A Mechanisation of Probabilistic Designs in Isabelle/UTP

Kangfeng Ye Simon Foster
Jim Woodcock
University of York, UK

{kangfeng.ye,simon.foster,jim.woodcock}@york.ac.uk

May 30, 2021

Abstract

Contents

Acknowledgements.

A Probabilistic Designs

This is the mechanisation of *probabilistic designs* [1, 2] in Isabelle/UTP.

theory *utp-prob-des*
imports *UTP−Calculi.utp-wprespec UTP−Designs.utp-designs HOL−Probability.Probability-Mass-Function HOL−Probability.SPMF*

begin recall-syntax

purge-notation *inner* (**infix** $\cdot$ *70*)

declare [[*coercion pmf*]]

alphabet *'s prss* =
  prob :: *'s pmf*

If the probabilities of two disjoint sample sets sums up to 1, then the probability of the first set is equal to 1 minus the probability of the second set.

lemma *pmf-disj-set*:
  assumes $X \cap Y = \{\}$
  shows $((\sum_a i{\in}(X \cup Y).\ pmf\ M\ i) = 1) = ((\sum_a i{\in}X.\ pmf\ M\ i) = 1 - (\sum_a i{\in}Y.\ pmf\ M\ i))$
  by (*metis assms diff-eq-eq infsetsum-Un-disjoint pmf-abs-summable*)

no-utp-lift *ndesign wprespec uwp*

Probabilistic designs ((*'s*, *'s*) *rel-pdes*), that map the standard state space to the probabilistic state space, are heterogeneous.

type-synonym (*'a*, *'b*) *rel-pdes* = (*'a*, *'b prss*) *rel-des*
type-synonym *'s hrel-pdes* = (*'s*, *'s*) *rel-pdes*
type-synonym *'s hrel-hpdes* = (*'s prss*, *'s prss*) *rel-des*

translations
  (*type*) (*'a*, *'b*) *rel-pdes* <= (*type*) (*'a*, *'b prss*) *rel-des*

forget-prob is a non-homogeneous design as a forgetful function that maps a discrete probability distribution $\boldsymbol{U}(\$prob)$ at initial observation to a final state.

definition *forget-prob* :: (*'s prss*, *'s*) *rel-des* (**fp**) **where**
[*upred-defs*]: $forget\text{-}prob = U(true \vdash_n (\$prob(\$\mathbf{v}') > 0))$

The weakest prespecification of a standard design D wrt **fp** is the weakest probabilistic design, as an embedding of D in the probabilistic world through $\mathcal{K}$.

definition *pemb* :: (*'a*, *'b*) *rel-des* ⇒ (*'a*, *'b*) *rel-pdes* ($\mathcal{K}$)
  where [*upred-defs*]: *pemb* D = **fp** $\backslash$ D

lemma *pemb-mono*: $P \sqsubseteq Q \Longrightarrow \mathcal{K}(P) \sqsubseteq \mathcal{K}(Q)$
  by (*metis* (*mono-tags*, *lifting*) *dual-order.trans order-refl pemb-def wprespec*)

lemma *wdprespec*: $(true \vdash_n R) \backslash (p \vdash_n Q) = (p \vdash_n (R \backslash Q))$
  by (*rel-auto*)

lemma *pemb-form*:
  fixes R :: (*'a*, *'b*) *urel*
  shows $\boldsymbol{U}((\$prob(\$\mathbf{v}') > 0) \backslash R) = \boldsymbol{U}((\sum_a i{\in}\{s'.(R\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob'\ i) = 1)$ (**is** *?lhs* = *?rhs*)
proof −
  have $?lhs = \boldsymbol{U}((\neg (\neg R) ;; (0 < \$prob'\$\mathbf{v})))$
    by (*rel-auto*)
  also have ... $= \boldsymbol{U}((\sum_a i{\in}\{s'.(R\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob'\ i) = 1)$
    apply (*rel-auto*)
    apply (*metis* (*no-types*, *lifting*) *infsetsum-pmf-eq-1 mem-Collect-eq pmf-positive subset-eq*)
   apply (*metis AE-measure-pmf-iff UNIV-I measure-pmf.prob-eq-1 measure-pmf-conv-infsetsum mem-Collect-eq set-pmf-eq' sets-measure-pmf*)

```
    done
  finally show ?thesis .
qed
```

Embedded standard designs are probabilistic designs [2, Theorem 1] and [1, Theorem 3.6].

```
lemma prob-lift [ndes-simp]:
  fixes R :: ('a, 'b) urel and p :: 'a upred
  shows 𝒦(p ⊢n R) = U(p ⊢n ((∑a i∈{s'.(R wp (&v = s'))<}. $prob´ i) = 1))
proof −
  have 1:𝒦(p ⊢n R) = U(p ⊢n (($prob($v´) > 0) \ R))
    by (rel-auto)
  have 2:U(($prob($v´) > 0) \ R) = U((∑a i∈{s'.(R wp (&v = s'))<}. $prob´ i) = 1)
    by (simp add: pemb-form)
  show ?thesis
    by (simp add: 1 2)
qed

no-utp-lift usubst (0) subst (1)
```

A.1 wplus

Two pmfs can be joined into one by their corresponding weights via $P +_w Q$ where w is the weight of P.

```
definition wplus :: 'a pmf ⇒ real ⇒ 'a pmf ⇒ 'a pmf ((- +_ -) [64, 0, 65] 64) where
wplus P w Q = join-pmf (pmf-of-list [(P, w), (Q, 1 − w)])
```

Query of the probability value of a state i in a joined probability distribution is just the summation of the query of i in P by its weight w and the query of i in Q by its weight $(1 - w)$.

```
lemma pmf-wplus:
  assumes w ∈ {0..1}
  shows pmf (P +w Q) i = pmf P i ∗ w + pmf Q i ∗ (1 − w)
proof −
  from assms have pmf-wf-list: pmf-of-list-wf [(P, w), (Q, 1 − w)]
    by (auto intro!: pmf-of-list-wfI)
  show ?thesis
  proof (cases w ∈ {0<..<1})
    case True
    hence set-pmf:set-pmf (pmf-of-list [(P, w), (Q, 1 − w)]) = {P, Q}
      by (subst set-pmf-of-list-eq, auto simp add: pmf-wf-list)
    thus ?thesis
    proof (cases P = Q)
      case True
      from assms show ?thesis
        apply (auto simp add: wplus-def join-pmf-def pmf-bind)
        apply (subst integral-measure-pmf[of {P, Q}])
          apply (auto simp add: set-pmf-of-list pmf-wf-list set-pmf pmf-pmf-of-list)
        apply (simp add: True)
        apply (metis distrib-right eq-iff-diff-eq-0 le-add-diff-inverse mult.commute mult-cancel-left1)
        done
    next
      case False
      then show ?thesis
        apply (auto simp add: wplus-def join-pmf-def pmf-bind)
```

```
      apply (subst integral-measure-pmf[of {P, Q}])
        apply (auto simp add: set-pmf-of-list pmf-wf-list set-pmf pmf-pmf-of-list)
      done
    qed
  next
    case False
    thm disjE
    with assms have w = 0 ∨ w = 1
      by (auto)
    with assms show ?thesis
    proof (erule-tac disjE, simp-all)
      assume w: w = 0
      with pmf-wf-list have set-pmf (pmf-of-list [(P, w), (Q, 1 − w)]) = {Q}
        apply (simp add: pmf-of-list-remove-zeros(2)[THEN sym])
        apply (subst set-pmf-of-list-eq, auto simp add: pmf-of-list-wf-def)
        done
      with w show pmf (P +0 Q) i = pmf Q i
      apply (auto simp add: wplus-def join-pmf-def pmf-bind pmf-wf-list pmf-of-list-remove-zeros(2)[THEN
sym])
        apply (subst integral-measure-pmf[of {Q}])
          apply (simp-all add: set-pmf-of-list-eq pmf-pmf-of-list pmf-of-list-wf-def)
        done
    next
      assume w: w = 1
      with pmf-wf-list have set-pmf (pmf-of-list [(P, w), (Q, 1 − w)]) = {P}
        apply (simp add: pmf-of-list-remove-zeros(2)[THEN sym])
        apply (subst set-pmf-of-list-eq, auto simp add: pmf-of-list-wf-def)
        done
      with w show pmf (P +1 Q) i = pmf P i
      apply (auto simp add: wplus-def join-pmf-def pmf-bind pmf-wf-list pmf-of-list-remove-zeros(2)[THEN
sym])
        apply (subst integral-measure-pmf[of {P}])
          apply (simp-all add: set-pmf-of-list-eq pmf-pmf-of-list pmf-of-list-wf-def)
        done
    qed
  qed
qed

lemma wplus-commute:
  assumes w ∈{0..1}
  shows P +w Q = Q +(1 − w) P
  using assms by (auto intro: pmf-eqI simp add: pmf-wplus)

lemma wplus-idem:
  assumes w ∈{0..1}
  shows P +w P = P
  using assms
  apply (rule-tac pmf-eqI)
  apply (simp add: pmf-wplus)
  by (metis le-add-diff-inverse mult.commute mult-cancel-left2 ring-class.ring-distribs(2))

lemma wplus-zero: P +0 Q = Q
  by (auto intro: pmf-eqI simp add: pmf-wplus)

lemma wplus-one: P +1 Q = P
```

by (*auto intro*: *pmf-eqI simp add*: *pmf-wplus*)

This is used to prove the associativity of probabilistic choice: *prob-choice-assoc.*

lemma *wplus-assoc*:
  assumes $w_1 \in \{0..1\}$ $w_2 \in \{0..1\}$
  assumes $(1-w_1)*(1-w_2)=(1-r_2)$ $w_1=r_1*r_2$
  shows $P +_{w_1} (Q +_{w_2} R) = (P +_{r_1} Q) +_{r_2} R$
proof (*cases* $w_1 = 0 \wedge w_2 = 0$)
  case *True*
  then show *?thesis*
    proof −
      from *assms*(*3*−*4*) **have** *t1*: $r_2=0$
        by (*simp add*: *True*)
      then show *?thesis*
        by (*simp add*: *wplus-zero True t1*)
    qed
next
  case *False*
  from *assms*(*3*) **have** *f1*: $r_2 = w_1+w_2-w_1*w_2$
    proof −
      have *f1*: $\forall r\ ra.\ (ra{::}real) + - r = 0 \vee \neg\ ra = r$
        by *simp*
      have *f2*: $\forall r\ ra\ rb\ rc.\ (rc{::}real) \cdot rb + - (ra \cdot r) = rc \cdot (rb + - r) + (rc + - ra) \cdot r$
        by (*simp add*: *mult-diff-mult*)
      have *f3*: $\forall r\ ra.\ (ra{::}real) + (r + - ra) = r + 0$
        by *fastforce*
      have *f4*: $\forall r\ ra.\ (ra{::}real) + ra \cdot r = ra \cdot (1 + r)$
        by (*simp add*: *distrib-left*)
      have *f5*: $\forall r\ ra.\ (ra{::}real) + - r + 0 = ra + - r$
        by *linarith*
      have *f6*: $\forall r\ ra.\ (0{::}real) + (ra + - r) = ra + - r$
        by *simp*
      have $1 + - w_2 + - (w_1 \cdot (1 + - w_2)) = 1 + (0 + - r_2)$
      using *f2 f1* **by** (*metis* (*no-types*) *add.left-commute add-uminus-conv-diff assms*(*3*) *mult.left-neutral*)
      then have $1 + (w_1 + w_1 \cdot - w_2 + - r_2) = 1 + - w_2$
        using *f6 f5 f4 f3* **by** (*metis* (*no-types*) *add.left-commute*)
    then show *?thesis*
    by *linarith*
    qed
  then have *f2*: $r_2 \in \{0..1\}$
    using *assms*(*1*−*2*) **by** (*smt assms*(*3*) *atLeastAtMost-iff mult-le-one sum-le-prod1*)
  from *f1* **have** *f2'*: $(w_1+w_2-w_1*w_2) \geq w_1$
    using *assms*(*1*) *assms*(*2*) *mult-left-le-one-le* **by** *auto*
  from *f1* **have** *f3*: $r_1 = w_1/(w_1+w_2-w_1*w_2)$
    by (*metis False add.commute add-diff-eq assms*(*4*) *diff-add-cancel*
        mult-zero-left mult-zero-right nonzero-eq-divide-eq)
  show *?thesis*
  proof (*cases* $w_1 = 0$)
    case *True*
    from *f3* **have** *ft1*: $r_1 = 0$
      by (*simp add*: *True*)
    from *f1* **have** *ft2*: $r_2 = w_2$
      by (*simp add*: *True*)
    then show *?thesis*
      using *ft1 ft2 assms*(*1*−*2*)

by (*simp add*: *True wplus-zero*)
next
case *False*
from *f3 f2′* **have** *ff1*: $r_1 \leq 1$
using *False*
by (*metis assms*(*4*) *atLeastAtMost-iff eq-iff f1 f2 le-cases le-numeral-extra*(*4*) *mult-cancel-right2 mult-right-mono*)
have *ff2*: $r_1 \geq 0$
by (*smt False assms*(*1*) *assms*(*4*) *atLeastAtMost-iff f2 mult-not-zero zero-le-mult-iff*)
from *ff1* **and** *ff2* **have** *ff3*: $r_1 \in \{0..1\}$
by *simp*
have *ff4*: $w_2 * (1 - w_1) = (1 - r_1) * r_2$
using *f1 f3 False assms*
by (*metis* (*no-types*, *hide-lams*) *add-diff-eq diff-add-eq-diff-diff-swap diff-diff-add diff-diff-eq2 eq-iff-diff-eq-0 mult.commute mult.right-neutral right-diff-distrib′ right-minus-eq*)
then show *?thesis*
using *assms*(*1*−*2*) *f2 ff3* **apply** (*rule-tac pmf-eqI*)
apply (*simp add*: *assms*(*1*−*2*) *f2 ff3 pmf-wplus*)
using *assms*(*3*−*4*) *ff4*
by (*metis* (*no-types*, *hide-lams*) *add.commute add.left-commute mult.assoc mult.commute*)
qed
qed

A.2 Probabilistic Choice

We use parallel-by-merge in UTP to define the probabilistic choice operator. The merge predicate is the join of two distributions by their weights.

definition *prob-merge* :: *real* ⇒ ((′*s*, ′*s prss*, ′*s prss*) *mrg*, ′*s prss*) *urel* (**PM**$_{\_}$) **where**
[*upred-defs*]: *prob-merge* r = ***U***($prob´ = $0:prob $+_{«r»}$ $1:prob)

lemma *swap-prob-merge*:
assumes $r \in \{0..1\}$
shows $swap_m$; ; $\mathbf{PM}_r = \mathbf{PM}_{1\ -\ r}$
by (*rel-auto*, (*metis assms wplus-commute*)+)

abbreviation *prob-des-merge* :: *real* ⇒ ((′*s des*, ′*s prss des*, ′*s prss des*) *mrg*, ′*s prss des*) *urel* (**PDM**$_{\_}$)
where
$\mathbf{PDM}_r \equiv \mathbf{DM}(\mathbf{PM}_r)$

lemma *swap-prob-des-merge*:
assumes $r \in \{0..1\}$
shows $swap_m$; ; $\mathbf{PDM}_r = \mathbf{PDM}_{1\ -\ r}$
by (*metis assms swap-des-merge swap-prob-merge*)

The probabilistic choice operator is defined conditionally in order to satisfy unit and zero laws (*prob-choice-one* and *prob-choice-zero*) below. The definition of the operator follows [1, Definition 3.14]. Actually use of $P \parallel^D{}_{\mathbf{PM}_r} Q$ directly for (r = 0) or (r = 1) cannot get the desired result (P or Q) as the precondition of merged designs cannot be discharged to the precondition of P or Q simply.

definition *prob-choice* :: ′*s hrel-pdes* ⇒ *real* ⇒ ′*s hrel-pdes* ⇒ ′*s hrel-pdes* ((- ⊕$_{\_}$ -) [*164*, *0*, *165*] *164*)
where [*upred-defs*]:
prob-choice P r Q ≡
if $r \in \{0<..<1\}$
then $P \parallel^D{}_{\mathbf{PM}_r} Q$

else (*if* $r = 0$
then Q
else (*if* $r = 1$
then P
else $\top_D$))

The r in $P \oplus_r Q$ is a real number (HOL terms). Sometimes, however, we want a similar operator of which the weight is a UTP expression (therefore it depends on the values of state variables). For example, $P \oplus_{U(1/real\ («N»-i))} Q$ in a uniform selection algorithms where $\&i$ is a state variable. Hence, $(P \oplus_{eE} Q)$ is defined below, which is inspired by Morgan's logical constant [3].

definition *prob-choice-r* :: $('a, 'a)$ *rel-pdes* $\Rightarrow$ (*real*, $'a$) *uexpr* $\Rightarrow$ $('a, 'a)$ *rel-pdes* $\Rightarrow$ $('a, 'a)$ *rel-pdes*
((- $\oplus_{e}$- -) [*164*, *0*, *165*] *164*)
where [*upred-defs*]:
prob-choice-r $P\ E\ Q \equiv (con_D\ R \cdot (II_D \triangleleft U(«R» = E) \triangleright_D \bot_D)\ ;;\ (P \oplus_R Q))$

lemma *prob-choice-commute*: $r \in \{0..1\} \Longrightarrow P \oplus_r Q = Q \oplus_{1\ -\ r} P$
by (*simp add*: *prob-choice-def swap-prob-des-merge*[*THEN sym*], *metis par-by-merge-commute-swap*)

lemma *prob-choice-one*:
$P \oplus_1 Q = P$
by (*simp add*: *prob-choice-def*)

lemma *prob-choice-zero*:
$P \oplus_0 Q = Q$
by (*simp add*: *prob-choice-def*)

lemma *prob-choice-r*:
$r \in \{0<..<1\} \Longrightarrow P \oplus_r Q = P \parallel^D_{\mathbf{PM}_r} Q$
by (*simp add*: *prob-choice-def*)

lemma *prob-choice-inf-simp*:
$(\bigsqcap\ r \in \{0<..<1\} \cdot (P \oplus_r Q)) = (\bigsqcap\ r \in \{0<..<1\} \cdot\ P \parallel^D_{\mathbf{PM}_r} Q)$
using *prob-choice-r*
apply (*simp add*: *prob-choice-def*)
by (*simp add*: *UINF-as-Sup-collect image-def*)

inf-is-exists helps to establish the fact that our theorem regarding nondeterminism [2, Sect. 8] is the same as He's [1, Theorem 3.10].

lemma *inf-is-exists*:
$(\bigsqcap\ r \in \{0<..<1\} \cdot\ (p \vdash_n P) \parallel^D_{\mathbf{PM}_r} (q \vdash_n Q))$
$= (\exists\ r \in \boldsymbol{U}(\{0<..<1\}) \cdot\ (p \vdash_n P) \parallel^D_{\mathbf{PM}_r} (q \vdash_n Q))$
by (*pred-auto*)

A.3 Kleisli Lifting and Sequential Composition

utp-lit-vars

The Kleisli lifting operator maps a probabilistic design $(p \vdash_n R)$ into a "lifted" design that maps from *prob* to *prob*. Therefore, one probabilistic design can be composed sequentially with another lifted design. The precondition of the definition specifies that all states of the initial distribution satisfy the predicate p. The postcondition specifies that there exists a function Q, that maps states to distributions, such that

- for any state s, if its probability in the initial distribution is larger than 0, then R(s, Q(s)) must be held;

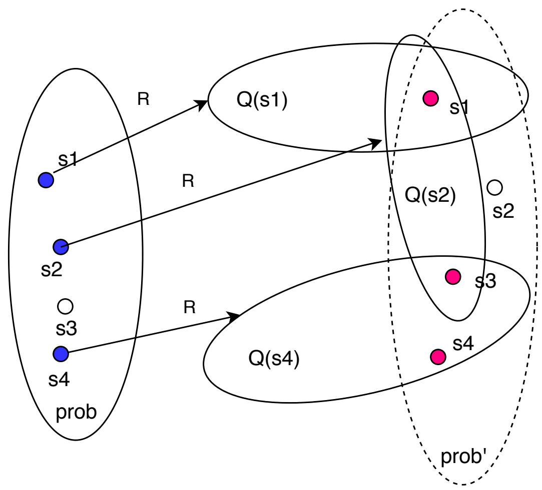

Figure 1: Illustration of Kleisli lifting

- any state *ss* in final distribution $\$prob´$ is equal to summation of all paths from any state t in its initial distribution to *ss* via $Q\ t$.

Figure 1 illustrates the lifting operation, provided that there are four states in the state space. The blue states in $\$prob$ denotes their initial probabilities are larger than 0, and the red states in $\$prob´$ denotes their final probabilities are larger than 0. Q is defined as

$$\{(s_1, Q(s_1)), (s_2, Q(s_2)), (s_4, Q(s_4))\}$$

and the relation between s_i and $Q(s_i)$ is established by R. In addition, the probability of s_1 in $Q(s_1)$ is larger than 0, that of s_1 and s_3 in $Q(s_2)$, and that of s_3 and s_4 in $Q(s_4)$. Finally, the finally distribution is given below.

$$prob'(s_1) = prob(s_1) * Q(s_1)(s_1) + prob(s_2) * Q(s_2)(s_1)$$
$$prob'(s_3) = prob(s_2) * Q(s_2)(s_3) + prob(s_4) * Q(s_4)(s_3)$$
$$prob'(s_4) = prob(s_2) * Q(s_2)(s_4) + prob(s_4) * Q(s_4)(s_4)$$

definition *kleisli-lift2*:: $'a\ upred \Rightarrow ('a,\ 'a\ prss)\ urel \Rightarrow ('a\ prss,\ 'a\ prss)\ rel\text{-}des$
  where *kleisli-lift2 p R* =
  $(\ \boldsymbol{U}((\sum_a\ i\in[\![p]\!]_p.\ \$prob\ i) = 1)$
    $\vdash_r$
  $(\exists\ Q \cdot ($
      $(\forall\ ss \cdot U((\$prob´\ ss) = (\sum_a\ t.\ ((\$prob\ t) * (pmf\ (Q\ t)\ ss))))) \wedge$
      $(\forall\ s \cdot (\neg(U(\$prob\ \$\mathbf{v}´ > 0 \wedge \$\mathbf{v}´ = s)\ ;;$
          $((((\neg R)\ ;;\ (\forall\ t \cdot U((\$prob\ t) = (pmf\ (Q\ s)\ t)))))))$
    $))$
  $)))$

named-theorems *kleisli-lift*

Alternatively, we can define the lifting operator as a normal design, instead of a design in previous definition.

definition *kleisli-lift2'*:: $'a\ upred \Rightarrow ('a,\ 'a\ prss)\ urel \Rightarrow ('a\ prss,\ 'a\ prss)\ rel\text{-}des$ **where**
[*kleisli-lift*]: *kleisli-lift2' p R* =
  $(\ \boldsymbol{U}((\sum_a\ i\in[\![p]\!]_p.\ \&prob\ i) = 1)$

$\vdash_n$
$(\exists\ Q \cdot ($
$(\forall\, ss \cdot U((\$prob\acute{}\ ss) = (\sum_a\ t.\ ((\$prob\ t) * (pmf\ \ (Q\ t)\ ss))))) \wedge$
$(\forall\, s \cdot (\neg(U(\$prob\ \$\mathbf{v}\acute{} > 0 \wedge \$\mathbf{v}\acute{} = s)\ ;;$
$((\neg R)\ ;;\ \ (\forall\, t \cdot U((\$prob\ t) = (pmf\ \ (Q\ s)\ t)))))$
$))$
$)))$

Two definitions actually are equal.

lemma *kleisli-lift2-eq*: *kleisli-lift2′ p R = kleisli-lift2 p R*
apply (*simp add*: *kleisli-lift2-def*)
apply (*simp add*: *utp-prob-des.kleisli-lift2′-def*)
by (*rel-auto*)

utp-expr-vars

Then the lifting operator ↑ is defined upon *kleisli-lift2*.

definition *kleisli-lift* (↑) **where**
$kleisli\text{-}lift\ P = kleisli\text{-}lift2\ (\lfloor pre_D(P)\rfloor_<)\ (pre_D(P) \wedge post_D(P))$

The alternative definition of the lifting operator ↑ is based on *kleisli-lift2′*.

lemma *kleisli-lift-alt-def*:
$kleisli\text{-}lift\ P = kleisli\text{-}lift2'\ (\lfloor pre_D(P)\rfloor_<)\ (pre_D(P) \wedge post_D(P))$
by (*simp add*: *kleisli-lift-def kleisli-lift2-eq*)

Sequential composition of two probabilistic designs (P and Q) is composition of P with the lifted Q through the Kleisli lifting operator.

abbreviation *pseqr* :: ('b, 'b) *rel-pdes* ⇒ ('b, 'b) *rel-pdes* ⇒ ('b, 'b) *rel-pdes* (**infix** $;;_p$ *60*)
where $pseqr\ P\ Q \equiv (P\ ;;\ (\uparrow Q))$

II_p is the identity of sequence of probabilistic designs.

abbreviation *skip-p* (II_p) **where**
$skip\text{-}p \equiv \mathcal{K}(II_D)$

The top of probabilistic designs is still the top of designs.

abbreviation *falsep* :: ('b, 'b) *rel-pdes* ($false_p$) **where**
falsep ≡ *false*

end

B (pmf) Laws

This section presents many proved laws regarding pmf to facilitate proof of algebraic laws of probabilistic designs.

theory *utp-prob-pmf-laws*
imports *UTP−Designs.utp-designs*
HOL−Probability.Probability-Mass-Function
utp-prob-des
begin recall-syntax

