# Von Neumann Morgenstern Utility Theorem * 

Julian Parsert Cezary Kaliszyk

September 13, 2023


#### 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 exists. 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

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

[^0]6 Definition of vNM-utility function ..... 47
7 Finite outcomes ..... 49
8 Related work ..... 53

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


## 1 Composition of Probability Mass functions

definition mix-pmf :: real $\Rightarrow{ }^{\prime}$ a $p m f \Rightarrow{ }^{\prime} a p m f \Rightarrow{ }^{\prime} a p m f$ where mix-pmf $\alpha$ p $q=($ bernoulli-pmf $\alpha) \geqslant(\lambda X$. if $X$ then $p$ else $q)$
lemma $p m f$-mix: $a \in\{0 . .1\} \Longrightarrow p m f($ mix-pmf $a p q) x=a * p m f p x+(1-$ a) * $p m f q x$
by (simp add: mix-pmf-def pmf-bind)
lemma pmf-mix-deeper: $a \in\{0 . .1\} \Longrightarrow p m f($ mix-pmf a $p q) x=a * p m f p x+$ $p m f q x-a * p m f q x$
by (simp add: left-diff-distrib' $p m f-m i x$ )
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: $a \in\{0<. .<1\} \Longrightarrow$ set-pmf (mix-pmf a p q) $=$ set-pmf $p \cup$ set-pmf $q$
by (auto simp add: mix-pmf-def split: if-splits)
lemma set-pmf-mix-eq: $a \in\{0 . .1\} \Longrightarrow$ mix-pmf a $p p=p$
by (simp add: mix-pmf-def)
lemma pmf-equiv-intro[intro]:
assumes $\bigwedge e . e \in \operatorname{set}-p m f p \Longrightarrow p m f p e=p m f q e$
assumes $\wedge e . e \in \operatorname{set}-p m f q \Longrightarrow p m f q e=p m f p e$

```
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 \e.e e set-pmf p\Longrightarrowpmf pe=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-eqals:
    assumes a \in{0..1}
    shows mix-pmf a p q= mix-pmf (1-a)qp
proof -
    have fst: }\forallx\in\mathrm{ set-pmf p.pmf (mix-pmf a p q) x = pmf (mix-pmf (1-a)q p) x
    proof
    fix }
    assume x \in set-pmf p
    have pmf (mix-pmf a pq) x =a*pmf p x + (1-a)*pmf q x
        using pmf-mix[of a p q x ] assms by blast
    also have ... =a*pmf px+pmf qx-a*pmf qx
        by (simp add: left-diff-distrib)
    from pmf-mix[of 1-a q p x] assms
    have pmf (mix-pmf (1-a) q p)x=(1-a)*pmf qx+(1-(1-a))*
pmf p x
            by auto
    then show pmf (mix-pmf a p q) x = pmf (mix-pmf (1 - a)q q ) x
        using calculation by auto
    qed
    have }\forallx\in\mathrm{ set-pmf q. pmf (mix-pmf a p q) x = pmf (mix-pmf (1-a)qp)x
    proof
        fix }
        assume x f set-pmf q
    have pmf (mix-pmf a p q) x = a*pmf p x + (1 - a)*pmf qx
            using pmf-mix[of a p q x] assms by blast
    also have ... =a*pmf px+pmfqx-a*pmf qx
        by (simp add: left-diff-distrib)
    from pmf-mix[of 1-a q p x] assms
    show pmf (mix-pmf a p q) x = pmf (mix-pmf (1 - a) q p)x
            using calculation by auto
qed
    then have }\forallx\inset-pmf (mix-pmf a p q).pmf (mix-pmf a pq) x = pmf
(mix-pmf (1-a)qp) 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::real)
```

```
    and }\beta\in{0..(1::real)
    assumes \alpha>\beta
    shows pmf (mix-pmf (\beta/\alpha) (mix-pmf \alpha pq)q) e= \beta*pmf pe+pmfqe-
\beta*pmfqe
proof
    let ?l = (mix-pmf (\beta/\alpha)(mix-pmf \alpha p q) q)
    have fst: pmf ?l e=(\beta/\alpha)*pmf (mix-pmf \alpha p q) e+(1-\beta/\alpha)*pmf 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 pmf (mix-pmf \alpha p q) e=\alpha*pmf pe+(1-\alpha)*pmf q e
    using pmf-mix[of \alpha p q] assms(2) assms(3) assms(1) by blast
    have pmf ?l e = (\beta/\alpha)*(\alpha*pmf pe+(1-\alpha)*pmf qe)+(1-\beta/\alpha)*pmf
q e
    using fst assms(1) pmf-mix by fastforce
    then have pmf ?l e = ((\beta/\alpha)*\alpha*pmf p e + (\beta/\alpha)*(1-\alpha)*pmf q e) +
(1-\beta/\alpha)*pmf q e
    using fst assms(1) by (metis mult.assoc ring-class.ring-distribs(1))
    then have *:pmf ?l e = ( \beta*pmf pe+(\beta/\alpha)*(1-\alpha)*pmf q e) + (1-\beta/\alpha)
* pmf q e
    using fst assms(1) assms(2) assms(3) by auto
    then have pmf ?l e e ( \beta*pmf pe+((\beta/\alpha)-(\beta/\alpha)*\alpha)*pmf q e) +(1-\beta/\alpha)
* pmf q e
            using fst assms(1) assms(2) assms(3) by (simp add: * diff-divide-distrib
right-diff-distrib')
    then have pmf ?l e = (\beta*pmf pe+((\beta/\alpha)-\beta)*pmf qe)+(1-\beta/\alpha)*pmf
q e
            using fst assms(1) assms(2) assms(3) by auto
    then have pmf ?l e=( }\beta*pmfpe+(\beta/\alpha)*pmfqe-\beta*pmfqe)+1
pmf q e- \beta/\alpha* pmf 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 mix-pmf ( }\beta/\alpha)(\mathrm{ mix-pmf 人 p q) q = mix-pmf }\beta\mathrm{ p q(is ?l = ?r)
proof (rule pmf-equiv-intro1[symmetric])
    fix }
    assume a: e\in set-pmf ?r
    have e\in 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 pmf ?l e = \beta*pmf pe-\beta*pmf qe+pmfqe
            using pmf-mix-deeper[of \beta/\alpha p q e] mix-pmf-comp-left-div[of \alpha \beta p q e] assms
by auto
    then show pmf(mix-pmf \beta pq) e= pmf(mix-pmf (\beta/\alpha)(mix-pmf \alpha 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 mix-pmf a (mix-pmf b p q) q= mix-pmf (a*b) pq
proof -
    define }\gamma\mathrm{ where g: }\gamma=(a*b
    define l where l:l=(mix-pmf b p q)
    define r where r:r = mix-pmf (a*b) p q
    have y:}\gamma\in{0..1
    using assms(2) mult-le-one assms g by auto
    have alt-: \foralle\in set-pmf l. pmfre r % * pmf pe+pmf qe-\gamma*pmf qe
    proof
    fix e
    have pmf re=\gamma*pmf pe+(1-\gamma)*pmf qe
        using <\gamma\in{0..1}> g pmf-mix r by fastforce
    moreover have ... = \gamma*pmf pe+1*pmf qe-\gamma*pmfqe
        by (simp add: algebra-simps)
    moreover have ... = pmf (mix-pmf \gamma pq)e
        using calculation gr by auto
    moreover have ... = \gamma * pmf pe+pmf qe-\gamma*pmf qe
        using calculation by auto
    ultimately show pmf re=\gamma*pmf pe+pmf qe-\gamma*pmf qe
        by auto
    qed
    have }\foralle\inset-pmfr.pmf l e=b*pmf pe+pmfqe-b*pmf q
        using allI pmf-mix-deeper assms(2) l by fastforce
    have mix-pmf a (mix-pmf b p q) q = mix-pmf (a*b) pq
    proof (rule ccontr)
    assume neg:\negmix-pmf a (mix-pmf b p q) q = mix-pmf (a*b) pq
    then have b: b\not=0
        by (metis (no-types) assms(1) mult-cancel-right2 pmf-mix-0 set-pmf-mix-eq)
    have f3: b-(a*b)>0\longrightarrow mix-pmf a (mix-pmf b pq) q= 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 (set-pmf (mix-pmf a p q))\subseteq set-pmf p\cup set-pmf q
proof -
```

```
    have \(a \in\{0<. .<1\} \Longrightarrow\) ?thesis
    by (simp add: set-pmf-mix)
    moreover have \(a=1 \Longrightarrow\) ?thesis
    by \(\operatorname{simp}\)
    moreover have \(a=0 \Longrightarrow\) ?thesis
    by \(\operatorname{simp}\)
ultimately show ?thesis
    using assms less-eq-real-def by auto
qed
lemma mix-pmf-preserves-finite-support:
    assumes \(a \in\{0 . .1\}\)
    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 \((\exists x . p m f p x=1) \longleftrightarrow\) card \((\) set-pmf \(p)=1\)
proof (rule iffI, goal-cases)
    case 1
    show ?case
    proof (rule ccontr)
        assume neg: \(\neg \operatorname{card}(\) set-pmf \(p)=1\)
        then have card \((\operatorname{set}-p m f p) \neq 1\)
        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: ( \(\sum i \in\) set-pmf \(p\).pmf \(\left.p i\right)=1\)
    using sum-pmf-eq-1[of set-pmf \(p\) p] by auto
    have set-pmf-nemtpy: set-pmf \(p \neq\{ \}\)
    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 \(\operatorname{card}(\) set-pmf \(p)>1 \longrightarrow\left(\sum i \in \operatorname{set}-p m f\right.\) p.pmf \(\left.p i\right)>1\)
    proof
    assume card (set-pmf p) > 1
    have \(\exists x y\).pmf \(p x=1 \wedge y \neq x \wedge y \in \operatorname{set}-p m f p\)
        using set-pmf-nemtpy is-singleton \(I^{\prime}\) is-singleton-altdef
        by (metis 1 neg)
    then show \(\left(\sum i \in\right.\) set-pmf p.pmf pi)>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
```

