A Formalization Of The Dutch Book Theorem

Matthew Doty

September 24, 2021

Contents

Chapter 1

Logical Foundations

The logical formulation of probability presented in §2 relies essentially on automated *classical propositional logic*. In order to provide this, we first develop an extensive theory of classical propositional logic up to completeness.

We first give the *pure implicational fragement of intuitionistic logic* as an elementary foundation. This is presented in §1.1. Following this will be referred to as *implication logic*.

Implication logic is extended to full *classical propositional logic* in §1.2. Completeness is presented in §1.3. Finally logical connectives are defined for classical logic in §1.5.

1.1 Implication Logic

theory *Implication-Logic*
  imports *Main*
begin

sledgehammer-params [*smt-proofs* = *false*]

This theory presents the pure implicational fragment of intuitionistic logic. That is to say, this is the fragment of intuitionistic logic containing *implication only*, and no other connectives nor *falsum*. It shall be referred to as *implication logic* in our discussions. For further reference see [24].

1.1.1 Axiomatization

Implication logic can be given by the a Hilbert-style axiom system, following Troelstra and Schwichtenberg [23, §1.3.9, pg. 33].

class *implication-logic* =
  fixes *deduction* :: $'a \Rightarrow bool$ ($\vdash$ - [*60*] *55*)

fixes *implication* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\rightarrow$ *70*)
assumes *axiom-k*: $\vdash \varphi \rightarrow \psi \rightarrow \varphi$
assumes *axiom-s*: $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
assumes *modus-ponens*: $\vdash \varphi \rightarrow \psi \Longrightarrow \vdash \varphi \Longrightarrow \vdash \psi$

1.1.2 Common Rules

lemma (**in** *implication-logic*) *trivial-implication*:
$\vdash \varphi \rightarrow \varphi$
by (*meson axiom-k axiom-s modus-ponens*)

lemma (**in** *implication-logic*) *flip-implication*:
$\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow \psi \rightarrow \varphi \rightarrow \chi$
by (*meson axiom-k axiom-s modus-ponens*)

lemma (**in** *implication-logic*) *hypothetical-syllogism*:
$\vdash (\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
by (*meson axiom-k axiom-s modus-ponens*)

lemma (**in** *implication-logic*) *flip-hypothetical-syllogism*:
$\vdash (\psi \rightarrow \varphi) \rightarrow (\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi)$
using *modus-ponens flip-implication hypothetical-syllogism* **by** *blast*

lemma (**in** *implication-logic*) *implication-absorption*:
$\vdash (\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \psi$
by (*meson axiom-k axiom-s modus-ponens*)

1.1.3 Lists of Assumptions

List Implication

Implication given a list of assumptions can be expressed recursively

primrec (**in** *implication-logic*)
list-implication :: $'a\ list \Rightarrow 'a \Rightarrow 'a$ (**infix** $:\rightarrow$ *80*) **where**
$[] :\rightarrow \varphi = \varphi$
$\mid (\psi \# \Psi) :\rightarrow \varphi = \psi \rightarrow \Psi :\rightarrow \varphi$

Deduction From a List of Assumptions

Deduction from a list of assumptions can be expressed in terms of $(:\rightarrow)$.

definition (**in** *implication-logic*) *list-deduction* :: $'a\ list \Rightarrow 'a \Rightarrow bool$ (**infix** $:\vdash$ *60*)
where
$\Gamma :\vdash \varphi \equiv \vdash \Gamma :\rightarrow \varphi$

List Deduction as Implication Logic

The relation $(:\vdash)$ may naturally be interpreted as a *deduction* predicate for an instance of implication logic for a fixed list of assumptions Γ.

Analogues of the two axioms of implication logic can be naturally stated using list implication.

lemma (**in** *implication-logic*) *list-implication-axiom-k*:
$\vdash \varphi \rightarrow \Gamma :\rightarrow \varphi$
by (*induct* Γ, (*simp*, *meson axiom-k axiom-s modus-ponens*)+)

lemma (**in** *implication-logic*) *list-implication-axiom-s*:
$\vdash \Gamma :\rightarrow (\varphi \rightarrow \psi) \rightarrow \Gamma :\rightarrow \varphi \rightarrow \Gamma :\rightarrow \psi$
by (*induct* Γ,
(*simp*, *meson axiom-k axiom-s modus-ponens hypothetical-syllogism*)+)

The lemmas $\vdash\ ?\varphi \rightarrow\ ?\Gamma :\rightarrow\ ?\varphi$ and $\vdash\ ?\Gamma :\rightarrow (?\varphi \rightarrow\ ?\psi) \rightarrow\ ?\Gamma :\rightarrow\ ?\varphi \rightarrow\ ?\Gamma :\rightarrow\ ?\psi$ jointly give rise to an interpretation of implication logic, where a list of assumptions Γ play the role of a *background theory* of $(:\vdash)$.

context *implication-logic* **begin**
interpretation *list-deduction-logic*:
implication-logic $\lambda\ \varphi.\ \Gamma :\vdash \varphi\ (\rightarrow)$
proof qed
(*meson*
list-deduction-def
axiom-k
axiom-s
modus-ponens
list-implication-axiom-k
list-implication-axiom-s)+
end

The following *weakening* rule can also be derived.

lemma (**in** *implication-logic*) *list-deduction-weaken*:
$\vdash \varphi \Longrightarrow \Gamma :\vdash \varphi$
unfolding *list-deduction-def*
using *modus-ponens list-implication-axiom-k*
by *blast*

In the case of the empty list, the converse may be established.

lemma (**in** *implication-logic*) *list-deduction-base-theory* [*simp*]:
$[] :\vdash \varphi \equiv \vdash \varphi$
unfolding *list-deduction-def*
by *simp*

lemma (**in** *implication-logic*) *list-deduction-modus-ponens*:
$\Gamma :\vdash \varphi \rightarrow \psi \Longrightarrow \Gamma :\vdash \varphi \Longrightarrow \Gamma :\vdash \psi$
unfolding *list-deduction-def*
using *modus-ponens list-implication-axiom-s*
by *blast*

1.1.4 The Deduction Theorem

One result in the meta-theory of implication logic is the *deduction theorem*, which is a mechanism for moving antecedents back and forth from collections of assumptions.

To develop the deduction theorem, the following two lemmas generalize $\vdash (?\varphi \to ?\psi \to ?\chi) \to ?\psi \to ?\varphi \to ?\chi$.

lemma (**in** *implication-logic*) *list-flip-implication1*:
$\vdash (\varphi \# \Gamma) :\to \chi \to \Gamma :\to (\varphi \to \chi)$
by (*induct* Γ,
(*simp*,
meson
axiom-k
axiom-s
modus-ponens
flip-implication
hypothetical-syllogism)+)

lemma (**in** *implication-logic*) *list-flip-implication2*:
$\vdash \Gamma :\to (\varphi \to \chi) \to (\varphi \# \Gamma) :\to \chi$
by (*induct* Γ,
(*simp*,
meson
axiom-k
axiom-s
modus-ponens
flip-implication
hypothetical-syllogism)+)

Together the two lemmas above suffice to prove a form of the deduction theorem:

theorem (**in** *implication-logic*) *list-deduction-theorem*:
$(\varphi \# \Gamma) :\vdash \psi = \Gamma :\vdash \varphi \to \psi$
unfolding *list-deduction-def*
by (*metis modus-ponens list-flip-implication1 list-flip-implication2*)

1.1.5 Monotonic Growth in Deductive Power

In logic, for two sets of assumptions Φ and Ψ, if $\Psi \subseteq \Phi$ then the latter theory Φ is said to be *stronger* than former theory Ψ. In principle, anything a weaker theory can prove a stronger theory can prove. One way of saying this is that deductive power increases monotonically with as the set of underlying assumptions grow.

The monotonic growth of deductive power can be expressed as a meta-theorem in implication logic.

The lemma $\vdash\ ?\Gamma :\rightarrow (?\varphi \rightarrow ?\chi) \rightarrow (?\varphi \# ?\Gamma) :\rightarrow ?\chi$ presents a means of *introducing* assumptions into a list of assumptions when those assumptions have arrived at by an implication. The next lemma presents a means of *discharging* those assumptions, which can be used in the monotonic growth theorem to be proved.

lemma (**in** *implication-logic*) *list-implication-removeAll*:
$\vdash \Gamma :\rightarrow \psi \rightarrow (\textit{removeAll}\ \varphi\ \Gamma) :\rightarrow (\varphi \rightarrow \psi)$
proof −
  have $\forall\ \psi.\ \vdash \Gamma :\rightarrow \psi \rightarrow (\textit{removeAll}\ \varphi\ \Gamma) :\rightarrow (\varphi \rightarrow \psi)$
  proof(*induct* Γ)
    case *Nil*
    then show *?case* **by** (*simp*, *meson axiom-k*)
  next
    case (*Cons* χ Γ)
    assume
      inductive-hypothesis: $\forall\ \psi.\ \vdash \Gamma :\rightarrow \psi \rightarrow \textit{removeAll}\ \varphi\ \Gamma :\rightarrow (\varphi \rightarrow \psi)$
    moreover {
      assume $\varphi \neq \chi$
      with *inductive-hypothesis*
      have $\forall\ \psi.\ \vdash (\chi \# \Gamma) :\rightarrow \psi \rightarrow \textit{removeAll}\ \varphi\ (\chi \# \Gamma) :\rightarrow (\varphi \rightarrow \psi)$
        by (*simp*, *meson modus-ponens hypothetical-syllogism*)
    }
    moreover {
      fix ψ
      assume *φ-equals-χ*: $\varphi = \chi$
      moreover with *inductive-hypothesis*
      have $\vdash \Gamma :\rightarrow (\chi \rightarrow \psi) \rightarrow \textit{removeAll}\ \varphi\ (\chi \# \Gamma) :\rightarrow (\varphi \rightarrow \chi \rightarrow \psi)$ **by** *simp*
      hence $\vdash \Gamma :\rightarrow (\chi \rightarrow \psi) \rightarrow \textit{removeAll}\ \varphi\ (\chi \# \Gamma) :\rightarrow (\varphi \rightarrow \psi)$
        by (*metis*
            calculation
            modus-ponens
            implication-absorption
            list-flip-implication1
            list-flip-implication2
            list-implication.simps(*2*))
      ultimately have $\vdash (\chi \# \Gamma) :\rightarrow \psi \rightarrow \textit{removeAll}\ \varphi\ (\chi \# \Gamma) :\rightarrow (\varphi \rightarrow \psi)$
        by (*simp*,
            metis
              modus-ponens
              hypothetical-syllogism
              list-flip-implication1
              list-implication.simps(*2*))
    }
    ultimately show *?case* **by** *simp*
  qed
  thus *?thesis* **by** *blast*
qed