B.1 Laws

```
lemma sum-pmf-eq-1:
  fixes M::'a pmf
  shows (∑a i::'a. pmf M i) = 1
  by (simp add: infsetsum-pmf-eq-1)

lemma pmf-not-the-one-is-zero:
  fixes M::'a pmf
  assumes pmf M xa = 1
  assumes xa ≠ xb
  shows pmf M xb = 0
proof (rule ccontr)
  assume a1: ¬ pmf M xb = (0::real)
  have f0: pmf M xb > 0
    using a1 by simp
  have f1: (∑a i∈{xa,xb}. pmf M i) = (pmf M xa + pmf M xb)
    apply (simp add: infsetsum-def)
    by (simp add: assms(2) lebesgue-integral-count-space-finite)
  have f2: (∑a i::'a. pmf M i) ≥ (∑a i∈{xa,xb}. pmf M i)
    by (metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum sum-pmf-eq-1)
  from f1 f2 have (∑a i::'a. pmf M i) > 1
    using assms(1) f0 by linarith
  then show False
    using sum-pmf-eq-1
    by (simp add: sum-pmf-eq-1)
qed

lemma pmf-not-in-the-one-is-zero:
  fixes M::'a pmf
  assumes (∑a xb::'a ∈ A. pmf M xb) = 1
  assumes xa ∉ A
  shows pmf M xa = 0
proof (rule ccontr)
  assume a1: ¬ pmf M xa = (0::real)
  have f0: pmf M xa > 0
    using a1 by simp
  have f1: (∑a i∈A ∪ {xa}. pmf M i) = ((∑a xb::'a ∈ A. pmf M xb) + (∑a xb::'a ∈ {xa}. pmf M
xb))
    unfolding infsetsum-altdef abs-summable-on-altdef
    apply (subst set-integral-Un, auto)
    using abs-summable-on-altdef assms(2) apply fastforce
    using abs-summable-on-altdef apply blast
    using abs-summable-on-altdef by blast
  then have f2: ... = 1 + pmf M xa
    using assms(1) by auto
  then have f3: ... > 1
    using f0 by linarith
  then show False
    by (metis f1 f2 measure-pmf.prob-le-1 measure-pmf-conv-infsetsum not-le)
qed

lemma pmf-not-in-the-two-is-zero:
  fixes M::'a pmf
  assumes a ∈ {0..1}
  assumes sa ≠ sb
```

assumes *pmf M sa* = *a*
assumes *pmf M sb* = *1* − *a*
assumes *sc* ∉ {*sa*, *sb*}
shows *pmf M sc* = *0*
proof −
have *f1*: *infsetsum* (*pmf M*) {*sa*, *sb*} = *infsetsum* (*pmf M*) {*sa*} + *infsetsum* (*pmf M*) {*sb*}
by (*simp add*: *assms*(*2*))
then have *f2*: ... = *pmf M sa* + *pmf M sb*
by *simp*
then have *f3*: ... = *1*
using *assms*(*3*) *assms*(*4*) **by** *auto*
show *?thesis*
apply (*rule pmf-not-in-the-one-is-zero*[**where** *A* = {*sa*, *sb*}])
using *f1 f2 f3* **apply** *linarith*
using *assms*(*5*) **by** *auto*
qed

lemma *infsetsum-single*:
fixes *y*::′*a*
shows ($\sum_a$*xb*::′*a*. (*if xb* = *y then xa else 0*)) = *xa*
proof −
have ($\sum_a$*xb*::′*a*. (*if xb* = *y then* (*xa*) *else 0*)) =
($\sum_a$*xb*∈({*y*} ∪ {*t*. ¬*t*=*y*}). (*if xb* = *y then* (*xa*) *else 0*))
proof −
have *UNIV* = {*y*} ∪ {*a*. ¬ *a* = *y*}
by *blast*
then show *?thesis*
by *presburger*
qed
also have ... = ($\sum_a$*xb*∈({*y*}). (*if xb* = *y then* (*xa*) *else 0*)) +
($\sum_a$*xb*∈({*t*. ¬*t*=*y*}). (*if xb* = *y then* (*xa*) *else 0*))
unfolding *infsetsum-altdef abs-summable-on-altdef*
apply (*subst set-integral-Un*, *auto*)
using *abs-summable-on-altdef* **apply** *fastforce*
using *abs-summable-on-altdef* **by** (*smt abs-summable-on-0 abs-summable-on-cong mem-Collect-eq*)
also have ... = (*xa*) + ($\sum_a$*xb*∈({*t*. ¬*t*=*y*}). (*if xb* = *y then* (*xa*) *else 0*))
by *simp*
also have ... = (*xa*)
by (*smt add-cancel-left-right infsetsum-all-0 mem-Collect-eq*)
then show *?thesis*
by (*simp add*: *calculation*)
qed

lemma *infsetsum-single*′:
fixes *xa*::′*a* **and** *y*::′*a*
shows ($\sum_a$*xb*::′*a*. (*if xb* = *y then P*(*xa*) *else 0*)) = *P*(*xa*)
by (*simp add*: *infsetsum-single*)

lemma *pmf-sum-single*:
fixes $prob_v$::′*a pmf*
shows ($\sum_a$*xb*::′*a*. (*if xb* = *xa then pmf* $prob_v$ *xa else 0*)) = *pmf* $prob_v$ *xa*
by (*simp add*: *infsetsum-single*)

lemma *infsetsum-two*:

```
  assumes ya ≠ yb
  shows (∑a xb::'a. (if xb = ya then va else (if xb = yb then vb else 0))) = va + vb
  proof −
    have (∑a xb::'a. (if xb = ya then va else (if xb = yb then vb else 0))) =
          (∑a xb∈({ya,yb} ∪ {t. ¬t=ya ∧ ¬t=yb}).
      (if xb = ya then va else (if xb = yb then vb else 0)))
      proof −
        have UNIV = ({ya, yb} ∪ {t. ¬t=ya ∧ ¬t=yb})
          by blast
        then show ?thesis
          by presburger
      qed
    also have ... = (∑a xb∈({ya,yb}). (if xb = ya then va else (if xb = yb then vb else 0))) +
      (∑a xb∈({t. ¬t=ya ∧ ¬t=yb}). (if xb = ya then va else (if xb = yb then vb else 0)))
      unfolding infsetsum-altdef abs-summable-on-altdef
      apply (subst set-integral-Un, auto)
      using abs-summable-on-altdef apply fastforce
     using abs-summable-on-altdef by (smt abs-summable-on-0 abs-summable-on-cong mem-Collect-eq)
    also have ... = (∑a xb∈({ya,yb}). (if xb = ya then va else (if xb = yb then vb else 0))) +
      0
      by (smt infsetsum-all-0 mem-Collect-eq)
    also have ... = (∑a xb∈({ya}). (if xb = ya then va else (if xb = yb then vb else 0))) +
      (∑a xb∈({yb}). (if xb = ya then va else (if xb = yb then vb else 0)))
      apply (simp add: infsetsum-Un-disjoint)
      using assms by auto
    also have ... = va + vb
      using assms by auto
    then show ?thesis
      by (simp add: calculation)
  qed

lemma infsetsum-two':
  assumes xa ≠ xb
  assumes pmf M xa + pmf M xb = (1::real)
  shows (∑a x::'a. (pmf M x) · (Q x)) = pmf M xa · (Q xa) + pmf M xb · (Q xb)
proof −
  have f1: ∀ xc. xc ∉ {xa, xb} ⟶ pmf M xc = 0
    apply (auto, rule pmf-not-in-the-two-is-zero[where sa=xa and sb=xb and a=pmf M xa])
    apply auto+
      apply (simp add: pmf-le-1)
    using assms by auto+
  have f2: (∑a x::'a. (pmf M x) · (Q x)) =
    (∑a x::'a. (if x = xa then (pmf M xa) · (Q xa) else
      (if x = xb then (pmf M xb) · (Q xb) else (pmf M x) · (Q x))))
    by metis
  have f3: ... = (∑a x::'a. (if x = xa then (pmf M xa) · (Q xa) else
      (if x = xb then (pmf M xb) · (Q xb) else 0)))
    using f1
    by (smt infsetsum-cong insertE mult-not-zero singleton-iff)
  show ?thesis
    using f2 f3
    by (simp add: assms(1) infsetsum-two)
qed

lemma pmf-sum-single':
```

fixes $prob_v$::$'a$ *pmf*
shows $(\sum_a x::'a.\ pmf\ prob_v\ x \cdot pmf\ (pmf\text{-}of\text{-}list\ [(x,\ 1::real)])\ xa) = pmf\ prob_v\ xa$
proof −
have $pmf\ (pmf\text{-}of\text{-}list\ [(xb,\ 1::real)])\ xa = (if\ xb = xa\ then\ 1\ else\ 0)$
by (*simp add*: *filter.simps*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list*)
then have $(pmf\ prob_v\ xb \cdot pmf\ (pmf\text{-}of\text{-}list\ [(xb,\ 1::real)])\ xa) = (if\ xb = xa\ then\ pmf\ prob_v\ xa\ else\ 0)$
by *simp*
then show *?thesis*
using *pmf-sum-single*
by (*smt filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.set*(*1*) *list.set*(*2*) *list.simps*(*8*) *list.simps*(*9*) *mult-cancel-left1 mult-cancel-right1 pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list.Nil sum-list-simps*(*2*))
qed

lemma *pmf-sum-single''*:
fixes $prob_v$::$'a$ *pmf*
shows $(\sum_a x::'a.\ pmf\ prob_v\ xa \cdot pmf\ (pmf\text{-}of\text{-}list\ [(y,\ 1::real)])\ x) = pmf\ prob_v\ xa$
proof −
have *f1*: $\forall x.\ pmf\ (pmf\text{-}of\text{-}list\ [(y,\ 1::real)])\ x = (if\ y = x\ then\ 1\ else\ 0)$
by (*simp add*: *filter.simps*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list*)
then have *f2*: $\forall x.\ (pmf\ prob_v\ xa \cdot pmf\ (pmf\text{-}of\text{-}list\ [(y,\ 1::real)])\ x) = (if\ y = x\ then\ pmf\ prob_v\ xa\ else\ 0)$
by *simp*
then have *f3*: $(\sum_a x::'a.\ pmf\ prob_v\ xa \cdot pmf\ (pmf\text{-}of\text{-}list\ [(y,\ 1::real)])\ x) = (\sum_a x::'a.\ (if\ y = x\ then\ pmf\ prob_v\ xa\ else\ 0))$
by *simp*
have *f4*: $(\sum_a x::'a.\ (if\ x = y\ then\ pmf\ prob_v\ xa\ else\ 0)) = pmf\ prob_v\ xa$
by (*simp add*: *infsetsum-single'*[*of y* $\lambda x.\ pmf\ prob_v\ x\ xa$])
then show *?thesis*
by (*smt f3 infsetsum-cong*)
qed

lemma *infsum-singleton-is-single*:
assumes $\forall xb.\ xb \neq xa \longrightarrow P\ xb = (0::real)$
shows $(\sum_a x::'a.\ P\ x \cdot Q\ x) = P\ xa \cdot Q\ xa$
proof −
have $\forall x.\ P\ x \cdot Q\ x = (if\ x = xa\ then\ P\ xa \cdot Q\ xa\ else\ 0)$
apply (*auto*)
using *assms* **by** *blast*
then have *f1*: $(\sum_a x::'a.\ P\ x \cdot Q\ x) = (\sum_a x::'a.\ (if\ x = xa\ then\ P\ xa \cdot Q\ xa\ else\ 0))$
by *auto*
show *?thesis*
apply (*simp add*: *f1*)
by (*rule infsetsum-single*)
qed

lemma *pmf-sum-singleton-is-single*:
fixes M::$'a$ *pmf*
assumes $pmf\ M\ xa = 1$
shows $(\sum_a x::'a.\ pmf\ M\ x \cdot Q\ x) = Q\ xa$
proof −
have $\forall x.\ pmf\ M\ x \cdot Q\ x = (if\ x = xa\ then\ Q\ xa\ else\ 0)$
using *assms pmf-not-the-one-is-zero* **by** *fastforce*
then have $(\sum_a x::'a.\ pmf\ M\ x \cdot Q\ x) = (\sum_a x::'a.\ (if\ x = xa\ then\ Q\ xa\ else\ 0))$

by *auto*
then show *?thesis*
by (*simp add*: *infsetsum-single*)
qed

lemma *pmf-out-of-list-is-zero*:
assumes $r \in \{0..1\}$ $\neg$ $xa = xb$ $\neg$ $ii = xa$ $\neg$ $ii = xb$
shows *pmf* (*pmf-of-list* [(*xa*, *r*), (*xb*, *1−r*)]) *ii* = (*0*::*real*)
using *assms*
by (*smt atLeastAtMost-iff empty-iff filter.simps*(*1*) *filter.simps*(*2*) *fst-conv insert-iff list.set*(*1*) *list.set*(*2*) *list.simps*(*8*) *list.simps*(*9*) *pmf-of-list-wf-def pmf-pmf-of-list snd-conv sum-list.Cons sum-list.Nil*)

lemma *pmf-instance-from-one-full-state*:
assumes *pmf M xa* = *1*
shows *M* = (*pmf-of-list* [(*xa*, *1*)])
proof −
have *f1*: ∀ *ii*. *pmf M ii* = *pmf* (*pmf-of-list* [(*xa*, *1*)]) *ii*
proof
fix *ii*::*'a*
show *pmf M ii* = *pmf* (*pmf-of-list* [(*xa*, *1*)]) *ii* (**is** *?LHS* = *?RHS*)
proof (*cases ii* = *xa*)
case *True*
have *f1*: *?LHS* = *1.0*
by (*simp add*: *assms*(*1*) *True*)
have *f2*: *?RHS* = *1.0*
apply (*subst pmf-pmf-of-list*)
using *assms* **apply** (*simp add*: *pmf-of-list-wf-def*)
by (*simp add*: *True*)
show *?thesis* **using** *f1 f2* **by** *simp*
next
case *False*
have *f1*: *?LHS* = *0*
using *False assms pmf-not-the-one-is-zero* **by** *fastforce*
have *f2*: *?RHS* = *0*
apply (*subst pmf-pmf-of-list*)
using *assms* **apply** (*simp add*: *pmf-of-list-wf-def*)
using *False* **by** *auto*
show *?thesis* **using** *f1 f2* **by** *simp*
qed
qed
show *?thesis*
using *f1 pmf-eq-iff* **by** *auto*
qed

lemma *pmf-instance-from-two-full-states*:
assumes *pmf M xa* = *1* − *pmf M xb*
assumes ¬ *xa* = *xb*
shows *M* = (*pmf-of-list* [(*xa*, *pmf M xa*), (*xb*, *pmf M xb*)])
proof −
let *?r* = *pmf M xa*
have *f1*: ∀ *ii*. *pmf M ii* = *pmf* (*pmf-of-list* [(*xa*, *?r*), (*xb*, *1−?r*)]) *ii*
proof
fix *ii*::*'a*
show *pmf M ii* = *pmf* (*pmf-of-list* [(*xa*, *?r*), (*xb*, *1−?r*)]) *ii* (**is** *?LHS* = *?RHS*)

```
      proof (cases ii = xa)
        case True
        have f1: ?LHS = ?r
          by (simp add: True)
        have f2: ?RHS = ?r
          apply (subst pmf-pmf-of-list)
          using assms apply (simp add: pmf-of-list-wf-def)
          apply (simp add: pmf-le-1)
          using True assms(2) by auto
        show ?thesis using f1 f2 by simp
      next
        case False
        then have F: ¬ ii = xa
          by blast
        show ?thesis
          proof (cases ii = xb)
            case True
            have f1: ?LHS = 1−?r
              using True by (simp add: assms(1))
            have f2: ?RHS = 1−?r
              apply (subst pmf-pmf-of-list)
              using assms apply (simp add: pmf-of-list-wf-def)
              apply (simp add: pmf-le-1)
              using True assms(2) by auto
            show ?thesis using f1 f2 by simp
          next
            case False
            have f1: ?LHS = 0
              proof (rule ccontr)
                assume aa1: ¬ pmf M ii = (0::real)
                have f1: (∑a i∈{xa,xb,ii}. pmf M i) = (pmf M xa + pmf M xb + pmf M ii)
                  apply (simp add: infsetsum-def)
                  using F False lebesgue-integral-count-space-finite
                  by (smt assms(2) finite.emptyI finite.insertI insert-absorb insert-iff integral-pmf
                      pmf.rep-eq singleton-insert-inj-eq' sum.insert)
                have f2: (∑a i. pmf M i) ≥ (∑a i∈{xa,xb,ii}. pmf M i)
                  by (metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum sum-pmf-eq-1)
                from f1 f2 have (∑a i. pmf M i) > 1
                  using pmf-pos aa1 assms(1) by fastforce
                then show False
                  by (simp add: sum-pmf-eq-1)
              qed
            have f2: ?RHS = 0
              apply (subst pmf-pmf-of-list)
              using assms apply (simp add: pmf-of-list-wf-def)
              apply (simp add: pmf-le-1)
              using F False by auto
            show ?thesis using f1 f2 by simp
          qed
      qed
    qed
  show ?thesis
    using f1 pmf-eq-iff
    by (metis assms(1) cancel-ab-semigroup-add-class.diff-right-commute diff-eq-diff-eq)
qed
```

```
lemma pmf-instance-from-two-full-states':
  assumes pmf M xa = 1 − pmf M xb
  assumes ¬ xa = xb
  shows M = (pmf-of-list [(xa, (1::real))]) +pmf M xa (pmf-of-list [(xb, (1::real))])
  apply (subst pmf-instance-from-two-full-states[of M xa xb])
  using assms apply blast
  using assms(2) apply simp
  proof −
    have f0: pmf M xa ∈ {0..1}
      by (simp add: pmf-le-1)
    have f1: ∀ ii. pmf (pmf-of-list [(xa, pmf M xa), (xb, pmf M xb)]) ii =
      pmf (pmf-of-list [(xa, 1::real)] +pmf M xa pmf-of-list [(xb, 1::real)]) ii
      apply (auto)
      using f0 apply (simp add: pmf-wplus)
      proof −
        fix ii::'a
        show pmf (pmf-of-list [(xa, pmf M xa), (xb, pmf M xb)]) ii =
         pmf (pmf-of-list [(xa, 1::real)]) ii · pmf M xa +
         pmf (pmf-of-list [(xb, 1::real)]) ii · ((1::real) − pmf M xa)
           (is ?LHS = ?RHS)
         proof (cases ii = xa)
           case True
           have f1: ?LHS = pmf M xa
             apply (subst pmf-pmf-of-list)
             apply (smt assms(1) insert-iff list.set(1) list.set(2) list.simps(8) list.simps(9)
                 pmf-nonneg pmf-of-list-wf-def prod.sel(2) singletonD sum-list.Cons sum-list.Nil)
             using True assms(2) by auto
           have f2: ?RHS = pmf M xa
             apply (subst pmf-pmf-of-list)
             using assms apply (simp add: pmf-of-list-wf-def)
             apply (subst pmf-pmf-of-list)
             using assms apply (simp add: pmf-of-list-wf-def)
             using True assms(2) by auto
           show ?thesis using f1 f2 by simp
         next
           case False
           then have F: ¬ ii = xa
             by blast
           show ?thesis
             proof (cases ii = xb)
               case True
               have f1: ?LHS = pmf M xb
                 apply (subst pmf-pmf-of-list)
                 apply (smt assms(1) insert-iff list.set(1) list.set(2) list.simps(8) list.simps(9)
                     pmf-nonneg pmf-of-list-wf-def prod.sel(2) singletonD sum-list.Cons sum-list.Nil)
                 using True assms(2) by auto
               have f2: ?RHS = pmf M xb
                 apply (subst pmf-pmf-of-list)
                 using assms apply (simp add: pmf-of-list-wf-def)
                 apply (subst pmf-pmf-of-list)
                 using assms apply (simp add: pmf-of-list-wf-def)
                 using True assms by auto
               show ?thesis using f1 f2 by simp
             next
```

case *False*
have *f1*: *?LHS* = *0*
using *pmf-out-of-list-is-zero* **by** (*smt F False assms*(*1*) *assms*(*2*) *f0*)
have *f2*: *?RHS* = *0*
by (*smt F False filter.simps*(*1*) *filter.simps*(*2*) *fst-conv list.set*(*1*) *list.set*(*2*)
list.simps(*8*) *list.simps*(*9*) *pmf-of-list-wf-def pmf-pmf-of-list singletonD snd-conv*
sum-list.Cons sum-list.Nil sum-list-mult-const)
show *?thesis* **using** *f1 f2* **by** *simp*
qed
qed
qed
show *pmf-of-list* [(*xa*, *pmf M xa*), (*xb*, *pmf M xb*)] =
pmf-of-list [(*xa*, *1*::*real*)] $+_{pmf\ M\ xa}$ *pmf-of-list* [(*xb*, *1*::*real*)]
using *f1 pmf-eqI* **by** *blast*
qed

lemma *pmf-comp-set*:
shows (($\sum_a$ *i*∈(*X*). *pmf M i*) = *1*) = (($\sum_a$ *i*∈ −*X*. *pmf M i*) = *0*)
using *pmf-disj-set*[*of X* −*X*]
by (*simp add*: *sum-pmf-eq-1*)

lemma *pmf-all-zero*:
assumes (($\sum_a$ *i*∈(*X*). *pmf M i*) = *0*)
shows ∀ *x*∈*X*. *pmf M x* = *0*
proof
fix *x*::*'a*
assume *a1*: *x* ∈ *X*
show *pmf M x* = (*0*::*real*)
proof (*rule ccontr*)
assume *a2*: ¬ *pmf M x* = (*0*::*real*)
have *f1*: *pmf M x* > (*0*::*real*)
using *pmf-nonneg a2* **by** *simp*
have *f2*: ($\sum_a$ *i*∈(*X*). *pmf M i*) ≥ ($\sum_a$ *i*∈{*x*}. *pmf M i*)
using *a1*
by (*meson empty-subsetI infsetsum-mono-neutral-left insert-subset order-refl pmf-abs-summable*
pmf-nonneg)
have *f3*: ($\sum_a$ *i*∈{*x*}. *pmf M i*) = *pmf M x*
by *simp*
have *f4*: ($\sum_a$ *i*∈(*X*). *pmf M i*) > *0*
using *f2 f3 f1* **by** *linarith*
show *False*
using *f4* **by** (*simp add*: *assms*)
qed
qed

lemma *pmf-utp-univ*:
fixes $prob_v$::*'a pmf*
shows ($\sum_a$ *x*::*'a* | $[\![P]\!]_e$ (*more*, *x*) ∨ $[\![\neg P]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*) = (*1*::*real*)
by (*simp add*: *infsetsum-pmf-eq-1 lit.rep-eq not-upred-def uexpr-appl.rep-eq uminus-uexpr-def*)

lemma *pmf-disj-set2*:
assumes *X* ∩ *Y* = {}
shows ($\sum_a$ *i*∈(*X* ∪ *Y*). *pmf M i*) = ($\sum_a$ *i*∈*X*. *pmf M i*) + ($\sum_a$ *i*∈*Y*. *pmf M i*)
by (*metis assms infsetsum-Un-disjoint pmf-abs-summable*)

lemma *pmf-disj-set2'*:
  fixes $prob_v$::'a *pmf*
  assumes $\neg$ ($\exists$ *x*. *P x* $\wedge$ *Q x*)
  shows ($\sum_a x$::'a | *P x* $\vee$ *Q x*. *pmf* $prob_v$ *x*) =
      ($\sum_a x$::'a | *P x*. *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | *Q x*. *pmf* $prob_v$ *x*)
  apply (*simp add*: *infsetsum-altdef*)
proof –
  have *1*: {*x*::'a. *P x* $\vee$ *Q x*} = {*x*::'a. *P x*} $\cup$ {*x*::'a. *Q x*}
    using *assms* **by** *blast*
  show *set-lebesgue-integral* (*count-space UNIV*) {*x*::'a. *P x* $\vee$ *Q x*} (*pmf* $prob_v$) =
    set-lebesgue-integral (*count-space UNIV*) (*Collect P*) (*pmf* $prob_v$) +
    set-lebesgue-integral (*count-space UNIV*) (*Collect Q*) (*pmf* $prob_v$)
    apply (*simp add*: *1*)
    unfolding *infsetsum-altdef abs-summable-on-altdef*
    apply (*subst set-integral-Un*, *auto*)
    using *assms* **apply** *blast*
    using *abs-summable-on-altdef* **apply** *blast*
    using *abs-summable-on-altdef* **by** *blast*
qed

lemma *pmf-utp-disj-set2*:
  fixes $prob_v$::'a *pmf*
  assumes $\neg$ ($\exists$ *x*. $[\![P]\!]_e$ (*more*, *x*) $\wedge$ $[\![Q]\!]_e$ (*more*, *x*))
  shows ($\sum_a x$::'a | $[\![P]\!]_e$ (*more*, *x*) $\vee$ $[\![Q]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*) =
      ($\sum_a x$::'a | $[\![P]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | $[\![Q]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*)
  using *assms* **by** (*rule pmf-disj-set2'*)

lemma *pmf-disj-set3*:
  fixes $prob_v$::'a *pmf*
  assumes *a1*: $\neg$ ($\exists$ *x*. *P x* $\wedge$ *Q x*)
  assumes *a2*: $\neg$ ($\exists$ *x*. *P x* $\wedge$ *R x*)
  assumes *a3*: $\neg$ ($\exists$ *x*. *Q x* $\wedge$ *R x*)
  shows ($\sum_a x$::'a | *P x* $\vee$ *Q x* $\vee$ *R x*. *pmf* $prob_v$ *x*) =
      ($\sum_a x$::'a | *P x*. *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | *Q x*. *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | *R x*. *pmf* $prob_v$ *x*)
proof –
  have *1*: ($\sum_a x$::'a | *P x* $\vee$ *Q x* $\vee$ *R x*. *pmf* $prob_v$ *x*) =
        ($\sum_a x$::'a | *P x*. *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | *Q x* $\vee$ *R x*. *pmf* $prob_v$ *x*)
    apply (*rule pmf-disj-set2'*)
    using *assms* **by** *blast*
  have *2*: ($\sum_a x$::'a | *Q x* $\vee$ *R x*. *pmf* $prob_v$ *x*) = ($\sum_a x$::'a | *Q x*. *pmf* $prob_v$ *x*) + ($\sum_a x$::'a | *R x*. *pmf* $prob_v$ *x*)
    apply (*rule pmf-disj-set2'*)
    using *assms* **by** *blast*
  from *1 2* **show** *?thesis*
    by *auto*
qed

lemma *pmf-utp-comp0*:
  fixes $prob_v$::'a *pmf*
  assumes ($\sum_a x$::'a | $[\![P]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*) = (*1*::*real*)
  shows ($\sum_a x$::'a | $[\![\neg P]\!]_e$ (*more*, *x*). *pmf* $prob_v$ *x*) = (*0*::*real*)
  using *pmf-utp-univ*
  by (*smt Collect-cong Compl-eq assms bool-Compl-def lit.rep-eq mem-Collect-eq not-upred-def*
    pmf-comp-set uexpr-appl.rep-eq uminus-uexpr-def)