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 $p m f$
assumes $y \in$ set-pmf $p$ set-pmf $p-\{y\} \neq\{ \}$
obtains $a q$ where $a \in\{0<. .<1\}$ set-pmf $q=$ set-pmf $p-\{y\}$ $p=$ mix-pmf a $q$ (return-pmf $y)$
proof -
from assms obtain $x$ where $x: x \in$ set-pmf $p-\{y\}$ by auto
define $a$ where $a=1-p m f$ p $y$
have $a-n 1: a \neq 1$
by (simp add: a-def assms(1) pmf-eq-0-set-pmf)
have 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 $y$ : pmf p $y<1$ using $p m f-l e-1[o f p y]$ unfolding $a$-def by linarith
hence $a$ : $a>0$ by (simp add: a-def)
define $q$ where $q=$ embed-pmf ( $\lambda z$. if $z=y$ then 0 else pmf $p z / a$ )
have $q$ : pmf $q z=($ if $z=y$ then 0 else pmf $p z / a)$ for $z$
unfolding $q$-def
proof (rule pmf-embed-pmf)
have $1=\left(\int^{+} x\right.$. ennreal (pmf px) dcount-space UNIV)
by (rule nn-integral-pmf-eq-1 [symmetric])
also have $\ldots=\left(\int+{ }^{+}\right.$. ennreal $(p m f p x) *$ indicator $\{y\} x+$
ennreal (pmf p $x$ ) * indicator $(-\{y\}) x$ dcount-space UNIV)
by (intro nn-integral-cong) (auto simp: indicator-def)
also have $\ldots=\left(\int^{+} x\right.$. ennreal $(p m f \quad p x) *$ indicator $\{y\} x$ dcount-space UNIV) +
$\left(\int{ }^{+}\right.$x. ennreal ( $p m f$ p $\left.x\right) *$ indicator $(-\{y\}) x$ dcount-space UNIV)
by (subst nn-integral-add) auto
also have $\left(\int^{+} x\right.$. ennreal $(p m f p x) *$ indicator $\{y\} x$ dcount-space UNIV $)=$ pmf $p y$
by (subst nn-integral-indicator-finite) auto
also have ennreal (pmf py) $+\left(\int^{+}\right.$x. ennreal $(p m f p x) *$ indicator $(-\{y\}) x$
dcount-space UNIV)

- ennreal $($ pmf $p y)=\left(\int+x\right.$. ennreal $(p m f p x) *$ indicator $(-\{y\})$
x Dcount-space UNIV)
by simp
also have $1-\operatorname{ennreal}(p m f p y)=\operatorname{ennreal}(1-p m f p y)$
by (subst ennreal-1 [symmetric], subst ennreal-minus) auto
finally have eq: $\left(\int^{+} x \in-\{y\}\right.$. ennreal (pmf $\left.p x\right)$ dcount-space UNIV) $=1-$ pmf p y ..
have $\left(\int^{+}\right.$x. ennreal (if $x=y$ then 0 else pmf $\left.p x / a\right)$ dcount-space UNIV) $=$
$\left(\int+x\right.$. inverse $a *($ ennreal (pmf px)*indicator $(-\{y\}) x)$ dcount-space UNIV)
using $a$ by (intro nn-integral-cong) (auto simp: divide-simps ennreal-mult' [symmetric])
also have $\ldots=$ inverse $a *\left(\int^{+} x \in-\{y\}\right.$. ennreal (pmf px) dcount-space UNIV)
using $a$ by (subst nn-integral-cmult [symmetric]) (auto simp: ennreal-mult') also note eq
also have ennreal (inverse a) * ennreal (1-pmf p y) $=$ ennreal $((1-p m f p$ y) $/ a)$
using a by (subst ennreal-mult' [symmetric]) (auto simp: field-simps)
also have $(1-p m f p y) / a=1$ using $y$ by (simp add: $a$-def)
finally show $\left(\int^{+} x\right.$. ennreal (if $x=y$ then 0 else pmf $p x / a$ ) dcount-space $U N I V)=1$
by simp
qed (insert a, auto)
have mix-pmf $(1-p m f p y) q($ return-pmf $y)=p$
using $y$ by (intro pmf-eqI) (auto simp: q pmf-mix pmf-le-1 a-def)
moreover have set-pmf $q=$ set-pmf $p-\{y\}$
using $y$ by (auto simp: q set-pmf-eq a-def)
ultimately show ?thesis using that $[$ of $1-p m f$ p $y$ q] y assms by (auto simp: set-pmf-eq)
qed
lemma pmf-mix-induct [consumes 2, case-names degenerate mix]:
assumes finite $A$ set-pmf $p \subseteq A$
assumes degenerate: $\bigwedge x . x \in A \Longrightarrow P($ return-pmf $x)$
assumes mix: $\quad \bigwedge p a y$. set-pmf $p \subseteq A \Longrightarrow a \in\{0<. .<1\} \Longrightarrow y \in A \Longrightarrow$ $P p \Longrightarrow P($ mix-pmf a $p($ return-pmf $y))$
shows $P p$
proof -
have finite (set-pmf $p$ ) set-pmf $p \neq\{ \}$ set-pmf $p \subseteq A$
using $\operatorname{assms}(1,2)$ 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 $p=$ 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 $x \in$ set-pmf $p$ set-pmf $p-\{x\} \neq\{ \}$ by auto
from mix-pmf-partition [OF this] obtain a $q$
where decomp: $a \in\{0<. .<1\}$ set-pmf $q=$ set-pmf $p-\{x\}$
$p=$ mix-pmf a $q$ (return-pmf $x$ ) by blast
have $P($ mix-pmf a $q($ return-pmf $x))$
using insert.prems decomp(1) insert.hyps
by (intro mix insert) (auto simp: decomp(2))
with decomp (3) show? ?ase by simp
qed
qed
lemma pmf-mix-induct' [consumes 2, case-names degenerate mix]:
assumes finite $A$ set-pmf $p \subseteq A$
assumes degenerate: $\bigwedge x . x \in A \Longrightarrow P($ return-pmf $x)$
assumes mix: $\quad \bigwedge p q a$. set-pmf $p \subseteq A \Longrightarrow$ set-pmf $q \subseteq A \Longrightarrow a \in\{0<. .<1\}$


## $\Longrightarrow$

$$
P p \Longrightarrow P q \Longrightarrow P(\text { mix-pmf a } p q)
$$

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 $\left(\sum i \in \operatorname{set}-p m f(\right.$ mix-pmf a p q). pmf (mix-pmf a p q) $i)=\left(\sum i \in s e t-p m f\right.$
p. $a * p m f p i)+\left(\sum i \in \operatorname{set}-p m f q .(1-a) * p m f q i\right)$
proof -
have fst: $\left(\sum i \in \operatorname{set}-p m f(\operatorname{mix}-p m f\right.$ a $p q) . p m f(\operatorname{mix}-p m f$ a $\left.p q) i\right)=1$
using sum-pmf-eq-1 assms by auto
have ( $\left.\sum i \in \operatorname{set}-p m f p . a * p m f p i\right)=a *\left(\sum i \in s e t-p m f\right.$ p. pmf $\left.p i\right)$
by (simp add: sum-distrib-left)
also have $\ldots=a * 1$
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 $\left(\sum i \in\right.$ set-pmf T. $\left.a * p m f p i * f i\right)=a *\left(\sum i \in s e t-p m f T\right.$.pmf $\left.p i * f i\right)$
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 $\left(\sum i \in T . a * p m f p i * f i\right)=\left(\sum i \in s e t-p m f p . a * p m f p i * f i\right)$
proof -
consider (eq) set-pmf $p=T \mid(s u b)$ set-pmf $p \subset T$
using assms by blast
then show ?thesis
proof (cases)
next
case sub
define $A$ where $A=T-($ 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 $A \cap$ set-pmf $p=\{ \}$ using $A$-def assms(1) by blast
ultimately have $*:\left(\sum i \in T . a * p m f p i * f i\right)=\left(\sum i \in s e t-p m f p . a * p m f p i\right.$ $* f i)+\left(\sum i \in A . a * p m f p i * f i\right)$
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 $\forall e \in A$. pmf $p e=0$
by (simp add: $A$-def pmf-eq-0-set-pmf)
hence $\left(\sum i \in A . a * p m f p i * f i\right)=0$
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 $*$ : ( $\sum i \in \operatorname{set}-p m f(m i x-p m f$ a $\left.p q) . a * p m f p i * f i\right)=a *\left(\sum i \in \operatorname{set}-p m f\right.$ (mix-pmf a p q). pmf $p i * f i$ )
by (metis (mono-tags, lifting) mult.assoc sum.cong sum-distrib-left)
have $* *:\left(\sum i \in \operatorname{set}-p m f(\right.$ mix-pmf a p $\left.q) .(1-a) * p m f q i * f i\right)=(1-a) *\left(\sum i\right.$ $\in \operatorname{set}-p m f($ 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 pq) $f=\left(\sum i \in\right.$ set-pmf (mix-pmf a p q). pmf (mix-pmf a pq) $i * f i)$
by (metis fn pmf-integral-code-unfold pmf-integral-pmf-set-integral pmf-set-integral-code-alt-finite)
also have $g: \ldots=\left(\sum i \in \operatorname{set}-\mathrm{pmf}(\operatorname{mix}-p m f\right.$ a $p q) .(a * p m f p i+(1-a) * p m f$ $q i) * f i)$
using $p m f-m i x[o f$ a $p q]$ assms(3) by auto
also have $* * * *: \ldots=\left(\sum i \in \operatorname{set}-\mathrm{pmf}(\operatorname{mix}-p m f\right.$ a $p q) . a * p m f p i * f i+(1-a)$ * pmf $q i * f i)$
by (simp add: mult.commute ring-class.ring-distribs(1))
also have $f: \ldots=\left(\sum i \in \operatorname{set}-\mathrm{pmf}(\operatorname{mix}-\mathrm{pmf}\right.$ a $\left.p q) . a * \operatorname{pmf} p i * f i\right)+\left(\sum i \in\right.$ set-pmf (mix-pmf a p q). $(1-a) * p m f q i * f i)$
by (simp add: sum.distrib)
also have $\ldots=a *\left(\sum i \in \operatorname{set}-\mathrm{pmf}(\operatorname{mix}-p m f\right.$ a p q). pmf $p i * f i)+(1-a) *$ ( $\sum i \in \operatorname{set}-p m f(m i x-p m f$ a $p q) . p m f q i * f i$ )
using $* * *$ by $\operatorname{simp}$
also have $h$ : $\ldots=a *\left(\sum i \in \operatorname{set}-p m f p . p m f p i * f i\right)+(1-a) *\left(\sum i \in \operatorname{set}-p m f\right.$ q. pmf $q i * f i)$

```
    proof -
```