From lemma above presents what is needed to prove that deductive power for lists is monotonic.

```
theorem (in implication-logic) list-implication-monotonic:
  set Σ ⊆ set Γ ⟹ ⊢ Σ :→ φ → Γ :→ φ
proof −
  assume set Σ ⊆ set Γ
  moreover have ∀ Σ φ. set Σ ⊆ set Γ ⟶ ⊢ Σ :→ φ → Γ :→ φ
  proof(induct Γ)
    case Nil
    then show ?case
      by (metis
            list-implication.simps(1)
            list-implication-axiom-k
            set-empty
            subset-empty)
  next
    case (Cons ψ Γ)
    assume
      inductive-hypothesis: ∀Σ φ. set Σ ⊆ set Γ ⟶ ⊢ Σ :→ φ → Γ :→ φ
    {
      fix Σ
      fix φ
      assume Σ-subset-relation: set Σ ⊆ set (ψ # Γ)
      have ⊢ Σ :→ φ → (ψ # Γ) :→ φ
      proof −
        {
          assume set Σ ⊆ set Γ
          hence ?thesis
            by (metis
                  inductive-hypothesis
                  axiom-k modus-ponens
                  flip-implication
                  list-implication.simps(2))
        }
        moreover {
          let ?Δ = removeAll ψ Σ
          assume ¬ (set Σ ⊆ set Γ)
          hence set ?Δ ⊆ set Γ
            using Σ-subset-relation by auto
          hence ⊢ ?Δ :→ (ψ → φ) → Γ :→ (ψ → φ)
            using inductive-hypothesis by auto
          hence ⊢ ?Δ :→ (ψ → φ) → (ψ # Γ) :→ φ
            by (metis
                  modus-ponens
                  flip-implication
                  list-flip-implication2
                  list-implication.simps(2))
          moreover have ⊢ Σ :→ φ → ?Δ :→ (ψ → φ)
            by (simp add: local.list-implication-removeAll)
          ultimately have ?thesis
```

using *modus-ponens hypothetical-syllogism* **by** *blast*
}
ultimately show *?thesis* **by** *blast*
qed
}
thus *?case* **by** *simp*
qed
ultimately show *?thesis* **by** *simp*
qed

A direct consequence is that deduction from lists of assumptions is monotonic as well:

theorem (**in** *implication-logic*) *list-deduction-monotonic*:
$set\ \Sigma \subseteq set\ \Gamma \Longrightarrow \Sigma :\vdash \varphi \Longrightarrow \Gamma :\vdash \varphi$
unfolding *list-deduction-def*
using *modus-ponens list-implication-monotonic*
by *blast*

1.1.6 The Deduction Theorem Revisited

The monotonic nature of deduction allows us to prove another form of the deduction theorem, where the assumption being discharged is completely removed from the list of assumptions.

theorem (**in** *implication-logic*) *alternate-list-deduction-theorem*:
$(\varphi \# \Gamma) :\vdash \psi = (removeAll\ \varphi\ \Gamma) :\vdash \varphi \to \psi$
by (*metis*
list-deduction-def
modus-ponens
filter-is-subset
list-deduction-monotonic
list-deduction-theorem
list-implication-removeAll
removeAll.simps(2)
removeAll-filter-not-eq)

1.1.7 Reflection

In logic the *reflection* principle sometimes refers to when a collection of assumptions can deduce any of its members. It is automatically derivable from $[\![set\ ?\Sigma \subseteq set\ ?\Gamma;\ ?\Sigma :\vdash\ ?\varphi]\!] \Longrightarrow\ ?\Gamma :\vdash\ ?\varphi$ among the other rules provided.

lemma (**in** *implication-logic*) *list-deduction-reflection*:
$\varphi \in set\ \Gamma \Longrightarrow \Gamma :\vdash \varphi$
by (*metis*
list-deduction-def
insert-subset
list.simps(15)

list-deduction-monotonic
list-implication.simps(2)
list-implication-axiom-k
order-refl)

1.1.8 The Cut Rule

Cut is a rule commonly presented in sequent calculi, dating back to Gerhard Gentzen's *Investigations in Logical Deduction* (1935) [12]

The cut rule is not generally necessary in sequent calculi. It can often be shown that the rule can be eliminated without reducing the power of the underlying logic. However, as demonstrated by George Boolos' *Don't Eliminate Cute* (1984) [6], removing the rule can often lead to very inefficient proof systems.

Here the rule is presented just as a meta theorem.

theorem (**in** *implication-logic*) *list-deduction-cut-rule*:
$(\varphi \ \# \ \Gamma) :\vdash \psi \Longrightarrow \Delta :\vdash \varphi \Longrightarrow \Gamma \ @ \ \Delta :\vdash \psi$
by (*metis* (*no-types*, *lifting*)
Un-upper1
Un-upper2
list-deduction-modus-ponens
list-deduction-monotonic
list-deduction-theorem
set-append)

The cut rule can also be strengthened to entire lists of propositions.

theorem (**in** *implication-logic*) *strong-list-deduction-cut-rule*:
$(\Phi \ @ \ \Gamma) :\vdash \psi \Longrightarrow \forall \ \varphi \in set \ \Phi. \ \Delta :\vdash \varphi \Longrightarrow \Gamma \ @ \ \Delta :\vdash \psi$
proof −
have $\forall \ \psi. \ (\Phi \ @ \ \Gamma :\vdash \psi \longrightarrow (\forall \ \varphi \in set \ \Phi. \ \Delta :\vdash \varphi) \longrightarrow \Gamma \ @ \ \Delta :\vdash \psi)$
proof(*induct* Φ)
case *Nil*
then show *?case*
by (*metis*
Un-iff
append.left-neutral
list-deduction-monotonic
set-append
subsetI)
next
case (*Cons* χ Φ) **assume** *inductive-hypothesis*:
$\forall \ \psi. \ \Phi \ @ \ \Gamma :\vdash \psi \longrightarrow (\forall \varphi \in set \ \Phi. \ \Delta :\vdash \varphi) \longrightarrow \Gamma \ @ \ \Delta :\vdash \psi$
{
fix ψ χ
assume $(\chi \ \# \ \Phi) \ @ \ \Gamma :\vdash \psi$
hence *A*: $\Phi \ @ \ \Gamma :\vdash \chi \rightarrow \psi$ **using** *list-deduction-theorem* **by** *auto*

assume $\forall\varphi \in$ *set* $(\chi \# \Phi)$. Δ :⊢ φ
hence *B*: $\forall$ $\varphi \in$ *set* Φ. Δ :⊢ φ
and *C*: Δ :⊢ χ **by** *auto*
from *A B* **have** Γ @ Δ :⊢ $\chi \to \psi$ **using** *inductive-hypothesis* **by** *blast*
with *C* **have** Γ @ Δ :⊢ ψ
by (*meson*
list.set-intros(*1*)
list-deduction-cut-rule
list-deduction-modus-ponens
list-deduction-reflection)
}
thus *?case* **by** *simp*
qed
moreover assume $(\Phi$ @ $\Gamma)$:⊢ ψ
moreover assume $\forall$ $\varphi \in$ *set* Φ. Δ :⊢ φ
ultimately show *?thesis* **by** *blast*
qed

1.1.9 Sets of Assumptions

While deduction in terms of lists of assumptions is straight-forward to define, deduction (and the *deduction theorem*) is commonly given in terms of *sets* of propositions. This formulation is suited to establishing strong completeness theorems and compactness theorems.

The presentation of deduction from a set follows the presentation of list deduction given for (:⊢).

1.1.10 Definition of Deduction

Just as deduction from a list (:⊢) can be defined in terms of (:→), deduction from a *set* of assumptions can be expressed in terms of (:⊢).

definition (**in** *implication-logic*) *set-deduction* :: *'a set* ⇒ *'a* ⇒ *bool* (**infix** ⊩ *60*)
where
$\Gamma \Vdash \varphi \equiv \exists\ \Psi.\ set(\Psi) \subseteq \Gamma \wedge \Psi$:⊢ φ

Interpretation as Implication Logic

As in the case of (:⊢), the relation (⊩) may be interpreted as *deduction* predicate for a fixed set of assumptions Γ.

The following lemma is given in order to establish this, which asserts that every implication logic tautology $\vdash \varphi$ is also a tautology for $\Gamma \Vdash \varphi$.

lemma (**in** *implication-logic*) *set-deduction-weaken*:
$\vdash \varphi \Longrightarrow \Gamma \Vdash \varphi$
using *list-deduction-base-theory set-deduction-def* **by** *fastforce*

In the case of the empty set, the converse may be established.

lemma (**in** *implication-logic*) *set-deduction-base-theory*:
  $\{\} \Vdash \varphi \equiv \vdash \varphi$
  using *list-deduction-base-theory set-deduction-def* **by** *auto*

Next, a form of *modus ponens* is provided for $(\Vdash)$.

lemma (**in** *implication-logic*) *set-deduction-modus-ponens*:
  $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$
proof –
  assume $\Gamma \Vdash \varphi \to \psi$
  then obtain Φ **where** *A*: *set* $\Phi \subseteq \Gamma$ **and** *B*: $\Phi :\vdash \varphi \to \psi$
    using *set-deduction-def* **by** *blast*
  assume $\Gamma \Vdash \varphi$
  then obtain Ψ **where** *C*: *set* $\Psi \subseteq \Gamma$ **and** *D*: $\Psi :\vdash \varphi$
    using *set-deduction-def* **by** *blast*
  from *B D* **have** Φ @ $\Psi :\vdash \psi$
    using *list-deduction-cut-rule list-deduction-theorem* **by** *blast*
  moreover from *A C* **have** *set* $(\Phi$ @ $\Psi) \subseteq \Gamma$ **by** *simp*
  ultimately show *?thesis*
    using *set-deduction-def* **by** *blast*
qed

context *implication-logic* **begin**
interpretation *set-deduction-logic*:
  implication-logic $\lambda\ \varphi.\ \Gamma \Vdash \varphi\ (\to)$
proof
   fix $\varphi\ \psi$
   show $\Gamma \Vdash \varphi \to \psi \to \varphi$ **by** (*metis axiom-k set-deduction-weaken*)
next
    fix $\varphi\ \psi\ \chi$
    show $\Gamma \Vdash (\varphi \to \psi \to \chi) \to (\varphi \to \psi) \to \varphi \to \chi$
      by (*metis axiom-s set-deduction-weaken*)
next
    fix $\varphi\ \psi$
    show $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$
      using *set-deduction-modus-ponens* **by** *metis*
qed
end

1.1.11 The Deduction Theorem

The next result gives the deduction theorem for $(\Vdash)$.