lemma *pmf-utp-comp0'*:
  fixes $prob_v$::'*a pmf*
  assumes $(\sum_a x{::}'a \mid P\ x.\ pmf\ prob_v\ x) = (1{::}real)$
  shows $(\sum_a x{::}'a \mid \neg\ P\ x.\ pmf\ prob_v\ x) = (0{::}real)$
  using *pmf-utp-univ*
  by (*metis Collect-neg-eq assms pmf-comp-set*)

lemma *pmf-utp-comp1*:
  fixes $prob_v$::'*a pmf*
  assumes $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (0{::}real)$
  shows $(\sum_a x{::}'a \mid [\![\neg P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
  using *pmf-utp-univ pmf-utp-comp0*
  by (*smt Collect-cong Compl-eq assms bool-Compl-def lit.rep-eq mem-Collect-eq not-upred-def pmf-comp-set uexpr-appl.rep-eq uminus-uexpr-def*)

lemma *pmf-comp1*:
  fixes $prob_v$::'*a pmf*
  assumes $(\sum_a x{::}'a \mid P\ x.\ pmf\ prob_v\ x) = (0{::}real)$
  shows $(\sum_a x{::}'a \mid \neg(P\ x).\ pmf\ prob_v\ x) = (1{::}real)$
  by (*smt Collect-cong Compl-eq assms bool-Compl-def lit.rep-eq mem-Collect-eq not-upred-def pmf-comp-set uexpr-appl.rep-eq uminus-uexpr-def*)

lemma *pmf-utp-comp1'*:
  fixes $prob_v$::'*a pmf*
  assumes $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (0{::}real)$
  shows $(\sum_a x{::}'a \mid \neg[\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
  by (*smt Collect-cong Compl-eq assms bool-Compl-def lit.rep-eq mem-Collect-eq not-upred-def pmf-comp-set uexpr-appl.rep-eq uminus-uexpr-def*)

lemma *pmf-utp-comp-not0*:
  fixes $prob_v$::'*a pmf*
  assumes $\neg\ (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
  shows $\neg\ (\sum_a x{::}'a \mid [\![\neg P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (0{::}real)$
  using *pmf-utp-univ pmf-utp-comp0 assms pmf-utp-comp1* **by** *fastforce*

lemma *pmf-utp-comp-not1*:
  fixes $prob_v$::'*a pmf*
  assumes $\neg\ (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (0{::}real)$
  shows $\neg\ (\sum_a x{::}'a \mid [\![\neg P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
  using *pmf-utp-univ pmf-utp-comp0 assms pmf-utp-comp1* **by** *fastforce*

term *count-space*
term *measure-space*
term *measure-of*
term *Abs-measure*
term *sigma-sets*
term *lebesgue-integral*
term *has-bochner-integral*

lemma *pmf-disj-leq*:
  fixes $prob_v$::'*a pmf* **and** *more*::'*a*
  shows $(\sum_a x{::}'a \mid P\ x.\ pmf\ prob_v\ x) \leq$
    $(\sum_a x{::}'a \mid P\ x \vee Q\ x.\ pmf\ prob_v\ x)$

by (*metis* (*mono-tags*, *lifting*) *infsetsum-mono-neutral-left le-less mem-Collect-eq pmf-abs-summable pmf-nonneg subsetI*)

lemma *pmf-disj-leq'*:
fixes $prob_v$::'a *pmf* **and** *more*::'a
shows $(\sum_a x::'a \mid P\ x.\ pmf\ prob_v\ x) \leq (\sum_a x::'a \mid Q\ x \vee P\ x.\ pmf\ prob_v\ x)$
by (*metis* (*mono-tags*, *lifting*) *infsetsum-mono-neutral-left le-less mem-Collect-eq pmf-abs-summable pmf-nonneg subsetI*)

lemma *pmf-utp-disj-leq*:
fixes $prob_v$::'a *pmf* **and** *P*::'a *hrel* **and** *Q*::'a *hrel* **and** *more*::'a
shows $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) \leq (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x)$
by (*simp add*: *pmf-disj-leq*)

lemma *pmf-utp-disj-eq-1*:
fixes $prob_v$::'a *pmf* **and** *P*::'a *hrel* **and** *Q*::'a *hrel* **and** *more*::'a
assumes $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
shows $(\sum_a x::'a \mid \exists\, v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) = (1::real)$
proof −
have *f1*: $(\sum_a x::'a \mid \exists\, v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) = (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x)$
by (*metis*)
have *f2*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) \leq 1$
by (*metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum*)
have *f3*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) \leq (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x)$
by (*rule pmf-utp-disj-leq*)
then have $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) \geq 1$
using *assms* **by** *auto*
then show *?thesis*
using *f2 f1* **by** *linarith*
qed

lemma *pmf-utp-disj-eq-1'*:
fixes $prob_v$::'a *pmf* **and** *P*::'a *hrel* **and** *Q*::'a *hrel* **and** *more*::'a
assumes $(\sum_a x::'a \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
shows $(\sum_a x::'a \mid \exists\, v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) = (1::real)$
proof −
have *f1*: $(\sum_a x::'a \mid \exists\, v::'a.\ [\![Q]\!]_e\ (more,\ x) \wedge v = x \vee [\![P]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) = (1::real)$
by (*simp add*: *assms pmf-utp-disj-eq-1*)
have $(\sum_a x::'a \mid \exists\, v::'a.\ [\![Q]\!]_e\ (more,\ x) \wedge v = x \vee [\![P]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) = (\sum_a x::'a \mid \exists\, v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x)$
by *meson*
then show *?thesis*
using *f1* **by** *auto*
qed

lemma *pmf-conj-eq-0*:
fixes $prob_v'$::'a *pmf* **and** $prob_v''$::'a *pmf*
assumes $(\sum_a x::'a \mid P\ x.\ pmf\ prob_v'\ x) = (0::real)$
assumes $(\sum_a x::'a \mid Q\ x.\ pmf\ prob_v''\ x) = (0::real)$

assumes $r \in \{0 {<}..{<} 1\}$
shows $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ (prob_v{}' +_r prob_v{}'')\ x) = (0{::}real)$
using *assms*(3) **apply** (*simp add*: *pmf-wplus*)
proof −
have $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}'\ x) = (0{::}real)$
using *assms infsetsum-nonneg*
by (*smt Collect-cong pmf-disj-leq pmf-nonneg*)
then have *1*: $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}'\ x \cdot r) = (0{::}real)$
using *assms*(3) **by** (*simp add*: *infsetsum-cmult-left pmf-abs-summable*)
have $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}''\ x) = (0{::}real)$
using *assms infsetsum-nonneg*
by (*smt Collect-cong pmf-disj-leq pmf-nonneg*)
then have *2*: $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}''\ x \cdot ((1{::}real) - r)) = (0{::}real)$
using *assms*(3) **by** (*simp add*: *infsetsum-cmult-left pmf-abs-summable*)
have $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}'\ x \cdot r + pmf\ prob_v{}''\ x \cdot ((1{::}real) - r))$
$= (\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}'\ x \cdot r) + (\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}''\ x \cdot ((1{::}real) - r))$
using *infsetsum-add* **by** (*simp add*: *infsetsum-add abs-summable-on-cmult-left pmf-abs-summable*)
then show $(\sum_a x{::}'a \mid P\ x \wedge Q\ x.\ pmf\ prob_v{}'\ x \cdot r + pmf\ prob_v{}''\ x \cdot ((1{::}real) - r)) = (0{::}real)$
using *1 2* **by** *linarith*
qed

lemma *pmf-utp-conj-eq-0*:
fixes $prob_v{}'{::}'a\ pmf$ **and** $prob_v{}''{::}'a\ pmf$ **and** $P{::}'a\ hrel$ **and** $Q{::}'a\ hrel$ **and** $more{::}'a$
assumes $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (0{::}real)$
assumes $(\sum_a x{::}'a \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x) = (0{::}real)$
assumes $r \in \{0 {<}..{<} 1\}$
shows $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (more,\ x).\ pmf\ (prob_v{}' +_r prob_v{}'')\ x) = (0{::}real)$
using *pmf-conj-eq-0 assms*(1) *assms*(2) *assms*(3) **by** *blast*

lemma *pmf-utp-disj-comm*:
fixes $prob_v{::}'a\ pmf$ **and** $P{::}'a\ hrel$ **and** $Q{::}'a\ hrel$ **and** $more{::}'a$
shows $(\sum_a x{::}'a \mid \exists\, v{::}'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) =$
$(\sum_a x{::}'a \mid \exists\, v{::}'a.\ [\![Q]\!]_e\ (more,\ x) \wedge v = x \vee [\![P]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x)$
by *meson*

lemma *pmf-utp-disj-imp*:
fixes $ok_v{::}bool$ **and** $more{::}'a$ **and** $ok_v{}'{::}bool$ **and** $prob_v{::}'a\ pmf$
assumes *a1*: $(\sum_a x{::}'a \mid \exists\, v{::}'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) =$
$(1{::}real)$
assumes *a2*: $\neg\ (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
assumes *a3*: $\neg\ (\sum_a x{::}'a \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
shows $(0{::}real) < (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \wedge \neg\ [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) \wedge$
$(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \wedge \neg\ [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) < (1{::}real)$
apply (*rule conjI*)
proof −
from *a1* **have** *f11*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
proof −
have $\{a.\ \exists\, aa.\ [\![P]\!]_e\ (more,\ a) \wedge aa = a \vee [\![Q]\!]_e\ (more,\ a) \wedge aa = a\} = \{a.\ [\![P]\!]_e\ (more,\ a) \vee$
$[\![Q]\!]_e\ (more,\ a)\}$
by *auto*
then show *?thesis*
using *a1* **by** *presburger*
qed
then have *f12*: $(\sum_a x{::}'a \mid ([\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (more,\ x)) \vee ([\![P]\!]_e\ (more,\ x) \wedge \neg[\![Q]\!]_e\ (more,$
$x)) \vee$

$(\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (1::real)$
by $(metis\ (no\text{-}types,\ lifting)\ Collect\text{-}cong)$
have $f13$: $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)) \vee (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,\ x)) \vee$
$(\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
$= (\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,\ x))\ .\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
apply $(rule\ pmf\text{-}disj\text{-}set3)$
by $blast+$
then have $f14$: $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,\ x))\ .\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (1::real)$
using $f12$ **by** $auto$

show $(0::real) < (\sum_a x::'a \mid \llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\ \llbracket Q\rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x)$
proof $(rule\ ccontr)$
assume $a11$: $\neg\ (0::real) < (\sum_a x::'a \mid \llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\ \llbracket Q\rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x)$
from $a11\ f14$ **have** $f111$: $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (1::real)$
by $(smt\ infsetsum\text{-}nonneg\ pmf\text{-}nonneg)$
have $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)) \vee (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf$
$prob_v\ x)$
$= (\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
apply $(rule\ pmf\text{-}disj\text{-}set2')$
by $blast$
then have $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)) \vee (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).$
$pmf\ prob_v\ x)$
$= (1::real)$
using $f111$ **by** $auto$
then have $(\sum_a x::'a \mid \llbracket Q\rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
by $(metis\ (mono\text{-}tags,\ lifting)\ Collect\text{-}cong)$
then show $False$
using $a3$ **by** $auto$
qed
next
from $a1$ **have** $f11$: $(\sum_a x::'a \mid \llbracket P\rrbracket_e\ (more,\ x) \vee \llbracket Q\rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
proof $-$
have $\{a.\ \exists\, aa.\ \llbracket P\rrbracket_e\ (more,\ a) \wedge aa = a \vee \llbracket Q\rrbracket_e\ (more,\ a) \wedge aa = a\} = \{a.\ \llbracket P\rrbracket_e\ (more,\ a) \vee$
$\llbracket Q\rrbracket_e\ (more,\ a)\}$
by $auto$
then show $?thesis$
using $a1$ **by** $presburger$
qed
then have $f12$: $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)) \vee (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,$
$x)) \vee$
$(\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (1::real)$
by $(metis\ (no\text{-}types,\ lifting)\ Collect\text{-}cong)$
have $f13$: $(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)) \vee (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,\ x)) \vee$
$(\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
$= (\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\llbracket P\rrbracket_e\ (more,\ x) \wedge \neg\llbracket Q\rrbracket_e\ (more,\ x))\ .\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg\llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
apply $(rule\ pmf\text{-}disj\text{-}set3)$
by $blast+$

then have *f14*: $(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \neg \llbracket Q \rrbracket_e\ (more,\ x))\ .\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg \llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (1::real)$
using *f12* **by** *auto*

show $(\sum_a x::'a \mid \llbracket P \rrbracket_e\ (more,\ x) \wedge \neg\ \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) < (1::real)$
proof (*rule ccontr*)
assume *a11*: $\neg\ (\sum_a x::'a \mid \llbracket P \rrbracket_e\ (more,\ x) \wedge \neg\ \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) < (1::real)$
from *a11* **have** *f110*: $(\sum_a x::'a \mid \llbracket P \rrbracket_e\ (more,\ x) \wedge \neg\ \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
by (*smt measure-pmf.prob-le-1 measure-pmf-conv-infsetsum*)
then have *f111*: $(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\neg \llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (0::real)$
using *f14* **by** *auto*
then have *f112*: $(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) = (0::real)$
by (*smt infsetsum-nonneg pmf-nonneg*)
have *f113*: $(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)) \vee (\llbracket P \rrbracket_e\ (more,\ x) \wedge \neg \llbracket Q \rrbracket_e\ (more,\ x)).$
$pmf\ prob_v\ x) =$
$(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x) +$
$(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \neg \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf\ prob_v\ x)$
apply (*rule pmf-disj-set2'*)
by *blast*
have $(\sum_a x::'a \mid (\llbracket P \rrbracket_e\ (more,\ x) \wedge \llbracket Q \rrbracket_e\ (more,\ x)) \vee (\llbracket P \rrbracket_e\ (more,\ x) \wedge \neg \llbracket Q \rrbracket_e\ (more,\ x)).\ pmf$
$prob_v\ x) =$
$(1::real)$
using *f112 f110* **by** (*simp add*: *f113*)
then have *f114*: $(\sum_a x::'a \mid \llbracket P \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
by (*metis* (*mono-tags*, *lifting*) *Collect-cong*)
then show *False*
using *a2* **by** *auto*
qed
qed

lemma *pmf-utp-disj-imp'*:
fixes ok_v::*bool* **and** *more*::*'a* **and** ok_v'::*bool* **and** $prob_v$::*'a pmf*
assumes *a1*: $(\sum_a x::'a \mid \exists\, v::'a.\ \llbracket P \rrbracket_e\ (more,\ x) \wedge v = x \vee \llbracket Q \rrbracket_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x) =$
$(1::real)$
assumes *a2*: $\neg\ (\sum_a x::'a \mid \llbracket P \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
assumes *a3*: $\neg\ (\sum_a x::'a \mid \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
shows $(0::real) < (\sum_a x::'a \mid \neg \llbracket P \rrbracket_e\ (more,\ x) \wedge\ \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) \wedge$
$(\sum_a x::'a \mid \neg \llbracket P \rrbracket_e\ (more,\ x) \wedge\ \llbracket Q \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) < (1::real)$
proof −
have $(0::real) < (\sum_a x::'a \mid \llbracket Q \rrbracket_e\ (more,\ x) \wedge\ \neg \llbracket P \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) \wedge$
$(\sum_a x::'a \mid \llbracket Q \rrbracket_e\ (more,\ x) \wedge\ \neg \llbracket P \rrbracket_e\ (more,\ x).\ pmf\ prob_v\ x) < (1::real)$
using *assms* **by** (*simp add*: *pmf-utp-disj-imp pmf-utp-disj-comm*)
then show *?thesis*
by (*metis* (*mono-tags*, *lifting*) *Collect-cong*)
qed

lemma *pmf-sum-subset-imp-1*:
assumes $P \subseteq Q$
assumes $(\sum_a i::'a \in P.\ pmf\ M\ i) = 1$
shows $(\sum_a i::'a \in Q.\ pmf\ M\ i) = 1$
proof −
have *f1*: *infsetsum* (*pmf M*) $P \leq$ *infsetsum* (*pmf M*) Q
apply (*rule infsetsum-mono-neutral-left*)

```
    apply (simp add: pmf-abs-summable)+
    apply (simp add: assms)
    by simp
  show ?thesis
    using f1 assms
    by (metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum order-class.order.antisym)
qed
```

B.2 Measures

Construct 0.prob and 1.prob from a supplied pmf P, and two sets A and B. We cannot modify the probability function in pmf since it has to satisfy a condition (*prob-space M*). But we can modify the function in the measure space by dropping P to a measure, then modifying measure function, afterwards lifting back to the probability space.

But when lifting, we need to prove additional laws *prob-space M ∧ sets M = UNIV ∧ (AE x in M. measure M {x} ≠ 0)* to ensure modified measure is a probability measure.

```
definition proj-f :: 'a set ⇒ 'a set ⇒ 'a pmf ⇒ 'a measure (ℱ) where
proj-f A B P = measure-of (space P) (sets P)
  (λAA.
    (   emeasure P (AA ∩ (A−B))∗(((∑a i∈B−A. pmf P i) + (∑a i∈A−B. pmf P i))/(∑a i∈A−B.
pmf P i))
      + emeasure P (AA ∩ (A ∩ B))
    )
  )


lemma emeasure-infsum-eq: emeasure (P::'a pmf) A = (∑a i∈A. pmf P i)
  by (simp add: measure-pmf.emeasure-eq-measure measure-pmf-conv-infsetsum)

lemma proj-f-sets: sets (ℱ A B P) = UNIV
   apply (simp add: proj-f-def)
  by auto

lemma proj-f-space: space (ℱ A B P) = UNIV
  by (simp add: proj-f-def)

lemma pmf-measure-zero:
  assumes ∀ i ∈ A. emeasure (measure-pmf P) {i} = (0::ennreal)
  shows emeasure (measure-pmf P) A = (0::ennreal)
 by (metis assms disjoint-iff-not-equal emeasure-Int-set-pmf emeasure-empty emeasure-pmf-single-eq-zero-iff)

lemma proj-f-emeasure: emeasure (ℱ A B P) C =
   (λAA. emeasure P (AA ∩ (A−B)) ∗ (((∑a i∈B−A . pmf P i) + (∑a i∈A−B . pmf P i))/(∑a
i∈A−B . pmf P i))
  + emeasure P (AA ∩ (A ∩ B))) C
  apply (simp add: proj-f-def)
  apply (intro emeasure-measure-of-sigma)
  apply (metis sets.sigma-algebra-axioms sets-measure-pmf space-measure-pmf)
  apply (simp add: positive-def)
  defer
  apply simp
  proof (rule countably-additiveI)
    fix Aa :: nat ⇒ 'a set
    let ?A-B = infsetsum (pmf P) (A−B)
```

```
  let ?B-A = infsetsum (pmf P) (B−A)
  let ?A-and-B = infsetsum (pmf P) (A ∩ B)
  let ?em-A-and-B = emeasure (measure-pmf P) (A ∩ B)
  let ?em-A-B = emeasure (measure-pmf P) (A − B)
  let ?em-B-A = emeasure (measure-pmf P) (B − A)
  assume *: range Aa ⊆ UNIV disjoint-family Aa ⋃ (range Aa) ∈ UNIV
  let ?f = λi::nat . emeasure (measure-pmf P) (Aa i ∩ (A − B)) ·
      ennreal ((?B-A + ?A-B) / ?A-B) +
      emeasure (measure-pmf P) (Aa i ∩ (A ∩ B))

  have f1: (∑ i::nat. ?f i) = (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A − B)) ·
      ennreal ((?B-A + ?A-B) / ?A-B)) +
     (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A ∩ B)))
   apply (rule sym, rule suminf-add)
   apply blast
   by blast
  have f2: (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A − B)) ·
      ennreal ((?B-A + ?A-B) / ?A-B))
     = (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A − B))) ·
      ennreal ((?B-A + ?A-B) / ?A-B)
   by simp
  have f2: (⋃ i. Aa i) = ⋃ (range Aa)
   by blast
  then have f3: ((⋃ i. Aa i) ∩ (A − B)) = (⋃ i. Aa i ∩ (A − B))
   by blast
  then have f3': ((⋃ i. Aa i) ∩ (A  ∩ B)) = (⋃ i. Aa i ∩ (A ∩ B))
   by blast
  have f4: (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A − B)))
   = emeasure (measure-pmf P) (⋃ i. Aa i ∩ (A − B))
   apply (rule suminf-emeasure)
   apply simp
  by (meson *(2) disjoint-family-subset semilattice-inf-class.inf.absorb-iff2 semilattice-inf-class.inf-left-idem)
  also have f4': ... = emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A − B))
   using f3 by simp
  have f5: (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A ∩ B)))
   = emeasure (measure-pmf P) (⋃ i. Aa i ∩ (A ∩ B))
   apply (rule suminf-emeasure)
   apply simp
  by (meson *(2) disjoint-family-subset semilattice-inf-class.inf.absorb-iff2 semilattice-inf-class.inf-left-idem)
  have f5': ... = emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A ∩ B))
   using f3' by simp
  have f6: (∑ i::nat. ?f i) = (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A − B))) ·
      ennreal ((?B-A + ?A-B) / ?A-B)
     + (∑ i::nat. emeasure (measure-pmf P) (Aa i ∩ (A ∩ B)))
   using f1 f2 by simp
  have f6': ... = emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A − B)) ·
      ennreal ((?B-A + ?A-B) / ?A-B)
     + emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A ∩ B))
   using f4 f4' f5 f5' by simp
  then show (∑ i::nat. ?f i) =
   emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A − B)) ·
   ennreal ((?B-A + ?A-B) / ?A-B) +
   emeasure (measure-pmf P) (⋃ (range Aa) ∩ (A ∩ B))
   using f6 by simp
qed
```

```
lemma prob-space-proj-f:
  fixes P::'a pmf and A::'a set and B::'a set
  assumes (∑a i∈A ∪ B . pmf P i) = (1::real)
  assumes (∑a i∈A−B . pmf P i) > (0::real)
  assumes (∑a i∈B−A . pmf P i) > (0::real)
  shows prob-space (ℱ A B P)
  apply (intro prob-spaceI)
  apply (simp add: prob-space-def proj-f-def)
  proof −
    let ?A-B = infsetsum (pmf P) (A−B)
    let ?B-A = infsetsum (pmf P) (B−A)
    let ?A-and-B = infsetsum (pmf P) (A ∩ B)
    let ?em-A-and-B = emeasure (measure-pmf P) (A ∩ B)
    let ?em-A-B = emeasure (measure-pmf P) (A − B)
    let ?em-B-A = emeasure (measure-pmf P) (B − A)
    have f0: (∑a i∈A ∪ B . pmf P i) = (∑a i∈(A ∩ B) ∪ (A−B) ∪ (B−A)  . pmf P i)
      by (simp add: Int-Diff-Un)
    also have f0': ...=?A-B + ?B-A + ?A-and-B
      by (smt Diff-Diff-Int Un-Diff-Int calculation infsetsum-Diff infsetsum-Un-Int
          lattice-class.inf-sup-aci(1) pmf-abs-summable semilattice-sup-class.sup-ge1)
    have f1: (space
          (measure-of UNIV UNIV
            (λAA::'a set.
                emeasure (measure-pmf P) (AA ∩ (A − B)) ·
                ennreal ((?B-A + ?A-B) / ?A-B) +
                emeasure (measure-pmf P) (AA ∩ (A ∩ B))))) = UNIV
      by (simp add: space-measure-of-conv)
    have f2: emeasure
          (measure-of UNIV UNIV
            (λAA::'a set.
                emeasure (measure-pmf P) (AA ∩ (A − B)) ·
                ennreal ((?B-A + ?A-B) / ?A-B) +
                emeasure (measure-pmf P) (AA ∩ (A ∩ B)))) UNIV =
          (λAA::'a set.
                emeasure (measure-pmf P) (AA ∩ (A − B)) ·
                ennreal ((?B-A + ?A-B) / ?A-B) +
                emeasure (measure-pmf P) (AA ∩ (A ∩ B))) UNIV
      using proj-f-emeasure by (metis proj-f-def sets-measure-pmf space-measure-pmf)
    have f3: ?em-A-B = ?A-B
      by (simp add: measure-pmf.emeasure-eq-measure measure-pmf-conv-infsetsum)
    have f4: ?em-A-B > 0
      using assms(2) by (simp add: f3)
    have f5: ?B-A = ?em-B-A
      by (simp add: measure-pmf.emeasure-eq-measure measure-pmf-conv-infsetsum)
    have f5': ?A-B + ?B-A
      = ?em-A-B + ?em-B-A
      by (simp add: f3 f5 infsetsum-nonneg)
    have f5'': (?A-B + ?B-A) / ?A-B
      = (?em-A-B + ?em-B-A) / ?em-A-B
      by (smt assms(2) assms(3) divide-ennreal f3 f5')
    have f5''': ?A-B · ((?B-A + ?A-B)/?A-B) = (?B-A + ?A-B)
      using assms(2) by auto
    have f6: (λAA::'a set.
                emeasure (measure-pmf P) (AA ∩ (A − B)) ·
```

ennreal ((*?B-A* + *?A-B*) / *?A-B*) +
emeasure (*measure-pmf P*) (*AA* ∩ (*A* ∩ *B*))) *UNIV*
= (
?em-A-B ·
ennreal ((*?B-A* + *?A-B*) / *?A-B*) +
?em-A-and-B)
by *auto*
have *f7*: ... = (
ennreal ?A-B · ((*?B-A* + *?A-B*) / *?A-B*) +
?em-A-and-B)
using *f3 f5 f5''* **by** (*simp add*: *add.commute*)
have *f8*: ... = (*ennreal ?A-B* · ((*?B-A* + *?A-B*) / *?A-B*) +
ennreal ?A-and-B)
by (*simp add*: *measure-pmf.emeasure-eq-measure measure-pmf-conv-infsetsum*)
have *f9*: ... = (*ennreal* (*?B-A* + *?A-B*) + *ennreal ?A-and-B*)
using *f5'''* **by** (*smt assms*(*2*) *ennreal-mult'*)
have *f10*: ...= *ennreal* (*?B-A* + *?A-B* + *?A-and-B*)
by (*simp add*: *infsetsum-nonneg*)
have *f11*: ... = *ennreal* (*1*)
using *f0 f0'* **by** (*simp add*: *assms*(*1*))
then show *emeasure*
(*measure-of UNIV UNIV*
(λ*AA*::*'a set*.
emeasure (*measure-pmf P*) (*AA* ∩ (*A* − *B*)) ·
ennreal ((*infsetsum* (*pmf P*) (*B* − *A*) + *infsetsum* (*pmf P*) (*A* − *B*)) / *infsetsum* (*pmf P*) (*A*
− *B*)) +
emeasure (*measure-pmf P*) (*AA* ∩ (*A* ∩ *B*))))
UNIV = (*1*::*ennreal*)
by (*simp add*: *f10 f2 f7 f8 f9*)
qed