    have \(\left(\sum i \in\right.\) set-pmf (mix-pmf a p q). pmf \(\left.p i * f i\right)=\left(\sum i \in\right.\) set-pmf p. pmf
    $p i * f i)$
using subsets sum-over-subset-pmf-support[of (mix-pmf a p q) p 1 f] fn by
auto
moreover have $\left(\sum i \in \operatorname{set}-p m f(m i x-p m f\right.$ a p q). pmf q $i * f i)=\left(\sum i \in\right.$
set-pmf q. pmf $q i * f i)$
using subsets sum-over-subset-pmf-support[of (mix-pmf a $p$ q) q $1 f] f n \mathbf{b y}$
auto
ultimately show ?thesis
by (simp)
qed
finally show?thesis
proof -
have $\left(\sum i \in\right.$ set-pmf $q$. pmf $\left.q i * f i\right)=$ 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 ( $\sum i \in$ set-pmf p.pmf $p i * f i$ ) 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 ( $\operatorname{simp} a d d$ : * ** *** **** $f g h)$
qed
qed
lemma expected-value-mix-pmf:
assumes finite (set-pmf p)
and finite (set-pmf q)
assumes $a \in\{0 . .1\}$
shows measure-pmf.expectation (mix-pmf a $p q$ ) $f=a *$ measure-pmf.expectation
$p f+(1-a) *$ measure-pmf.expectation $q f$
proof -
consider (0) $a=0|(b) a \in\{0<. .<1\}|(1) a=1$
using assms(3) less-eq-real-def by auto
then show?thesis
proof (cases)
case 0
have $($ mix-pmf a p $q)=q$
using 0 pmf-mix-0 by blast
have measure-pmf.expectation $q f=(1-a) *$ measure-pmf.expectation $q f$
by ( $\operatorname{simp}$ 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
```

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

```
begin
```


## 2 Lotteries

definition lotteries-on
where

```
lotteries-on Oc ={p.(set-pmf p)\subseteqOc}
```

lemma lotteries-on-subset:
assumes $A \subseteq B$
shows lotteries-on $A \subseteq$ lotteries-on $B$
by (metis (no-types, lifting) Collect-mono assms gfp.leq-trans lotteries-on-def)
lemma support-in-outcomes:
$\forall o c . \forall p \in$ lotteries-on oc. $\forall a \in$ set-pmf $p . a \in o c$
by (simp add: lotteries-on-def subsetD)
lemma lotteries-on-nonempty:
assumes outcomes $\neq\{ \}$
shows lotteries-on outcomes $\neq\{ \}$
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 $\forall l \in$ 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 $\forall l \in$ lotteries-on outcomes. card $($ set-pmf $l)=1$

## proof

fix $l$
assume $l \in$ lotteries-on outcomes

```
    have finite outcomes
    using assms card.infinite by force
    then have l\inlotteries-on outcomes \longrightarrow1= card (set-pmf l)
    by (metis assms card-eq-O-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 card (set-pmf l)=1
    by (simp add: <l \in lotteries-on outcomes`)
qed
lemma finite-nempty-ex-degernate-in-lotteries:
    assumes out }\not={
    assumes finite out
    shows }\existse\in\mathrm{ lotteries-on out. }\existsx\in\mathrm{ out.pmf e x=1
proof (rule ccontr)
    assume a: }\neg(\exists e\inlotteries-on out. \existsx\inout.pmf e x=1)
    then have subset: }\foralle\in\mathrm{ lotteries-on out. set-pmf e }\subseteq\mathrm{ out
        using lotteries-on-def by (simp add: lotteries-on-def)
    then have }\foralle.e\inlotteries-on out \longrightarrow((\sumi\inset-pmf e.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 card (set-pmf p)=1
    shows }\foralle\in\mathrm{ set-pmf p. 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 card outcomes = 1
    shows card (lotteries-on outcomes) = 1
proof -
    obtain x :: ' }a\mathrm{ where
    x:outcomes ={x}
    using assms card-1-singletonE by blast
    have exl: \existsl\inlotteries-on outcomes. pmf l x = 1
    by (metis x assms card.infinite empty-iff
        finite-nempty-ex-degernate-in-lotteries insert-iff zero-neq-one)
    have pmfs:}\foralll\in\mathrm{ lotteries-on outcomes. set-pmf l}={x
    by (simp add: lotteries-on-def set-pmf-subset-singleton x)
    have }\foralll\inlotteries-on outcomes. pmf lx=
    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:
```



```
proof-
    have {return-pmf x |x.x\inS} = return-pmf 'S
        by blast
    also have card ... = card S
        by (intro card-image) (auto simp: inj-on-def)
    finally show card {return-pmf x |x.x 隹}= card S .
qed
lemma mix-pmf-in-lotteries:
    assumes p\inlotteries-on A
        and q\in lotteries-on A
    and }a\in{0<..<1
    shows (mix-pmf a p q) \in lotteries-on A
proof -
    have set-pmf (mix-pmf a p q)= set-pmf p\cup 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 card {x\inlotteries-on out. card (set-pmf x)=1}=card out
proof -
    consider (empty) out ={}|(not-empty) out }\not={
    by blast
    then show ?thesis
    proof (cases)
    case not-empty
    define }DG\mathrm{ where
            DG:DG={x\in lotteries-on out. card (set-pmf x)=1}
    define AP where
        AP:AP}={\mathrm{ return-pmf }x|x.x\in\mathrm{ out }
    have **: card AP = card out
            using AP return-pmf-card-equals-set by blast
    have *: }\foralld\inDG.d\inA
    proof
            fix l
            assume l 
            then have l\inlotteries-on out }\wedge1=\operatorname{card}(\mathrm{ set-pmf l)
                using DG by force
            then obtain }x\mathrm{ where
                x:x\in out 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\inAP
            using AP x (1) by blast
        qed
        moreover have AP=DG
        proof
            have }\foralle\inAP.e\in lotteries-on ou
                by(auto simp: lotteries-on-def AP)
        then show AP\subseteqDG 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 mix-pmf and the element to the left q/p changes

```
definition independent-vnm
    where
        independent-vnm C P =
    (\forallp\inC.\forallq\inC.\forallr\inC.\forall(\alpha::real) }\in{0<..1}.p\succeq[P]q\longleftrightarrow mix-pmf \alpha 
r\succeq[P] mix-pmf \alphaqr)
lemma independent-vnmI1:
    assumes (\forallp\inC.\forallq\inC.\forallr\inC.\forall\alpha\in{0<..1}.p\succeq[P]q\longleftrightarrow mix-pmf \alpha
pr\succeq[P] mix-pmf \alpha qr)
    shows independent-vnm C P
    using assms independent-vnm-def by blast
```