theorem (**in** *implication-logic*) *set-deduction-theorem*:
  insert $\varphi\ \Gamma \Vdash \psi = \Gamma \Vdash \varphi \to \psi$
proof –
  have $\Gamma \Vdash \varphi \to \psi \Longrightarrow$ *insert* $\varphi\ \Gamma \Vdash \psi$
    by (*metis*
        set-deduction-def

insert-mono
list.simps(*15*)
list-deduction-theorem)
moreover {
assume *insert* φ $\Gamma \Vdash \psi$
then obtain Φ **where** *set* $\Phi \subseteq$ *insert* φ Γ **and** Φ $:\vdash \psi$
using *set-deduction-def* **by** *auto*
hence *set* (*removeAll* φ Φ) $\subseteq \Gamma$ **by** *auto*
moreover from ‹$\Phi :\vdash \psi$› **have** *removeAll* φ Φ $:\vdash \varphi \to \psi$
using *modus-ponens list-implication-removeAll list-deduction-def*
by *blast*
ultimately have $\Gamma \Vdash \varphi \to \psi$
using *set-deduction-def* **by** *blast*
}
ultimately show *insert* φ $\Gamma \Vdash \psi = \Gamma \Vdash \varphi \to \psi$ **by** *metis*
qed

1.1.12 Monotonic Growth in Deductive Power

In contrast to the ($:\vdash$) relation, the proof that the deductive power of ($\Vdash$) grows monotonically with its assumptions may be fully automated.

theorem *set-deduction-monotonic*:
$\Sigma \subseteq \Gamma \Longrightarrow \Sigma \Vdash \varphi \Longrightarrow \Gamma \Vdash \varphi$
by (*meson dual-order.trans set-deduction-def*)

1.1.13 The Deduction Theorem Revisited

As a consequence of the fact that $[\![?\Sigma \subseteq ?\Gamma;\ ?\Sigma \Vdash ?\varphi]\!] \Longrightarrow ?\Gamma \Vdash ?\varphi$ is automatically provable, an alternate *deduction theorem* where the discharged assumption is completely removed from the set of assumptions is just a consequence of the more conventional *insert* $?\varphi$ $?\Gamma \Vdash ?\psi = ?\Gamma \Vdash ?\varphi \to ?\psi$ rule and some basic set identities.

theorem (**in** *implication-logic*) *alternate-set-deduction-theorem*:
insert φ $\Gamma \Vdash \psi = \Gamma - \{\varphi\} \Vdash \varphi \to \psi$
by (*metis insert-Diff-single set-deduction-theorem*)

1.1.14 Reflection

Just as in the case of ($:\vdash$), deduction from sets of assumptions makes true the *reflection principle* and is automatically provable.

theorem (**in** *implication-logic*) *set-deduction-reflection*:
$\varphi \in \Gamma \Longrightarrow \Gamma \Vdash \varphi$
by (*metis*
Set.set-insert
list-implication.simps(*1*)
list-implication-axiom-k
set-deduction-theorem

set-deduction-weaken)

1.1.15 The Cut Rule

The final principle of ($\Vdash$) presented is the *cut rule*.

First, the weak form of the rule is established.

theorem (**in** *implication-logic*) *set-deduction-cut-rule*:
  insert φ $\Gamma \Vdash \psi \Longrightarrow \Delta \Vdash \varphi \Longrightarrow \Gamma \cup \Delta \Vdash \psi$
proof −
  assume *insert* φ $\Gamma \Vdash \psi$
  hence $\Gamma \Vdash \varphi \to \psi$ **using** *set-deduction-theorem* **by** *auto*
  hence $\Gamma \cup \Delta \Vdash \varphi \to \psi$ **using** *set-deduction-def* **by** *auto*
  moreover assume $\Delta \Vdash \varphi$
  hence $\Gamma \cup \Delta \Vdash \varphi$ **using** *set-deduction-def* **by** *auto*
  ultimately show *?thesis* **using** *set-deduction-modus-ponens* **by** *metis*
qed

Another lemma is shown next in order to establish the strong form of the cut rule. The lemma shows the existence of a *covering list* of assumptions Ψ in the event some set of assumptions Δ proves everything in a finite set of assumptions Φ.

lemma (**in** *implication-logic*) *finite-set-deduction-list-deduction*:
  assumes *finite* Φ
  and $\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi$
  shows $\exists\,\Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall\varphi \in \Phi.\ \Psi :\vdash \varphi)$
  using *assms*
proof(*induct* Φ *rule*: *finite-induct*)
  case *empty* **thus** *?case* **by** (*metis all-not-in-conv empty-subsetI set-empty*)
next
  case (*insert* χ Φ)
  assume $\forall\varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow \exists\,\Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall\varphi \in \Phi.\ \Psi :\vdash \varphi)$
    and $\forall\varphi \in$ *insert* χ $\Phi.\ \Delta \Vdash \varphi$
  hence $\exists\,\Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall\varphi{\in}\Phi.\ \Psi :\vdash \varphi)$ **and** $\Delta \Vdash \chi$ **by** *simp+*
  then obtain Ψ_1 Ψ_2 **where**
    set $(\Psi_1\ @\ \Psi_2) \subseteq \Delta$
    $\forall\varphi \in \Phi.\ \Psi_1 :\vdash \varphi$
    $\Psi_2 :\vdash \chi$
    using *set-deduction-def* **by** *auto*
  moreover from *this* **have** $\forall\varphi \in$ (*insert* χ Φ). $\Psi_1\ @\ \Psi_2 :\vdash \varphi$
    by (*metis*
        insert-iff
        le-sup-iff
        list-deduction-monotonic
        order-refl set-append)
  ultimately show *?case* **by** *blast*
qed

With $[\![finite\ ?\Phi;\ \forall\varphi\in?\Phi.\ ?\Delta \Vdash \varphi]\!] \Longrightarrow \exists\Psi.\ set\ \Psi \subseteq\ ?\Delta \wedge (\forall\varphi\in?\Phi.\ \Psi :\vdash \varphi)$ the strengthened form of the cut rule can be given.

theorem (**in** *implication-logic*) *strong-set-deduction-cut-rule*:
  assumes $\Phi \cup \Gamma \Vdash \psi$
  and $\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi$
  shows $\Gamma \cup \Delta \Vdash \psi$
proof −
  obtain Σ **where**
    A: *set* $\Sigma\ \subseteq \Phi \cup \Gamma$ **and**
    B: $\Sigma :\vdash \psi$
    using *assms*(*1*) *set-deduction-def*
    by *auto*+
  obtain $\Phi'\ \Gamma'$ **where**
    C: *set* $\Phi' =$ *set* $\Sigma \cap \Phi$ **and**
    D: *set* $\Gamma' =$ *set* $\Sigma \cap \Gamma$
    by (*metis inf-sup-aci*(*1*) *inter-set-filter*)+
  then have *set* $(\Phi'\ @\ \Gamma') =$ *set* Σ **using** *A* **by** *auto*
  hence *E*: $\Phi'\ @\ \Gamma' :\vdash \psi$ **using** *B list-deduction-monotonic* **by** *blast*
  hence $\forall\ \varphi \in$ *set* $\Phi'.\ \Delta \Vdash \varphi$ **using** *assms*(*2*) *C* **by** *auto*
  from *this* **obtain** Δ' **where** *set* $\Delta' \subseteq \Delta$ **and** $\forall\ \varphi \in$ *set* $\Phi'.\ \Delta' :\vdash \varphi$
    using *finite-set-deduction-list-deduction* **by** *blast*
  with *strong-list-deduction-cut-rule D E*
  have *set* $(\Gamma'\ @\ \Delta') \subseteq \Gamma \cup \Delta$ **and** $\Gamma'\ @\ \Delta' :\vdash \psi$ **by** *auto*
  thus *?thesis* **using** *set-deduction-def* **by** *blast*
qed

1.1.16 Maximally Consistent Sets For Implication Logic

Maximally Consistent Sets are a common construction for proving completeness of logical calculi. For a classic presentation, see Dirk van Dalen's *Logic and Structure* (2013, §1.5, pgs. 42–45) [25].

Maximally consistent sets will form the foundation of all of the model theory we will employ in this text. In fact, apart from classical logic semantics, conventional model theory will not be used at all.

The models we are centrally concerned are derived from maximally consistent sets. These include probability measures used in completeness theorems of probability logic found in §2.7, as well as arbitrage opportunities stipulated by the *Dutch Book Theorem* in §3.2.

Since implication logic does not have *falsum*, consistency is defined relative to a formula φ.

definition (**in** *implication-logic*)
  formula-consistent :: $'a \Rightarrow\ 'a\ set \Rightarrow bool$ (*-−consistent* - [*100*] *100*)
  where
    [*simp*]: φ−*consistent* $\Gamma \equiv \neg\ (\Gamma \Vdash \varphi)$

Since consistency is defined relative to some φ, *maximal consistency* is presented as asserting that either ψ or $\psi \to \varphi$ is in the consistent set Γ, for all ψ. This coincides with the traditional definition in classical logic when φ is *falsum*.

definition (**in** *implication-logic*)
formula-maximally-consistent-set-def :: $'a \Rightarrow {'a}\ set \Rightarrow bool$ (-—*MCS* - [*100*] *100*)
where
[*simp*]: φ—*MCS* $\Gamma \equiv$ (φ—*consistent* Γ) $\wedge$ ($\forall\ \psi.\ \psi \in \Gamma \vee (\psi \to \varphi) \in \Gamma$)

Every consistent set Γ may be extended to a maximally consistent set.

However, no assumption is made regarding the cardinality of the types of an instance of *implication-logic*.