lemma *proj-f-AE*:
fixes *P*::*'a pmf* **and** *A*::*'a set* **and** *B*::*'a set*
assumes ($\sum_a$ *i*∈*A* ∪ *B* . *pmf P i*) = (*1*::*real*)
assumes ($\sum_a$ *i*∈*A*−*B* . *pmf P i*) > (*0*::*real*)
assumes ($\sum_a$ *i*∈*B*−*A* . *pmf P i*) > (*0*::*real*)
shows *AE x*::*'a in* $\mathcal{F}$ *A B P*. ¬ *Sigma-Algebra.measure* ($\mathcal{F}$ *A B P*) {*x*} = (*0*::*real*)
apply (*rule AE-I*[**where** *N*={*x*::*'a*. ((
emeasure (*measure-pmf P*) ({*x*} ∩ (*A*−*B*)) = *0*) ∧
(*emeasure* (*measure-pmf P*) ({*x*} ∩ *A* ∩ *B*) = *0*))}])
proof −
have {*x*::*'a*. *x* ∈ *space* ($\mathcal{F}$ *A B P*) ∧ ¬ ¬ *Sigma-Algebra.measure* ($\mathcal{F}$ *A B P*) {*x*} = (*0*::*real*)}
= {*x*::*'a*. *Sigma-Algebra.measure* ($\mathcal{F}$ *A B P*) {*x*} = (*0*::*real*)}
by (*simp add*: *proj-f-space*)
also have ... =
{*x*::*'a*. *Sigma-Algebra.measure* (*measure-of UNIV UNIV*
(λ*AA*. *emeasure P* (*AA* ∩ (*A*−*B*)) ∗ ((($\sum_a$ *i*∈*B*−*A* . *pmf P i*) + ($\sum_a$ *i*∈*A*−*B* . *pmf P i*))/($\sum_a$
i∈*A*−*B* . *pmf P i*))
+ *emeasure P* (*AA* ∩ (*A* ∩ *B*)))) {*x*} = (*0*::*real*)}
by (*simp add*: *proj-f-def*)
also have ... = {*x*::*'a*. *enn2real* ((λ*AA*::*'a set*.
emeasure (*measure-pmf P*) (*AA* ∩ (*A*−*B*)) ·
ennreal ((*infsetsum* (*pmf P*) (*A*−*B*) + *infsetsum* (*pmf P*) (*B*−*A*)) / *infsetsum* (*pmf P*) (*A*−*B*))
+
emeasure (*measure-pmf P*) (*AA* ∩ (*A* ∩ *B*))) {*x*}) = (*0*::*real*)}

apply (*simp add*: *measure-def*)
by (*smt Collect-cong Sigma-Algebra.measure-def UNIV-I calculation proj-f-emeasure proj-f-space*)
also have ... = $\{x::'a.$ (($\lambda AA::'a$ *set*.
emeasure (*measure-pmf P*) ($AA \cap (A-B)$) $\cdot$
ennreal ((*infsetsum* (*pmf P*) ($A-B$) + *infsetsum* (*pmf P*) ($B-A$)) / *infsetsum* (*pmf P*) ($A-B$))
+
emeasure (*measure-pmf P*) ($AA \cap (A \cap B)$))) $\{x\}$) = (0::*real*)$\}$
apply (*simp add*: *enn2real-eq-0-iff*)
using *ennreal-mult-eq-top-iff* **by** *auto*
also have ... = $\{x::'a.$ (($\lambda AA::'a$ *set*.
emeasure (*measure-pmf P*) ($AA \cap (A-B)$) $\cdot$
ennreal ((*infsetsum* (*pmf P*) ($A-B$) + *infsetsum* (*pmf P*) ($B-A$)) / *infsetsum* (*pmf P*) ($A-B$)))
$\{x\}$ = 0) $\wedge$
(($\lambda AA::'a$ *set*. *emeasure* (*measure-pmf P*) ($AA \cap (A \cap B)$)) $\{x\}$ = 0)$\}$
by *simp*
also have ... = $\{x::'a.$ (($\lambda AA::'a$ *set*.
emeasure (*measure-pmf P*) ($AA \cap (A-B)$)) $\{x\}$ = 0) $\wedge$
(($\lambda AA::'a$ *set*. *emeasure* (*measure-pmf P*) ($AA \cap (A \cap B)$)) $\{x\}$ = 0)$\}$
using *assms*(2) *assms*(3) **by** *force*
also have ... = $\{x::'a.$ (
emeasure (*measure-pmf P*) ($\{x\} \cap (A-B)$) = 0) $\wedge$
(*emeasure* (*measure-pmf P*) ($\{x\} \cap (A \cap B)$) = 0)$\}$
by *blast*
then show $\{x::'a.\ x \in$ *space* ($\mathcal{F}$ A B P) $\wedge \neg \neg$ *Sigma-Algebra.measure* ($\mathcal{F}$ A B P) $\{x\}$ = (0::*real*)$\}$
$\subseteq \{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\}$
by (*metis* (*no-types*, *lifting*) *Collect-mono-iff Int-assoc calculation*)
next
have *f1*: *emeasure* ($\mathcal{F}$ A B P)
$\{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\}$
= (λAA. *emeasure* P ($AA \cap (A-B)$) $*$
((($\sum_a i{\in}B-A$. *pmf* P i) + ($\sum_a i{\in}A-B$. *pmf* P i))/($\sum_a i{\in}A-B$. *pmf* P i))
+ *emeasure* P ($AA \cap (A \cap B)$)))
$\{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\}$
by (*rule proj-f-emeasure*)
have *f2*: $\forall\, i \in \{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\}$.
emeasure (*measure-pmf P*) ($\{i\} \cap (A - B)$) = (0::*ennreal*)
by *blast*
have *f3*: $\forall\, i \in \{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\}$.
emeasure (*measure-pmf P*) ($\{i\} \cap A \cap B$) = (0::*ennreal*)
by *blast*
have *f4*: *emeasure* P ($\{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\} \cap (A-B)$) = 0
apply (*rule pmf-measure-zero*)
by (*simp add*: *Int-insert-right lattice-class.inf-sup-aci*(1))
have *f5*: *emeasure* P ($\{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$
emeasure (*measure-pmf P*) ($\{x\} \cap A \cap B$) = (0::*ennreal*)$\} \cap (A \cap B)$) = 0
apply (*rule pmf-measure-zero*)
by (*simp add*: *Int-insert-right lattice-class.inf-sup-aci*(1))
show *emeasure* ($\mathcal{F}$ A B P)
$\{x::'a.$ *emeasure* (*measure-pmf P*) ($\{x\} \cap (A - B)$) = (0::*ennreal*) $\wedge$

emeasure (*measure-pmf* *P*) ({*x*} ∩ *A* ∩ *B*) = (*0*::*ennreal*)} = (*0*::*ennreal*)
using *f1 f4 f5* **by** *simp*
next
show {*x*::′*a*.
emeasure (*measure-pmf* *P*) ({*x*} ∩ (*A* − *B*)) = (*0*::*ennreal*) ∧
emeasure (*measure-pmf* *P*) ({*x*} ∩ *A* ∩ *B*) = (*0*::*ennreal*)}
∈ *sets* (ℱ *A B P*)
by (*simp add*: *proj-f-sets*)
qed

lemma *proj-f-measure-pmf*:
fixes *P*::′*a pmf* **and** *A*::′*a set* **and** *B*::′*a set*
assumes ($\sum_a$ *i*∈*A* ∪ *B* . *pmf P i*) = (*1*::*real*)
assumes ($\sum_a$ *i*∈*A*−*B* . *pmf P i*) > (*0*::*real*)
assumes ($\sum_a$ *i*∈*B*−*A* . *pmf P i*) > (*0*::*real*)
shows (*measure-pmf* (*Abs-pmf* (ℱ *A B P*))) = ℱ *A B P*
apply (*rule pmf.Abs-pmf-inverse*)
apply (*auto*)
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *prob-space-proj-f* **apply** *blast*
apply (*simp add*: *proj-f-sets*)
using *assms*(*1*) *assms*(*2*) *assms*(*3*) *proj-f-AE* **by** *blast*

lemma *enn2real-distrib*: *enn2real* (*A*∗*c* + *A*∗*d*) = *enn2real* (*A*∗(*c*+*d*))
by (*simp add*: *distrib-left*)

lemma *proj-f-sum-eq-1*:
fixes *P*::′*a pmf* **and** *A*::′*a set* **and** *B*::′*a set*
assumes ($\sum_a$ *i*∈*A* ∪ *B* . *pmf P i*) = (*1*::*real*)
assumes ($\sum_a$ *i*∈*A*−*B* . *pmf P i*) > (*0*::*real*)
assumes ($\sum_a$ *i*∈*B*−*A* . *pmf P i*) > (*0*::*real*)
shows ($\sum_a$ *x*::′*a* | *x* ∈ *A* . *pmf* (*Abs-pmf* (ℱ *A B P*)) *x*) = (*1*::*real*)
proof −
have *f1*: ($\sum_a$ *x*::′*a* | *x* ∈ *A* . *pmf* (*Abs-pmf* (ℱ *A B P*)) *x*)
= *measure* (*measure-pmf* (*Abs-pmf* (ℱ *A B P*))) *A*
by (*simp add*: *measure-pmf-conv-infsetsum*)
then have *f2*: ... = *measure* (ℱ *A B P*) *A*
using *assms* **by** (*simp add*: *proj-f-measure-pmf*)
then have *f3*: ... = *enn2real* (*emeasure* (*measure-of* (*space P*) (*sets P*)
(λ*AA*. *emeasure P* (*AA* ∩ (*A*−*B*)) ∗ (
(($\sum_a$ *i*∈*B*−*A* . *pmf P i*) + ($\sum_a$ *i*∈*A*−*B* . *pmf P i*))/($\sum_a$ *i*∈*A*−*B* . *pmf P i*))
+ *emeasure P* (*AA* ∩ (*A* ∩ *B*)))) *A*)
by (*simp add*: *proj-f-def measure-def*)
then have *f4*: ... = *enn2real* ((λ*AA*. *emeasure P* (*AA* ∩ (*A*−*B*)) ∗
((($\sum_a$ *i*∈*B*−*A* . *pmf P i*) + ($\sum_a$ *i*∈*A*−*B* . *pmf P i*))/($\sum_a$ *i*∈*A*−*B* . *pmf P i*))
+ *emeasure P* (*AA* ∩ (*A* ∩ *B*))) *A*)
by (*simp add*: *Sigma-Algebra.measure-def proj-f-emeasure*)
then have *f5*: ... = *enn2real* (*emeasure P* ((*A*−*B*)) ∗
((($\sum_a$ *i*∈*B*−*A* . *pmf P i*) + ($\sum_a$ *i*∈*A*−*B* . *pmf P i*))/($\sum_a$ *i*∈*A*−*B* . *pmf P i*))
+ *emeasure P* ((*A* ∩ *B*)))
by (*metis* (*no-types*, *lifting*) *Int-Diff semilattice-inf-class.inf.idem*
semilattice-inf-class.inf-left-idem)
then show *?thesis*
by (*metis Int-commute Sigma-Algebra.measure-def assms*(*1*) *assms*(*2*) *assms*(*3*)

bounded-semilattice-inf-top-class.inf-top.right-neutral emeasure-pmf-UNIV enn2real-eq-1-iff f1 proj-f-emeasure proj-f-measure-pmf)
qed

end

C Probabilistic Designs Laws

theory *utp-prob-des-laws*
imports *UTP−Calculi.utp-wprespec*
UTP−Designs.utp-designs
HOL−Probability.Probability-Mass-Function

utp-prob-des

utp-prob-pmf-laws
begin recall-syntax

C.1 Probability Embedding

Inverse of $\mathcal{K}$ [1, Corollary 3.7]: embedding a standard design (P) in the probabilistic world then forgetting its probability distribution is equal to P itself.

lemma *pemp-inv*:
assumes P *is* **N**
shows $\mathcal{K}(P)$; ; **fp** $= P$
proof −
have *1*: $P \sqsubseteq \mathcal{K}(P)$; ; **fp**
apply (*simp add*: *pemb-def forget-prob-def*)
by (*simp add*: *wprespec1*)
also have *2*: $\mathcal{K}(P)$; ; **fp** $\sqsubseteq P$
proof −
obtain pre_P $post_P$
where p:$P = (pre_P \vdash_n post_P)$
using *assms* **by** (*metis ndesign-form*)
have $\mathcal{K}(P)$; ; **fp** $= \mathcal{K}(pre_P \vdash_n post_P)$; ; **fp**
using *p* **by** *auto*
also have $\mathcal{K}(pre_P \vdash_n post_P)$; ; **fp** $\sqsubseteq pre_P \vdash_n post_P$
apply (*simp add*: *pemb-def forget-prob-def wprespec-def*)
apply (*rel-simp*)
proof −
fix ok_v::*bool* **and** *more*::'*a* **and** ok_v'::*bool* **and** *morea*::'*b*
assume *a1*: $ok_v \wedge [\![pre_P]\!]_e\ more \longrightarrow ok_v' \wedge [\![post_P]\!]_e\ (more, morea)$
show $\exists (ok_v''::bool)\ prob_v::'b\ pmf.$
$([\![pre_P]\!]_e\ more \longrightarrow$
$ok_v \longrightarrow$
$(\forall (ok_v::bool)\ morea::'b.$
$ok_v \wedge [\![post_P]\!]_e\ (more, morea) \vee ok_v'' \wedge (ok_v \longrightarrow \neg\ (0::real) < pmf\ prob_v\ morea))) \wedge$
$(ok_v'' \longrightarrow ok_v' \wedge (0::real) < pmf\ prob_v\ morea)$
apply (*rule-tac* x=ok_v' **in** *exI*)
apply (*rule-tac x*=*pmf-of-list* [(*morea*, *1.0*)] **in** *exI*)
apply (*auto*)
using *a1* **apply** *blast*
using *a1* **apply** *blast*
apply (*rename-tac* ok_v'' *moreaa*)

```
      proof −
        fix okv''::bool and moreaa::'b
        assume a21: ⟦preP⟧e more
        assume a22: okv
        assume a23: okv''
        assume a2: (0::real) < pmf (pmf-of-list [(morea, (1::real))]) moreaa
        have f1: moreaa = morea
          proof (rule ccontr)
            assume a3: ¬ moreaa = morea
            have f2: pmf-of-list-wf [(morea, (1::real))]
              by (simp add: pmf-of-list-wf-def)
            have f3: pmf (pmf-of-list [(morea, (1::real))]) moreaa =
                  sum-list (map snd (filter (λz. fst z = moreaa) [(morea, (1::real))]))
              by (simp add: f2 pmf-pmf-of-list)
            then have ... = 0
              using a3 by auto
            then show False
              using a2 f3 by linarith
          qed
        show ⟦postP⟧e (more, moreaa)
          using a1 a21 a22 a23 a2 f1 by blast
      next
        show (0::real) < pmf (pmf-of-list [(morea, 1::real)]) morea
          by (simp add: pmf-of-list-wf-def pmf-pmf-of-list)
      qed
    qed
    then show ?thesis
      by (simp add: p)
  qed
  show ?thesis
    using 1 2 by simp
qed

lemma pemp-bot: 𝒦(⊥D) = ⊥D
  apply (simp add: upred-defs)
  by (rel-auto)

lemma pemp-bot': 𝒦(⊥D) = true
  apply (simp add: upred-defs)
  by (rel-auto)

lemma pemp-assigns: 𝒦(⟨σ⟩D) = U(true ⊢n ($prob´((σ † &v)<) = 1))
  by (simp add: assigns-d-ndes-def prob-lift wp usubst, rel-auto)

lemma pemp-skip: 𝒦(IID) = U(true ⊢n ($prob´($v) = 1))
  by (simp only: assigns-d-id[THEN sym] pemp-assigns usubst, rel-auto)

lemma pemp-assign:
  fixes e :: (-, -) uexpr
  shows 𝒦(x :=D e) = U(true ⊢n ($prob´($v⟦e</$x⟧) = 1))
  by (simp add: pemp-assigns wp usubst, rel-auto)

lemma pemp-cond:
  assumes P is N Q is N
  shows 𝒦(P ◁ b ▷D Q) = 𝒦(P) ◁ b ▷D 𝒦(Q)
```

apply (*ndes-simp cls*: *assms*)
by (*rel-auto*)

C.1.1 Demonic choice

lemma *pemb-intchoice*:
shows $\mathcal{K}((p \vdash_n P) \sqcap (q \vdash_n Q))$
$= \mathcal{K}(p \vdash_n P) \sqcap \mathcal{K}(q \vdash_n Q) \sqcap (\bigsqcap\ r \in \{0{<}..{<}1\} \cdot (\mathcal{K}(p \vdash_n P) \oplus_r \mathcal{K}(q \vdash_n Q)))$
(**is** *?LHS* = *?RHS*)
apply (*simp add*: *prob-choice-inf-simp*)
apply (*rule-tac eq-split*)
defer
apply (*simp add*: *prob-lift ndesign-choice*)
apply (*simp add*: *upred-defs*)
apply (*rel-auto*)
apply (*simp add*: *pmf-utp-disj-eq-1*)
proof −
fix ok_v :: *bool* **and** *more* :: *'a* **and** $ok_v{}'$ :: *bool* **and** $prob_v$ :: *'a pmf*
assume $(\sum_a x \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = 1$
then have *infsetsum* $(pmf\ prob_v)\ \{a.\ \exists\ aa.\ [\![Q]\!]_e\ (more,\ a) \wedge aa = a \vee [\![P]\!]_e\ (more,\ a) \wedge aa = a\} =$
1
by (*simp add*: *pmf-utp-disj-eq-1*)
then show $(\sum_a a \mid \exists\ aa.\ [\![P]\!]_e\ (more,\ a) \wedge aa = a \vee [\![Q]\!]_e\ (more,\ a) \wedge aa = a.\ pmf\ prob_v\ a) = 1$
by (*simp add*: *pmf-utp-disj-comm*)
next
fix ok_v::*bool* **and** *more*::*'a* **and** $ok_v{}'$::*bool* **and** *r*::*real* **and** $ok_v{}''$::*bool* **and** $ok_v{}'''$::*bool*
and $prob_v{}'$::*'a pmf* **and** $ok_v{}''''$::*bool* **and** $prob_v{}''$::*'a pmf* **and** $ok_v{}'''''$::*bool*
assume *a1*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (1{::}real)$
assume *a2*: $(\sum_a x{::}'a \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x) = (1{::}real)$
assume *a3*: $(0{::}real) < r$
assume *a4*: $r < (1{::}real)$
show $(\sum_a x{::}'a \mid \exists\ v{::}'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ (prob_v{}' +_r prob_v{}'')$
$x) =$
$(1{::}real)$
using *a3 a4* **apply** (*simp add*: *pmf-wplus*)
proof −
have *f1*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (1{::}real)$
using *a1* **by** (*metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum order-class.order.antisym*
pmf-disj-leq)
have $(\sum_a x{::}'a \mid [\![Q]\!]_e\ (more,\ x) \vee [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x) = (1{::}real)$
using *a2* **by** (*metis measure-pmf.prob-le-1 measure-pmf-conv-infsetsum order-class.order.antisym*
pmf-disj-leq)
then have *f2*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x) = (1{::}real)$
by (*metis* (*no-types*, *lifting*) *Collect-cong*)
have $(\sum_a x{::}'a \mid \exists\ v{::}'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.$
$pmf\ prob_v{}'\ x \cdot r + pmf\ prob_v{}''\ x \cdot ((1{::}real) - r))$
$= (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x \cdot r + pmf\ prob_v{}''\ x \cdot ((1{::}real) -$
$r))$
by *metis*
also have ... $= (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x \cdot r)$
$+ (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x \cdot ((1{::}real) - r))$
by (*simp add*: *abs-summable-on-cmult-left infsetsum-add pmf-abs-summable*)
also have ... $= (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) \cdot r$
$+ (\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v{}''\ x) \cdot ((1{::}real) - r)$
by (*simp add*: *infsetsum-cmult-left pmf-abs-summable*)
also have *f3*: ... $= (1{::}real)$

using *f1 f2 a3 a4* **by** *simp*
show $(\sum_a x::'a \mid \exists v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.$
$pmf\ prob_v{}'\ x \cdot r + pmf\ prob_v{}''\ x \cdot ((1::real) - r)) = (1::real)$
using *f3* **by** (*simp add*: *calculation*)
qed
next
let $?LHS = \ \boldsymbol{U}((p \wedge q) \vdash_n (\ (\exists\ a \in \{0<..<1\}\ .\ \exists\ b \in \{0<..<1\}\ .$
$(\sum_a\ i{\in}\{s'.((P \vee Q)\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob\'\ i) = 1\ \wedge$
$(\sum_a\ i{\in}\{s'.((P \wedge \neg Q)\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob\'\ i) = a\ \wedge$
$(\sum_a\ i{\in}\{s'.((\neg P \wedge Q)\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob\'\ i) = b)))$
let $?RHS = \ \boldsymbol{U}((p \wedge q) \vdash_n (\ (\exists\ r \in \{0<..<1\}\ .\ \exists\ prob_0\ .\ \exists\ prob_1\ .$
$((\sum_a\ i{\in}\{s'.((P)\ wp\ (\&\mathbf{v} = s'))^<\}.\ (pmf\ prob_0\ i)) = (1::real))\ \wedge$
$((\sum_a\ i{\in}\{s'.((Q)\ wp\ (\&\mathbf{v} = s'))^<\}.\ (pmf\ prob_1\ i)) = (1::real))\ \wedge$
$\$prob\' = prob_0 +_r prob_1$
$)))$
let $?B = \ \boldsymbol{U}((p \wedge q) \vdash_n$
$(((\sum_a\ i{\in}\{s'.((P)\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob\'\ i) = 1)$
$\vee (\sum_a\ i{\in}\{s'.((Q)\ wp\ (\&\mathbf{v} = s'))^<\}.\ \$prob\'\ i) = 1))$
have *f1*: $\mathcal{K}\ ((p \vdash_n P) \sqcap (q \vdash_n Q)) = (?B \sqcap ?LHS)$
apply (*simp add*: *prob-lift ndesign-choice*)
apply (*rel-auto*)
apply (*simp add*: *pmf-utp-disj-imp*)+
apply (*simp add*: *pmf-utp-disj-imp'*)+
apply (*simp add*: *pmf-utp-disj-eq-1*)
by (*simp add*: *pmf-utp-disj-eq-1'*)

have *f2*: $?RHS \sqsubseteq ?LHS$
apply (*rel-simp*)
proof −
fix $ok_v::bool$ **and** $more::'a$ **and** $ok_v{}'::bool$ **and** $prob_v::'a\ pmf$
let $?a = (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \wedge \neg\ [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x)$
let $?b = (\sum_a x::'a \mid \neg\ [\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x)$
let $?b1 = (infsetsum\ (pmf\ prob_v)\ (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))$
let $?a1 = infsetsum\ (pmf\ prob_v)\ (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})$
let $?prob_0 = Abs\text{-}pmf\ (\mathcal{F}\ \{s.\ [\![P]\!]_e\ (more,\ s)\}\ \{s.\ [\![Q]\!]_e\ (more,\ s)\}\ prob_v)$
let $?prob_1 = Abs\text{-}pmf\ (\mathcal{F}\ \{s.\ [\![Q]\!]_e\ (more,\ s)\}\ \{s.\ [\![P]\!]_e\ (more,\ s)\}\ prob_v)$
assume *a1*: $(\sum_a x::'a \mid \exists v::'a.\ [\![P]\!]_e\ (more,\ x) \wedge v = x \vee [\![Q]\!]_e\ (more,\ x) \wedge v = x.\ pmf\ prob_v\ x)$
$= (1::real)$
assume *a2*: $(0::real) < ?a$
assume *a3*: $?a < (1::real)$
assume *a4*: $(0::real) < ?b$
assume *a5*: $?b < (1::real)$

from *a1* **have** *a1'*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x) \vee [\![Q]\!]_e\ (more,\ x).\ pmf\ prob_v\ x) = (1::real)$
by (*smt Collect-cong*)
from *a1'* **have** *a1''*:
$infsetsum\ (pmf\ prob_v)\ (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} \cup \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}) = (1::real)$
by (*simp add*: *Collect-disj-eq*)
have *b-eq*: $?b1 = ?b$
by (*smt Collect-cong mem-Collect-eq set-diff-eq*)
have *a-eq*: $?a1 = ?a$
by (*smt Collect-cong mem-Collect-eq set-diff-eq*)
from *a2* **have** *a2'*:
$(0::real) < infsetsum\ (pmf\ prob_v)\ (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})$
by (*smt Collect-cong mem-Collect-eq set-diff-eq*)

```
from a4 have a4':
  (0::real) < infsetsum (pmf prob_v) ({s::'a. ⟦Q⟧_e (more, s)} − {s::'a. ⟦P⟧_e (more, s)})
  by (smt Collect-cong mem-Collect-eq set-diff-eq)
have f21: ?a/(?a+?b) ∈ {0::real<..<1::real}
  using a2 a3 a4 a5 by auto
have f211: ?b/(?a+?b) ∈ {0::real<..<1::real}
  using a2 a3 a4 a5 by auto
have f21': 1 − (?a/(?a+?b)) = ((?a+?b)/(?a+?b)) − (?a/(?a+?b))
  using a2 a4 by auto
then have f21'': ... = ?b/(?a+?b)
  by (smt add-divide-distrib)
have f222: ((?b1 + ?a1) / ?a1)∗(?a/(?a+?b)) = ((?b + ?a)/?a)∗(?a/(?a+?b))
  using a-eq b-eq by simp
then have f222': ... = 1
by (smt f21' f211 greaterThanLessThan-iff nonzero-mult-divide-mult-cancel-right2 times-divide-times-eq)
have f223: ((?b1 + ?a1) / ?b1)∗(?b/(?a+?b)) = ((?b + ?a)/?b)∗(?b/(?a+?b))
  using a-eq b-eq by simp
then have f223': ... = 1
  by (smt a4 f21' nonzero-mult-divide-mult-cancel-right2 times-divide-times-eq)

have f22: (∑_a x::'a | x ∈ {x::'a. ⟦P⟧_e (more, x)} .
  (pmf (Abs-pmf (ℱ {s::'a. ⟦P⟧_e (more, s)} {s::'a. ⟦Q⟧_e (more, s)} prob_v))) x) = (1::real)
  apply (rule proj-f-sum-eq-1[of prob_v {s::'a. ⟦P⟧_e (more, s)} {s::'a. ⟦Q⟧_e (more, s)}])
  using a1'' apply blast
  using a2' apply blast
  using a4' by blast

 then have f23: infsetsum (pmf (Abs-pmf (ℱ {s::'a. ⟦P⟧_e (more, s)} {s::'a. ⟦Q⟧_e (more, s)}
prob_v)))
     {x::'a. ⟦P⟧_e (more, x)} = (1::real)
  by simp
have f24: ∀ i::'a. pmf prob_v i = pmf (?prob_0 +_?a/(?a+?b) ?prob_1) i
  apply (auto)
  proof −
   fix i::'a
   have P-notQ: {s::'a. ⟦P⟧_e (more, s)} − {s::'a. ⟦Q⟧_e (more, s)} = {s::'a. ⟦P⟧_e (more, s) ∧ ¬
⟦Q⟧_e (more, s)}
     by blast
   have Q-notP: {s::'a. ⟦Q⟧_e (more, s)} − {s::'a. ⟦P⟧_e (more, s)} = {s::'a. ⟦Q⟧_e (more, s) ∧ ¬
⟦P⟧_e (more, s)}
     by blast
    have P-and-Q: {s::'a. ⟦P⟧_e (more, s)} ∩ {s::'a. ⟦Q⟧_e (more, s)} = {s::'a. ⟦P⟧_e (more, s) ∧
⟦Q⟧_e (more, s)}
     by blast
  have f240: emeasure (measure-pmf prob_v) ({i} ∩ ({s::'a. ⟦P⟧_e (more, s)} ∩ {s::'a. ⟦Q⟧_e (more,
s)})) ∗ (?a/(?a+?b)) +
      emeasure (measure-pmf prob_v) ({i} ∩ ({s::'a. ⟦P⟧_e (more, s)} ∩ {s::'a. ⟦Q⟧_e (more, s)})) ∗
(?b/(?a+?b))
     = emeasure (measure-pmf prob_v) ({i} ∩ ({s::'a. ⟦P⟧_e (more, s)} ∩ {s::'a. ⟦Q⟧_e (more, s)}))∗
      ((?a/(?a+?b)) + (?b/(?a+?b)))
     by (smt distrib-left ennreal-plus f21 f211 greaterThanLessThan-iff)
  then have f240': ... = emeasure (measure-pmf prob_v) ({i} ∩ ({s::'a. ⟦P⟧_e (more, s)} ∩ {s::'a.
⟦Q⟧_e (more, s)}))
     by (smt ennreal-1 f21' f21'' mult.right-neutral)
  let ?P-Q = emeasure (measure-pmf prob_v) ({i} ∩ ({s::'a. ⟦P⟧_e (more, s)} − {s::'a. ⟦Q⟧_e (more,
```