lemma independent-vnmI2:
assumes $\bigwedge p q r \alpha . p \in C \Longrightarrow q \in C \Longrightarrow r \in C \Longrightarrow \alpha \in\{0<. .1\} \Longrightarrow p \succeq[P]$
$q \longleftrightarrow$ mix-pmf $\alpha$ pr $\succeq[P]$ 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 $C P \longleftrightarrow(\forall p \in C . \forall q \in C . \forall r \in C . \forall \alpha \in\{0<. .<1\}$.
$p \succeq[P] q \longleftrightarrow$ mix-pmf $\alpha$ pr $\succeq[P]$ mix-pmf $\alpha q r$ ) (is ? $L \longleftrightarrow ? R)$
proof (rule iffI)
assume $a$ : ? R
have independent-vnm $C P$
$\mathbf{b y}$ (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 $(\forall p \in C . \forall q \in C . \forall r \in C . \forall(\alpha::$ real $) \in\{0<. .1\} . p \succeq[P] q \longleftrightarrow$ mix-pmf
$\alpha p r \succeq[P]$ 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 as $4:(\alpha::$ real $) \in\{0<. .1\}$
then show $p \succeq[P] q=$ mix-pmf $\alpha p r \succeq[P]$ 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 $(\forall p \in C . \forall q \in C . \forall r \in C . \forall \alpha \in\{0<. .1\} . p \succeq[P] q \longleftrightarrow$ mix-pmf $\alpha p$ $r \succeq[P]$ 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 mix-pmf $\alpha$ pr$\succeq[P]$ mix-pmf $\alpha q r$
using assms independece-dest-alt by blast

```
lemma independent-vnmD3:
    fixes pqr \alpha
    assumes }\alpha\in{0<..1
        and p\inC
        and q\inC
        and}r\in
    assumes independent-vnm C P
    assumes mix-pmf \alpha pr\succeq[P] mix-pmf \alpha qr
    shows p\succeq[P] q
    using assms independece-dest-alt by blast
lemma independent-vnmD4:
    assumes independent-vnm C P
    assumes refl-on C P
    assumes p\inC
        and q\inC
        and r}\in
        and}\alpha\in{0..1
        and p\succeq[P]q
    shows mix-pmf \alpha pr\succeq[P] mix-pmf \alpha q r
    using assms
    by (cases }\alpha=0\vee\alpha\in{0<..1},metis assms(1,2,3,4
        independece-dest-alt pmf-mix-0 refl-onD, auto)
lemma approx-indep-ge:
    assumes }x\approx[\mathcal{R}]
    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 }\forallr\inlotteries-on outcomes. (mix-pmf \alpha y r)\succeq[\mathcal{R}](mix-pmf \alpha xr),
proof
    fix r
    assume a:r\in lotteries-on outcomes (is r f ?lo)
    have clct: }y\succeq[\mathcal{R}]x\wedge\mathrm{ independent-vnm ?lo }\mathcal{R}\wedgey\in?lo \wedge x\in?lo \wedger\in?l
        by (meson a assms(1) assms(2) atLeastAtMost-iff greaterThanAtMost-iff
        ind preference-def rational-preference-def rpr)
    then have in-lo: mix-pmf \alpha y r f ?lo (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\vee0<\alpha
        using assms by auto
    then show mix-pmf \alpha y r\succeq[\mathcal{R}] 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}]
    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 \forallr lotteries-on outcomes. (mix-pmf \alpha yr) \approx[\mathcal{R}](mix-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}]
    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-pmf \alpha x y) \succeq[\mathcal{R}] y
proof -
    have xy-p:x\in(lotteries-on outcomes) y\in(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-pmf \alpha x y) (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: }\forallr\in\mathrm{ ?lot. (mix-pmf < x r) }\succeq[\mathcal{R}] (mix-pmf \alpha y r)
        by (metis assms assms(2) atLeastAtMost-iff greaterThanAtMost-iff indepen-
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}]
    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-pmf \alpha y x)\succeq[\mathcal{R}] y
proof -
    define }\beta\mathrm{ where
        b: \beta=1-\alpha
    have }\beta\in{0..1
        using assms(4) b by auto
    then have mix-pmf \beta xy\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-pmf \betaxy=mix-pmf \alpha y x
    by (metis assms(4) b pmf-inverse-switch-eqals)
    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::real) \(\}\)
    shows \((m i x-p m f \alpha x y) \succ[\mathcal{R}] y\)
proof -
    have \(x y\) - \(p: x \in\) (lotteries-on outcomes) \(y \in\) (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-pmf \(\alpha x y) \in(\) 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. ( mix-pmf \(\alpha x r) \succeq[\mathcal{R}](\) mix- \(p m f \alpha y r)\)
        by (metis assms \((1,3,4)\) independece-dest-alt ind \(x y\)-p)
    have ( 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. ( mix-pmf \(\alpha x r) \succ[\mathcal{R}](\) mix-pmf \(\alpha\) y \(r)\)
        using assms(1) assms(4) ind independece-dest-alt \(x y-p(1) x y-p(2)\) 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 \(C P=(\forall p \in C . \forall q \in C . \forall r \in C . p \succeq[P] q \wedge q \succeq[P] r \longrightarrow\) \((\exists \alpha \in\{0 . .1\} .(\) mix-pmf \(\alpha p r) \approx[P] q))\)
lemma continuous-vnmD:
assumes continuous-vnm \(C P\)
shows \((\forall p \in C . \forall q \in C . \forall r \in C . p \succeq[P] q \wedge q \succeq[P] r \longrightarrow\) \((\exists \alpha \in\{0 . .1\} .(\) mix-pmf \(\alpha p r) \approx[P] q))\)
using continuous-vnm-def assms by blast
lemma continuous-vnmI:
assumes \(\wedge p q r . p \in C \Longrightarrow q \in C \Longrightarrow r \in C \Longrightarrow p \succeq[P] q \wedge q \succeq[P] r \Longrightarrow\) \(\exists \alpha \in\{0 . .1\} .(\) mix-pmf \(\alpha\) pr) \(\approx[P] q\)
shows continuous-vnm \(C P\)
by (simp add: assms continuous-vnm-def)
lemma mix-in-lot:
assumes \(x \in\) lotteries-on outcomes
and \(y \in\) lotteries-on outcomes
and \(\alpha \in\{0 . .1\}\)
```

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

lemma non-unique-continuous-unfolding:
assumes cnt: continuous-vnm (lotteries-on outcomes) $\mathcal{R}$
assumes 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 $\exists \alpha \in\{0 . .1\} . q \approx[\mathcal{R}]$ mix-pmf $\alpha p r$
using assms(1) assms(2) cnt continuous-vnmD assms
proof -
have $\forall p$. $p \in$ (lotteries-on outcomes $) \wedge q \in($ lotteries-on outcomes $) \longleftrightarrow p \succeq[\mathcal{R}]$
$q \vee q \succeq[\mathcal{R}] p$
using assms rational-preference.compl[of lotteries-on outcomes $\mathcal{R}]$
by (metis (no-types, opaque-lifting) preference-def rational-preference-def)
then show ?thesis
using continuous-vnmD $[$ OF assms(1) $]$ by (metis assms(3) 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 outcomes :: 'a set
    fixes }\mathcal{R
    assumes rpr: rational-preference (lotteries-on outcomes) \mathcal{R}
    assumes ind: independent-vnm (lotteries-on outcomes) }\mathcal{R
begin
abbreviation \mathcal{P}\equivlotteries-on outcomes
lemma relation-in-carrier:
    x\succeq[\mathcal{R}] y\Longrightarrowx\in\mathcal{P}\wedge y\in\mathcal{P}
    by (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 mix-pmf \alpha pr \succeq[\mathcal{R}] mix-pmf \alpha qr
    using ind by (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}]
    shows mix-pmf \alpha p r}\succ[\mathcal{R}] mix-pmf \alpha qr
    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 mix-pmf \alpha pr\succeq[\mathcal{R}] mix-pmf \alpha qr
    shows p\succeq[\mathcal{R}]q
proof -
    have mix-pmf \alpha pr 
        using assms mix-in-lot relation-in-carrier by blast
    moreover have mix-pmf \alpha qr < \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}]
    assumes }b\in{0<..(1::\mathrm{ real )}
    shows }x\succ[\mathcal{R}] mix-pmf b y x
proof -
    consider b=1 | b\not=1
    by blast
    then show ?thesis
    proof (cases)
    case 2
    have sg: (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 mix-pmf b x y \in\mathcal{P}
        by (meson sg preference-def rational-preference-def rpr)
    have mix-pmf b x x f \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 mix-pmf b x x \succ[\mathcal{R}] mix-pmf b y x
        using mix-pmf-preferred-independence[of x b] assms
        by (meson <b \in{0<..<1}> 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}]
    assumes }\alpha\in{0<..<1
    shows u\succ[\mathcal{R}] mix-pmf \alpha uv
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-eqals x-sg-y-sg-mpmf-right)
qed
lemma neumann-3B-b-non-strict:
    assumes }u\succeq[\mathcal{R}]
    assumes }\alpha\in{0..1
    shows u\succeq[\mathcal{R}] mix-pmf \alphauv
proof -
    have f2: mix-pmf \alpha (u::'a pmf) v = mix-pmf (1-\alpha) vu
        using pmf-inverse-switch-eqals 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}]
    assumes \alpha}\in{0<..<(1::real)
    and }\beta\in{0<..<(1::\mathrm{ real ) }
    assumes \beta>\alpha
    shows (mix-pmf \beta}vu)\succ[\mathcal{R}](\mathrm{ mix-pmf 人 vu)
proof -
    define t where
    t:t= mix-pmf \betavu
    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<
```

```
    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}vu\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)vu
    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-eqals x-sg-y-sg-mpmf-right)
    have }t\succ[\mathcal{R}]
    using assms(1) assms(3) ind rpr sg-imp-mix-sg t by fastforce
    hence t-s:t\succ[\mathcal{R}](mix-pmf \gamma tu)
    proof -
        have (mix-pmf \gamma tu)\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 vu)
        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 \betavu}\succ[\mathcal{R}] mix-pmf \alphav
    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 $(e x) \alpha=1 \wedge \beta=0 \mid(m) \alpha \neq 1 \vee \beta \neq 0$
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 \(m x-p:(m i x-p m f \alpha x y) \in \mathcal{P}\)
        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 \(m x-p\) 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) \(2 \operatorname{assms(3)~by~auto~}\)
    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: \(\operatorname{assms}(1)\) )
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 $\alpha$ pr $\approx[\mathcal{R}]$ 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 mix-pmf \(\alpha\) r \(p \approx[\mathcal{R}]\) mix-pmf \(\alpha r q\)
proof -
    have 1: \(\alpha \leq 1 \wedge \alpha \geq 01-\alpha \in\{0 . .1\}\)
        using assms(3) by auto+
    have fst: mix-pmf \(\alpha\) r \(p \approx[\mathcal{R}]\) mix-pmf \((1-\alpha)\) pr
        using assms by (metis mix-in-lot pmf-inverse-switch-eqals
        rational-preference.compl relation-in-carrier rpr)
    moreover have mix-pmf \(\alpha\) r \(p \approx[\mathcal{R}]\) mix-pmf \(\alpha\) r \(q\)
        using approx-remains-after-same-comp[of - - \(\alpha\) ] pmf-inverse-switch-eqals[of \(\alpha\)
p q] 1
            pmf-inverse-switch-eqals 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 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 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 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}] mix-pmf \alpha pq
proof -
    consider }p\succeq[\mathcal{R}]q|q\succeq[\mathcal{R}]
        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}] mix-pmf \alpha pq
            using assms(3) neumann-3B-b-non-strict calculation by blast
        moreover show ?thesis
            using rpr preference.transitivity[of \mathcal{P }
            by (meson assms(1) calculation(2) rational-preference-def transE)
    next
        case 2
        moreover have q\succeq[\mathcal{R}] 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 \mathcal{P }
            by (meson assms(2) calculation(2) rational-preference-def transE)
        qed
qed
private definition degenerate-lotteries where
    degenerate-lotteries }={x\in\mathcal{P}.card (set-pmf x)=1
private definition best where
    best }={x\in\mathcal{P}.(\forally\in\mathcal{P}.x\succeq[\mathcal{R}]y)
private definition worst where
    worst }={x\in\mathcal{P}.(\forally\in\mathcal{P}.y\succeq[\mathcal{R}]x)
lemma degenerate-total:
    \foralle\indegenerate-lotteries. }\forallm\in\mathcal{P}.e\succeq[\mathcal{R}]m\veem\succeq[\mathcal{R}]
    using degenerate-lotteries-def rational-preference.compl rpr by fastforce
lemma degen-outcome-cardinalities:
    card degenerate-lotteries = card outcomes
    using card-degen-lotteries-equals-outcomes degenerate-lotteries-def by auto
lemma degenerate-lots-subset-all: degenerate-lotteries }\subseteq\mathcal{P
    by (simp add: degenerate-lotteries-def)
```