As a result, typical proofs that assume a countable domain are not suitable. Our proof leverages *Zorn's lemma*.

lemma (**in** *implication-logic*) *formula-consistent-extension*:
assumes φ—*consistent* Γ
shows (φ—*consistent* (*insert* ψ Γ)) $\vee$ (φ—*consistent* (*insert* ($\psi \to \varphi$) Γ))
proof –
{
assume $\neg$ φ—*consistent insert* ψ Γ
hence $\Gamma \Vdash \psi \to \varphi$
using *set-deduction-theorem*
unfolding *formula-consistent-def*
by *simp*
hence φ—*consistent insert* ($\psi \to \varphi$) Γ
by (*metis Un-absorb assms formula-consistent-def set-deduction-cut-rule*)
}
thus *?thesis* **by** *blast*
qed

theorem (**in** *implication-logic*) *formula-maximally-consistent-extension*:
assumes φ—*consistent* Γ
shows $\exists\ \Omega$. (φ—*MCS* Ω) $\wedge$ $\Gamma \subseteq \Omega$
proof –
let *?*Γ*-extensions* = $\{\Sigma$. (φ—*consistent* Σ) $\wedge$ $\Gamma \subseteq \Sigma\}$
have $\exists\ \Omega \in$ *?*Γ*-extensions*. $\forall \Sigma \in$ *?*Γ*-extensions*. $\Omega \subseteq \Sigma \longrightarrow \Sigma = \Omega$
proof (*rule subset-Zorn*)
fix $\mathcal{C}$ :: $'a\ set\ set$
assume *subset-chain-*$\mathcal{C}$: *subset.chain ?*Γ*-extensions* $\mathcal{C}$
hence $\mathcal{C}$: $\forall\ \Sigma \in \mathcal{C}.\ \Gamma \subseteq \Sigma\ \forall\ \Sigma \in \mathcal{C}.$ φ—*consistent* Σ
unfolding *subset.chain-def*
by *blast+*
show $\exists\ \Omega \in$ *?*Γ*-extensions*. $\forall\ \Sigma \in \mathcal{C}.\ \Sigma \subseteq \Omega$
proof *cases*
assume $\mathcal{C} = \{\}$ **thus** *?thesis* **using** *assms* **by** *blast*
next
let *?*$\Omega = \bigcup\ \mathcal{C}$

assume $\mathcal{C} \neq \{\}$
hence $\Gamma \subseteq ?\Omega$ **by** (*simp add*: $\mathcal{C}(1)$ *less-eq-Sup*)
moreover have φ−*consistent* $?\Omega$
proof −
  {
    assume $\neg$ φ−*consistent* $?\Omega$
    then obtain ω **where** ω:
      finite ω
      $\omega \subseteq ?\Omega$
      $\neg$ φ−*consistent* ω
      unfolding
        formula-consistent-def
        set-deduction-def
      by *auto*
    from $\omega(1)$ $\omega(2)$ **have** $\exists\ \Sigma \in \mathcal{C}.\ \omega \subseteq \Sigma$
    proof (*induct* ω *rule*: *finite-induct*)
      case *empty* **thus** *?case* **using** ‹$\mathcal{C} \neq \{\}$› **by** *blast*
    next
      case (*insert* ψ ω)
      from *this* **obtain** Σ_1 Σ_2 **where**
        Σ_1:
          $\omega \subseteq \Sigma_1$
          $\Sigma_1 \in \mathcal{C}$
        and Σ_2:
          $\psi \in \Sigma_2$
          $\Sigma_2 \in \mathcal{C}$
        by *auto*
      hence $\Sigma_1 \subseteq \Sigma_2 \vee \Sigma_2 \subseteq \Sigma_1$
        using *subset-chain-*$\mathcal{C}$
        unfolding *subset.chain-def*
        by *blast*
      hence (*insert* ψ ω) $\subseteq \Sigma_1 \vee$ (*insert* ψ ω) $\subseteq \Sigma_2$
        using Σ_1 Σ_2 **by** *blast*
      thus *?case* **using** Σ_1 Σ_2 **by** *blast*
    qed
    hence $\exists\ \Sigma \in \mathcal{C}.$ (φ−*consistent* Σ) $\wedge$ $\neg$ (φ−*consistent* Σ)
      using $\mathcal{C}(2)$ $\omega(3)$
      unfolding
        formula-consistent-def
        set-deduction-def
      by *auto*
    hence *False* **by** *auto*
  }
  thus *?thesis* **by** *blast*
qed
ultimately show *?thesis* **by** *blast*
qed
qed
then obtain Ω **where** Ω:

$\Omega \in$ *?*Γ-*extensions*
$\forall \Sigma \in$ *?*Γ-*extensions*. $\Omega \subseteq \Sigma \longrightarrow \Sigma = \Omega$
by *auto+*
{
fix ψ
have ($\varphi-$*consistent insert* ψ Ω) $\vee$ ($\varphi-$*consistent insert* ($\psi \to \varphi$) Ω)
$\Gamma \subseteq$ *insert* ψ Ω
$\Gamma \subseteq$ *insert* ($\psi \to \varphi$) Ω
using Ω(*1*) *formula-consistent-extension formula-consistent-def*
by *auto*
hence *insert* ψ $\Omega \in$ *?*Γ-*extensions*
$\vee$ *insert* ($\psi \to \varphi$) $\Omega \in$ *?*Γ-*extensions*
by *blast*
hence $\psi \in \Omega \vee (\psi \to \varphi) \in \Omega$ **using** Ω(*2*) **by** *blast*
}
thus *?thesis*
using Ω(*1*)
unfolding *formula-maximally-consistent-set-def-def*
by *blast*
qed

Finally, maximally consistent sets contain anything that can be deduced from them, and model a form of *modus ponens*.

lemma (**in** *implication-logic*) *formula-maximally-consistent-set-def-reflection*:
$\varphi-$*MCS* $\Gamma \Longrightarrow \psi \in \Gamma = \Gamma \Vdash \psi$
proof $-$
assume $\varphi-$*MCS* Γ
{
assume $\Gamma \Vdash \psi$
moreover from ‹$\varphi-$*MCS* Γ› **have** $\psi \in \Gamma \vee (\psi \to \varphi) \in \Gamma$ $\neg$ $\Gamma \Vdash \varphi$
unfolding
formula-maximally-consistent-set-def-def
formula-consistent-def
by *auto*
ultimately have $\psi \in \Gamma$
using *set-deduction-reflection set-deduction-modus-ponens*
by *metis*
}
thus $\psi \in \Gamma = \Gamma \Vdash \psi$
using *set-deduction-reflection*
by *metis*
qed

theorem (**in** *implication-logic*) *formula-maximally-consistent-set-def-implication-elimination*:
assumes $\varphi-$*MCS* Ω
shows $(\psi \to \chi) \in \Omega \Longrightarrow \psi \in \Omega \Longrightarrow \chi \in \Omega$
using
assms
formula-maximally-consistent-set-def-reflection

set-deduction-modus-ponens
by *blast*

This concludes our introduction to implication logic.

end

1.2 Classical Propositional Logic

theory *Classical-Logic*
imports ../*Intuitionistic*/*Implication-Logic*
begin

sledgehammer-params [*smt-proofs* = *false*]

This theory presents *classical propositional logic*, which is classical logic without quantifiers.

1.2.1 Axiomatization

Classical propositional logic can be given by the following Hilbert-style axiom system. It is *implication-logic* extended with *falsum* and double negation.

class *classical-logic* = *implication-logic* +
fixes *falsum* :: $'a$ ($\bot$)
assumes *double-negation*: $\vdash (((\varphi \to \bot) \to \bot) \to \varphi)$

In some cases it is useful to assume consistency as an axiom:

class *consistent-classical-logic* = *classical-logic* +
assumes *consistency*: $\neg \vdash \bot$

1.2.2 Common Rules

There are many common tautologies in classical logic. Once we have established *completeness* in §1.3, we will be able to leverage Isabelle/HOL's automation for proving these elementary results.

In order to bootstrap completeness, we develop some common lemmas using classical deduction alone.

lemma (**in** *classical-logic*)
ex-falso-quodlibet: $\vdash \bot \to \varphi$
using *axiom-k double-negation modus-ponens hypothetical-syllogism*
by *blast*

lemma (**in** *classical-logic*)
Contraposition: $\vdash ((\varphi \to \bot) \to (\psi \to \bot)) \to \psi \to \varphi$
proof −

have $[\varphi \to \bot, \psi, (\varphi \to \bot) \to (\psi \to \bot)] :\vdash \bot$
using *flip-implication list-deduction-theorem list-implication.simps(1)*
unfolding *list-deduction-def*
by *presburger*
hence $[\psi, (\varphi \to \bot) \to (\psi \to \bot)] :\vdash (\varphi \to \bot) \to \bot$
using *list-deduction-theorem* **by** *blast*
hence $[\psi, (\varphi \to \bot) \to (\psi \to \bot)] :\vdash \varphi$
using *double-negation list-deduction-weaken list-deduction-modus-ponens*
by *blast*
thus *?thesis*
using *list-deduction-base-theory list-deduction-theorem* **by** *blast*
qed

lemma (**in** *classical-logic*)
double-negation-converse: $\vdash \varphi \to (\varphi \to \bot) \to \bot$
by (*meson axiom-k modus-ponens flip-implication*)

The following lemma is sometimes referred to as *The Principle of Pseudo-Scotus*[3].

lemma (**in** *classical-logic*)
pseudo-scotus: $\vdash (\varphi \to \bot) \to \varphi \to \psi$
using *ex-falso-quodlibet modus-ponens hypothetical-syllogism* **by** *blast*

Another popular lemma is attributed to Charles Sanders Peirce, and has come to be known as *Peirces Law*[19].

lemma (**in** *classical-logic*) *Peirces-law*:
$\vdash ((\varphi \to \psi) \to \varphi) \to \varphi$
proof −
have $[\varphi \to \bot, (\varphi \to \psi) \to \varphi] :\vdash \varphi \to \psi$
using
pseudo-scotus
list-deduction-theorem
list-deduction-weaken
by *blast*
hence $[\varphi \to \bot, (\varphi \to \psi) \to \varphi] :\vdash \varphi$
by (*meson*
list.set-intros(1)
list-deduction-reflection
list-deduction-modus-ponens
set-subset-Cons
subsetCE)
hence $[\varphi \to \bot, (\varphi \to \psi) \to \varphi] :\vdash \bot$
by (*meson*
list.set-intros(1)
list-deduction-modus-ponens
list-deduction-reflection)
hence $[(\varphi \to \psi) \to \varphi] :\vdash (\varphi \to \bot) \to \bot$
using *list-deduction-theorem* **by** *blast*
hence $[(\varphi \to \psi) \to \varphi] :\vdash \varphi$

using *double-negation*
list-deduction-modus-ponens
list-deduction-weaken
by *blast*
thus *?thesis*
using *list-deduction-def*
by *auto*
qed

lemma (**in** *classical-logic*) *excluded-middle-elimination*:
$\vdash (\varphi \to \psi) \to ((\varphi \to \bot) \to \psi) \to \psi$
proof $-$
let $?\Gamma = [\psi \to \bot, \varphi \to \psi, (\varphi \to \bot) \to \psi]$
have $?\Gamma :\vdash (\varphi \to \bot) \to \psi$
$?\Gamma :\vdash \psi \to \bot$
by (*simp add*: *list-deduction-reflection*)+
hence $?\Gamma :\vdash (\varphi \to \bot) \to \bot$
by (*meson*
flip-hypothetical-syllogism
list-deduction-base-theory
list-deduction-monotonic
list-deduction-theorem
set-subset-Cons)
hence $?\Gamma :\vdash \varphi$
using
double-negation
list-deduction-modus-ponens
list-deduction-weaken
by *blast*
hence $?\Gamma :\vdash \psi$
by (*meson*
list.set-intros(*1*)
list-deduction-modus-ponens
list-deduction-reflection
set-subset-Cons subsetCE)
hence $[\varphi \to \psi, (\varphi \to \bot) \to \psi] :\vdash \psi$
using
Peirces-law
list-deduction-modus-ponens
list-deduction-theorem
list-deduction-weaken
by *blast*
thus *?thesis*
unfolding *list-deduction-def*
by *simp*
qed

1.2.3 Maximally Consistent Sets For Classical Logic

Relativized maximally consistent sets were introduced in §1.1.16. Often this is exactly what we want in a proof. A completeness theorem typically starts by assuming φ is not provable, then finding a φ−*MCS* Γ which gives rise to a model which does not make φ true.