$s)\}))$

let $?Q\text{-}P = emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))$

let $?PQ = emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))$

have $f241$: $pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ prob_v))\ i \cdot ?a/(?a+?b) +$
$pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ prob_v))\ i \cdot$
$((1::real) - ?a/(?a+?b))$
$= measure\ (measure\text{-}pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ prob_v)))\ \{i\}$
$\cdot\ ?a/(?a+?b) +$
$measure\ (measure\text{-}pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ prob_v)))\ \{i\} \cdot$
$((1::real) - ?a/(?a+?b))$
by (*simp add*: *pmf.rep-eq*)

also have $f242$: ... $= measure\ ((\mathcal{F}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ prob_v))\ \{i\}$
$\cdot\ ?a/(?a+?b) +$
$measure\ ((\mathcal{F}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ prob_v))\ \{i\} \cdot$
$((1::real) - ?a/(?a+?b))$
by (*simp add*: *Un-commute a1'' a2' a4' proj-f-measure-pmf*)

also have $f243$: ... $= enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?b1 + ?a1)\ /\ ?a1) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}))) \cdot$
$(?a/(?a+?b)) +$
$enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![P]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?a1 + ?b1)\ /\ ?b1) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))) \cdot$
$((1::real) - (?a/(?a+?b)))$
apply (*simp only*: *measure-def*)
by (*simp add*: *proj-f-emeasure*)

also have $f244$: ... $= enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?b1 + ?a1)\ /\ ?a1) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}))) \cdot$
$(?a/(?a+?b)) +$
$enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![P]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?a1 + ?b1)\ /\ ?b1) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))) \cdot$
$((?b/(?a+?b)))$
using *f21' f21''* **by** *simp*

also have $f245$: ... $= enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?b1 + ?a1)\ /\ ?a1) *(?a/(?a+?b)) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![P]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\})) \cdot$
$(?a/(?a+?b))) +$
$enn2real$
$(emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} - \{s::'a.\ [\![P]\!]_e\ (more,\ s)\})) \cdot$
$ennreal\ ((?a1 + ?b1)\ /\ ?b1) +$
$emeasure\ (measure\text{-}pmf\ prob_v)\ (\{i\} \cap (\{s::'a.\ [\![Q]\!]_e\ (more,\ s)\} \cap \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}))) \cdot$
$((?b/(?a+?b)))$
by (*smt distrib-right' enn2real-ennreal enn2real-mult f21 greaterThanLessThan-iff*)

also have *f246*: ... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
ennreal ((*?b1* + *?a1*) / *?a1*) ∗(*?a*/(*?a*+*?b*)) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
(*?a*/(*?a*+*?b*))) +
enn2real
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) ·
ennreal ((*?a1* + *?b1*) / *?b1*) ∗(*?b*/(*?a*+*?b*)) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) ·
(*?b*/(*?a*+*?b*)))
by (*smt distrib-right' enn2real-ennreal enn2real-mult f211 greaterThanLessThan-iff*)
also have *f247*: ... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
1 +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
(*?a*/(*?a*+*?b*))) +
enn2real
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) ·
1 +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) ·
(*?b*/(*?a*+*?b*)))
using *f222 f222' f223 f223'* **by** (*smt ennreal-1 ennreal-mult'' f21 f211 greaterThanLessThan-iff*
mult.assoc)
also have *f248*: ... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
(*?a*/(*?a*+*?b*)) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) ·
(*?b*/(*?a*+*?b*)))
by (*smt enn2real-plus ennreal-add-eq-top ennreal-mult-eq-top-iff ennreal-neq-top*
measure-pmf.emeasure-subprob-space-less-top mult.right-neutral order-top-class.less-top)
also have *f249*: ... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
(*?a*/(*?a*+*?b*)) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ·
(*?b*/(*?a*+*?b*)))
by (*simp add*: *Int-commute*)
also have *f2410*:... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ∗
(*?a*/(*?a*+*?b*)) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) ∗
(*?b*/(*?a*+*?b*)))

by (*simp add*: *add.assoc add.left-commute*)
also have *f2411*: ... = *enn2real*
(*emeasure* (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)} − {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf prob*$_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*)} ∩ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*)}))
)
using *f240 f240'* **by** (*simp add*: *add.assoc*)

also have *f2412*: ... = *enn2real*
(*emeasure* (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ⟦*Q*⟧$_e$ (*more*, *s*)}))
)
by (*simp add*: *P-notQ P-and-Q Q-notP*)
have *f2413*: *emeasure* (*measure-pmf* $prob_v$) {*i*} = *enn2real*
(*emeasure* (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*Q*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*P*⟧$_e$ (*more*, *s*)})) +
emeasure (*measure-pmf* $prob_v$) ({*i*} ∩ ({*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ⟦*Q*⟧$_e$ (*more*, *s*)}))
)
proof (*cases i* ∈ {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*Q*⟧$_e$ (*more*, *s*)})
case *True*
then show *?thesis*
by (*simp add*: *ennreal-enn2real-if*)
next
case *False*
then have *Ff*: *i* ∉ {*s*::'*a*. ⟦*P*⟧$_e$ (*more*, *s*) ∧ ¬ ⟦*Q*⟧$_e$ (*more*, *s*)}
by *auto*
then show *?thesis*
proof (*cases i* ∈ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*) ∧ ¬⟦*P*⟧$_e$ (*more*, *s*)})
case *True*
then show *?thesis* **by** (*simp add*: *ennreal-enn2real-if*)
next
case *False*
then have *Fff*: *i* ∉ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*) ∧ ¬⟦*P*⟧$_e$ (*more*, *s*)}
by *auto*
then show *?thesis*
proof (*cases i* ∈ {*s*::'*a*. ⟦*Q*⟧$_e$ (*more*, *s*) ∧ ⟦*P*⟧$_e$ (*more*, *s*)})
case *True*
then show *?thesis*
by (*metis* (*no-types*, *lifting*) *Int-insert-left-if0 Int-insert-left-if1*
Sigma-Algebra.measure-def add.left-neutral
bounded-lattice-bot-class.inf-bot-left emeasure-empty
measure-pmf.emeasure-eq-measure mem-Collect-eq)
next
case *False*
then have *Ffff*: *i* ∈ {*s*::'*a*. ¬(⟦*P*⟧$_e$ (*more*, *s*) ∨ ⟦*Q*⟧$_e$ (*more*, *s*))}
using *Ff Fff* **by** *blast*
from *a1* **have** *g1*: ($\sum_a$*x*::'*a* | ⟦*P*⟧$_e$ (*more*, *x*) ∨ ⟦*Q*⟧$_e$ (*more*, *x*). *pmf* $prob_v$ *x*) =
(*1*::*real*)
using *a1'* **by** *blast*
then have *g2*: ($\sum_a$*x*::'*a* | ¬(⟦*P*⟧$_e$ (*more*, *x*) ∨ ⟦*Q*⟧$_e$ (*more*, *x*)). *pmf* $prob_v$ *x*) =
(*0*::*real*)
by (*rule pmf-utp-comp0'*[*of* $prob_v$ λ*x*. (⟦*P*⟧$_e$ (*more*, *x*) ∨ ⟦*Q*⟧$_e$ (*more*, *x*))])
have *g4*: ($\sum_a$*x*::'*a* | (λ*x*. *x* = *i*) *x*. *pmf* $prob_v$ *x*) ≤
($\sum_a$*x*::'*a* | (λ*x*. *x* = *i*) *x* ∨ ¬(⟦*P*⟧$_e$ (*more*, *x*) ∨ ⟦*Q*⟧$_e$ (*more*, *x*)). *pmf* $prob_v$ *x*)
by (*rule pmf-disj-leq*[*of* $prob_v$ (λ*x*. *x* = *i*) -])
then have *g5*: ($\sum_a$*x*::'*a* | (λ*x*. *x* = *i*) *x*. *pmf* $prob_v$ *x*) ≤
($\sum_a$*x*::'*a* | ¬(⟦*P*⟧$_e$ (*more*, *x*) ∨ ⟦*Q*⟧$_e$ (*more*, *x*)). *pmf* $prob_v$ *x*)
using *Ffff* **by** (*smt Collect-cong mem-Collect-eq*)
then have *g6*: ($\sum_a$*x*::'*a* | (λ*x*. *x* = *i*) *x*. *pmf* $prob_v$ *x*) = *0*
using *g2* **by** *simp*
have ($\sum_a$*x*::'*a* | *x* = *i*. *pmf* $prob_v$ *x*) = *pmf* $prob_v$ *i*
by *auto*

then have *g7*: $(pmf\ prob_v)\ i = 0$
using *g6* **by** *linarith*
then show *?thesis* **using** *g7*
by (*simp add*: *emeasure-pmf-single pmf-measure-zero*)
qed
qed
qed
have *f241*: $pmf\ prob_v\ i =$
$pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ prob_v))\ i \cdot ?a/(?a+?b)\ +$
$pmf\ (Abs\text{-}pmf\ (\mathcal{F}\ \{s::'a.\ [\![Q]\!]_e\ (more,\ s)\}\ \{s::'a.\ [\![P]\!]_e\ (more,\ s)\}\ prob_v))\ i \cdot ((1::real) -$
$?a/(?a+?b))$
by (*metis* (*no-types*, *lifting*) *P-and-Q P-notQ Q-notP Sigma-Algebra.measure-def calculation ennreal-add-eq-top ennreal-enn2real f2413 measure-pmf.emeasure-subprob-space-less-top order-top-class.less-top pmf.rep-eq*)
show $pmf\ prob_v\ i = pmf\ (?prob_0 +_{?a/(?a+?b)} ?prob_1)\ i$
using *f21* **apply** (*simp add*: *f21 pmf-wplus*)
using *f241* **by** *blast*
qed
have *f25*: $prob_v = (?prob_0 +_{?a/(?a+?b)} ?prob_1)$
apply (*rule pmf-eqI*)
using *f24* **by** *blast*
show $\exists\, x::real \in \{0::real<..<1::real\}.$
$\exists\, xa::'a\ pmf.$
$(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ xa\ x) = (1::real)\ \wedge$
$(\exists\, xb::'a\ pmf.\ (\sum_a x::'a \mid [\![Q]\!]_e\ (more,\ x).\ pmf\ xb\ x) = (1::real) \wedge prob_v = xa +_x xb)$
apply (*simp add*: *Set.Bex-def*)
apply (*rule-tac* $x = ?a/(?a+?b)$ **in** *exI*)
apply (*rule conjI*)
using *f21* **apply** *simp*
apply (*rule conjI*)
using *f21* **apply** *simp*
apply (*rule-tac* $x = ?prob_0$ **in** *exI*)
apply (*rule-tac conjI*)
using *f23* **apply** *blast*
apply (*rule-tac* $x = ?prob_1$ **in** *exI*)
apply (*rule-tac conjI*)
apply (*metis Collect-mem-eq Un-commute a1'' a2' a4' proj-f-sum-eq-1*)
using *f25* **by** *blast*
qed
then have *f3*: $(?B \sqcap ?RHS) \sqsubseteq (?B \sqcap ?LHS)$
by (*smt sup-bool-def sup-uexpr.rep-eq upred-ref-iff*)

have *f4*: $(?B \sqcap ?RHS)$
$= \mathcal{K}\ (p \vdash_n P) \sqcap \mathcal{K}\ (q \vdash_n Q) \sqcap (\bigsqcap\ r::real \in \{0::real<..<1::real\} \cdot \mathcal{K}\ (p \vdash_n P) \parallel^D{}_{\mathbf{PM}_r} \mathcal{K}\ (q \vdash_n$
$Q))$
apply (*simp add*: *prob-lift ndesign-choice*)
apply (*simp add*: *upred-defs*)
apply (*rel-auto*)
apply *blast*
using *greaterThanLessThan-iff* **by** *blast*

show ‘$\mathcal{K}\ ((p \vdash_n P) \sqcap (q \vdash_n Q)) \Rightarrow$
$\mathcal{K}\ (p \vdash_n P) \sqcap \mathcal{K}\ (q \vdash_n Q) \sqcap (\bigsqcap\ r::real \in \{0::real<..<1::real\} \cdot \mathcal{K}\ (p \vdash_n P) \parallel^D{}_{\mathbf{PM}_r} \mathcal{K}\ (q \vdash_n Q))$‘
using *f1 f3 f4 refBy-order* **by** (*metis* (*mono-tags*, *lifting*))
qed

lemma *pemb-intchoice′*:
assumes P *is* **N** Q *is* **N**
shows $\mathcal{K}(P \sqcap Q)$
$= \mathcal{K}(P) \sqcap \mathcal{K}(Q) \sqcap (\bigsqcap\ r \in \{0{<}..{<}1\} \cdot (\mathcal{K}(P) \oplus_r \mathcal{K}(Q)))$
(**is** *?LHS* = *?RHS*)
proof –
obtain pre_p $post_p$ pre_q $post_q$
where $p{:}P = (pre_p \vdash_n post_p)$ **and**
$q{:}Q = (pre_q \vdash_n post_q)$
using *assms* **by** (*metis ndesign-form*)
have $\mathcal{K}((pre_p \vdash_n post_p) \sqcap (pre_q \vdash_n post_q))$
$= \mathcal{K}(pre_p \vdash_n post_p) \sqcap \mathcal{K}(pre_q \vdash_n post_q) \sqcap (\bigsqcap\ r \in \{0{<}..{<}1\} \cdot (\mathcal{K}(pre_p \vdash_n post_p) \oplus_r \mathcal{K}(pre_q \vdash_n post_q)))$
by (*simp add*: *pemb-intchoice*)
then show *?thesis*
using p q **by** *auto*
qed

lemma *pemb-dem-choice-refinedby-prochoice*:
assumes $r \in \{0..1\}$ P *is* **N** Q *is* **N**
shows $\mathcal{K}(P \sqcap Q) \sqsubseteq (\mathcal{K}(P) \oplus_r \mathcal{K}(Q))$
proof (*cases* $r \in \{0{::}real{<}..{<}1{::}real\}$)
case *True*
show *?thesis*
using *assms* **apply** (*simp add*: *pemb-intchoice′*)
apply (*simp add*: *UINF-as-Sup-collect*)
by (*meson SUP-le-iff True semilattice-sup-class.sup-ge2*)
next
case *False*
then show *?thesis*
by (*metis assms(1) atLeastAtMost-iff greaterThanLessThan-iff less-le pemb-mono prob-choice-one prob-choice-zero semilattice-sup-class.sup-ge1 semilattice-sup-class.sup-ge2*)
qed

C.1.2 Kleisli Lift and Sequential Composition

lemma *kleisli-lift-skip-unit*: $\uparrow (\mathcal{K}(II_D)) = kleisli\text{-}lift2\ true\ (U(\$prob'(\$\mathbf{v}) = 1))$
by (*simp add*: *kleisli-lift-def pemp-skip*)

lemma *kleisli-lift-skip*:
$kleisli\text{-}lift2\ true\ (U(\$prob'(\$\mathbf{v}) = 1)) = \boldsymbol{U}(true \vdash_n (\$prob' = \$prob))$
apply (*simp add*: *kleisli-lift2-def ndesign-def*)
apply (*rel-auto*)
apply (*metis* (*full-types*) *equalityI lit.rep-eq mem-Collect-eq order-top-class.top-greatest subsetI upred-ref-iff upred-set.rep-eq sum-pmf-eq-1*)
apply (*metis* (*full-types*) *lit.rep-eq mem-Collect-eq order-top-class.top.extremum-unique subsetI upred-ref-iff upred-set.rep-eq sum-pmf-eq-1*)
proof –
fix $ok_v{::}bool$ **and** $prob_v{::}'a\ pmf$ **and** $ok_v'{::}bool$ **and** $prob_v'{::}'a\ pmf$ **and** $x{::}'a \Rightarrow 'a\ pmf$
assume $a1$: $\forall xa{::}'a.\ pmf\ prob_v'\ xa = (\sum_a xb{::}'a.\ pmf\ prob_v\ xb \cdot pmf\ (x\ xb)\ xa)$
assume $a2$: $\forall xa{::}'a.$
$(\exists prob_v{::}'a\ pmf.\ \neg\ pmf\ prob_v\ xa = (1{::}real) \wedge (\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \longrightarrow \neg\ (0{::}real) < pmf\ prob_v\ xa$
from $a2$ **have** $f1$: $\forall xa{::}'a.\ (pmf\ (x\ xa)\ xa = 1\ ) \vee \neg\ (0{::}real) < pmf\ prob_v\ xa$
by *blast*

```
  then have f2: ∀xa::'a. (pmf (x xa) xa = 1 ) ∨ (0::real) = pmf prob_v xa
    by auto
  have f3: ∀xa. (pmf prob_v xb · pmf (x xb) xa) = (if xb = xa then pmf prob_v xa else 0)
    apply (rule allI)
    proof −
      fix xa::'a
      show pmf prob_v xb · pmf (x xb) xa = (if xb = xa then pmf prob_v xa else (0::real))
      proof (cases xb = xa)
        case True
        then show ?thesis
          using f2 by auto
      next
        case False
        then have f: ¬xb = xa
          by simp
        then show ?thesis
        proof (cases pmf prob_v xb = 0)
          case True
          then show ?thesis
            by auto
        next
          case False
          then have pmf (x xb) xb = 1
            using f2 by auto
          then have pmf (x xb) xa = 0
            using f apply (simp add: pmf-def)
            by (simp add: measure-pmf-single pmf-not-the-one-is-zero)
          then show ?thesis
            by (simp add: f)
        qed
      qed
    qed
  have f4: ∀xa. (∑_a xb::'a. pmf prob_v xb · pmf (x xb) xa) =
                 (∑_a xb::'a. (if xb = xa then pmf prob_v xa else 0))
    using f3
    by (smt f2 infsetsum-cong mult-cancel-left2 mult-not-zero pmf-not-the-one-is-zero)
  have f5: ∀xa. (∑_a xb::'a. (if xb = xa then pmf prob_v xa else 0)) =  pmf prob_v xa
    by (simp add: pmf-sum-single)
  have f6: ∀xa. pmf prob_v' xa = pmf prob_v xa
    using f4 f5 a1 by simp
  show prob_v' = prob_v
    using f6 by (simp add: pmf-eqI)
next
  fix ok_v::bool and prob_v::'a pmf and ok_v'::bool
  show ∃x::'a ⇒ 'a pmf.
          (∀xa::'a. pmf prob_v xa = (∑_a xb::'a. pmf prob_v xb · pmf (x xb) xa)) ∧
          (∀xa::'a.
              (∃prob_v::'a pmf. ¬ pmf prob_v xa = (1::real) ∧ (∀xb::'a. pmf prob_v xb = pmf (x xa) xb))
⟶
              ¬ (0::real) < pmf prob_v xa)
    apply (rule-tac x=λs::'a. pmf-of-list([(s, 1.0)]) in exI)
    apply (rule conjI, auto)
    apply (simp add: pmf-sum-single')
    by (smt filter.simps(1) filter.simps(2) list.map(1) list.map(2) list.set(1) list.set(2)
        pmf-of-list-wf-def pmf-pmf-of-list prod.sel(1) prod.sel(2) singletonD sum-list.Nil
```

sum-list-simps(*2*))
qed

lemma *kleisli-lift-skip*′:
$\uparrow (\mathcal{K}(II_D)) \;=\; \boldsymbol{U}(true \vdash_n (\$prob' = \$prob))$
by (*simp add*: *kleisli-lift-skip kleisli-lift-skip-unit*)

lemma *kleisli-lift-skip-left-unit*:
assumes P *is* **N**
shows $(\mathcal{K}(II_D))\,;;\ \uparrow P = P$
proof −
obtain pre_p $post_p$ **where** $p{:}P = (pre_p \vdash_n post_p)$
using *assms* **by** (*metis ndesign-form*)
have *f1*: $(\mathcal{K}(II_D))\,;;\ \uparrow (pre_p \vdash_n post_p) = (pre_p \vdash_n post_p)$
apply (*simp add*: *pemp-skip kleisli-lift-def kleisli-lift2-def upred-set-def*)
apply (*rel-auto*)
apply (*metis* (*full-types*) *Compl-iff infsetsum-all-0 mem-Collect-eq pmf-comp-set pmf-not-the-one-is-zero upred-set.rep-eq*)
apply (*metis Compl-iff infsetsum-all-0 mem-Collect-eq pmf-comp-set pmf-not-the-one-is-zero upred-set.rep-eq*)
proof −
fix ok_v::*bool* **and** *more*::′*a* **and** $prob_v$::′*a pmf* **and** $ok_v{}'$::*bool* **and** $ok_v{}''$::*bool* **and** $prob_v{}'$::′*a pmf* **and** x::′*a* ⇒ ′*a pmf*
assume *a1*: $[\![pre_p]\!]_e$ *more*
assume *a2*: $pmf\ prob_v{}'\ more = (1{::}real)$
assume *a3*: $\forall xa{::}'a.\ pmf\ prob_v\ xa = (\sum_a xb{::}'a.\ pmf\ prob_v{}'\ xb \cdot pmf\ (x\ xb)\ xa)$
assume *a4*: $\forall xa{::}'a.$
$(\exists prob_v{::}'a\ pmf.\ ([\![pre_p]\!]_e\ xa \longrightarrow \neg\ [\![post_p]\!]_e\ (xa, (\!|prob_v = prob_v|\!))) \wedge (\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \longrightarrow$
$\neg\ (0{::}real) < pmf\ prob_v{}'\ xa$
from *a4* **have** *f1*:
$(\exists prob_v{::}'a\ pmf.\ \neg\ [\![post_p]\!]_e\ (more, (\!|prob_v = prob_v|\!)) \wedge (\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ more)\ xb)) \longrightarrow$
$\neg\ (0{::}real) < pmf\ prob_v{}'\ more$
using *a1* **by** *blast*
then have *f2*: $\neg(\exists prob_v{::}'a\ pmf.\ \neg\ [\![post_p]\!]_e\ (more, (\!|prob_v = prob_v|\!)) \wedge (\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ more)\ xb))$
using *a2* **by** *simp*
then have *f3*: $(\forall prob_v{::}'a\ pmf.\ [\![post_p]\!]_e\ (more, (\!|prob_v = prob_v|\!)) \vee \neg(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ more)\ xb))$
by *blast*
then have *f4*: $[\![post_p]\!]_e\ (more, (\!|prob_v = prob_v|\!)) \vee \neg(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ more)\ xb)$
by *blast*
from *a3 a2* **have** *f5*: $(\forall xa{::}'a.\ (\sum_a xb{::}'a.\ pmf\ prob_v{}'\ xb \cdot pmf\ (x\ xb)\ xa) =$
$(\sum_a xb{::}'a.\ if\ xb = more\ then\ pmf\ (x\ more)\ xa\ else\ 0))$
by (*smt infsetsum-cong mult-cancel-left mult-cancel-right1 pmf-not-the-one-is-zero*)
have *f6*: $(\forall xa{::}'a.\ (\sum_a xb{::}'a.\ if\ xb = more\ then\ pmf\ (x\ more)\ xa\ else\ 0) = pmf\ (x\ more)\ xa)$
apply (*rule allI*)
proof −
fix xa::′*a*
show $(\sum_a xb{::}'a.\ if\ xb = more\ then\ pmf\ (x\ more)\ xa\ else\ (0{::}real)) = pmf\ (x\ more)\ xa$
by (*simp add*: *infsetsum-single*′[*of more* $\lambda y.\ pmf\ (x\ y)\ xa\ more$])
qed
have *f7*: $(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ more)\ xb)$
using *f6 f5 a3* **by** *simp*