```
lemma alt-definition-of-degenerate-lotteries[iff]:
    {return-pmf x |x. x\in outcomes } = degenerate-lotteries
proof (standard, goal-cases)
    case 1
```



```
    proof
        fix }
        assume a: }x\in{\mathrm{ return-pmf x |x. x 趹comes }
        then have card (set-pmf x)=1
        by auto
    moreover have set-pmf x\subseteq 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\mathrm{ degenerate-lotteries
        by (simp add: degenerate-lotteries-def)
    qed
    then show ?case by blast
next
    case 2
    have }\forallx\in\mathrm{ degenerate-lotteries. }x\in{\mathrm{ return-pmf }x|x.x\in\mathrm{ outcomes }
    proof
        fix }
    assume a:x degenerate-lotteries
    hence card (set-pmf x)=1
        using degenerate-lotteries-def by blast
    moreover have set-pmf x}\subseteq\mathrm{ outcomes
        by (meson a degenerate-lots-subset-all subset-iff support-in-outcomes)
    moreover obtain e where {e} = set-pmf x
        using calculation
        by (metis card-1-singletonE)
    moreover have e\in outcomes
        using calculation(2) calculation(3) by blast
    moreover have x = return-pmf e
        using calculation(3) set-pmf-subset-singleton by fast
```



```
        by blast
    qed
    then show ?case by blast
qed
lemma best-indifferent:
    \forallx\in best. }\forally\in\mathrm{ best. }x\approx[\mathcal{R}]
    by (simp add: best-def)
lemma worst-indifferent:
    \forallx\in worst. }\forally\in\mathrm{ worst. }x\approx[\mathcal{R}]
    by (simp add: worst-def)
```

```
lemma best-worst-indiff-all-indiff:
    assumes \(b \in\) best
        and \(w \in\) 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 \(\forall e \in\) local. \(\mathcal{P} . e \approx[\mathcal{R}] b\)
    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 \(\alpha>\beta \longleftrightarrow\) mix-pmf \(\alpha b w \succ[\mathcal{R}]\) mix-pmf \(\beta b w(\) is \(? L \longleftrightarrow ? R)\)
    using assms step-1-most-general[ of b w \(\alpha \beta \beta]\)
    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 mix-pmf \alpha b w\succeq[\mathcal{R}] mix-pmf \betabw
proof-
    have (0::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:
$\{x \in$ degenerate-lotteries. $(\forall y \in$ degenerate-lotteries. $x \succeq[\mathcal{R}] y)\} \neq\{ \}($ is $? l \neq$ \{\})
proof
assume $a: ? l=\{ \}$
let $? D G=$ degenerate-lotteries
obtain $R$ where
$R$ : rational-preference ? $D G R \quad$ $\subseteq \mathcal{R}$
using degenerate-lots-subset-all rational-preference.all-carrier-ex-sub-rel rpr by blast
then have $\exists e \in ? D G . \forall e^{\prime} \in ? D G . e \succeq[\mathcal{R}] e^{\prime}$
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:
$\{x \in$ degenerate-lotteries. $(\forall y \in$ degenerate-lotteries. $y \succeq[\mathcal{R}] x)\} \neq\{ \}$ (is ?l $\neq$ \{\})
proof
assume $a: ? l=\{ \}$
let ? $D G=$ degenerate-lotteries
obtain $R$ where
$R$ : rational-preference ? $D G R \quad R \subseteq \mathcal{R}$
using degenerate-lots-subset-all rational-preference.all-carrier-ex-sub-rel rpr by

```
blast
    have }\existse\in?,DG.\forall\mp@subsup{e}{}{\prime}\in?,DG. \mp@subsup{e}{}{\prime}\succeq[\mathcal{R}]
        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:
    \existsx\in degenerate-lotteries. }\forally\in\mathrm{ degenerate-lotteries. }x\succeq[\mathcal{R}]
    using degenerate-has-max-preferred by blast
lemma exists-worst-degenerate:
    \existsx\in degenerate-lotteries. }\forally\in\mathrm{ degenerate-lotteries. }y\succeq[\mathcal{R}]
    using degenerate-has-min-preferred by blast
lemma best-degenerate-in-best-overall:
    \existsx\in degenerate-lotteries.}\forally\in\mathcal{P}.x\succeq[\mathcal{R}]
proof -
    obtain b where
        b:b}\mathrm{ degenerate-lotteries }\forally\in\mathrm{ degenerate-lotteries. }b\succeq[\mathcal{R}]
        using exists-best-degenerate by blast
    have asm: finite outcomes set-pmf b\subseteqoutcomes
    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: }\foralld\in\mathrm{ outcomes. }b\succeq[\mathcal{R}] return-pmf d
        using alt-definition-of-degenerate-lotteries b(2) by blast
    define P where
    P=(\lambdap.p\in\mathcal{P}\longrightarrow\mathrm{ return-pmf B}\succeq[\mathcal{R}] p)
    have P p for p
    proof -
        consider set-pmf p\subseteqoutcomes | \negset-pmf p\subseteq outcomes
            by blast
        then show ?thesis
        proof (cases)
            case 1
            have finite outcomes set-pmf p\subseteqoutcomes
                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:
    \(\exists x \in\) degenerate-lotteries. \(\forall y \in \mathcal{P} . y \succeq[\mathcal{R}] x\)
proof -
    obtain \(b\) where
        \(b: b \in\) degenerate-lotteries \(\forall y \in\) 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: \(\forall d \in\) outcomes. return-pmf \(d \succeq[\mathcal{R}] b\)
    using alt-definition-of-degenerate-lotteries b(2) by blast
    define \(P\) where
        \(P=(\lambda p . p \in \mathcal{P} \longrightarrow p \succeq[\mathcal{R}]\) return-pmf \(B)\)
    have \(P p\) for \(p\)
    proof -
    consider set-pmf \(p \subseteq\) outcomes \(\mid \neg\) set-pmf \(p \subseteq\) 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 }\not={
    using degenerate-lots-subset-all worst-def worst-degenerate-in-worst-overall by
auto
lemma trans-approx:
    assumes }x\approx[\mathcal{R}]
        and }y\approx[\mathcal{R}]
    shows }x\approx[\mathcal{R}]
    using preference.indiff-trans[of \mathcal{P }\mathcal{R}xyz] 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 degenerate-lotteries }\wedgex\in\mathrm{ best )
private definition some-worst where
    some-worst =(SOME x. x d degenerate-lotteries ^ x\in worst)
private definition my-U :: 'a pmf => real
    where
        my-U p=(SOME \alpha. \alpha\in{0..1}\wedge p\approx[\mathcal{R}] mix-pmf \alpha some-best some-worst)
lemma exists-best-and-degenerate: degenerate-lotteries \cap best }\not={
    using best-def best-degenerate-in-best-overall degenerate-lots-subset-all by blast
lemma exists-worst-and-degenerate: degenerate-lotteries }\cap\mathrm{ worst }\not={
    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 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 mix-pmf $\alpha$ some-best some-worst $\succeq[\mathcal{R}]$ mix-pmf $\beta$ 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 $(1::$ 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 $\ldots=(1-\beta-(\alpha-\beta)) /(1-\beta)$
by (metis diff-divide-distrib)
also have $\ldots=(1-\alpha) /(1-\beta)$
by $\operatorname{simp}$
finally show ?thesis.
qed
lemma all-mix-pmf-indiff-indiff-best-worst:
assumes $l \in \mathcal{P}$
assumes $b \in$ best
assumes $w \in$ worst
assumes $b \approx[\mathcal{R}] w$
shows $\forall \alpha \in\{0 . .1\} . l \approx[\mathcal{R}]$ 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 some-best $\succ[\mathcal{R}]$ some-worst
assumes $\alpha \in\{0 . .1\}$
assumes $\beta \in\{0 . .1\}$
assumes mix-pmf $\beta$ some-best some-worst $\approx[\mathcal{R}]$ mix-pmf $\alpha$ 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 some-best $\succ[\mathcal{R}]$ some-worst
assumes $\alpha \in\{0 . .1\}$
and $\beta \in\{0 . .1\}$
assumes mix-pmf $\beta$ some-best some-worst $\succeq[\mathcal{R}]$ mix-pmf $\alpha$ some-best some-worst
shows $\beta \geq \alpha$
proof -
have $*$ : mix-pmf $\beta$ some-best some-worst $\approx[\mathcal{R}]$ mix-pmf $\alpha$ some-best some-worst $\Longrightarrow \alpha \leq \beta$
using assms(1) assms(2) assms(3) indiff-imp-same-utility-value by blast

```
    consider mix-pmf \beta some-best some-worst }\succ[\mathcal{R}] mix-pmf \alpha some-best some-wors
|
    mix-pmf \beta some-best some-worst }\approx[\mathcal{R}] mix-pmf \alpha some-best some-wors
    using assms(4) by blast
    then show ?thesis
    by(cases) (meson assms(2) assms(3) geq-mix-imp-weak-pref le-cases *)+
qed
lemma ge-mix-pmf-preferred:
    assumes }x\succ[\mathcal{R}]
    assumes \alpha \in{0..1}
        and \beta\in{0..1}
assumes \alpha\geq\beta
shows (mix-pmf \alphaxy)\succeq[\mathcal{R}](mix-pmf \betaxy)
using assms(1) assms(2) assms(3) assms(4) by blast
```


### 5.2 Add continuity to assumptions