A more conventional presentation says that Γ is maximally consistent if and only if $\neg\ \Gamma \Vdash \bot$ and $\forall\,\psi.\ \psi \in \Gamma \lor \psi \to \varphi \in \Gamma$. This conventional presentation will come up when formulating MaxSAT in §2.6. This in turn allows us to formulate MaxSAT completeness for probability inequalities in §2.8 and a form of the *Dutch Book Theorem* in §3.2.1.

definition (**in** *classical-logic*)
  consistent :: $'a$ *set* $\Rightarrow$ *bool* **where**
    [*simp*]: *consistent* $\Gamma \equiv \bot-$*consistent* Γ

definition (**in** *classical-logic*)
  maximally-consistent-set :: $'a$ *set* $\Rightarrow$ *bool* (*MCS*) **where**
    [*simp*]: *MCS* $\Gamma \equiv \bot-$*MCS* Γ

lemma (**in** *classical-logic*)
  formula-maximally-consistent-set-def-negation: $\varphi-$*MCS* $\Gamma \Longrightarrow \varphi \to \bot \in \Gamma$
proof −
  assume $\varphi-$*MCS* Γ
  {
    assume $\varphi \to \bot \notin \Gamma$
    hence $(\varphi \to \bot) \to \varphi \in \Gamma$
      using ‹$\varphi-$*MCS* Γ›
      unfolding *formula-maximally-consistent-set-def-def*
      by *blast*
    hence $\Gamma \Vdash (\varphi \to \bot) \to \varphi$
      using *set-deduction-reflection*
      by *simp*
    hence $\Gamma \Vdash \varphi$
      using
        Peirces-law
        set-deduction-modus-ponens
        set-deduction-weaken
      by *metis*
    hence *False*
      using ‹$\varphi-$*MCS* Γ›
      unfolding
        formula-maximally-consistent-set-def-def
        formula-consistent-def
      by *simp*
  }
  thus *?thesis* **by** *blast*
qed

Relative maximal consistency and conventional maximal consistency in fact coincide in classical logic.

lemma (**in** *classical-logic*)
  formula-maximal-consistency: $(\exists\,\varphi.\ \varphi{-}\mathit{MCS}\ \Gamma) = \mathit{MCS}\ \Gamma$
proof –
  {
    fix φ
    have $\varphi{-}\mathit{MCS}\ \Gamma \Longrightarrow \mathit{MCS}\ \Gamma$
    proof –
      assume $\varphi{-}\mathit{MCS}\ \Gamma$
      have *consistent* Γ
        using
          $\langle\varphi{-}\mathit{MCS}\ \Gamma\rangle$
          ex-falso-quodlibet [**where** $\varphi{=}\varphi$]
          set-deduction-weaken [**where** $\Gamma{=}\Gamma$]
          set-deduction-modus-ponens
        unfolding
          formula-maximally-consistent-set-def-def
          consistent-def
          formula-consistent-def
        by *metis*
      moreover {
        fix ψ
        have $\psi \to \bot \notin \Gamma \Longrightarrow \psi \in \Gamma$
        proof –
          assume $\psi \to \bot \notin \Gamma$
          hence $(\psi \to \bot) \to \varphi \in \Gamma$
            using $\langle\varphi{-}\mathit{MCS}\ \Gamma\rangle$
            unfolding *formula-maximally-consistent-set-def-def*
            by *blast*
          hence $\Gamma \Vdash (\psi \to \bot) \to \varphi$
            using *set-deduction-reflection*
            by *simp*
          also have $\Gamma \Vdash \varphi \to \bot$
            using $\langle\varphi{-}\mathit{MCS}\ \Gamma\rangle$
                formula-maximally-consistent-set-def-negation
                set-deduction-reflection
            by *simp*
          hence $\Gamma \Vdash (\psi \to \bot) \to \bot$
            using *calculation*
                hypothetical-syllogism
                  [**where** $\varphi{=}\psi \to \bot$ **and** $\psi{=}\varphi$ **and** $\chi{=}\bot$]
                set-deduction-weaken
                  [**where** $\Gamma{=}\Gamma$]
                set-deduction-modus-ponens
            by *metis*
          hence $\Gamma \Vdash \psi$
            using *double-negation*
                  [**where** $\varphi{=}\psi$]

```
          set-deduction-weaken
            [where Γ=Γ]
          set-deduction-modus-ponens
       by metis
     thus ?thesis
       using ‹φ−MCS Γ›
          formula-maximally-consistent-set-def-reflection
       by blast
   qed
  }
  ultimately show ?thesis
    unfolding maximally-consistent-set-def
          formula-maximally-consistent-set-def-def
          formula-consistent-def
          consistent-def
    by blast
 qed
}
thus ?thesis
  unfolding maximally-consistent-set-def
  by metis
qed
```

Finally, classical logic allows us to strengthen $[\![?\varphi{-}MCS\ ?\Omega;\ ?\psi \to ?\chi \in ?\Omega;\ ?\psi \in ?\Omega]\!] \Longrightarrow ?\chi \in ?\Omega$ to a biconditional.

```
lemma (in classical-logic)
  formula-maximally-consistent-set-def-implication:
  assumes φ−MCS Γ
  shows ψ → χ ∈ Γ = (ψ ∈ Γ ⟶ χ ∈ Γ)
proof −
  {
    assume hypothesis: ψ ∈ Γ ⟶ χ ∈ Γ
    {
      assume ψ ∉ Γ
      have ∀ψ. φ → ψ ∈ Γ
        by (meson assms
              formula-maximally-consistent-set-def-negation
              formula-maximally-consistent-set-def-implication-elimination
              formula-maximally-consistent-set-def-reflection
              pseudo-scotus set-deduction-weaken)
      then have ∀χ ψ. insert χ Γ ⊩ ψ ∨ χ → φ ∉ Γ
        by (meson assms
              axiom-k
              formula-maximally-consistent-set-def-reflection
              set-deduction-modus-ponens
              set-deduction-theorem
              set-deduction-weaken)
      hence ψ → χ ∈ Γ
        by (meson ‹ψ ∉ Γ›
```

```
              assms
              formula-maximally-consistent-set-def-def
              formula-maximally-consistent-set-def-reflection
              set-deduction-theorem)
  }
  moreover {
    assume χ ∈ Γ
    hence ψ → χ ∈ Γ
      by (metis assms
              calculation
              insert-absorb
              formula-maximally-consistent-set-def-reflection
              set-deduction-theorem)
  }
  ultimately have  ψ → χ ∈ Γ using hypothesis by blast
 }
 thus ?thesis
  using assms
      formula-maximally-consistent-set-def-implication-elimination
  by metis
qed

end
```

1.3 Classical Soundness and Completeness

```
theory Classical-Logic-Completeness
 imports Classical-Logic
begin

sledgehammer-params [smt-proofs = false]
```

The following presents soundness completeness of basic propositional logic for propositional semantics. A concrete algebraic data type is given for propositional formulae in §1.3.1. Logic for these formulae is defined inductively. The Tarski truth relation $\models_{prop}$ is also defined inductively, and is presented in §1.3.3.

The most significant results here are the *embedding theorems*. These theorems show that the propositional calculus can be embedded in any logic extending *classical-logic*. These theorems are proved in §1.3.5.

1.3.1 Syntax

```
datatype 'a classical-propositional-formula =
   Falsum (⊥)
  | Proposition 'a (⟨ - ⟩ [45])
  | Implication
```

$'a$ *classical-propositional-formula*
$'a$ *classical-propositional-formula* (**infixr** $\rightarrow$ *70*)

1.3.2 Propositional Calculus

named-theorems *classical-propositional-calculus*
Rules for the Propositional Calculus

inductive *classical-propositional-calculus* ::
$'a$ *classical-propositional-formula* $\Rightarrow$ *bool* ($\vdash_{prop}$ - [*60*] *55*)
where
axiom-k [*classical-propositional-calculus*]:
$\vdash_{prop} \varphi \rightarrow \psi \rightarrow \varphi$
| *axiom-s* [*classical-propositional-calculus*]:
$\vdash_{prop} (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
| *double-negation* [*classical-propositional-calculus*]:
$\vdash_{prop} ((\varphi \rightarrow \bot) \rightarrow \bot) \rightarrow \varphi$
| *modus-ponens* [*classical-propositional-calculus*]:
$\vdash_{prop} \varphi \rightarrow \psi \Longrightarrow \vdash_{prop} \varphi \Longrightarrow \vdash_{prop} \psi$

instantiation *classical-propositional-formula*
:: (*type*) *classical-logic*
begin
definition [*simp*]: $\bot = \bot$
definition [*simp*]: $\vdash \varphi = \vdash_{prop} \varphi$
definition [*simp*]: $\varphi \rightarrow \psi = \varphi \rightarrow \psi$
instance by *standard* (*simp add*: *classical-propositional-calculus*)+
end

1.3.3 Propositional Semantics

primrec *classical-propositional-semantics* ::
$'a$ *set* $\Rightarrow$ $'a$ *classical-propositional-formula* $\Rightarrow$ *bool*
(**infix** $\models_{prop}$ *65*)
where
$\mathfrak{M} \models_{prop}$ *Proposition* $p = (p \in \mathfrak{M})$
| $\mathfrak{M} \models_{prop} \varphi \rightarrow \psi = (\mathfrak{M} \models_{prop} \varphi \longrightarrow \mathfrak{M} \models_{prop} \psi)$
| $\mathfrak{M} \models_{prop} \bot = False$

theorem *classical-propositional-calculus-soundness*:
$\vdash_{prop} \varphi \Longrightarrow \mathfrak{M} \models_{prop} \varphi$
by (*induct rule*: *classical-propositional-calculus.induct*, *simp*+)

1.3.4 Soundness and Completeness Proofs

definition *strong-classical-propositional-deduction* ::
$'a$ *classical-propositional-formula set*
$\Rightarrow$ $'a$ *classical-propositional-formula* $\Rightarrow$ *bool*
(**infix** $\Vdash_{prop}$ *65*)
where

[*simp*]: $\Gamma \Vdash_{prop} \varphi \equiv \Gamma \Vdash \varphi$

definition *strong-classical-propositional-tarski-truth* ::
'a classical-propositional-formula set
$\Rightarrow$ *'a classical-propositional-formula* $\Rightarrow$ *bool*
(**infix** $\models_{prop}$ *65*)
where
[*simp*]: $\Gamma \models_{prop} \varphi \equiv \forall\ \mathfrak{M}.(\forall\ \gamma \in \Gamma.\ \mathfrak{M} \models_{prop} \gamma) \longrightarrow \mathfrak{M} \models_{prop} \varphi$

definition *theory-propositions* ::
'a classical-propositional-formula set $\Rightarrow$ *'a set* ($\{\!\!\{$ - $\}\!\!\}$ [*50*])
where
[*simp*]: $\{\!\!\{\ \Gamma\ \}\!\!\} = \{p\ .\ \Gamma \Vdash_{prop} \mathit{Proposition}\ p\}$

Below we give the main lemma for completeness: the *truth lemma*. This proof connects the maximally consistent sets developed in §1.1.16 and §1.2.3 with the semantics given in §1.3.3.