show $[\![post_p]\!]_e$ (*more*, ($\!|prob_v = prob_v|\!$))
using *f7 f4* **by** *blast*
next
fix ok_v::*bool* **and** *more*::'*a* **and** $prob_v$::'*a pmf* **and** ok_v'::*bool*
assume *a1*: $\forall (ok_v''::bool)\ prob_v'::'a\ pmf.$
$ok_v \wedge (ok_v'' \longrightarrow \neg\ pmf\ prob_v'\ more = (1::real)) \vee$
$ok_v'' \wedge$
$infsetsum\ (pmf\ prob_v')\ (Collect\ [\![pre_p]\!]_e) = (1::real) \wedge$
$(ok_v' \longrightarrow$
$(\forall x::'a \Rightarrow 'a\ pmf.$
$(\exists xa::'a.\ \neg\ pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ prob_v'\ xb \cdot pmf\ (x\ xb)\ xa)) \vee$
$(\exists xa::'a.$
$(\exists prob_v::'a\ pmf.\ ([\![pre_p]\!]_e\ xa \longrightarrow \neg\ [\![post_p]\!]_e\ (xa,\ (\!|prob_v = prob_v|\!))) \wedge (\forall xb::'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \wedge$
$(0::real) < pmf\ prob_v'\ xa)))$
let $?prob_v' = (pmf\text{-}of\text{-}list\ [(more,1.0)])$
have *f1*: $\neg pmf\ ?prob_v'\ more = (1::real) \vee infsetsum\ (pmf\ ?prob_v')\ (Collect\ [\![pre_p]\!]_e) = (1::real)$
using *a1* **by** *blast*
have *f2*: $pmf\ ?prob_v'\ more = (1::real)$
by (*smt divide-self-if filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.map*(*1*)
list.map(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*)
prod.sel(*2*) *singletonD sum-list-simps*(*1*) *sum-list-simps*(*2*))
have *f3*: $infsetsum\ (pmf\ ?prob_v')\ (Collect\ [\![pre_p]\!]_e) = (1::real)$
using *f1 f2* **by** *blast*
then have *f4*: $infsetsum\ (\lambda x.\ if\ x = more\ then\ 1\ else\ 0)\ (Collect\ [\![pre_p]\!]_e) = (1::real)$
by (*smt div-self filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.map*(*1*) *list.map*(*2*)
list.set(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*)
singletonD sum-list-simps(*1*) *sum-list-simps*(*2*))
then have *f8*: $more \in (Collect\ [\![pre_p]\!]_e)$
by (*smt infsetsum-all-0*)
show $[\![pre_p]\!]_e$ *more*
using *f8* **by** *blast*
next
fix ok_v::*bool* **and** *more*::'*a* **and** $prob_v$::'*a pmf* **and** ok_v'::*bool*
assume *a1*: $[\![post_p]\!]_e\ (more,\ (\!|prob_v = prob_v|\!))$
let $?prob_v = (pmf\text{-}of\text{-}list\ [(more,1.0)])$
have *f0*: $\forall xa::'a.\ pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ ?prob_v\ xb \cdot pmf\ prob_v\ xa)$
apply (*auto*)
proof −
fix *xa*::'*a*
have *f1*: $(\sum_a xb::'a.\ pmf\ (pmf\text{-}of\text{-}list\ [(more,\ 1::real)])\ xb \cdot pmf\ prob_v\ xa) =$
$(\sum_a xb::'a.\ pmf\ prob_v\ xa \cdot pmf\ (pmf\text{-}of\text{-}list\ [(more,\ 1::real)])\ xb)$
by (*meson mult.commute*)
have *f2*: $(\sum_a xb::'a.\ pmf\ prob_v\ xa \cdot pmf\ (pmf\text{-}of\text{-}list\ [(more,\ 1::real)])\ xb) = pmf\ prob_v\ xa$
by (*simp add*: *pmf-sum-single''*)
show $pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ (pmf\text{-}of\text{-}list\ [(more,\ 1::real)])\ xb \cdot pmf\ prob_v\ xa)$
apply (*rule sym*)
using *pmf-sum-single' f1* **by** (*simp add*: *f2*)
qed
show $\exists (ok_v'::bool)\ prob_v'::'a\ pmf.$
$(ok_v \longrightarrow ok_v' \wedge pmf\ prob_v'\ more = (1::real)) \wedge$
$(ok_v' \wedge infsetsum\ (pmf\ prob_v')\ (Collect\ [\![pre_p]\!]_e) = (1::real) \longrightarrow$
$(\exists x::'a \Rightarrow 'a\ pmf.$
$(\forall xa::'a.\ pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ prob_v'\ xb \cdot pmf\ (x\ xb)\ xa)) \wedge$
$(\forall xa::'a.$

($\exists\, prob_v$::$'a$ *pmf*.
([[pre_p]]$_e$ *xa* $\longrightarrow \neg$ [[$post_p$]]$_e$ (*xa*, (|$prob_v$ = $prob_v$|))) $\wedge$
($\forall\, xb$::$'a$. *pmf* $prob_v$ *xb* = *pmf* (*x xa*) *xb*)) $\longrightarrow$
$\neg$ (*0*::*real*) < *pmf* $prob_v{}'$ *xa*)))

apply (*rule-tac x* = *True* **in** *exI*)
apply (*rule-tac x* = (*pmf-of-list* [(*more*,*1.0*)]) **in** *exI*)
apply (*rule conjI*)
apply (*smt div-self filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.map*(*1*) *list.map*(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list-simps*(*1*) *sum-list-simps*(*2*))
apply (*auto*)
proof −
assume *a11*: *infsetsum* (*pmf* (*pmf-of-list* [(*more*, *1*::*real*)])) (*Collect* [[pre_p]]$_e$) = (*1*::*real*)
show $\exists\, x$::$'a \Rightarrow 'a$ *pmf*.
($\forall\, xa$::$'a$. *pmf* $prob_v$ *xa* = ($\sum_a xb$::$'a$. *pmf* (*pmf-of-list* [(*more*, *1*::*real*)]) *xb* $\cdot$ *pmf* (*x xb*) *xa*))
$\wedge$
($\forall\, xa$::$'a$.
($\exists\, prob_v$::$'a$ *pmf*.
([[pre_p]]$_e$ *xa* $\longrightarrow \neg$ [[$post_p$]]$_e$ (*xa*, (|$prob_v$ = $prob_v$|))) $\wedge$
($\forall\, xb$::$'a$. *pmf* $prob_v$ *xb* = *pmf* (*x xa*) *xb*)) $\longrightarrow$
$\neg$ (*0*::*real*) < *pmf* (*pmf-of-list* [(*more*, *1*::*real*)]) *xa*)
apply (*rule-tac x* = λx. $prob_v$ **in** *exI*)
apply (*rule conjI*)
using *f0* **apply** *auto*[*1*]
apply *auto*
proof −
fix *xa*::$'a$ **and** $prob_v{}'$::$'a$ *pmf*
assume *a111*: $\forall\, xb$::$'a$. *pmf* $prob_v{}'$ *xb* = *pmf* $prob_v$ *xb*
assume *a112*: (*0*::*real*) < *pmf* (*pmf-of-list* [(*more*, *1*::*real*)]) *xa*
assume *a113*: $\neg$ [[pre_p]]$_e$ *xa*
from *a112* **have** *f111*: *xa* = *more*
by (*smt filter.simps*(*1*) *filter.simps*(*2*) *list.map*(*1*) *list.map*(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list.Nil sum-list-simps*(*2*))
from *a11* **have** *f112*: [[pre_p]]$_e$ *more*
by (*smt a112 a113 filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-all-0 list.set*(*1*) *list.set*(*2*) *list.simps*(*8*) *list.simps*(*9*) *mem-Collect-eq pmf-of-list-wf-def pmf-pmf-of-list singletonD snd-conv sum-list.Cons sum-list.Nil*)
show *False*
using *a113 f111 f112* **by** *blast*
next
fix *xa*::$'a$ **and** $prob_v{}'$::$'a$ *pmf*
assume *a111*: $\forall\, xb$::$'a$. *pmf* $prob_v{}'$ *xb* = *pmf* $prob_v$ *xb*
assume *a112*: (*0*::*real*) < *pmf* (*pmf-of-list* [(*more*, *1*::*real*)]) *xa*
assume *a113*: $\neg$ [[$post_p$]]$_e$ (*xa*, (|$prob_v$ = $prob_v{}'$|))
from *a112* **have** *f111*: *xa* = *more*
by (*smt filter.simps*(*1*) *filter.simps*(*2*) *list.map*(*1*) *list.map*(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list.Nil sum-list-simps*(*2*))
from *a111* **have** *f112*: $prob_v{}' = prob_v$
by (*simp add*: *pmf-eqI*)
then show *False*
using *a113 a1 f111* **by** *blast*
qed
qed

qed
show *?thesis*
using *f1* **by** (*simp add*: *p*)
qed

lemma *kleisli-lift-skip-right-unit*:
assumes *P is* **N**
shows $P\ ;;_p\ (II_p) = P$
proof −
obtain pre_p $post_p$ **where** $p{:}P = (pre_p \vdash_n post_p)$
using *assms* **by** (*metis ndesign-form*)
have *f1*: $(pre_p \vdash_n post_p)\ ;;_p\ (II_p) = (pre_p \vdash_n post_p)$
apply (*simp add*: *kleisli-lift-skip′*)
by (*rel-auto*)
show *?thesis*
using *p f1* **by** *simp*
qed

term *x abs-summable-on A*
term *integrable*
term *has-bochner-integral M f x*
term $integral^L$ *M f* = (*if* ∃ *x*. *has-bochner-integral M f x then THE x*. *has-bochner-integral M f x else 0*)
term *infsetsum f A* = *lebesgue-integral* (*count-space A*) *f*
term *measure-of*

term *infsetsum* (λ*x*.
(*infsetsum*
(λ*xa*. *if pmf* $prob_v{}'$ *xa* > *0 then pmf* $prob_v{}'$ *xa* · *pmf* (*xx xa*) *x else 0*)
UNIV))
({*t*. ∃ *y*::′*b*. $[\![P]\!]_e$ (*more*, *y*) ∧ $[\![Q]\!]_e$ (*y*, *t*)})
term *simple-bochner-integrable x a*
term *sum*
thm *sum.If-cases*
thm *sum.Sigma*
thm *sum.swap*
term *ennreal*
term *ereal*

lemma *sum-ennreal-extract*:
assumes ∀ *x*. *P x* ≥ *0*
shows *sum* (λ*x*. *ennreal* (*P x*)) *A* = (*ennreal* (*sum* (λ*x*. *P x*) *A*))
using *assms* **by** *auto*

lemma *sum-uniform-value*:
assumes *A* ≠ {} *finite A*
shows *sum* (λ*x*. *C*/(*card A*)) *A* = *C*
using *assms* **by** *simp*

lemma *sum-uniform-value′*:
assumes ∀ *y*. *finite* (*A y*) ∀ *y* ∈ *B*. (*A y* ≠ {})
shows *sum* (λ*y*. *sum* (λ*x*. *C y*/(*card* (*A y*))) (*A y*)) *B* = (*sum* (λ*y*. *C y*) *B*)
using *assms* **by** (*simp add*: *sum-uniform-value*)

lemma *sum-uniform-value-zero*:
assumes $A = \{\}$ *finite A*
shows *sum* $(\lambda x.\ C/(card\ A))\ A = 0$
using *assms* **by** *simp*

lemma *pemb-seq-comp*:
fixes *D1*::('*a*, '*a*) *rel-des* **and** *D2*::('*a*, '*a*) *rel-des*
— He Jifeng's original paper doesn't explicitly mention the finiteness condition, but implicitly in the construction of f(u,v) where a *card* function is used. Without this condition, we are not able to prove this lemmas now because of subgoals 2 and 5 below which needs this condition to transform infsetsum to sum. More importantly, swap summation operators like *sum x*. (*sum y*. (*f x y*)) to *sum y*. (*sum x*. (*f x y*)) in order to expand some expressions.
assumes *finite* (*UNIV*::'*a set*)
assumes *D1 is* **N** *D2 is* **N**
shows $\mathcal{K}(D1\ ;;\ D2) = \mathcal{K}(D1)\ ;;\ (\uparrow (\mathcal{K}(D2)))$
proof −
obtain *p P q Q*
where $p{:}D1 = (p \vdash_n P)$ **and**
$q{:}D2 = (q \vdash_n Q)$
using *assms* **by** (*metis ndesign-form*)
have *seq-comp-ndesign*: $\mathcal{K}((p \vdash_n P)\ ;;\ (q \vdash_n Q)) = \mathcal{K}((p \vdash_n P))\ ;;\ (\uparrow (\mathcal{K}((q \vdash_n Q))))$
apply (*simp add*: *ndesign-composition-wp prob-lift*)
apply (*simp add*: *kleisli-lift2-def kleisli-lift-def upred-set-def*)
apply (*rel-auto*)
— Five subgoals to prove: 1, 3, 4 regarding preconditions and 2,5 for postconditions. Subgoal 2 and 5 are nontrivial.
proof −
fix ok_v::*bool* **and** *more*::'*a* **and** ok_v'::*bool* **and** $prob_v$::'*a pmf* **and** *y*::'*a*
assume *a1*: $\forall (ok_v''::bool)\ prob_v'::'a\ pmf.$
$ok_v \wedge [\![p]\!]_e\ more \wedge (ok_v'' \longrightarrow \neg (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v'\ x) = (1::real)) \vee$
$ok_v'' \wedge$
$infsetsum\ (pmf\ prob_v')\ (Collect\ [\![q]\!]_e) = (1::real) \wedge$
$(ok_v' \longrightarrow$
$(\forall x::'a \Rightarrow 'a\ pmf.$
$(\exists xa::'a.\ \neg\ pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ prob_v'\ xb \cdot pmf\ (x\ xb)\ xa)) \vee$
$(\exists xa::'a.$
$(\exists prob_v::'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x::'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1::real)) \wedge$
$(\forall xb::'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \wedge$
$(0::real) < pmf\ prob_v'\ xa)))$
assume *a2*: $[\![P]\!]_e\ (more,\ y)$
— Since a1 holds for every $prob_v'$, we choose a simple distribution $?prob_v'$, a point distribution.
let $?ok_v'' = True$
let $?prob_v' = (pmf\text{-}of\text{-}list\ [(y,1.0)])$
have *f1*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ (?prob_v')\ x) =$
$(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ if\ x = y\ then\ 1\ else\ 0)$
by (*smt divide-self-if filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.map*(*1*) *list.map*(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list-simps*(*1*) *sum-list-simps*(*2*))
also have *f2*: $... = (\sum_a x \in \{y\} \cup \{t.\ [\![P]\!]_e\ (more,\ t) \wedge t \neq y\}.\ if\ x = y\ then\ 1\ else\ 0)$
using *a2* **by** (*smt Collect-cong Un-insert-left bounded-semilattice-sup-bot-class.sup-bot.left-neutral insert-compr mem-Collect-eq*)

also have *f3*: ... $= (\sum_a x \in \{y\}.\ \textit{if } x = y \textit{ then 1 else 0}) +$
$(\sum_a x \in \{t.\ [\![P]\!]_e\ (\textit{more},\ t) \wedge t \neq y\}.\ \textit{if } x = y \textit{ then 1 else 0})$
unfolding *infsetsum-altdef abs-summable-on-altdef*
apply (*subst set-integral-Un*, *auto*)
apply (*meson abs-summable-on-altdef abs-summable-on-empty abs-summable-on-insert-iff*)
using *abs-summable-on-altdef* **by** (*smt abs-summable-on-0 abs-summable-on-cong mem-Collect-eq*)
also have *f4*: ... $= (1{::}\textit{real})$
by (*smt finite.emptyI finite.insertI infsetsum-all-0 infsetsum-finite insert-absorb insert-not-empty mem-Collect-eq sum.insert*)
have *f5*: $(ok_v \wedge [\![p]\!]_e\ \textit{more} \wedge$
$(\textit{True} \longrightarrow \neg\ (\sum_a x{::}'a \mid [\![P]\!]_e\ (\textit{more},\ x).\ \textit{pmf}\ (?\textit{prob}_v{}')\ x) = (1{::}\textit{real}))) = \textit{False}$
using *calculation f4* **by** *auto*
from *f5* **have** *f6*: $\textit{infsetsum}\ (\textit{pmf}\ ?\textit{prob}_v{}')\ (\textit{Collect}\ [\![q]\!]_e) = (1{::}\textit{real})$
using *a1* **by** *blast*
then have *f7*: $\textit{infsetsum}\ (\lambda x.\ \textit{if } x = y \textit{ then 1 else 0})\ (\textit{Collect}\ [\![q]\!]_e) = (1{::}\textit{real})$
by (*smt div-self filter.simps*(*1*) *filter.simps*(*2*) *infsetsum-cong list.map*(*1*) *list.map*(*2*) *list.set*(*1*) *list.set*(*2*) *pmf-of-list-wf-def pmf-pmf-of-list prod.sel*(*1*) *prod.sel*(*2*) *singletonD sum-list-simps*(*1*) *sum-list-simps*(*2*))
then have *f8*: $y \in (\textit{Collect}\ [\![q]\!]_e)$
by (*smt infsetsum-all-0*)
show $[\![q]\!]_e\ y$
using *f8* **by** *auto*
next

— Subgoal 2: postcondition implied from LHS to RHS: $\textit{prob}'(P;\ Q){=}1$ implies there exists an intermediate distribution ϱ and a function (Q in He's paper) from intermediate states to the distribution on final states.

fix $ok_v{::}\textit{bool}$ **and** $\textit{more}{::}'a$ **and** $ok_v{}'{::}\textit{bool}$ **and** $\textit{prob}_v{::}'a\ \textit{pmf}$
assume *a1*: $(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (\textit{more},\ y) \wedge [\![Q]\!]_e\ (y,\ x).\ \textit{pmf prob}_v\ x) = (1{::}\textit{real})$

— $?f(s',\ s_0)$, $?p$ and $?Q$ are corresponding functions to construct f, p and Q in He's paper.

let $?f = \lambda\ s'\ s_0.\ (\textit{if } [\![P]\!]_e\ (\textit{more},\ s_0) \wedge [\![Q]\!]_e\ (s_0,\ s') \textit{ then}$
$(\textit{pmf prob}_v\ s'/(\textit{card}\ \{t.\ [\![P]\!]_e\ (\textit{more},\ t) \wedge [\![Q]\!]_e\ (t,\ s')\}))$
$\textit{else } 0)$
let $?p = \lambda s_0\ .\ (\sum_a\ s'{::}'a\ .\ ?f\ s'\ s_0)$

— The else branch is not defined in He's paper. It couldn't be zero here as $?Q$ is used to give a witness $(\lambda s.\ \textit{embed-pmf}\ \ (?Q\ s))$ for $\exists\, x{::}'a \Rightarrow 'a\ \textit{pmf}$. The type of x is from states to a pmf distribution. If the else branch gives zero, it couldn't be able to construct a pmf distribution (sum is equal to 1). Therefore, we choose a uniform distribution upon whole state space if $?p\ s_0$ is equal to 0.

let $?Q = \lambda s_0\ s'.\ (\textit{if } ?p\ s_0 > 0 \textit{ then } (?f\ s'\ s_0\ /\ ?p\ s_0) \textit{ else } (1/\textit{card}\ (\textit{UNIV}{::}'a\ \textit{set})))$

— We construct a witness for $\textit{prob}_v{}'$ by embeding $?p$ function using *embed-pmf*. After that, we also need to expand $\textit{pmf}\ (\textit{embed-pmf}\ ?p)\ x$ to $?p\ x$ by *pmf-embed-pmf* which also needs to prove *nonneg* and *prob* assumptions. *p-prob* is for the *prob* condition.

have *p-prob*: $(\sum a{::}'a{\in}\textit{UNIV}.\ \ \textit{ennreal}\ (\sum x{::}'a{\in}\textit{UNIV}.$
$\textit{if } [\![P]\!]_e\ (\textit{more},\ a) \wedge [\![Q]\!]_e\ (a,\ x) \textit{ then pmf prob}_v\ x\ /\ \textit{real}\ (\textit{card}\ \{t{::}'a.\ [\![P]\!]_e\ (\textit{more},\ t) \wedge [\![Q]\!]_e\ (t,\ x)\})$
$\textit{else}\ (0{::}\textit{real}))) = (1{::}\textit{ennreal})$
proof −
from *a1* **have** *f11*: $(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (\textit{more},\ y) \wedge [\![Q]\!]_e\ (y,\ x).\ \textit{pmf prob}_v\ x) =$
$(\sum x \in \{t.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (\textit{more},\ y) \wedge [\![Q]\!]_e\ (y,\ t)\}.\ \textit{pmf prob}_v\ x)$
using *assms*(*1*) **apply** (*simp*)
by (*metis* (*no-types*, *lifting*) *finite-subset infsetsum-finite subset-UNIV*)
then have *f12*: $(\sum x \in \{t.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (\textit{more},\ y) \wedge [\![Q]\!]_e\ (y,\ t)\}.\ \textit{pmf prob}_v\ x) = (1{::}\textit{real})$
using *a1* **by** *linarith*

have *prob-ennreal-extract*: ($\sum$ *a*::*'a*∈*UNIV*. *ennreal*
($\sum$ *x*::*'a*∈*UNIV*.
if ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)
then pmf prob$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}) *else* (*0*::*real*)))
= (*ennreal* ($\sum$ *a*::*'a*∈*UNIV*.
($\sum$ *x*::*'a*∈*UNIV*. ((
if ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)
then pmf prob$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}) *else* (*0*::*real*))))))
apply (*rule sum-ennreal-extract*)
by (*simp add*: *sum-nonneg*)
have *prob-swap*: ($\sum$ *a*::*'a*∈*UNIV*.
($\sum$ *x*::*'a*∈*UNIV*. ((
if ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)
then pmf prob$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}) *else* (*0*::*real*)))))
= ($\sum$ *x*::*'a*∈*UNIV*.
($\sum$ *a*::*'a*∈*UNIV*. (
if ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)
then pmf prob$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}) *else* (*0*::*real*))))
by (*rule sum.swap*)
have *prob-if-cases*: ... = ($\sum$ *x*::*'a*∈*UNIV*.
((*sum* (λ*a*. *pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}))
({*a*. ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)}))))
using *assms*(*1*) **by** (*simp add*: *sum.If-cases*)
have *prob-set-split*: ... = ($\sum$ *x*::*'a*∈({*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)} ∪
−{*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)}).
((*sum* (λ*a*. *pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}))
({*a*. ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)}))))
by *simp*
have *prob-disjoint-union*: ... = ($\sum$ *x*::*'a*∈({*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)}).
((*sum* (λ*a*. *pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}))
({*a*. ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)})))) +
($\sum$ *x*::*'a*∈(−{*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)}).
((*sum* (λ*a*. *pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}))
({*a*. ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)}))))
by (*metis* (*mono-tags*, *lifting*) *Compl-iff IntE assms*(*1*)
boolean-algebra-class.sup-compl-top finite-Un sum.union-inter-neutral)
have *prob-elim-zero*: ... = ($\sum$ *x*::*'a*∈({*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)}).
((*sum* (λ*a*. *pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧ ⟦*Q*⟧$_e$ (*t*, *x*)}))
({*a*. ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*)}))))
apply (*simp add*: *sum-uniform-value-zero*)
by (*smt Compl-eq card-eq-sum mem-Collect-eq sum.not-neutral-contains-not-neutral*)
have *prob-uniform-value*: ... = ($\sum$ *x*::*'a*∈({*x*. ∃ *y*::*'a*. ⟦*P*⟧$_e$ (*more*, *y*) ∧ ⟦*Q*⟧$_e$ (*y*, *x*)}).
(*pmf prob*$_v$ *x*))
apply (*rule sum-uniform-value'*)
using *assms*(*1*) *rev-finite-subset* **apply** *auto*[*1*]
by *blast*
have *prob-eq-1*: ... = (*1*::*real*)
using *f12* **by** *auto*
show ($\sum$ *a*::*'a*∈*UNIV*. *ennreal*
($\sum$ *x*::*'a*∈*UNIV*.
if ⟦*P*⟧$_e$ (*more*, *a*) ∧ ⟦*Q*⟧$_e$ (*a*, *x*) *then pmf prob*$_v$ *x* / *real* (*card* {*t*::*'a*. ⟦*P*⟧$_e$ (*more*, *t*) ∧
⟦*Q*⟧$_e$ (*t*, *x*)})
else (*0*::*real*))) = (*1*::*ennreal*)
using *ennreal-1 prob-disjoint-union prob-elim-zero prob-ennreal-extract prob-eq-1*
prob-if-cases prob-set-split prob-swap prob-uniform-value **by** *presburger*

qed

— This is the subgoal 2. We need *?p* and *?Q* to construct witnesses for $prob_v{}'$ and x respectively.