```
context
    assumes cnt: continuous-vnm (lotteries-on outcomes)}\mathcal{R
begin
```

In Literature this is referred to as step 2.
lemma step-2-unique-continuous-unfolding:
assumes $p \succeq[\mathcal{R}] q$
and $q \succeq[\mathcal{R}] r$
and $p \succ[\mathcal{R}] r$
shows $\exists!\alpha \in\{0 . .1\} . q \approx[\mathcal{R}]$ mix-pmf $\alpha$ pr
proof (rule ccontr)
assume neg-a: $\nexists!\alpha . \alpha \in\{0 . .1\} \wedge q \approx[\mathcal{R}]$ mix-pmf $\alpha$ pr
have $\exists \alpha \in\{0 . .1\} . q \approx[\mathcal{R}]$ mix-pmf $\alpha p r$
using non-unique-continuous-unfolding[of outcomes $\mathcal{R} p q r]$
assms cnt rpr by blast
then obtain $\alpha \beta$ :: real where

$$
a-b: \alpha \in\{0 . .1\} \beta \in\{0 . .1\} \quad q \approx[\mathcal{R}] \text { mix-pmf } \alpha \text { pr } q \approx[\mathcal{R}] \text { mix-pmf } \beta \text { pr } \alpha \neq \beta
$$

using neg-a by blast
consider $\alpha>\beta \mid \beta>\alpha$
using $a-b$ by linarith
then show False
proof (cases)
case 1
with step-1-most-general $[$ of $p$ r $\alpha \beta$ assms
have mix-pmf $\alpha$ pr $\succ[\mathcal{R}]$ mix-pmf $\beta$ pr 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)
next
case 2
with step-1-most-general[of pr $\beta \alpha]$ assms have mix-pmf $\beta$ pr$\succ[\mathcal{R}]$ mix-pmf

## $\alpha p r$

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 folowing two lemmas are referred to sometimes called step 2.
lemma create-unique-indiff-using-distinct-best-worst:
assumes $l \in \mathcal{P}$
assumes $b \in$ best
assumes $w \in$ worst
assumes $b \succ[\mathcal{R}] w$
shows $\exists!\alpha \in\{0 . .1\} . l \approx[\mathcal{R}]$ 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 $\exists!\alpha \in\{0 . .1\} . l \approx[\mathcal{R}] \operatorname{mix}-p m f \alpha b w$ 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 $b \in$ best
assumes $w \in$ worst
shows $\exists \alpha \in\{0 . .1\} . l \approx[\mathcal{R}]$ 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 my-U $p \in\{0 . .1\} p \approx[\mathcal{R}]$ mix-pmf (my-U $p$ ) some-best some-worst
proof -
have some-best $\in$ best
by (simp add: some-best-in-best)
moreover have some-worst $\in$ worst
by (simp add: some-worst-in-worst)
with exists-element-bw-mix-is-approx [of p some-best some-worst] calculation assms

```
    have e: \exists\alpha\in{0..1}. p\approx[\mathcal{R}] mix-pmf \alpha some-best some-worst by blast
    then show my-U p\in{0..1}
    by (metis (mono-tags, lifting) my-U-def someI-ex)
    show p}\approx[\mathcal{R}] mix-pmf (my-U p) 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 mix-pmf (my-U p) some-best some-worst \succeq[\mathcal{R}] mix-pmf (my-U q) 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 mix-pmf (my-U p) some-best some-worst }\succ[\mathcal{R}] mix-pmf (my-U q
some-best some-worst
    shows my-U p>my-U q
proof (rule ccontr)
    assume }\negmy-Up>my-U
    then consider my-U p=my-Uq|my-U p<my-U q
    by linarith
    then show False
    proof (cases)
    case 1
    then have mix-pmf (my-U p) some-best some-worst }\approx[\mathcal{R}] mix-pmf (my-U q)
some-best some-worst
            using assms by auto
    then show ?thesis using assms by blast
    next
    case 2
    moreover have my-U q\in{0..1}
            using assms(2) my-U-is-defined(1) by blast
    moreover have my-U p\in{0..1}
                using assms(1) my-U-is-defined(1) by blast
    moreover have mix-pmf (my-U q) some-best some-worst \succeq[\mathcal{R}] mix-pmf (my-U
p) 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 some-best }\succ[\mathcal{R}] some-wors
    assumes my-U p\geqmy-U q
```

```
    shows p\succeq[\mathcal{R}]q
proof -
    have p-q:my-U p\in{0..1} my-U q\in{0..1}
        using assms my-U-is-defined(1) by blast+
    with ge-mix-pmf-preferred[of some-best some-worst my-U p my-U q]
        p-q assms(1) assms(3) assms(4)
    have mix-pmf (my-U p) some-best some-worst }\succeq[\mathcal{R}] mix-pmf (my-U q) some-bes
some-worst by blast
    consider my-U p=my-U q| my-U p>my-Uq
    using assms by linarith
    then show ?thesis
    proof (cases)
        case 2
        then show ?thesis
        by (meson assms(1) assms(2) assms(3) p-q(1) p-q(2) rational-preference.compl
        rpr step-1-most-general weak-pref-mix-with-my-U-weak-pref)
    qed (metis assms(1) assms(2) my-U-is-defined(2) trans-approx)
qed
lemma my-U-represents-pref:
    assumes some-best }\succ[\mathcal{R}] some-wors
    assumes p\in\mathcal{P}
        and q\in\mathcal{P}
    shows }p\succeq[\mathcal{R}]q\longleftrightarrowmy-U p\geqmy-U q(is ?L\longleftrightarrow \longleftrightarrow?R
proof -
    have p-def:my-U p\in{0..1} my-U q\in{0..1}
        using assms my-U-is-defined by blast+
    show ?thesis
    proof
        assume a:?L
            hence mix-pmf (my-U p) some-best some-worst \succeq[\mathcal{R}] mix-pmf (my-U q)
some-best some-worst
            using weak-pref-mix-with-my-U-weak-pref by auto
            then show ?R using leq-mix-imp-weak-inferior[of my-U p my-U q] p-def a
                assms(1) leq-mix-imp-weak-inferior by blast
    next
        assume ?R
        then show ?L using geq-my-U-imp-weak-preference
            using assms(1) assms(2) assms(3) by blast
    qed
qed
lemma first-iff-u-greater-strict-preff:
    assumes p\in\mathcal{P}
        and q\in\mathcal{P}
    assumes some-best }\succ[\mathcal{R}]\mathrm{ some-worst
    shows my-U p>my-U q\longleftrightarrow mix-pmf (my-U p) some-best some-worst }\succ[\mathcal{R}
mix-pmf (my-U q) some-best some-worst
```

```
proof
    assume a:my-U p>my-Uq
    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-wors
    shows mix-pmf (my-U p) some-best some-worst }\succ[\mathcal{R}] mix-pmf (my-U q) some-bes
some-worst \longleftrightarrowp\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}]
        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}]
    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}]\mathrm{ some-worst
    shows my-U (mix-pmf \alphapq)=\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 $U p$ where $U p: U p=m y-U p$
define $U q$ where $U q: U q=m y-U q$
have long-in: $(\alpha * U p+(1-\alpha) * U q) \in\{0 . .1\}$
proof -
have $U p \in\{0 . .1\}$
using assms Up my-U-is-defined(1) by blast
moreover have $U q \in\{0 . .1\}$
using assms Uq my-U-is-defined(1) by blast
moreover have $\alpha * U p \in\{0 . .1\}$
using $\langle U p \in\{0 . .1\}$ 〉 assms(3) mult-le-one by auto
moreover have $1-\alpha \in\{0 . .1\}$
using assms(3) by auto
moreover have $(1-\alpha) * U q \in\{0 . .1\}$
using mult-le-one $[$ of $1-\alpha$ Uq] calculation(2) calculation(4) by auto
ultimately show ?thesis
using add-nonneg-nonneg[of $\alpha * U p(1-\alpha) * U q]$
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 $p] B W U p$ by simp
have snd: $q \approx[\mathcal{R}]($ mix-pmf $U q B W)$
using assms my-U-is-defined[of $q] B W U q$ 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 $U p B W) q$
using fst assms(2) assms(3) mix-pmf-preferred-independence by blast
have $* *$ : mix-pmf $\alpha$ (mix-pmf Up $B W$ ) $($ mix-pmf $U q B W)=$
mix-pmf $(\alpha * U p+(1-\alpha) * U q) B W($ is ? $L=? R)$
proof -
let ? $\operatorname{mixP} Q=($ mix-pmf $(\alpha * U p+(1-\alpha) * U q) B W)$
have $\forall e \in s e t-p m f$ ? L. pmf (?L) $e=p m f ? m i x P Q e$
proof
fix $e$
assume asm: $e \in$ set-pmf ? $L$
have i1: pmf (?L) $e=\alpha * p m f(m i x-p m f U p B W) e+$ pmf (mix-pmf UqBW)e- $\alpha$ *pmf (mix-pmf Uq B W) e
using pmf-mix-deeper[of $\alpha$ mix-pmf $U p B W$ (mix-pmf $U q B W$ ) e] assms(3)
by blast
have $i 3: \ldots=\alpha * U p * p m f B e+\alpha * p m f W e-\alpha * U p * p m f W e+U q$