All together, the technique we are using essentially follows Blackburn et al.'s approach [2, §4.2, pgs. 196-201].

lemma *truth-lemma*:
assumes *MCS* Γ
shows $\Gamma \Vdash_{prop} \varphi \equiv \{\!\!\{\ \Gamma\ \}\!\!\} \models_{prop} \varphi$
proof (*induct* φ)
case *Falsum*
then show *?case* **using** *assms* **by** *auto*
next
case (*Proposition* x)
then show *?case* **by** *simp*
next
case (*Implication* ψ χ)
thus *?case*
unfolding *strong-classical-propositional-deduction-def*
by (*metis*
assms
maximally-consistent-set-def
formula-maximally-consistent-set-def-implication
classical-propositional-semantics.simps(*2*)
implication-classical-propositional-formula-def
set-deduction-modus-ponens
set-deduction-reflection)
qed

Here the truth lemma above is combined with $?\varphi-\mathit{consistent}\ ?\Gamma \Longrightarrow \exists\,\Omega.\ ?\varphi-\mathit{MCS}\ \Omega \wedge\ ?\Gamma \subseteq \Omega$ proven in §1.3.3. These theorems together give rise to completeness for the propositional calculus.

theorem *classical-propositional-calculus-strong-soundness-and-completeness*:
$\Gamma \Vdash_{prop} \varphi = \Gamma \models_{prop} \varphi$
proof –

have *soundness*: $\Gamma \Vdash_{prop} \varphi \Longrightarrow \Gamma \models_{prop} \varphi$
proof −
  assume $\Gamma \Vdash_{prop} \varphi$
  from *this* **obtain** Γ' **where** Γ': *set* $\Gamma' \subseteq \Gamma$ $\Gamma' :\vdash \varphi$
  by (*simp add*: *set-deduction-def*, *blast*)
  {
    fix $\mathfrak{M}$
    assume $\forall\ \gamma \in \Gamma.\ \mathfrak{M} \models_{prop} \gamma$
    hence $\forall\ \gamma \in$ *set* $\Gamma'.\ \mathfrak{M} \models_{prop} \gamma$ **using** $\Gamma'(1)$ **by** *auto*
    hence $\forall\ \varphi.\ \Gamma' :\vdash \varphi \longrightarrow \mathfrak{M} \models_{prop} \varphi$
    proof (*induct* Γ')
      case *Nil*
      then show *?case*
        by (*simp add*:
            classical-propositional-calculus-soundness
            list-deduction-def)
    next
      case (*Cons* ψ Γ')
      thus *?case* **using** *list-deduction-theorem* **by** *fastforce*
    qed
    with $\Gamma'(2)$ **have** $\mathfrak{M} \models_{prop} \varphi$ **by** *blast*
  }
  thus $\Gamma \models_{prop} \varphi$
    using *strong-classical-propositional-tarski-truth-def* **by** *blast*
qed
have *completeness*: $\Gamma \models_{prop} \varphi \Longrightarrow \Gamma \Vdash_{prop} \varphi$
proof (*erule contrapos-pp*)
  assume $\neg\ \Gamma \Vdash_{prop} \varphi$
  hence $\exists\ \mathfrak{M}.\ (\forall\ \gamma \in \Gamma.\ \mathfrak{M} \models_{prop} \gamma) \land \neg\ \mathfrak{M} \models_{prop} \varphi$
  proof −
    from $\langle\neg\ \Gamma \Vdash_{prop} \varphi\rangle$ **obtain** Ω **where** Ω: $\Gamma \subseteq \Omega$ $\varphi-MCS$ Ω
      by (*meson*
          formula-consistent-def
          formula-maximally-consistent-extension
          strong-classical-propositional-deduction-def)
    hence $(\varphi \to \bot) \in \Omega$
      using *formula-maximally-consistent-set-def-negation* **by** *blast*
    hence $\neg\ \{\!|\ \Omega\ |\!\}\ \models_{prop} \varphi$
      using Ω
          formula-consistent-def
          formula-maximal-consistency
          formula-maximally-consistent-set-def-def
          truth-lemma
      unfolding *strong-classical-propositional-deduction-def*
      by *blast*
    moreover have $\forall\ \gamma \in \Gamma.\ \{\!|\ \Omega\ |\!\}\ \models_{prop} \gamma$
      using
        formula-maximal-consistency
        truth-lemma

Ω
set-deduction-reflection
unfolding *strong-classical-propositional-deduction-def*
by *blast*
ultimately show *?thesis* **by** *auto*
qed
thus $\neg\ \Gamma \VDash_{prop} \varphi$
unfolding *strong-classical-propositional-tarski-truth-def*
by *simp*
qed
from *soundness completeness* **show** $\Gamma \Vdash_{prop} \varphi = \Gamma \VDash_{prop} \varphi$
by *linarith*
qed

theorem *classical-propositional-calculus-soundness-and-completeness*:
$\vdash_{prop} \varphi = (\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} \varphi)$
using *classical-propositional-calculus-soundness* [**where** φ=φ]
classical-propositional-calculus-strong-soundness-and-completeness
[**where** φ=φ **and** Γ={}]
strong-classical-propositional-deduction-def
[**where** φ=φ **and** Γ={}]
strong-classical-propositional-tarski-truth-def
[**where** φ=φ **and** Γ={}]
deduction-classical-propositional-formula-def [**where** φ=φ]
set-deduction-base-theory [**where** φ=φ]
by *metis*

instantiation *classical-propositional-formula*
:: (*type*) *consistent-classical-logic*
begin
instance by *standard*
(*simp add*: *classical-propositional-calculus-soundness-and-completeness*)
end

1.3.5 Embedding Theorem For the Propositional Calculus

primrec (**in** *classical-logic*)
classical-propositional-formula-embedding
:: *'a classical-propositional-formula* $\Rightarrow$ *'a* (($\llparenthesis$ - $\rrparenthesis$ [*50*]) **where**
$\llparenthesis\ \langle\ p\ \rangle\ \rrparenthesis = p$
| $\llparenthesis\ \varphi \rightarrow \psi\ \rrparenthesis = \llparenthesis\ \varphi\ \rrparenthesis \rightarrow \llparenthesis\ \psi\ \rrparenthesis$
| $\llparenthesis\ \bot\ \rrparenthesis = \bot$

theorem (**in** *classical-logic*) *propositional-calculus*:
$\vdash_{prop} \varphi \Longrightarrow \vdash \llparenthesis\ \varphi\ \rrparenthesis$
by (*induct rule*: *classical-propositional-calculus.induct*,
(*simp add*: *axiom-k axiom-s double-negation modus-ponens*)+)

theorem (**in** *classical-logic*) *propositional-semantics*:

```
  ∀𝔐. 𝔐 ⊨prop φ ⟹ ⊢ ⦇ φ ⦈
  by (simp add:
        classical-propositional-calculus-soundness-and-completeness
        propositional-calculus)

end
```

1.4 Digression: List Utility Theorems

Throughout our work it will be necessary to reuse common lemmas regarding lists and multisets. These results are proved in the following section and reused by subsequent lemmas and theorems.

```
theory List-Utilities
  imports
    HOL−Library.Permutation
begin

sledgehammer-params [smt-proofs = false]
```

1.4.1 Multiset Coercion

```
lemma length-sub-mset:
  assumes mset Ψ ⊆# mset Γ
      and length Ψ >= length Γ
    shows mset Ψ = mset Γ
  using assms
proof −
  have ∀ Ψ. mset Ψ ⊆# mset Γ
            ⟶ length Ψ >= length Γ
            ⟶ mset Ψ = mset Γ
  proof (induct Γ)
    case Nil
    then show ?case by simp
  next
    case (Cons γ Γ)
    {
      fix Ψ
      assume mset Ψ ⊆# mset (γ # Γ) length Ψ >= length (γ # Γ)
      have γ ∈ set Ψ
      proof (rule ccontr)
        assume γ ∉ set Ψ
        hence ◇: remove1 γ Ψ = Ψ
          by (simp add: remove1-idem)
        have mset Ψ ⊆# mset (γ # Γ)
          using ‹mset Ψ ⊆# mset (γ # Γ)› by auto
        hence mset Ψ ⊆# mset (remove1 γ (γ # Γ))
          by (metis ◇ mset-le-perm-append perm-remove-perm remove1-append)
        hence mset Ψ ⊆# mset Γ
```

```
        by simp
      hence mset Ψ = mset Γ
        using ‹length (γ # Γ) ≤ length Ψ› size-mset-mono by fastforce
      hence length Ψ = length Γ
        by (metis size-mset)
      hence length Γ ≥ length (γ # Γ)
        using ‹length (γ # Γ) ≤ length Ψ› by auto
      thus False by simp
    qed
    hence ♡: mset Ψ = mset (γ # (remove1 γ Ψ))
      by simp
    hence length (remove1 γ Ψ) >= length Γ
      by (metis
            ‹length (γ # Γ) ≤ length Ψ›
            drop-Suc-Cons
            drop-eq-Nil
            length-Cons
            mset-eq-length)
    moreover have mset (remove1 γ Ψ) ⊆# mset Γ
      by (simp,
          metis
            ♡
            ‹mset Ψ ⊆# mset (γ # Γ)›
            mset.simps(2)
            mset-remove1
            mset-subset-eq-add-mset-cancel)
    ultimately have mset (remove1 γ Ψ) = mset Γ using Cons by blast
    with ♡ have mset Ψ = mset (γ # Γ) by simp
  }
  thus ?case by blast
 qed
 thus ?thesis using assms by blast
qed

lemma set-exclusion-mset-simplify:
 assumes ¬ (∃ ψ ∈ set Ψ. ψ ∈ set Σ)
    and mset Ψ ⊆# mset (Σ @ Γ)
  shows mset Ψ ⊆# mset Γ
using assms
proof (induct Σ)
 case Nil
 then show ?case by simp
next
 case (Cons σ Σ)
 then show ?case
  by (cases σ ∈ set Ψ,
      fastforce,
      metis
        add.commute
```

```
      add-mset-add-single
      diff-single-trivial
      in-multiset-in-set
      mset.simps(2)
      notin-set-remove1
      remove-hd
      subset-eq-diff-conv
      union-code
      append-Cons)
qed
```