show $\exists (ok_v{}'::bool)\ prob_v{}'::'a\ pmf.$
$(ok_v \wedge [\![p]\!]_e\ more \longrightarrow ok_v{}' \wedge (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (1::real)) \wedge$
$(ok_v{}' \wedge infsetsum\ (pmf\ prob_v{}')\ (Collect\ [\![q]\!]_e) = (1::real) \longrightarrow$
$(\exists x::'a \Rightarrow 'a\ pmf.$
$(\forall xa::'a.\ pmf\ prob_v\ xa = (\sum_a xb::'a.\ pmf\ prob_v{}'\ xb \cdot pmf\ (x\ xb)\ xa)) \wedge$
$(\forall xa::'a.$
$(\exists prob_v::'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x::'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1::real)) \wedge$
$(\forall xb::'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \longrightarrow$
$\neg (0::real) < pmf\ prob_v{}'\ xa)))$
apply (*rule-tac* x = *True* **in** *exI*)
— Construct a witness for $prob_v{}'$ by *?p*
apply (*rule-tac* x = *embed-pmf* (*?p*) **in** *exI*)
apply (*auto*)
proof −
have *f1*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).$
$pmf\ (embed\text{-}pmf$
$(\lambda s_0::'a.$
$\sum_a s'::'a.$
$if\ [\![P]\!]_e\ (more,\ s_0) \wedge [\![Q]\!]_e\ (s_0,\ s')$
$then\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ s')\})$
$else\ (0::real)))\ x)$
$= (\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ ?p\ x)$
apply (*subst pmf-embed-pmf*)
apply (*simp add*: *infsetsum-nonneg*)
apply (*simp add*: *assms*(*1*) *nn-integral-count-space-finite*)
defer
apply (*simp*)
using *p-prob* **by** *blast*
have *f2*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ ?p\ x) = (1::real)$
proof −
have *P-infset-to-fset*: $(\sum_a x::'a \mid [\![P]\!]_e\ (more,\ x).\ ?p\ x) =$
$(\sum x::'a \mid [\![P]\!]_e\ (more,\ x).\ (\sum s'::'a{\in}UNIV.\ ?f\ s'\ x))$
using *assms*(*1*)
by (*smt boolean-algebra-class.sup-compl-top finite-Un infsetsum-finite sum-mono*)
have *P-swap*: ... $= (\sum s'::'a{\in}UNIV.\ \sum x::'a \mid [\![P]\!]_e\ (more,\ x).\ ?f\ s'\ x)$
by (*rule sum.swap*)
have *P-if-cases*: ... $= (\sum s'::'a{\in}UNIV.$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ s')\}))$
$(\{x.\ [\![P]\!]_e\ (more,\ x)\} \cap \{x.\ [\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (x,\ s')\}))))$
using *assms*(*1*) **apply** (*subst sum.If-cases*)
using *rev-finite-subset* **apply** *blast*
by *simp*
have *P-if-cases'*: ... $= (\sum s'::'a{\in}UNIV.$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ s')\}))$
$(\{x.\ [\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (x,\ s')\}))))$
by (*simp add*: *Collect-conj-eq*)
have *P-split*: ... $= (\sum s'::'a{\in}(\{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\} \cup$
$-\{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\}).$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ s')\}))$
$(\{x.\ [\![P]\!]_e\ (more,\ x) \wedge [\![Q]\!]_e\ (x,\ s')\}))))$
by *simp*

have *P-disjoint-union*: ... = $(\sum s'::'a \in (\{x.\ \exists y::'a.\ \llbracket P\rrbracket_e\ (more,\ y) \wedge \llbracket Q\rrbracket_e\ (y,\ x)\}).$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ \llbracket P\rrbracket_e\ (more,\ t) \wedge \llbracket Q\rrbracket_e\ (t,\ s')\}))$
$(\{x.\ \llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (x,\ s')\})))) +$
$(\sum s'::'a \in (-\{x.\ \exists y::'a.\ \llbracket P\rrbracket_e\ (more,\ y) \wedge \llbracket Q\rrbracket_e\ (y,\ x)\}).$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ \llbracket P\rrbracket_e\ (more,\ t) \wedge \llbracket Q\rrbracket_e\ (t,\ s')\}))$
$(\{x.\ \llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (x,\ s')\}))))$
by (*meson Compl-iff Int-iff assms*(*1*) *finite-subset subset-UNIV sum.union-inter-neutral*)
have *P-elim-zero*: ... = $(\sum s'::'a \in (\{x.\ \exists y::'a.\ \llbracket P\rrbracket_e\ (more,\ y) \wedge \llbracket Q\rrbracket_e\ (y,\ x)\}).$
$((sum\ (\lambda x.\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ \llbracket P\rrbracket_e\ (more,\ t) \wedge \llbracket Q\rrbracket_e\ (t,\ s')\}))$
$(\{x.\ \llbracket P\rrbracket_e\ (more,\ x) \wedge \llbracket Q\rrbracket_e\ (x,\ s')\}))))$
apply (*simp add*: *sum-uniform-value-zero*)
by (*smt Compl-eq card-eq-sum mem-Collect-eq sum.not-neutral-contains-not-neutral*)
have *P-sum-elim*: ... = $(\sum s'::'a \in (\{x.\ \exists y::'a.\ \llbracket P\rrbracket_e\ (more,\ y) \wedge \llbracket Q\rrbracket_e\ (y,\ x)\}).\ pmf\ prob_v$
$s')$
apply (*rule sum-uniform-value'*)
using *assms*(*1*) *rev-finite-subset* **apply** *auto*[*1*]
by *blast*
have *prob-eq-1*: ... = (*1*::*real*)
by (*metis* (*no-types*, *lifting*) *Compl-partition a1 assms*(*1*) *finite-Un infsetsum-finite*)
show *?thesis*
using *P-disjoint-union P-elim-zero P-if-cases P-if-cases' P-infset-to-fset*
P-split P-sum-elim P-swap prob-eq-1 **by** *linarith*
qed
show $(\sum_a x::'a \mid \llbracket P\rrbracket_e\ (more,\ x).$
$pmf\ (embed\text{-}pmf$
$(\lambda s_0::'a.$
$\sum_a s'::'a.$
$if\ \llbracket P\rrbracket_e\ (more,\ s_0) \wedge \llbracket Q\rrbracket_e\ (s_0,\ s')$
$then\ pmf\ prob_v\ s'\ /\ real\ (card\ \{t::'a.\ \llbracket P\rrbracket_e\ (more,\ t) \wedge \llbracket Q\rrbracket_e\ (t,\ s')\})$
$else\ (0::real)))$
$x) = (1::real)$
by (*simp add*: *f1 f2*)
next
assume *a-sum-q*: *infsetsum* (*pmf* (*embed-pmf* (*?p*))) (*Collect* $\llbracket q\rrbracket_e$) = (*1*::*real*)
have *f01*: $\forall s.\ (\sum a::'a \in UNIV.\ (?Q\ s)\ a) = (1::real)$
proof −
have *Q-cond-ext*: $\forall s.\ (\sum a::'a \in UNIV.\ (?Q\ s)\ a) =$
$(if\ (0::real) < ?p\ s$
$then\ \sum a::'a \in UNIV.\ ?f\ a\ s\ /\ ?p\ s$
$else\ \sum a::'a \in UNIV.\ (1::real)\ /\ real\ CARD('a))$
by *auto*
have *Q-uniform-dis*: $(\sum a::'a \in UNIV.\ (1::real)\ /\ real\ CARD('a)) = 1$
by (*simp add*: *assms*(*1*))
have *Q-sum-div-ext*: $\forall s.\ (if\ (0::real) < ?p\ s$
$then\ \sum a::'a \in UNIV.\ ?f\ a\ s\ /\ ?p\ s$
$else\ \sum a::'a \in UNIV.\ (1::real)\ /\ real\ CARD('a)) =$
$(if\ (0::real) < ?p\ s$
$then\ (\sum a::'a \in UNIV.\ ?f\ a\ s)\ /\ ?p\ s$
$else\ \sum a::'a \in UNIV.\ (1::real)\ /\ real\ CARD('a))$
by (*simp add*: *sum-divide-distrib*)
have *Q-eq-1*: $\forall s.\ (if\ (0::real) < ?p\ s$
$then\ (\sum a::'a \in UNIV.\ ?f\ a\ s)\ /\ ?p\ s$
$else\ \sum a::'a \in UNIV.\ (1::real)\ /\ real\ CARD('a)) = 1$
by (*simp add*: *assms*(*1*))
show *?thesis*

by (*simp add*: *Q-cond-ext Q-eq-1 Q-sum-div-ext*)
qed
have *P-simp*: $\forall x.$ *pmf* (*embed-pmf* (*?p*)) x = *?p* x
apply (*subst pmf-embed-pmf*)
apply (*simp add*: *infsetsum-nonneg*)
apply (*simp add*: *assms*(*1*) *nn-integral-count-space-finite*)
defer
apply (*simp*)
using *p-prob* **by** *blast*
from *a-sum-q* **have** *a-sum-q'*: *infsetsum ?p* (*Collect* $[\![q]\!]_e$) = (*1*::*real*)
using *P-simp* **by** *auto*
have *Q-simp*: $\forall x. \forall s.$ *pmf* (*embed-pmf* (*?Q s*)) x = (*?Q s*) x
apply (*subst pmf-embed-pmf*)
apply (*simp add*: *infsetsum-nonneg*)
apply (*simp add*: *assms*(*1*) *nn-integral-count-space-finite*)
defer
apply (*simp*)
using *f01* **by** (*simp add*: *assms*(*1*))
have *f02*: ($\forall$ *xa*::*'a*.
pmf prob$_v$ *xa* = ($\sum_a$*xb*::*'a*. *pmf* (*embed-pmf* (*?p*)) *xb* $\cdot$ *pmf* (*embed-pmf* (*?Q xb*)) *xa*))
proof −
have *f021*: $\forall$ *xa*::*'a*. ($\sum_a$*xb*::*'a*. *pmf* (*embed-pmf* (*?p*)) *xb* $\cdot$ *pmf* (*embed-pmf* (*?Q xb*)) *xa*)
= ($\sum_a$*xb*::*'a*. (*?p xb*) $\cdot$ *pmf* (*embed-pmf* (*?Q xb*)) *xa*)
using *P-simp* **by** *auto*
have *f022*: $\forall$ *xa*::*'a*. ($\sum_a$*xb*::*'a*. (*?p xb*) $\cdot$ *pmf* (*embed-pmf* (*?Q xb*)) *xa*) =
($\sum_a$*xb*::*'a*. (*?p xb*) $\cdot$ (*?Q xb*) *xa*)
using *Q-simp* **by** *auto*
have *f023*: $\forall$ *xa*::*'a*. ($\sum_a$*xb*::*'a*. (*?p xb*) $\cdot$ (*?Q xb*) *xa*) =
($\sum_a$*xb*::*'a*.
(*if* (*0*::*real*) < (*?p xb*)
then ((*?p xb*) $\cdot$ (*?f xa xb* / *?p xb*))
else ((*?p xb*) $\cdot$ ((*1*::*real*) / *real CARD*(*'a*)))))
using *assms*(*1*)
by (*smt div-by-1 infsetsum-cong nonzero-eq-divide-eq times-divide-eq-right*)
have *p-leq-zero*: $\forall$ *xb*. (*?p xb*)$\geq$ *0*
by (*simp add*: *infsetsum-nonneg*)
have *f024*: $\forall$ *xa*::*'a*. ($\sum_a$*xb*::*'a*.
(*if* (*0*::*real*) < (*?p xb*)
then ((*?p xb*) $\cdot$ (*?f xa xb* / *?p xb*))
else ((*?p xb*) $\cdot$ ((*1*::*real*) / *real CARD*(*'a*))))) =
($\sum_a$*xb*::*'a*. (*if* (*0*::*real*) < (*?p xb*) *then* (*?f xa xb*) *else 0*))
using *p-leq-zero*
by (*smt divide-cancel-right infsetsum-cong mult-not-zero nonzero-mult-div-cancel-left*)
have *f025*: $\forall$ *xa*::*'a*. ($\sum_a$*xb*::*'a*. (*if* (*0*::*real*) < (*?p xb*) *then* (*?f xa xb*) *else 0*)) =
($\sum$*xb*::*'a*$\in$\{*xb*. (*0*::*real*) < (*?p xb*)\}. (*?f xa xb*))
using *assms*(*1*) **by** (*simp add*: *sum.If-cases*)
have *f026*: $\forall$ *xa*::*'a*. ($\sum$*xb*::*'a*$\in$\{*xb*. (*0*::*real*) < (*?p xb*)\}. (*?f xa xb*))
= ($\sum$*xb*::*'a*$\in$(\{*xb*. (*0*::*real*) < (*?p xb*)\} $\cap$ \{*xb*. $[\![P]\!]_e$ (*more*, *xb*) $\wedge$ $[\![Q]\!]_e$ (*xb*, *xa*)\}).
(*pmf prob*$_v$ *xa* / *real* (*card* \{*t*::*'a*. $[\![P]\!]_e$ (*more*, *t*) $\wedge$ $[\![Q]\!]_e$ (*t*, *xa*)\})))
using *assms*(*1*) **apply** (*subst sum.If-cases*)
using *rev-finite-subset* **apply** *blast*
by *simp*
have *f028*: $\forall$ *xa*::*'a*. ($\sum$*xb*::*'a*$\in$(\{*xb*. (*0*::*real*) < (*?p xb*)\} $\cap$
\{*xb*. $[\![P]\!]_e$ (*more*, *xb*) $\wedge$ $[\![Q]\!]_e$ (*xb*, *xa*)\}).
(*pmf prob*$_v$ *xa* / *real* (*card* \{*t*::*'a*. $[\![P]\!]_e$ (*more*, *t*) $\wedge$ $[\![Q]\!]_e$ (*t*, *xa*)\}))) = *pmf prob*$_v$ *xa*

apply (*rule allI*)
proof −
  fix $xa::'a$
  show $(\sum xb::'a\in(\{xb.\ (0::real) < (?p\ xb)\} \cap$
    $\{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\}).$
   $(pmf\ prob_v\ xa\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\}))) = pmf\ prob_v\ xa$
   proof (*cases* $pmf\ prob_v\ xa = 0$)
    case *True*
    then show *?thesis*
     by *simp*
   next
    case *False*
    then have *notneg*: $pmf\ prob_v\ xa > 0$
     by *simp*
    from *a1* **have** *comp-set*:
     $(\sum_a x::'a \in -\{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\}.\ pmf\ prob_v\ x) = (0::real)$
     using *pmf-comp-set* **by** *blast*
    then have *all-zero*: $\forall x \in -\{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\}.\ pmf\ prob_v\ x$
$= 0$
     using *pmf-all-zero* **by** *blast*
    have *not-in*: $xa \notin -\{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\}$
     using *notneg all-zero False* **by** *blast*
    then have *is-in*: $xa \in \{x.\ \exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x)\}$
     by *blast*
    then have *exist*: $\exists y::'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ xa)$
     by *blast*
    then have *card-not-zero*: $real\ (card\ \{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\}) \neq 0$
     by (*metis* (*no-types*, *lifting*) *Collect-empty-eq assms*(*1*) *card-0-eq*
       finite-subset of-nat-0-eq-iff order-top-class.top-greatest)
    have *ff*: $\{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\} \subseteq \{xb.\ (0::real) < (?p\ xb)\}$
     apply *auto*
     proof −
      fix $x::'a$
      assume *a11*: $[\![P]\!]_e\ (more,\ x)$
      assume *a12*: $[\![Q]\!]_e\ (x,\ xa)$
      let $?fx = \lambda xb.\ if\ [\![Q]\!]_e\ (x,\ xb)\ then\ pmf\ prob_v\ xb\ /$
       $real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xb)\})\ else\ (0::real)$
      have *ff0*: $\forall xb.\ ?fx\ xb \geq 0$
       by *simp*
      then have *ff1*:$(\sum xb::'a\in\{xa\}.\ ?fx\ xb) \leq (\sum xa::'a\in UNIV.\ ?fx\ xa)$
       using *assms*(*1*) **apply** (*subst sum-mono2*)
       apply *blast*
       apply *blast*
       apply *blast*
       by *auto*
      then have *ff2*:$(\sum_a xb::'a\in\{xa\}.\ ?fx\ xb) \leq (\sum_a\ xa::'a.\ ?fx\ xa)$
       using *assms*(*1*) **by** *simp*
      have *card-no-zero*: $(card\ \{t::'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\}) > 0$
       using *a11 a12*
       by (*metis* (*mono-tags*, *lifting*) *Collect-empty-eq assms*(*1*) *card-gt-0-iff*
        finite-subset order-top-class.top-greatest)
     have *ff3*:$(\sum_a xb::'a\in\{xa\}.\ ?fx\ xb) = pmf\ prob_v\ xa\ /\ real\ (card\ \{t::'a.\ [\![P]\!]_e\ (more,$
$t) \wedge [\![Q]\!]_e\ (t,\ xa)\})$
       using *a12* **by** *auto*
      have *ff4*:... > 0

using *notneg card-no-zero*
by *simp*
show $(0{::}real) < (\sum_a xa{::}'a.\ if\ [\![Q]\!]_e\ (x,\ xa)\ then\ pmf\ prob_v\ xa\ /$
$real\ (card\ \{t{::}'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\})\ else\ (0{::}real))$
using *ff2 ff3 ff4* **by** *linarith*
qed

have *ff1*: $(\sum xb{::}'a{\in}(\{xb.\ (0{::}real) < (?p\ xb)\} \cap$
$\{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\}).$
$(pmf\ prob_v\ xa\ /\ real\ (card\ \{t{::}'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\}))) =$
$(\sum xb{::}'a{\in}(\{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\}).$
$(pmf\ prob_v\ xa\ /\ real\ (card\ \{t{::}'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\})))$
using *ff*
by (*simp add*: *semilattice-inf-class.inf.absorb-iff2*)
have *ff2*: ... =
$(real\ (card\ \{xb.\ [\![P]\!]_e\ (more,\ xb) \wedge [\![Q]\!]_e\ (xb,\ xa)\}) *$
$(pmf\ prob_v\ xa\ /\ real\ (card\ \{t{::}'a.\ [\![P]\!]_e\ (more,\ t) \wedge [\![Q]\!]_e\ (t,\ xa)\})))$
by *simp*
have *ff3*: ... $= pmf\ prob_v\ xa$
using *card-not-zero* **by** *simp*
show *?thesis*
using *ff1 ff2 ff3* **by** *linarith*
qed
qed
show *?thesis*
using *f021 f022 f023 f024 f025 f026 f028* **by** *auto*
qed
show $\exists x{::}'a \Rightarrow 'a\ pmf.$
$(\forall xa{::}'a.$
$pmf\ prob_v\ xa = (\sum_a xb{::}'a.\ pmf\ (embed\text{-}pmf\ (?p))\ xb \cdot pmf\ (x\ xb)\ xa)) \wedge$
$(\forall xa{::}'a.$
$(\exists prob_v{::}'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1{::}real)) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \longrightarrow$
$\neg (0{::}real) < pmf\ (embed\text{-}pmf\ (?p))\ xa)$
apply (*rule-tac* $x = \lambda s.\ embed\text{-}pmf\ (?Q\ s)$ **in** *exI*)
apply (*rule conjI*)
using *f02* **apply** *blast*
proof
fix $xa{::}'a$
have *f10*: $(\exists prob_v{::}'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1{::}real)) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = (?Q\ xa)\ xb)) \longrightarrow$
$\neg (0{::}real) < ?p\ xa$
apply (*rule impI*)
proof –
assume *aa*: $(\exists prob_v{::}'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1{::}real)) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = (?Q\ xa)\ xb))$
have $(([\![q]\!]_e\ xa \longrightarrow \neg (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ (?Q\ xa)\ x) = (1{::}real)))$
using *aa* **by** *auto*
then have $\neg [\![q]\!]_e\ xa \vee ([\![q]\!]_e\ xa \wedge \neg (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ (?Q\ xa)\ x) = (1{::}real))$
by (*simp add*: *disjCI*)
then show $\neg (0{::}real) < ?p\ xa$
proof

assume *aa*: $\neg[\![q]\!]_e$ *xa*
from *a-sum-q′* **have** *infsetsum ?p* $(-$*Collect* $[\![q]\!]_e) = (0$::*real*)
by (*metis* (*no-types*, *lifting*) *P-simp infsetsum-cong pmf-comp-set*)
then show $\neg$ (*0*::*real*) < *?p xa*
using *a-sum-q′ pmf-all-zero aa*
by (*smt Compl-iff P-simp infsetsum-cong mem-Collect-eq*)
next
assume *aa1*: $([\![q]\!]_e$ *xa* $\wedge \neg (\sum_a x$::′*a* $\mid [\![Q]\!]_e$ (*xa*, *x*). (*?Q xa*) *x*) = (*1*::*real*))
show $\neg$ (*0*::*real*) < *?p xa*
proof (*rule ccontr*)
assume *ac*: $\neg\neg$(*0*::*real*) < *?p xa*
from *ac* **have** $[\![P]\!]_e$ (*more*, *xa*)
by *force*
have *fc*: $(\sum_a x$::′*a* $\mid [\![Q]\!]_e$ (*xa*, *x*). (*?Q xa*) *x*) =
$(\sum_a x$::′*a* $\mid [\![Q]\!]_e$ (*xa*, *x*). (*?f x xa* / *?p xa*))
using *ac* **by** *auto*
have *fc1*: ... = $(\sum_a x$::′*a* $\mid [\![Q]\!]_e$ (*xa*, *x*). (*?f x xa*))/*?p xa*
proof −
have $\forall$ *r A f*. *infsetsum f A* / (*r*::*real*) = $(\sum_a a \in A$. *f* (*a*::′*a*) / *r*)
by (*metis assms*(*1*) *finite-subset infsetsum-finite subset-UNIV*
sum-divide-distrib)
then show *?thesis*
by *presburger*
qed
have *fc2*: ... = $(\sum_a x$::′*a* $\in$ (*UNIV* $-(-\{x.\ [\![Q]\!]_e$ (*xa*, *x*)})). (*?f x xa*))/*?p xa*
by *simp*
have *fc3*: ... = $((\sum_a x$::′*a* $\in$ (*UNIV*). (*?f x xa*)) −
$(\sum_a x$::′*a* $\in (-\{x.\ [\![Q]\!]_e$ (*xa*, *x*)}). (*?f x xa*)))/*?p xa*
using *assms*(*1*)
by (*smt Compl-eq-Diff-UNIV DiffE IntE boolean-algebra-class.sup-compl-top*
finite-Un infsetsum-finite sum.not-neutral-contains-not-neutral
sum.union-inter)
have *fc4*: ... = $((\sum_a x$::′*a* $\in$ (*UNIV*). (*?f x xa*))/*?p xa*) −
$(\sum_a x$::′*a* $\in (-\{x.\ [\![Q]\!]_e$ (*xa*, *x*)}). (*?f x xa*))/*?p xa*
using *diff-divide-distrib* **by** *blast*
have *fc5*: ... = *1*
by (*smt ComplD aa1 ac div-self fc fc1 fc2 fc3 infsetsum-all-0 mem-Collect-eq*)
show *False*
using *aa1 fc5 fc fc1 fc2 fc3 fc4* **by** *linarith*
qed
qed
qed

show ($\exists$ *prob*$_v$::′*a pmf*.
$([\![q]\!]_e$ *xa* $\longrightarrow \neg (\sum_a x$::′*a* $\mid [\![Q]\!]_e$ (*xa*, *x*). *pmf prob*$_v$ *x*) = (*1*::*real*)) $\wedge$
($\forall$ *xb*::′*a*. *pmf prob*$_v$ *xb* = *pmf* (*embed-pmf* (*?Q xa*)) *xb*)) $\longrightarrow$
$\neg$ (*0*::*real*) < *pmf* (*embed-pmf* (*?p*)) *xa*
using *P-simp Q-simp f10* **by** *auto*
qed
qed
next
fix *ok*$_v$::*bool* **and** *more*::′*a* **and** *ok*$_v$′::*bool* **and** *ok*$_v$″::*bool* **and** *prob*$_v$′::′*a pmf*
assume *a1*: $\forall$ *y*::′*a*. $[\![P]\!]_e$ (*more*, *y*) $\longrightarrow [\![q]\!]_e$ *y*
assume *a2*: $(\sum_a x$::′*a* $\mid [\![P]\!]_e$ (*more*, *x*). *pmf prob*$_v$′ *x*) = (*1*::*real*)
assume *a3*: $\neg$ *infsetsum* (*pmf prob*$_v$′) (*Collect* $[\![q]\!]_e$) = (*1*::*real*)

from *a1* **have** *f1*: $\{t.\ [\![P]\!]_e\ (more,\ t)\} \subseteq \{t.\ [\![q]\!]_e\ t\}$
by *blast*
have *f2*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (\sum_a x \in \{t.\ [\![P]\!]_e\ (more,\ t)\}.\ pmf\ prob_v{}'\ x)$
by *blast*
have *f3*: $(\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x) = (\sum_a x \in \{t.\ [\![q]\!]_e\ t\}.\ pmf\ prob_v{}'\ x)$
by *blast*
have *f4*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) \le (\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x)$
using *f2 f3 f1*
by (*meson infsetsum-mono-neutral-left order-refl pmf-abs-summable pmf-nonneg*)
have *f5*: $(\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x) = 1$
using *a2 f4*
by (*smt measure-pmf.prob-le-1 measure-pmf-conv-infsetsum*)
from *f5* **have** *f1*: $infsetsum\ (pmf\ prob_v{}')\ (Collect\ [\![q]\!]_e) = (1{::}real)$
by *blast*
show $ok_v{}'$
using *f1 a3* **by** *blast*
next
fix $ok_v{::}bool$ **and** $more{::}'a$ **and** $prob_v{::}'a\ pmf$ **and** $ok_v{}''{::}bool$ **and** $prob_v{}'{::}'a\ pmf$
assume *a1*: $\forall y{::}'a.\ [\![P]\!]_e\ (more,\ y) \longrightarrow [\![q]\!]_e\ y$
assume *a2*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (1{::}real)$
assume *a3*: $\neg\ infsetsum\ (pmf\ prob_v{}')\ (Collect\ [\![q]\!]_e) = (1{::}real)$
from *a1* **have** *f1*: $\{t.\ [\![P]\!]_e\ (more,\ t)\} \subseteq \{t.\ [\![q]\!]_e\ t\}$
by *blast*
have *f2*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (\sum_a x \in \{t.\ [\![P]\!]_e\ (more,\ t)\}.\ pmf\ prob_v{}'\ x)$
by *blast*
have *f3*: $(\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x) = (\sum_a x \in \{t.\ [\![q]\!]_e\ t\}.\ pmf\ prob_v{}'\ x)$
by *blast*
have *f4*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) \le (\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x)$
using *f2 f3 f1*
by (*meson infsetsum-mono-neutral-left order-refl pmf-abs-summable pmf-nonneg*)
have *f5*: $(\sum_a x{::}'a \mid [\![q]\!]_e\ x.\ pmf\ prob_v{}'\ x) = 1$
using *a2 f4*
by (*smt measure-pmf.prob-le-1 measure-pmf-conv-infsetsum*)
from *f5* **have** *f1*: $infsetsum\ (pmf\ prob_v{}')\ (Collect\ [\![q]\!]_e) = (1{::}real)$
by *blast*
show $(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x).\ pmf\ prob_v\ x) = (1{::}real)$
using *f1 a3* **by** *blast*
next
— Subgoal 5: postcondition implied from RHS to LHS: An intermediate distribution $prob_v{}'$ and a function xx from intermediate states to the distribution on final states implies $prob'(P;\ Q){=}1$.
fix $ok_v{::}bool$ **and** $more{::}'a$ **and** $ok_v{}'{::}bool$ **and** $prob_v{::}'a\ pmf$ **and** $ok_v{}''{::}bool$ **and**
$prob_v{}'{::}'a\ pmf$ **and** $xx{::}'a \Rightarrow 'a\ pmf$
assume *a1*: $[\![p]\!]_e\ more$
assume *a2*: $\forall y{::}'a.\ [\![P]\!]_e\ (more,\ y) \longrightarrow [\![q]\!]_e\ y$
assume *a3*: $(\sum_a x{::}'a \mid [\![P]\!]_e\ (more,\ x).\ pmf\ prob_v{}'\ x) = (1{::}real)$
assume *a4*: $\forall xa{::}'a.\ pmf\ prob_v\ xa = (\sum_a xb{::}'a.\ pmf\ prob_v{}'\ xb \cdot pmf\ (xx\ xb)\ xa)$
assume *a5*: $\forall xa{::}'a.$
$(\exists\, prob_v{::}'a\ pmf.$
$([\![q]\!]_e\ xa \longrightarrow \neg\ (\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ prob_v\ x) = (1{::}real)) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (xx\ xa)\ xb)) \longrightarrow$
$\neg\ (0{::}real) < pmf\ prob_v{}'\ xa$
let $?A = \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}$
let $?f = \lambda x\ xa.\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x$