* pmf $B e+$ $p m f W e-U q * p m f W e-\alpha * U q * p m f B e-\alpha * p m f W e+\alpha * U q *$ $p m f W e$
using left-diff-distrib' pmf-mix-deeper[of Up B We] 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 $j_{4}$ : pmf ? $\operatorname{mixPQ} e=(\alpha * U p+(1-\alpha) * U q) * p m f B e+$
$p m f W e-(\alpha * U p+(1-\alpha) * U q) * p m f W e$
using pmf-mix-deeper $[o f(\alpha * U p+(1-\alpha) * U q) B W e]$ long-in by blast
then show $p m f(? L) e=p m f ? m i x P Q e$

```
    by (simp add: i1 i3 mult.commute right-diff-distrib' ring-class.ring-distribs(1))
    qed
    then show ?thesis using pmf-equiv-intro1 by blast
    qed
    have mix-pmf \alpha (mix-pmf Up 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 pq\approx[\mathcal{R}] mix-pmf \alpha (mix-pmf (my-U p)B W) (mix-pmf
(my-Uq) B W)
    using Up Uq fO trans-approx by blast
    have mix-pmf \alpha (mix-pmf (my-U p)B W)(mix-pmf (my-U q) B W)=?R
        using Up Uq** by blast
    hence my-U(mix-pmf \alpha p q) =\alpha*Up+(1-\alpha)*Uq
    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 Up Uq 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 $q)=\alpha *($ general- $U p)+(1-\alpha) *($ general- $U$
q)
proof -
consider some-best $\succ[\mathcal{R}]$ some-worst $\mid$ 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}]\mathrm{ 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-wors
        using a by auto
    then show }x\succeq[\mathcal{R}]y=(\mathrm{ general-U y s general-U }x\mathrm{ )
        using 1 my-U-represents-pref[of x y] general-U-def by simp
    next
    case b
    have general-U }x=1\mathrm{ general- U y=1
        by (simp add: b general-U-def)+
    moreover have }x\approx[\mathcal{R}]y\mathrm{ 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=(\mathrm{ general-}Uy\leqgeneral-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 \(\exists u . u(\) 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 \(\mid\) 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 \((m i x-p m f \alpha p q)=\alpha *\) ? \(u p+(1-\alpha) *\) ? \(u q\)
    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 $o c U$
where

```
ocU p = general-U (return-pmf p)
```

lemma geral-U-is-expected-value-of-oc $U$ :
assumes set-pmf $p \subseteq$ outcomes
shows general- $U p=$ measure-pmf.expectation $p$ oc $U$
using fnt assms
proof (induct rule: pmf-mix-induct')
case ( $\operatorname{mix} p q a$ )
hence general- $U$ (mix-pmf a p $q)=a *$ general- $U p+(1-a) *$ general- $U q$
using general- U-is-linear-function $\left[\begin{array}{llll}o f & p & q & a\end{array}\right]$ mix.hyps assms lotteries-on-def mix.hyps by auto
also have $\ldots=a *$ measure-pmf.expectation $p$ oc $U+(1-a) *$ measure-pmf.expectation $q$ oc $U$
by (simp add: mix.hyps(4) mix.hyps(5))
also have $\ldots=$ measure-pmf.expectation (mix-pmf a p q) oc $U$
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 $x$ oc $U)$
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 $x \succeq[\mathcal{R}] y \Longrightarrow$ (measure-pmf.expectation $y$ oc $U \leq$ measure-pmf.expectation
$x$ oc $U$ )
proof -
assume $x \succeq[\mathcal{R}] y$
have general- $U x \geq$ general- $U y$
by (meson $\langle x \succeq[\mathcal{R}] y>$ general-U-ordinal-Utility ordinal-utility-def)
then show (measure-pmf.expectation y oc $U \leq$ measure-pmf.expectation $x$ oc $U$ )
using geral- $U$-is-expected-value-of-oc $U$ ocs by auto
qed
moreover have (measure-pmf.expectation $y$ oc $U \leq$ measure-pmf.expectation $x$ $o c U) \Longrightarrow x \succeq[\mathcal{R}] y$
proof -
assume (measure-pmf.expectation y oc $U \leq$ measure-pmf.expectation $x$ oc $U$ )

```
    then have general-U }x\geq\mathrm{ general- }U\mathrm{ y
        by (simp add: geral-U-is-expected-value-of-ocU ocs(1) ocs(2))
    then show }x\succeq[\mathcal{R}]
    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':
    \existsu. ordinal-utility \mathcal{P }}\mathcal{R}(\lambdax.measure-pmf.expectation x u
    using ordinal-utility-expected-value by blast
lemma ocU-is-expected-utility-bernoulli:
    shows }\forallx\in\mathcal{P}.\forally\in\mathcal{P}.x\succeq[\mathcal{R}]y
    measure-pmf.expectation x ocU \geq measure-pmf.expectation y oc U
    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 }\not={
    assumes }x\inlotteries-on outcomes and y lotteries-on outcome
    assumes ordinal-utility (lotteries-on outcomes) \mathcal{R}}(\lambdax\mathrm{ . measure-pmf.expectation
x u)
    shows measure-pmf.expectation x }u\geq\mathrm{ measure-pmf.expectation y }u\longleftrightarrowx\succeq[\mathcal{R}
y ( \text { is ? L } \longleftrightarrow \text { ?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 }\not={
    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 }\existsu\mathrm{ . ordinal-utility (lotteries-on outcomes) }\mathcal{R}(\lambdax.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 }\not={
    assumes \existsu. ordinal-utility (lotteries-on outcomes)}\mathcal{R}(\lambdax.measure-pmf.expectation
x u)
    shows rational-preference (lotteries-on outcomes) \mathcal{R}
    using assms(3) ordinal-util-imp-rat-prefs by blast
theorem vNM-utility-implies-independence:
    assumes fnt: finite outcomes and outcomes }\not={
    assumes \existsu. ordinal-utility (lotteries-on outcomes) \mathcal{R}}(\lambdax\mathrm{ . measure-pmf.expectation
x u)
    shows independent-vnm (lotteries-on outcomes) }\mathcal{R
proof (rule independent-vnmI2)
    fix p qr
        and \alpha::real
    assume a1: p\in\mathcal{P}\mathrm{ outcomes}
    assume a2: q\in\mathcal{P}\mathrm{ outcomes}
    assume a3: r \in\mathcal{P}\mathrm{ outcomes}
    assume a4: \alpha { {0<..1}
```



```
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\mathrm{ where
        u: ordinal-utility (lotteries-on outcomes) \mathcal{R}}(\lambdax.measure-pmf.expectation x u
        using assms by blast
    have p\succeq[\mathcal{R}]q\Longrightarrow mix-pmf \alpha pr\succeq[\mathcal{R}] mix-pmf \alpha qr
    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 pr) u\geq measure-pmf.expectation
(mix-pmf \alpha qr)u
    proof -
            have measure-pmf.expectation (mix-pmf \alpha pr) u=
                \alpha* measure-pmf.expectation pu+(1-\alpha)* measure-pmf.expectation r u
                using expected-value-mix-pmf-distrib[of p r a u] assms fnts a& by fastforce
            moreover have measure-pmf.expectation (mix-pmf \alphaqr)u=
                \alpha* measure-pmf.expectation qu+(1-\alpha)* 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 a& by auto
    qed
```

```
    then show mix-pmf \alpha pr\succeq[\mathcal{R}] 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 mix-pmf \alpha pr\succeq[\mathcal{R}] mix-pmf \alpha qr\Longrightarrowp\succeq[\mathcal{R}]q
    proof -
    assume mix-pmf \alpha pr\succeq[\mathcal{R}] mix-pmf \alphaqr
    hence f:measure-pmf.expectation (mix-pmf \alpha pr) u\geq measure-pmf.expectation
(mix-pmf \alpha q r)u
        using ordinal-utility.ordinal-utility-left u by fastforce
    hence measure-pmf.expectation p u\geq measure-pmf.expectation q u
    proof -
        have measure-pmf.expectation (mix-pmf \alpha pr) u=
            \alpha* measure-pmf.expectation p u + (1-\alpha)* measure-pmf.expectation r u
            using expected-value-mix-pmf-distrib[of p r a u] assms fnts a4 by fastforce
        moreover have measure-pmf.expectation (mix-pmf \alphaqr)u=
            \alpha* measure-pmf.expectation q u + (1-\alpha)* measure-pmf.expectation r u
            using expected-value-mix-pmf-distrib[of q r a u] assms fnts a4 by fastforce
            ultimately show ?thesis using f using a& by auto
    qed
    then show }p\succeq[\mathcal{R}]
        using a1 a2 ordinal-utility.util-def-conf u by fastforce
    qed
    ultimately show }p\succeq[\mathcal{R}]q=mix-pmf \alpha pr\succeq[\mathcal{R}] mix-pmf \alphaqr
        by blast
qed
lemma exists-weight-for-equality:
    assumes }a>c\mathrm{ and }a\geqb\mathrm{ and b \c
    shows }\exists(e::\mathrm{ real ) }\in{0..1}.(1-e)*a+e*c=
proof -
    from assms have b\inclosed-segment a c
    by (simp add: closed-segment-eq-real-ivl)
    thus ?thesis by (auto simp: closed-segment-def)
qed
lemma vNM-utilty-implies-continuity:
    assumes fnt: finite outcomes and outcomes }\not={
    assumes \existsu. ordinal-utility (lotteries-on outcomes)}\mathcal{R}(\lambdax.measure-pmf.expectation
x u)
    shows continuous-vnm (lotteries-on outcomes)}\mathcal{R
proof (rule continuous-vnmI)
    fix p qr
    assume a1: p \in\mathcal{P}\mathrm{ outcomes}
    assume a2: q\in\mathcal{P}\mathrm{ outcomes}
    assume a3: r\in\mathcal{P}\mathrm{ outcomes}
    assume a4:p\succeq[\mathcal{R}]q\wedgeq\succeq[\mathcal{R}]r
    then have g: 
            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 $a 4 u$ 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 $\alpha$ pr $\approx[\mathcal{R}] q$
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 bc by blast+
then obtain $e$ ::real where
$e: e \in\{0 . .1\}(1-e) * a+e * b=c$
using exists-weight-for-equality[ $\left[\begin{array}{lll} & b & a\end{array}\right]\langle b<a\rangle$ 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 prreu] fnts by fastforce
also have $\ldots=(1-e) * a+e * b$
using $a b$ by auto
also have $\ldots=c$
using $c e$ by auto
finally have $f$ : measure-pmf.expectation (mix-pmf (1-e) pr)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) prq $\mathcal{R}$
$u$ ] *
proof -
have mix-pmf $(1-e) p r \in \mathcal{P}$ outcomes
using $\langle 1-e \in\{0 . .1\}$ 〉 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(3) 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 }\not={
    shows rational-preference (lotteries-on outcomes)}\mathcal{R}
        independent-vnm (lotteries-on outcomes) \mathcal{R}\wedge
        continuous-vnm (lotteries-on outcomes) \mathcal{R}\longleftrightarrow
    ( \existsu. ordinal-utility (lotteries-on outcomes)\mathcal{R}(\lambdax. 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-utilty-implies-continuity[OF assms, of \mathcal{R}]
    ordinal-util-imp-rat-prefs[of lotteries-on outcomes }\mathcal{R}]\mathrm{ 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 \(v N M\)-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 \(\bigwedge p q . \quad p \in\) lotteries-on outcomes \(\Longrightarrow\)
                        \(q \in\) lotteries-on outcomes \(\Longrightarrow\)
        \(p \succeq[\) 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 \(\Longrightarrow q \in\) lotteries-on outcomes \(\Longrightarrow\)
```