1.4.2 List Mapping

The following notation for permutations is slightly nicer when formatted in LaTeX.

```
notation perm (infix ⇌ 50)

lemma map-perm:
  assumes A ⇌ B
  shows map f A ⇌ map f B
  by (metis assms mset-eq-perm mset-map)

lemma map-monotonic:
  assumes mset A ⊆# mset B
  shows mset (map f A) ⊆# mset (map f B)
  by (simp add: assms image-mset-subseteq-mono)

lemma perm-map-perm-list-exists:
  assumes A ⇌ map f B
  shows ∃ B'. A = map f B' ∧ B' ⇌ B
proof −
  have ∀ B. A ⇌ map f B ⟶ (∃ B'. A = map f B' ∧ B' ⇌ B)
  proof (induct A)
    case Nil
    then show ?case by simp
  next
    case (Cons a A)
    {
      fix B
      assume a # A ⇌ map f B
      from this obtain b where b:
        b ∈ set B
        f b = a
        by (metis
              (full-types)
              imageE
              list.set-intros(1)
              mset-eq-perm
              set-map
```

set-mset-mset)
hence $A \rightleftharpoons$ (*remove1* (*f b*) (*map f B*))
$B \rightleftharpoons b$ # *remove1 b B*
by (*metis*
‹a # $A \rightleftharpoons$ *map f B*›
perm-remove-perm
remove-hd,
meson b(*1*) *perm-remove*)
hence $A \rightleftharpoons$ (*map f* (*remove1 b B*))
by (*metis* (*no-types*)
list.simps(*9*)
mset-eq-perm
mset-map
mset-remove1
remove-hd)
from *this* **obtain** B' **where** B':
A = *map f* B'
$B' \rightleftharpoons$ (*remove1 b B*)
using *Cons.hyps* **by** *blast*
with *b* **have** a # A = *map f* (b # B')
by *simp*
moreover have $B \rightleftharpoons b$ # B'
by (*meson*
B'(*2*)
b(*1*)
cons-perm-eq
perm.trans
perm-remove
perm-sym)
ultimately have $\exists B'$. a # A = *map f* $B' \wedge B' \rightleftharpoons B$
by (*meson perm-sym*)
}
thus *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma *mset-sub-map-list-exists*:
assumes *mset* $\Phi \subseteq$# *mset* (*map f* Γ)
shows $\exists\ \Phi'$. *mset* $\Phi' \subseteq$# *mset* $\Gamma \wedge \Phi$ = (*map f* Φ')
proof −
have $\forall\ \Phi$. *mset* $\Phi \subseteq$# *mset* (*map f* Γ)
$\longrightarrow$ ($\exists\ \Phi'$. *mset* $\Phi' \subseteq$# *mset* $\Gamma \wedge \Phi$ = (*map f* Φ'))
proof (*induct* Γ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* γ Γ)
{

fix Φ
assume *mset* $\Phi \subseteq\#$ *mset* (*map f* (γ # Γ))
have $\exists \Phi'$. *mset* $\Phi' \subseteq\#$ *mset* (γ # Γ) $\wedge$ Φ = *map f* Φ'
proof *cases*
  assume *f* $\gamma \in$ *set* Φ
  hence *f* γ # (*remove1* (*f* γ) Φ) $\rightleftharpoons \Phi$
    by (*simp add*: *perm-remove perm-sym*)
  with ‹*mset* $\Phi \subseteq\#$ *mset* (*map f* (γ # Γ))›
  have *mset* (*remove1* (*f* γ) Φ) $\subseteq\#$ *mset* (*map f* Γ)
    by (*metis*
        insert-subset-eq-iff
        list.simps(*9*)
        mset.simps(*2*)
        mset-eq-perm
        mset-remove1
        remove-hd)
  from *this Cons* **obtain** Φ' **where** Φ':
    mset $\Phi' \subseteq\#$ *mset* Γ
    remove1 (*f* γ) Φ = *map f* Φ'
    by *blast*
  hence *mset* (γ # Φ') $\subseteq\#$ *mset* (γ # Γ)
    and *f* γ # (*remove1* (*f* γ) Φ) = *map f* (γ # Φ')
    by *simp*+
  hence $\Phi \rightleftharpoons$ *map f* (γ # Φ')
    using ‹*f* $\gamma \in$ *set* Φ› *perm-remove* **by** *force*
  from *this* **obtain** Φ'' **where** Φ'':
    Φ = *map f* Φ''
    $\Phi'' \rightleftharpoons \gamma$ # Φ'
    using *perm-map-perm-list-exists*
    by *blast*
  hence *mset* $\Phi'' \subseteq\#$ *mset* (γ # Γ)
    by (*metis* ‹*mset* (γ # Φ') $\subseteq\#$ *mset* (γ # Γ)› *mset-eq-perm*)
  thus *?thesis* **using** Φ'' **by** *blast*
next
  assume *f* $\gamma \notin$ *set* Φ
  have *mset* Φ − {#*f* γ#} = *mset* Φ
    by (*metis* (*no-types*)
        ‹*f* $\gamma \notin$ *set* Φ›
        diff-single-trivial
        set-mset-mset)
  moreover
  have *mset* (*map f* (γ # Γ))
        = *add-mset* (*f* γ) (*image-mset f* (*mset* Γ))
    by *simp*
  ultimately have *mset* $\Phi \subseteq\#$ *mset* (*map f* Γ)
    by (*metis* (*no-types*)
        Diff-eq-empty-iff-mset
        ‹*mset* $\Phi \subseteq\#$ *mset* (*map f* (γ # Γ))›
        add-mset-add-single

cancel-ab-semigroup-add-class.diff-right-commute
diff-diff-add mset-map)
with *Cons* **show** *?thesis*
by (*metis*
diff-subset-eq-self
mset-remove1
remove-hd
subset-mset.order.trans)
qed
}
thus *?case* **using** *Cons* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

1.4.3 Laws for Searching a List

lemma *find-Some-predicate*:
assumes *find* P Ψ = *Some* ψ
shows P ψ
using *assms*
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ω Ψ)
then show *?case* **by** (*cases* P ω, *fastforce*+)
qed

lemma *find-Some-set-membership*:
assumes *find* P Ψ = *Some* ψ
shows $\psi \in$ *set* Ψ
using *assms*
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ω Ψ)
then show *?case* **by** (*cases* P ω, *fastforce*+)
qed

1.4.4 Permutations

lemma *perm-count-list*:
assumes $\Phi \rightleftharpoons \Psi$
shows *count-list* Φ φ = *count-list* Ψ φ
proof −
have $\forall\,\Psi.\ \Phi \rightleftharpoons \Psi \longrightarrow$ *count-list* Φ φ = *count-list* Ψ φ
proof (*induct* Φ)
case *Nil*

```
    then show ?case
      by simp
  next
    case (Cons χ Φ)
    {
      fix Ψ
      assume χ # Φ ⇌ Ψ
      hence χ ∈ set Ψ
        using perm-set-eq by fastforce
      hence Ψ ⇌ χ # (remove1 χ Ψ)
        by (simp add: perm-remove)
      hence Φ ⇌ (remove1 χ Ψ)
        using ‹χ # Φ ⇌ Ψ› perm.trans by auto
      hence ◇: count-list Φ φ = count-list (remove1 χ Ψ) φ
        using Cons.hyps by blast
      have count-list (χ # Φ) φ = count-list Ψ φ
      proof cases
        assume χ = φ
        hence count-list (χ # Φ) φ = count-list Φ φ + 1 by simp
        with ◇ have count-list (χ # Φ) φ = count-list (remove1 χ Ψ) φ + 1
          by simp
        moreover have count-list (remove1 χ Ψ) φ + 1 = count-list Ψ φ
          using ‹χ = φ› ‹χ ∈ set Ψ›
          by (induct Ψ, simp, auto)
        ultimately show ?thesis by simp
      next
        assume χ ≠ φ
        with ◇ have count-list (χ # Φ) φ = count-list (remove1 χ Ψ) φ
          by simp
        moreover have count-list (remove1 χ Ψ) φ = count-list Ψ φ
          using ‹χ ≠ φ›
          by (induct Ψ, simp+)
        ultimately show ?thesis by simp
      qed
    }
    then show ?case
      by blast
  qed
  with assms show ?thesis by blast
qed

lemma count-list-append:
  count-list (A @ B) a = count-list A a + count-list B a
  by (induct A, simp, simp)

lemma append-set-containment:
  assumes a ∈ set A
      and A ⇌ B @ C
    shows a ∈ set B ∨ a ∈ set C
```

```
  using assms
  by (simp add: perm-set-eq)

lemma concat-remove1:
  assumes Ψ ∈ set ℒ
  shows concat ℒ ⇌ Ψ @ concat (remove1 Ψ ℒ)
    using assms
    by (induct ℒ,
        simp,
        simp,
        metis append.assoc
              perm.trans
              perm-append1
              perm-append-swap)

lemma concat-set-membership-mset-containment:
  assumes concat Γ ⇌ Λ
  and     Φ ∈ set Γ
  shows   mset Φ ⊆# mset Λ
  using assms
  by (induct Γ,
      simp,
      meson concat-remove1 mset-le-perm-append perm.trans perm-sym)

lemma (in comm-monoid-add) perm-list-summation:
  assumes Ψ ⇌ Φ
  shows (∑ψ'←Ψ. f ψ') = (∑φ'←Φ. f φ')
proof −
  have ∀ Φ. Ψ ⇌ Φ ⟶ (∑ψ'←Ψ. f ψ') = (∑φ'←Φ. f φ')
  proof (induct Ψ)
    case Nil
    then show ?case by simp
  next
    case (Cons ψ Ψ)
    {
      fix Φ
      assume hypothesis: ψ # Ψ ⇌ Φ
      hence Ψ ⇌ (remove1 ψ Φ)
        by (metis perm-remove-perm remove-hd)
      hence (∑ψ' ← Ψ. f ψ') = (∑φ' ← (remove1 ψ Φ). f φ')
        using Cons.hyps by blast
      moreover have ψ ∈ set Φ
        using hypothesis perm-set-eq by fastforce
      hence (∑φ' ← (ψ # (remove1 ψ Φ)). f φ') = (∑φ'←Φ. f φ')
      proof (induct Φ)
        case Nil
        then show ?case by simp
      next
        case (Cons φ Φ)
```

