{R} \wedge$  independent-vnm ( $\mathcal{P}$  outcomes)*  
 *$\mathcal{R} \wedge$  continuous-vnm ( $\mathcal{P}$  outcomes)  $\mathcal{R}$*   
**with** *Von-Neumann-Morgenstern-Utility-Theorem[of outcomes  $\mathcal{R}$ , OF assms]*  
**have**  
   ( $\exists u. \text{ordinal-utility } (\mathcal{P} \text{ outcomes}) \ \mathcal{R} \ (\lambda x. \text{measure-pmf.expectation } x \ u)$ ) **using**  
*assms by blast*  
**then obtain** *u where*  
   *u: ordinal-utility ( $\mathcal{P}$  outcomes)  $\mathcal{R} \ (\lambda x. \text{measure-pmf.expectation } x \ u)$*   
   **by** *auto*  
**have** *vNM-utility outcomes  $\mathcal{R} \ u$*   
**proof** (*standard, goal-cases*)  
**case** 1  
**then show** *?case*  
   **using** *u ordinal-utility.relation-subset-crossp* **by** *blast*  
**next**  
**case** (2  $p \ q$ )

```

then show ?case
  using assms(2) expected-value-is-utility-function fnt u by blast
qed
then show  $\exists u. vNM\text{-utility outcomes } \mathcal{R} u$ 
  by blast
next
assume a:  $\exists u. vNM\text{-utility outcomes } \mathcal{R} u$ 
then have rational-preference (P outcomes) R
  using vnm-utility-imp-reational-prefs by auto
moreover have independent-vnm (P outcomes) R
  using a by (meson assms(2) fnt vNM-utility-implies-independence vnm-utility-is-ordinal-utility)
moreover have continuous-vnm (P outcomes) R
  using a by (meson assms(2) fnt vNM-utility-implies-continuity vnm-utility-is-ordinal-utility)
ultimately show rational-preference (P outcomes) R  $\wedge$  independent-vnm (P outcomes) R  $\wedge$  continuous-vnm (P outcomes) R
  by auto
qed
end

```

## 8 Related work

Formalizations in Social choice theory has been formalized by Wiedijk [13], Nipkow [7], and Gammie [4, 5]. Vestergaard [12], Le Roux, Martin-Dorel, and Soloviev [10, 11] provide formalizations of results in game theory. A library for algorithmic game theory in Coq is described in[1].

Related work in economics includes the verification of financial systems [9], binomial pricing models [3], and VCG-Auctions [6]. In microeconomics we discussed a formalization of two economic models and the First Welfare Theorem [8].

To our knowledge the only work that uses expected utility theory is that of Eberl [2]. Since we focus on the underlying theory of expected utility, we found that there is only little overlap.

## References

- [1] A. Bagnall, S. Merten, and G. Stewart. A library for algorithmic game theory in *ssreflect/coq*. *Journal of Formalized Reasoning*, 10(1):67–95, 2017.
- [2] M. Eberl. Randomised social choice theory. *Archive of Formal Proofs*, May 2016. [http://isa-afp.org/entries/Randomised\\_Social\\_Choice.shtml](http://isa-afp.org/entries/Randomised_Social_Choice.shtml), Formal proof development.

- [3] M. Echenim and N. Peltier. The binomial pricing model in finance: A formalization in Isabelle. In L. de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *LNCS*, pages 546–562. Springer, 2017.
- [4] P. Gammie. Some classical results in social choice theory. *Archive of Formal Proofs*, Nov. 2008. <http://isa-afp.org/entries/SenSocialChoice.html>, Formal proof development.
- [5] P. Gammie. Stable matching. *Archive of Formal Proofs*, Oct. 2016. [http://isa-afp.org/entries/Stable\\_Matching.html](http://isa-afp.org/entries/Stable_Matching.html), Formal proof development.
- [6] M. Kerber, C. Lange, C. Rowat, and W. Windsteiger. Developing an auction theory toolbox. *AISB 2013*, pages 1–4, 2013.
- [7] T. Nipkow. Arrow and Gibbard-Satterthwaite. *Archive of Formal Proofs*, 2008.
- [8] J. Parsert and C. Kaliszyk. Formal Microeconomic Foundations and the First Welfare Theorem. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2018, pages 91–101. ACM, 2018.
- [9] G. O. Passmore and D. Ignatovich. Formal verification of financial algorithms. In L. de Moura, editor, *Automated Deduction - CADE 26*, pages 26–41. Springer, 2017.
- [10] S. L. Roux. Acyclic Preferences and Existence of Sequential Nash Equilibria: A formal and constructive equivalence. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, volume 5674 of *LNCS*, pages 293–309. Springer, 2009.
- [11] S. L. Roux, É. Martin-Dorel, and J. Smaus. An existence theorem of Nash Equilibrium in Coq and Isabelle. In P. Bouyer, A. Orlandini, and P. S. Pietro, editors, *Proceedings Eighth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2017, Roma, Italy, 20-22 September 2017.*, volume 256 of *EPTCS*, pages 46–60, 2017.
- [12] R. Vestergaard. A constructive approach to sequential nash equilibria. *Inf. Process. Lett.*, 97(2):46–51, 2006.
- [13] F. Wiedijk. Formalizing Arrow’s theorem. *Sadhana*, 34(1):193–220, Feb 2009.