from *a5* **have** *f1-0*: $\forall\, xa{::}'a.\ (0{::}real) < pmf\ prob_v{}'\ xa \longrightarrow$
$(\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ (xx\ xa)\ x) = (1{::}real)$
by *blast*
from *a3* **have** *f1-1*: $\forall\, xa{::}'a.\ (0{::}real) < pmf\ prob_v{}'\ xa \longrightarrow [\![P]\!]_e\ (more,\ xa)$
using *pmf-all-zero pmf-utp-comp0′* **by** *fastforce*
have *f1-2*: $\forall\, xa{::}'a.\ (0{::}real) < pmf\ prob_v{}'\ xa \longrightarrow$
$\{x.\ [\![Q]\!]_e\ (xa,\ x)\} \subseteq\ ?A$
using *f1-1* **by** *blast*
then have *f1-3*: $\forall\, xa{::}'a.\ (0{::}real) < pmf\ prob_v{}'\ xa \longrightarrow$
$(\sum x \in\ ?A.\ pmf\ (xx\ xa)\ x) \geq$
$(\sum_a x{::}'a \mid [\![Q]\!]_e\ (xa,\ x).\ pmf\ (xx\ xa)\ x)$
by (*metis* (*no-types*, *lifting*) *assms*(*1*) *boolean-algebra-class.sup-compl-top finite-Un infsetsum-finite pmf-nonneg sum-mono2*)
then have *f2*: $\forall\, xa{::}'a.\ (0{::}real) < pmf\ prob_v{}'\ xa \longrightarrow$
$(\sum x \in\ ?A.\ pmf\ (xx\ xa)\ x) = 1$
using *f1-0*
by (*smt assms*(*1*) *infsetsum-finite pmf-nonneg subset-UNIV sum-mono2 sum-pmf-eq-1*)

have *f3*: $(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x).\ \sum_a xa{::}'a.\ ?f\ x\ xa) =$
$(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x).$
$\sum_a xa{::}'a.\ if\ pmf\ prob_v{}'\ xa > 0\ then\ ?f\ x\ xa\ else\ 0)$
by (*smt infsetsum-cong mult-not-zero pmf-nonneg*)
also have *f4*: ... =
$(\sum_a x \in \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}.$
$\sum_a xa \in UNIV.\ if\ pmf\ prob_v{}'\ xa > 0\ then\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x\ else\ 0)$
by *blast*
also have *f5*: ... =
$(\sum x \in \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}.$
$\sum xa \in UNIV.\ if\ pmf\ prob_v{}'\ xa > 0\ then\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x\ else\ 0)$
using *assms*(*1*)
by (*metis* (*no-types*, *lifting*) *finite-subset infsetsum-finite subset-UNIV sum.cong*)
have *f6*: ... = $(\sum xa \in UNIV.\ \sum x \in \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}.$
$if\ pmf\ prob_v{}'\ xa > 0\ then\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x\ else\ 0)$
using *assms*(*1*) **apply** (*subst sum.swap*)
by *blast*
have *f7*: ... = $(\sum xa \in UNIV.\ if\ pmf\ prob_v{}'\ xa > 0\ then$
$(\sum x \in \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}.\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x)\ else\ 0)$
by (*smt sum.cong sum.not-neutral-contains-not-neutral*)
have *f8*: ... = $(\sum xa \in UNIV.\ if\ pmf\ prob_v{}'\ xa > 0\ then$
$pmf\ prob_v{}'\ xa \cdot (\sum x \in \{s'.\ \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ s')\}.\ pmf\ (xx\ xa)\ x)\ else\ 0)$
by (*metis* (*no-types*) *sum-distrib-left*)
have *f9*: ... = $(\sum xa \in UNIV.\ if\ pmf\ prob_v{}'\ xa > 0\ then\ pmf\ prob_v{}'\ xa\ else\ 0)$
using *f2* **by** (*metis* (*no-types*, *lifting*) *mult-cancel-left2*)
have *f10*: ... = $(\sum xa \in UNIV.\ pmf\ prob_v{}'\ xa)$
by (*meson less-linear pmf-not-neg*)
then show $(\sum_a x{::}'a \mid \exists\, y{::}'a.\ [\![P]\!]_e\ (more,\ y) \wedge [\![Q]\!]_e\ (y,\ x).$
$\sum_a xa{::}'a.\ pmf\ prob_v{}'\ xa \cdot pmf\ (xx\ xa)\ x) = (1{::}real)$
by (*smt assms*(*1*) *f3 f5 f6 f7 f8 f9 infsetsum-finite pmf-pos sum.cong sum-pmf-eq-1*)

qed
show *?thesis*
using *p q seq-comp-ndesign* **by** *blast*

qed

lemma *kleisli-left-mono*:
assumes $P \sqsubseteq Q$
assumes P *is* **N** Q *is* **N**
shows $\uparrow P \sqsubseteq \uparrow Q$
proof $-$
obtain $pre_p\ post_p\ pre_q\ post_q$
where $p{:}P = (pre_p \vdash_n post_p)$ **and**
$q{:}Q = (pre_q \vdash_n post_q)$
using *assms* **by** (*metis ndesign-form*)
have *f1*: $[\![\lfloor pre_D\ P\rfloor_<]\!]_p \subseteq [\![\lfloor pre_D\ Q\rfloor_<]\!]_p$
apply (*simp add*: *upred-set.rep-eq*)
using *assms*
by (*smt Collect-mono H1-H3-impl-H2 arestr.rep-eq rdesign-ref-monos*(*1*) *upred-ref-iff*)
have *f2*: $`pre_p \Rightarrow pre_q`$
using *p q assms* **by** (*simp add*: *ndesign-refinement'*)
have *f2'*: $post_p \sqsubseteq ?[pre_p]\ ;;\ post_q$
using *p q assms* **by** (*simp add*: *ndesign-refinement'*)
have *f3*: $[\![pre_p]\!]_p \subseteq [\![pre_q]\!]_p$
apply (*simp add*: *upred-set.rep-eq*)
apply (*rule Collect-mono*)
using *assms* **by** (*meson f2 impl.rep-eq taut.rep-eq*)
have *f4*: $\uparrow(pre_p \vdash_n post_p) \sqsubseteq \uparrow(pre_q \vdash_n post_q)$
apply (*simp add*: *kleisli-lift-alt-def kleisli-lift2'-def*)
apply (*simp add*: *ndesign-refinement*)
apply (*auto*)
apply (*pred-simp*)
using *f3 pmf-sum-subset-imp-1* **apply** *blast*
apply (*rel-simp*)
proof $-$
fix $prob_v{::}'a\ pmf$ **and** $prob_v'{::}'a\ pmf$ **and** $x{::}'a \Rightarrow 'a\ pmf$
assume *a1*: $infsetsum\ (pmf\ prob_v)\ [\![pre_p]\!]_p = (1{::}real)$
assume *a2*: $\forall xa{::}'a.\ pmf\ prob_v'\ xa = (\sum_a xb{::}'a.\ pmf\ prob_v\ xb \cdot pmf\ (x\ xb)\ xa)$
assume *a3*: $\forall xa{::}'a.$
$(\exists prob_v{::}'a\ pmf.$
$([\![pre_q]\!]_e\ xa \longrightarrow \neg\ [\![post_q]\!]_e\ (xa,\ (\!|prob_v = prob_v|\!))) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (x\ xa)\ xb)) \longrightarrow$
$\neg\ (0{::}real) < pmf\ prob_v\ xa$
show $\exists xa{::}'a \Rightarrow 'a\ pmf.$
$(\forall xb{::}'a.\ (\sum_a xa{::}'a.\ pmf\ prob_v\ xa \cdot pmf\ (x\ xa)\ xb) = (\sum_a x{::}'a.\ pmf\ prob_v\ x \cdot pmf\ (xa\ x)\ xb)) \wedge$
$(\forall x{::}'a.$
$(\exists prob_v{::}'a\ pmf.$
$([\![pre_p]\!]_e\ x \longrightarrow \neg\ [\![post_p]\!]_e\ (x,\ (\!|prob_v = prob_v|\!))) \wedge$
$(\forall xb{::}'a.\ pmf\ prob_v\ xb = pmf\ (xa\ x)\ xb)) \longrightarrow$
$\neg\ (0{::}real) < pmf\ prob_v\ x)$
apply (*rule-tac* $x = x$ **in** *exI*, *rule conjI*)
apply (*metis a1 mem-Collect-eq order-less-irrefl pmf-all-zero pmf-utp-comp0' upred-set.rep-eq*)
apply (*auto*)

using *a1 pmf-all-zero pmf-comp-set upred-set.rep-eq* **apply** *fastforce*
proof −
fix *xa*::*'a* **and** $prob_v$*'*::*'a pmf*
assume *a11*: ∀ *xb*::*'a*. *pmf* $prob_v{}'$ *xb* = *pmf* (*x xa*) *xb*
assume *a12*: (*0*::*real*) < *pmf* $prob_v$ *xa*
assume *a13*: ¬ $[\![post_p]\!]_e$ (*xa*, ⦇$prob_v = prob_v{}'$⦈)
from *a11* **have** *f11*: $prob_v{}' = x\ xa$
by (*simp add*: *pmf-eqI*)
from *a12* **have** *f12*: $[\![pre_p]\!]_e$ *xa*
using *a3* **by** (*smt Compl-iff a1 mem-Collect-eq pmf-all-zero pmf-comp-set upred-set.rep-eq*)
from *f12 f2* **have** *f13*: $[\![pre_q]\!]_e$ *xa*
using *a12 a3* **by** *blast*
have *f14*: $[\![post_q]\!]_e$ (*xa*, ⦇$prob_v = x\ xa$⦈)
using *a3 a12* **by** *blast*
have *f15*: $[\![post_p]\!]_e$ (*xa*, ⦇$prob_v = x\ xa$⦈)
using *f2'* **apply** (*rel-auto*)
by (*simp add*: *f12 f14*)
show *False*
using *a13 f11 f15* **by** *auto*
qed
qed
show *?thesis*
using *f4* **by** (*simp add*: *p q*)
qed

lemma *kleisli-left-monotonic*:
assumes ∀ *x*. *P x is* **N**
assumes *mono P*
shows *mono* (λ*X*. ↑(*P X*))
apply (*simp add*: *mono-def*, *auto*)
proof −
fix *x*::*'a* **and** *y*::*'a*
assume *a1*: $x \le y$
show ↑ (*P y*) ⊑ ↑ (*P x*)
apply (*subst kleisli-left-mono*)
using *a1 assms*(*2*) **apply** (*simp add*: *monoD*)
using *assms*(*1*) **by** *blast+*
qed

lemma *kleisli-left-H*:
assumes *P is* **H**
shows ↑*P is* **H**
by (*simp add*: *kleisli-lift2'-def kleisli-lift-alt-def ndesign-def rdesign-is-H1-H2*)

lemma *kleisli-left-N*:
assumes *P is* **N**
shows ↑*P is* **N**
apply (*simp add*: *kleisli-lift2'-def kleisli-lift-alt-def*)
using *ndesign-H1-H3* **by** *blast*

C.1.3 Recursion

C.2 Conditional Choice

declare [[*show-types*]]

lemma *cond-idem*:
  fixes P::$'s$ *hrel-pdes*
  shows $P \triangleleft b \triangleright_D P = P$
  by *auto*

lemma *cond-inf-distr*:
  fixes P::$'s$ *hrel-pdes* **and** Q::$'s$ *hrel-pdes* **and** R::$'s$ *hrel-pdes*
  shows $P \sqcap (Q \triangleleft b \triangleright_D R) = (P \sqcap Q) \triangleleft b \triangleright_D (P \sqcap R)$
  by (*rel-auto*)

C.3 Probabilistic Choice

lemma *prob-choice-inf-distr*:
  assumes $r \in \{0..1\}$ P *is* **N** Q *is* **N** R *is* **N**
  shows $(P \sqcap Q) \oplus_r R = ((P \oplus_r R) \sqcap (Q \oplus_r R))$ (**is** *?LHS* = *?RHS*)
proof −
  obtain pre_p $post_p$ pre_q $post_q$ pre_r $post_r$
    where p:$P = (pre_p \vdash_n post_p)$ **and**
      q:$Q = (pre_q \vdash_n post_q)$ **and**
      r:$R = (pre_r \vdash_n post_r)$
    using *assms* **by** (*metis ndesign-form*)
  hence *lhs*: $?LHS = ((pre_p \vdash_n post_p) \sqcap (pre_q \vdash_n post_q)) \oplus_r (pre_r \vdash_n post_r)$
    by *auto*
  have *rhs*: $?RHS = (((pre_p \vdash_n post_p) \oplus_r (pre_r \vdash_n post_r)) \sqcap ((pre_q \vdash_n post_q) \oplus_r (pre_r \vdash_n post_r)))$
    by (*simp add*: *p q r*)
  show *?thesis*
    apply (*simp add*: *p q r lhs rhs prob-choice-def*)
    apply (*ndes-simp cls*: *assms*)
    apply (*rel-auto*)
    apply *auto*[*1*]
    by *auto*
qed

lemma *prob-choice-inf-distl*:
  assumes $r \in \{0..1\}$ P *is* **N** Q *is* **N** R *is* **N**
  shows $P \oplus_r (Q \sqcap R) = ((P \oplus_r Q) \sqcap (P \oplus_r R))$ (**is** *?LHS* = *?RHS*)
proof −
  obtain pre_p $post_p$ pre_q $post_q$ pre_r $post_r$
    where p:$P = (pre_p \vdash_n post_p)$ **and**
      q:$Q = (pre_q \vdash_n post_q)$ **and**
      r:$R = (pre_r \vdash_n post_r)$
    using *assms* **by** (*metis ndesign-form*)
  hence *lhs*: $?LHS = ((pre_p \vdash_n post_p)) \oplus_r ((pre_q \vdash_n post_q) \sqcap (pre_r \vdash_n post_r))$
    by *auto*
  have *rhs*: $?RHS = (((pre_p \vdash_n post_p) \oplus_r (pre_q \vdash_n post_q)) \sqcap ((pre_p \vdash_n post_p) \oplus_r (pre_r \vdash_n post_r)))$
    by (*simp add*: *p q r*)
  show *?thesis*
    apply (*simp add*: *p q r lhs rhs prob-choice-def*)
    apply (*ndes-simp cls*: *assms*)
    apply (*rel-auto*)

apply *auto*[*1*]
by *auto*
qed

lemma *prob-choice-assoc*:
assumes $w_1 \in \{0..1\}$ $w_2 \in \{0..1\}$
$(1-w_1)*(1-w_2)=(1-r_2)$ $w_1=r_1*r_2$
P *is* **N** Q *is* **N** R *is* **N**
shows $(P \oplus_{w_1} (Q \oplus_{w_2} R)) = ((P \oplus_{r_1} Q) \oplus_{r_2} R)$ (**is** *?LHS* = *?RHS*)
proof −
obtain pre_p $post_p$ pre_q $post_q$ pre_r $post_r$
where p:$P = (pre_p \vdash_n post_p)$ **and**
q:$Q = (pre_q \vdash_n post_q)$ **and**
r:$R = (pre_r \vdash_n post_r)$
using *assms* **by** (*metis ndesign-form*)
hence *rhs*: $?RHS = ((pre_p \vdash_n post_p) \oplus_{r_1} (pre_q \vdash_n post_q)) \oplus_{r_2} (pre_r \vdash_n post_r)$
by *auto*
have *lhs*: $?LHS = (pre_p \vdash_n post_p) \oplus_{w_1} ((pre_q \vdash_n post_q) \oplus_{w_2} (pre_r \vdash_n post_r))$
by (*simp add*: *p q r*)
show *?thesis*
proof (*cases* $w_1 = 0 \lor w_1 = 1 \lor w_2 = 0 \lor w_2 = 1$)
case *True*
then show *?thesis*
proof (*cases* $w_1 = 0 \lor w_1 = 1$)
case *True*
then show *?thesis*
using *True prob-choice-one prob-choice-zero assms*(*3*−*4*)
by (*smt mult-cancel-left1 mult-cancel-right1 no-zero-divisors*)
next
case *False*
then show *?thesis*
using *False prob-choice-one prob-choice-zero assms*(*3*−*4*)
by (*smt True mult-cancel-left1 mult-cancel-right1*)
qed
next
case *False*
have *f1*: $w_1 \in \{0<..<1\}$
using *False assms*(*1*) **by** *auto*
have *f2*: $w_2 \in \{0<..<1\}$
using *False assms*(*2*) **by** *auto*
have *f3*: $(P \oplus_{w_1} (Q \oplus_{w_2} R)) = P \parallel^D{}_{\mathbf{PM}_{w_1}} (Q \parallel^D{}_{\mathbf{PM}_{w_2}} R)$
using *f1 f2* **by** (*simp add*: *prob-choice-r*)
from *assms*(*3*) **have** *f4*: $r_2 = w_1+w_2-w_1*w_2$
proof −
have *f1*: $\forall r\ ra.\ (ra::real) + - r = 0 \lor \neg\ ra = r$
by *simp*
have *f2*: $\forall r\ ra\ rb\ rc.\ (rc::real) \cdot rb + - (ra \cdot r) = rc \cdot (rb + - r) + (rc + - ra) \cdot r$
by (*simp add*: *mult-diff-mult*)
have *f3*: $\forall r\ ra.\ (ra::real) + (r + - ra) = r + 0$
by *fastforce*
have *f4*: $\forall r\ ra.\ (ra::real) + ra \cdot r = ra \cdot (1 + r)$
by (*simp add*: *distrib-left*)
have *f5*: $\forall r\ ra.\ (ra::real) + - r + 0 = ra + - r$
by *linarith*
have *f6*: $\forall r\ ra.\ (0::real) + (ra + - r) = ra + - r$

by *simp*
have $1 + - w_2 + - (w_1 \cdot (1 + - w_2)) = 1 + (0 + - r_2)$
using *f2 f1* **by** (*metis* (*no-types*) *add.left-commute add-uminus-conv-diff assms*(*3*) *mult.left-neutral*)
then have $1 + (w_1 + w_1 \cdot - w_2 + - r_2) = 1 + - w_2$
using *f6 f5 f4 f3* **by** (*metis* (*no-types*) *add.left-commute*)
then show *?thesis*
by *linarith*
qed
then have *f5*: $r_2 \in \{0<..<1\}$
using *f1 f2 assms*(*1*−*2*) *assms*(*3*) *f4*
by (*smt greaterThanLessThan-iff mult-left-le mult-nonneg-nonneg no-zero-divisors*)
from *f4* **have** *f6*: $(w_1+w_2-w_1*w_2) > w_1$
using *assms*(*1*) *assms*(*2*) *mult-left-le-one-le False* **by** *auto*
from *f4* **have** *f7*: $r_1 = w_1/(w_1+w_2-w_1*w_2)$
by (*metis False assms*(*4*) *mult-zero-right nonzero-eq-divide-eq*)
from *f6 f7* **have** *f8*: $r_1 \in \{0<..<1\}$
using *False f1 f2 assms*(*1*−*4*)
by (*metis divide-less-eq-1-pos f5 greaterThanLessThan-iff*
less-asym mult-zero-left nonzero-mult-div-cancel-left zero-less-divide-iff)
have *f9*: $((P \oplus_{r_1} Q) \oplus_{r_2} R) = (P \parallel^D{}_{\mathbf{PM}_{r_1}} Q) \parallel^D{}_{\mathbf{PM}_{r_2}} R$
using *f5 f8 f2* **by** (*simp add*: *prob-choice-r*)
show *?thesis*
apply (*simp add*: *f3 f9*)
apply (*simp add*: *p q r lhs rhs*)
apply (*ndes-simp cls*: *assms*)
apply (*rel-auto*)
apply (*metis assms*(*1*) *assms*(*2*) *assms*(*4*) *wplus-assoc*)
apply *blast*
apply (*metis assms*(*1*) *assms*(*2*) *assms*(*4*) *wplus-assoc*)
by *blast*
qed
qed

lemma *prob-choice-one'*:
assumes P *is* **N** Q *is* **N**
shows $(P \oplus_1 Q) = P$
by (*simp add*: *prob-choice-one*)

lemma *prob-choice-cond-distl*:
assumes $r \in \{0..1\}$ P *is* **N** Q *is* **N** R *is* **N**
shows $P \oplus_r (Q \triangleleft b \triangleright_D R) = ((P \oplus_r Q) \triangleleft b \triangleright_D (P \oplus_r R))$ (**is** *?LHS* = *?RHS*)
proof −
obtain pre_p $post_p$ pre_q $post_q$ pre_r $post_r$
where $p{:}P = (pre_p \vdash_n post_p)$ **and**
$q{:}Q = (pre_q \vdash_n post_q)$ **and**
$r{:}R = (pre_r \vdash_n post_r)$
using *assms* **by** (*metis ndesign-form*)
hence *lhs*: $?LHS = ((pre_p \vdash_n post_p)) \oplus_r ((pre_q \vdash_n post_q) \triangleleft b \triangleright_D (pre_r \vdash_n post_r))$
by *auto*
also have *lhs'*: $... = (pre_p \vdash_n post_p) \oplus_r (((pre_q \triangleleft b \triangleright pre_r) \vdash_n (post_q \triangleleft b \triangleright_r post_r)))$
by (*ndes-simp*)
have *rhs*: $?RHS = (((pre_p \vdash_n post_p) \oplus_r (pre_q \vdash_n post_q)) \triangleleft b \triangleright_D ((pre_p \vdash_n post_p) \oplus_r (pre_r \vdash_n post_r)))$
by (*simp add*: *p q r*)

show *?thesis*
  apply (*simp add*: *p q r lhs' rhs*)
  apply (*ndes-simp cls*: *assms*)
  by (*rel-auto*)
qed

C.3.1 UTP expression as weight

lemma *log-const-metasubt-eq*:
 assumes $\forall x.\ P\ x\ is\ \mathbf{N}$
 shows $(P\ r)[\![r \rightarrow \lceil \lceil E \rceil_< \rceil_D]\!] = (con_D\ R \cdot (II_D \triangleleft U(«R» = E) \triangleright_D \bot_D) \ ;;\ P\ R)$
proof –
 have *p*: $P\ r = (pre_D(P\ r) \vdash_r post_D(P\ r))$
  using *assms* **by** (*metis H1-H3-commute H1-H3-is-rdesign H3-idem Healthy-def*)
 have *f1*: $(pre_D(P\ r) \vdash_r post_D(P\ r))[\![r \rightarrow \lceil \lceil E \rceil_< \rceil_D]\!] = msubst\ (\lambda r.\ (pre_D(P\ r) \vdash_r post_D(P\ r)))\ \lceil \lceil E \rceil_< \rceil_D$
  by *simp*
 then have *f2*: ... = $msubst\ (\lambda r.\ P\ r)\ \lceil \lceil E \rceil_< \rceil_D$
  using *p* **apply** (*simp add*: *ext*)
  by (*metis* (*no-types*) *H1-H2-eq-rdesign H2-H3-absorb Healthy-def assms ndesign-form ndesign-is-H3*)
 have *f3*: $(pre_D(P\ r) \vdash_r post_D(P\ r))[\![r \rightarrow \lceil \lceil E \rceil_< \rceil_D]\!] =$
  $(con_D\ R \cdot (II_D \triangleleft U(«R» = E) \triangleright_D \bot_D) \ ;;\ (pre_D(P\ R) \vdash_r post_D(P\ R)))$
  by (*rel-auto*)
 show *?thesis*
  using *f1 f2 f3*
  by (*smt USUP-all-cong assms ndesign-def ndesign-form ndesign-pre*)
qed

lemma *log-const-metasubt-eq'*:
 shows $(P0 \vdash_n (P1\ r))[\![r \rightarrow \lceil \lceil E \rceil_< \rceil_D]\!] = (con_D\ R \cdot (II_D \triangleleft U(«R» = E) \triangleright_D \bot_D) \ ;;\ (P0 \vdash_n (P1\ R)))$
 apply (*ndes-simp*)
 by (*rel-auto*)

C.3.2 Assignment

C.4 Sequence

lemma *sequence-cond-distr*:
 assumes $P\ is\ \mathbf{N}\ Q\ is\ \mathbf{N}\ R\ is\ \mathbf{N}$
 shows $(P \triangleleft b \triangleright_D Q) \ ;;\ R\ = ((P \ ;;\ R) \triangleleft b \triangleright_D (Q \ ;;\ R))$ (**is** *?LHS = ?RHS*)
 by (*rel-auto*)

lemma *sequence-inf-distr*:
 assumes $P\ is\ \mathbf{N}\ Q\ is\ \mathbf{N}\ R\ is\ \mathbf{N}$
 shows $(P \sqcap Q) \ ;;\ R\ = ((P \ ;;\ R) \sqcap (Q \ ;;\ R))$ (**is** *?LHS = ?RHS*)
 by (*rel-auto*)

end

References

[1] J. He, C. Morgan, and A. McIver, "Deriving probabilistic semantics via the 'weakest completion'," in *Formal Methods and Software Engineering*, J. Davies, W. Schulte, and M. Barnett, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 131–145.

[2] J. C. P. Woodcock, A. L. C. Cavalcanti, S. Foster, A. Mota, and K. Ye, "Probabilistic semantics for RoboChart: A weakest completion approach," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science. Springer, 2019, p. to appear.

[3] C. C. Morgan, *Programming from Specifications.* Prentice-Hall, 1990.