```
    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 \subseteqordinal-utility (lotteries-on outcomes) relation ( }\lambda\mathrm{ 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\mathrm{ lotteries-on outcomes
    assumes q\in lotteries-on outcomes
    shows }p\succ[\mathrm{ relation] q}\longleftrightarrow\mathrm{ measure-pmf.expectation pu> measure-pmf.expectation
qu
    by (meson assms(1) assms(2) less-eq-real-def not-le util-def)
lemma pos-distrib-left:
```

```
    assumes c>0
    shows (\sumz\inoutcomes.pmf qz*(c*uz))=c*(\sumz\inoutcomes.pmf qz*(u
z))
proof -
    have (\sumz\inoutcomes.pmf qz*(c*uz))=(\sumz\inoutcomes.pmf qz*c*uz)
        by (simp add: ab-semigroup-mult-class.mult-ac(1))
    also have ... =(\sumz\inoutcomes. c*pmf qz*uz)
        by (simp add: mult.commute)
    also have ... =c*(\sumz\inoutcomes.pmf qz*uz)
    by (simp add: ab-semigroup-mult-class.mult-ac(1) sum-distrib-left)
    finally show ?thesis.
qed
lemma sum-pmf-util-commute:
    (\suma\inoutcomes.pmf pa*ua)=(\suma\inoutcomes. u a * pmf pa)
    by (simp add: mult.commute)
```


## 7 Finite outcomes

## context

assumes fnt: finite outcomes
begin
lemma sum-equals-pmf-expectation:
assumes $p \in$ lotteries-on outcomes
$\operatorname{shows}\left(\sum z \in\right.$ outcomes. $($ pmf $\left.p z) *(u z)\right)=$ measure-pmf.expectation $p u$
proof -
have fnt: finite outcomes by (simp add: vNM-utilityD(1) fnt)
have measure-pmf.expectation $p u=\left(\sum a \in\right.$ outcomes. pmf $\left.p a * u a\right)$
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 $p \in$ lotteries-on outcomes and $q \in$ lotteries-on outcomes
shows $p \succeq[$ relation $] \longleftrightarrow\left(\sum z \in\right.$ outcomes. $\left.(p m f p z) *(u z)\right) \geq\left(\sum z \in\right.$ outcomes. $(p m f q z) *(u z))$
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 $p \in$ lotteries-on outcomes
and $q \in$ lotteries-on outcomes
shows $p \succeq q \longleftrightarrow\left(\left(\sum a \in\right.\right.$ outcomes. pmf $\left.q a * u a\right)-\left(\sum a \in\right.$ outcomes. pmf $p a$

$$
* u a) \leq 0)
$$

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 $p \in$ lotteries-on outcomes
and $q \in$ lotteries-on outcomes
shows $p \succ[$ relation $] q \longleftrightarrow$ 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 $(\lambda x . c * u x)$
proof (standard, goal-cases)
case 1
then show?case using $v N M$-utility-axioms $v N M$-utility-def by blast
next
case (2pq)
have $q \in$ lotteries-on outcomes and $p \in$ lotteries-on outcomes using 2(2) by (simp add: fnt 2(1))+
then have $*: p \succeq q=$ (measure-pmf.expectation $q u \leq$ measure-pmf.expectation pu)
using expected-utility-weak-preference[of p q] assms by blast
have dist-c: $\left(\sum z \in\right.$ outcomes. $\left.(p m f q z) *(c * u z)\right)=c *\left(\sum z \in\right.$ outcomes. (pmf $q z) *(u z))$
using pos-distrib-left[of c q] assms by blast
have dist- $c^{\prime}$ : $\left(\sum z \in\right.$ outcomes. $\left.(p m f p z) *(c * u z)\right)=c *\left(\sum z \in\right.$ outcomes. (pmf $p z) *(u z))$
using pos-distrib-left[of c p] assms by blast
have $p \succeq q \longleftrightarrow\left(\left(\sum z \in\right.\right.$ outcomes. $\left.(p m f q z) *(c * u z)\right) \leq\left(\sum z \in\right.$ outcomes. $(p m f$ $p z) *(c * u z)))$
proof (rule iffI)
assume $p \succeq q$
then have $\left(\sum z \in\right.$ outcomes. pmf $\left.q z *(u z)\right) \leq\left(\sum z \in\right.$ outcomes. pmf $p z *(u$
z))
using utility-ge
using 2(1) 2(2) sum-equals-pmf-expectation by presburger
then show $\left(\sum z \in\right.$ outcomes. pmf $\left.q z *(c * u z)\right) \leq\left(\sum z \in\right.$ outcomes. pmf $p z *$ $(c * u z))$
using dist-c dist-c'
by ( simp add: assms)
next
assume $\left(\sum z \in\right.$ outcomes. pmf $\left.q z *(c * u z)\right) \leq\left(\sum z \in\right.$ outcomes. pmf $p z *(c$ * $u z$ ))
then have $\left(\sum z \in\right.$ outcomes. pmf $\left.q z *(u z)\right) \leq\left(\sum z \in\right.$ 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 $p \in$ lotteries-on outcomes
and $q \in$ lotteries-on outcomes
shows $p \succ$ [relation $] ~ q \longleftrightarrow$
$\left(\sum z \in\right.$ outcomes. $($ pmf $\left.p z) *(u z)\right)>\left(\sum z \in\right.$ outcomes. $\left.(p m f q z) *(u z)\right)$
using sum-equals-pmf-expectation[of $p$, OF assms(1)] assms(1) assms(2)
sum-equals-pmf-expectation[of $q$, OF assms(2)] strict-prefernce-iff-strict-utility by presburger
lemma strict-alt-def-utility-g: assumes $p \succ$ [relation] $q$
shows $\left(\sum z \in\right.$ outcomes. $($ pmf $\left.p z) *(u z)\right)>\left(\sum z \in\right.$ outcomes. $\left.(p m f q z) *(u z)\right)$
using assms not-outside(1) not-outside(2) strict-alt-def
by meson
end
end
lemma vnm-utility-is-ordinal-utility:
assumes $v N M$-utility outcomes relation $u$
shows ordinal-utility (lotteries-on outcomes) relation ( $\lambda$ p. 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 $v N M$-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\mathrm{ p. ( \z=outcomes. (pmf pz)*(uz)))]
            assms vnm-utility-is-ordinal-utility
    using ordinal-utility.util-imp-total by blast
    then show?case
    by simp
qed
theorem expected-utilty-theorem-form-vnm-utility:
    assumes fnt: finite outcomes and outcomes }\not={
    shows rational-preference (lotteries-on outcomes)}\mathcal{R}
        independent-vnm (lotteries-on outcomes) \mathcal{R}\wedge
        continuous-vnm (lotteries-on outcomes)}\mathcal{R}
        ( }\existsu\mathrm{ .vNM-utility outcomes }\mathcal{R}u
proof
    assume rational-preference (\mathcal{P outcomes) \mathcal{R}}\wedge\mathrm{ independent-vnm( }\mathcal{P}\mathrm{ outcomes)}
\mathcal { R } \wedge \text { continuous-vnm ( } \mathcal { P } \text { outcomes) } \mathcal { R }
    with Von-Neumann-Morgenstern-Utility-Theorem[of outcomes \mathcal{R},OF assms]
have
    ( }\exists\mathrm{ u. ordinal-utility (P outcomes) }\mathcal{R}(\lambdax.measure-pmf.expectation x u)) using
assms by blast
    then obtain u where
        u: ordinal-utility (\mathcal{P outcomes) }\mathcal{R}(\lambdax.measure-pmf.expectation x u)
        by auto
    have vNM-utility outcomes }\mathcal{R}
    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 \existsu.vNM-utility outcomes \mathcal{R }u
    by blast
next
    assume a: \existsu.vNM-utility outcomes }\mathcal{R}
    then have rational-preference (\mathcal{P}\mathrm{ outcomes) }\mathcal{R}
        using vnm-utility-imp-reational-prefs by auto
    moreover have independent-vnm (\mathcal{P}\mathrm{ outcomes) }\mathcal{R}
    using a by (meson assms(2) fnt vNM-utility-implies-independence vnm-utility-is-ordinal-utility)
    moreover have continuous-vnm (\mathcal{P}\mathrm{ outcomes) }\mathcal{R}
    using a by (meson assms(2) fnt vNM-utilty-implies-continuity vnm-utility-is-ordinal-utility)
    ultimately show rational-preference (\mathcal{P}\mathrm{ outcomes) }\mathcal{R}\wedge independent-vnm (\mathcal{P}
outcomes) }\mathcal{R}\wedge\mathrm{ continuous-vnm ( }\mathcal{P}\mathrm{ outcomes) }\mathcal{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, 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 $L N C S$, 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, 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 $L N C S$, 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.


[^0]:    *This work is supported by the Austrian Science Fund (FWF) project P26201 and the European Research Council (ERC) grant no 714034 SMART.