show *?case*
proof *cases*
assume $\varphi = \psi$
then show *?thesis* by *simp*
next
assume $\varphi \neq \psi$
hence $\psi \in set\ \Phi$
using *Cons.prems* by *auto*
hence $(\sum \varphi' \leftarrow (\psi \# (remove1\ \psi\ \Phi)).\ f\ \varphi') = (\sum \varphi' \leftarrow \Phi.\ f\ \varphi')$
using *Cons.hyps* by *blast*
hence $(\sum \varphi' \leftarrow (\varphi \# \Phi).\ f\ \varphi')$
$= (\sum \varphi' \leftarrow (\psi \# \varphi \# (remove1\ \psi\ \Phi)).\ f\ \varphi')$
by (*simp add*: *add.left-commute*)
moreover
have $(\psi \# (\varphi \# (remove1\ \psi\ \Phi))) = (\psi \# (remove1\ \psi\ (\varphi \# \Phi)))$
using $\langle \varphi \neq \psi \rangle$ by *simp*
ultimately show *?thesis*
by *simp*
qed
qed
ultimately have $(\sum \psi' \leftarrow (\psi \# \Psi).\ f\ \psi') = (\sum \varphi' \leftarrow \Phi.\ f\ \varphi')$
by *simp*
}
then show *?case* by *blast*
qed
with *assms* show *?thesis* by *blast*
qed

1.4.5 List Duplicates

primrec *duplicates* :: $'a\ list \Rightarrow 'a\ set$
where
duplicates $[] = \{\}$
| *duplicates* $(x \# xs) =$
(*if* $(x \in set\ xs)$
then insert x (*duplicates xs*)
else duplicates xs)

lemma *duplicates-subset*:
duplicates $\Phi \subseteq set\ \Phi$
by (*induct* Φ, *simp*, *auto*)

lemma *duplicates-alt-def*:
duplicates xs $= \{x.\ count\text{-}list\ xs\ x \geq 2\}$
proof (*induct xs*)
case *Nil*
then show *?case* by *simp*
next
case (*Cons x xs*)

assume *inductive-hypothesis*: $duplicates\ xs = \{x.\ 2 \leq count\text{-}list\ xs\ x\}$
then show *?case*
proof *cases*
assume $x \in set\ xs$
hence $count\text{-}list\ (x \# xs)\ x \geq 2$
by (*simp*, *induct xs*, *simp*, *simp*, *blast*)
hence $\{y.\ 2 \leq count\text{-}list\ (x \# xs)\ y\}$
$= insert\ x\ \{y.\ 2 \leq count\text{-}list\ xs\ y\}$
by (*simp*, *blast*)
thus *?thesis* **using** *inductive-hypothesis* ‹$x \in set\ xs$›
by *simp*
next
assume $x \notin set\ xs$
hence $\{y.\ 2 \leq count\text{-}list\ (x \# xs)\ y\} = \{y.\ 2 \leq count\text{-}list\ xs\ y\}$
by (*simp*, *auto*)
thus *?thesis* **using** *inductive-hypothesis* ‹$x \notin set\ xs$›
by *simp*
qed
qed

1.4.6 List Subtraction

primrec *list-subtract* :: $'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$ (**infixl** $\ominus$ *70*)
where
$xs \ominus [] = xs$
$|\ xs \ominus (y \# ys) = (remove1\ y\ (xs \ominus ys))$

lemma *list-subtract-mset-homomorphism* [*simp*]:
$mset\ (A \ominus B) = mset\ A - mset\ B$
by (*induct B*, *simp*, *simp*)

lemma *list-subtract-empty* [*simp*]:
$[] \ominus \Phi = []$
by (*induct* Φ, *simp*, *simp*)

lemma *list-subtract-remove1-cons-perm*:
$\Phi \ominus (\varphi \# \Lambda) \rightleftharpoons (remove1\ \varphi\ \Phi) \ominus \Lambda$
by (*induct* Λ, *simp*, *simp*, *metis perm-remove-perm remove1-commute*)

lemma *list-subtract-cons*:
assumes $\varphi \notin set\ \Lambda$
shows $(\varphi \# \Phi) \ominus \Lambda = \varphi \# (\Phi \ominus \Lambda)$
using *assms*
by (*induct* Λ, *simp*, *simp*, *blast*)

lemma *list-subtract-cons-absorb*:
assumes $count\text{-}list\ \Phi\ \varphi \geq count\text{-}list\ \Lambda\ \varphi$
shows $\varphi \# (\Phi \ominus \Lambda) \rightleftharpoons (\varphi \# \Phi) \ominus \Lambda$
using *assms*

```
proof -
  have ∀ Φ. count-list Φ φ ≥ count-list Λ φ
             ⟶ φ # (Φ ⊖ Λ) ⇌ (φ # Φ) ⊖ Λ
  proof (induct Λ)
    case Nil
    thus ?case using list-subtract-cons by fastforce
  next
    case (Cons ψ Λ)
    assume inductive-hypothesis:
          ∀Φ. count-list Λ φ ≤ count-list Φ φ
               ⟶ φ # Φ ⊖ Λ ⇌ (φ # Φ) ⊖ Λ
    {
      fix Φ :: 'a list
      assume count-list (ψ # Λ) φ ≤ count-list Φ φ
      have count-list Λ φ ≤ count-list (remove1 ψ Φ) φ
      proof (cases φ = ψ)
        case True
        hence 1 + count-list Λ φ ≤ count-list Φ φ
          using ‹count-list (ψ # Λ) φ ≤ count-list Φ φ›
          by auto
        moreover from this have φ ∈ set Φ
          using not-one-le-zero by fastforce
        hence Φ ⇌ φ # (remove1 ψ Φ)
          using True
          by (simp add: True perm-remove)
        ultimately show ?thesis by (simp add: perm-count-list)
      next
        case False
        hence count-list (ψ # Λ) φ = count-list Λ φ
           by simp
        moreover have count-list Φ φ = count-list (remove1 ψ Φ) φ
        proof (induct Φ)
          case Nil
          then show ?case by simp
        next
          case (Cons φ' Φ)
          show ?case
          proof (cases φ' = φ)
            case True
            with ‹φ ≠ ψ›
            have count-list (φ' # Φ) φ = 1 + count-list Φ φ
                 count-list (remove1 ψ (φ' # Φ)) φ
                   = 1 + count-list (remove1 ψ Φ) φ
              by simp+
            with Cons show ?thesis by linarith
          next
            case False
            with Cons show ?thesis by (cases φ' = ψ, simp+)
          qed
```

```
      qed
      ultimately show ?thesis
        using ‹count-list (ψ # Λ) φ ≤ count-list Φ φ›
        by auto
    qed
    hence φ # ((remove1 ψ Φ) ⊖ Λ) ⇌ (φ # (remove1 ψ Φ)) ⊖ Λ
        using inductive-hypothesis by blast
    moreover have φ # ((remove1 ψ Φ) ⊖ Λ) ⇌ φ # (Φ ⊖ (ψ # Λ))
      by (induct Λ, simp, simp add: perm-remove-perm remove1-commute)
    ultimately have ⋆: φ # (Φ ⊖ (ψ # Λ)) ⇌ (φ # (remove1 ψ Φ)) ⊖ Λ
      by (meson perm.trans perm-sym)
    have φ # (Φ ⊖ (ψ # Λ)) ⇌ (φ # Φ) ⊖ (ψ # Λ)
    proof cases
      assume φ = ψ
      hence (φ # Φ) ⊖ (ψ # Λ) ⇌ Φ ⊖ Λ
        using list-subtract-remove1-cons-perm by fastforce
      moreover have φ ∈ set Φ
        using
          ‹φ = ψ›
          ‹count-list (ψ # Λ) φ ≤ count-list Φ φ›
          leD
        by force
      hence Φ ⊖ Λ ⇌ (φ # (remove1 φ Φ)) ⊖ Λ
        by (induct Λ, simp add: perm-remove, simp add: perm-remove-perm)
      ultimately show ?thesis
        using ⋆
        by (metis ‹φ = ψ› mset-eq-perm)
    next
      assume φ ≠ ψ
      hence (φ # (remove1 ψ Φ)) ⊖ Λ ⇌ (φ # Φ) ⊖ (ψ # Λ)
        by (induct Λ, simp, simp add: perm-remove-perm remove1-commute)
      then show ?thesis using ⋆ by blast
    qed
  }
  then show ?case by blast
 qed
 with assms show ?thesis by blast
qed

lemma list-subtract-remove1-perm:
 assumes φ ∈ set Λ
 shows  Φ ⊖ Λ ⇌ (remove1 φ Φ) ⊖ (remove1 φ Λ)
proof –
 from ‹φ ∈ set Λ›
 have mset (Φ ⊖ Λ) = mset ((remove1 φ Φ) ⊖ (remove1 φ Λ))
   by simp
 thus ?thesis
   using mset-eq-perm by blast
qed
```

lemma *list-subtract-cons-remove1-perm*:
  assumes $\varphi \in$ *set* Λ
  shows $(\varphi \# \Phi) \ominus \Lambda \rightleftharpoons \Phi \ominus$ (*remove1* φ Λ)
  using *assms list-subtract-remove1-perm* **by** *fastforce*

lemma *list-subtract-removeAll-perm*:
  assumes *count-list* Φ $\varphi \leq$ *count-list* Λ φ
  shows $\Phi \ominus \Lambda \rightleftharpoons$ (*removeAll* φ Φ) $\ominus$ (*removeAll* φ Λ)
proof −
  have $\forall$ Λ. *count-list* Φ $\varphi \leq$ *count-list* Λ φ
        $\longrightarrow \Phi \ominus \Lambda \rightleftharpoons$ (*removeAll* φ Φ) $\ominus$ (*removeAll* φ Λ)
  proof (*induct* Φ)
    case *Nil*
    thus *?case* **by** *auto*
  next
    case (*Cons* ξ Φ)
    {
      fix Λ
      assume *count-list* $(\xi \# \Phi)$ $\varphi \leq$ *count-list* Λ φ
      hence $\Phi \ominus \Lambda \rightleftharpoons$ (*removeAll* φ Φ) $\ominus$ (*removeAll* φ Λ)
        by (*metis*
            Cons.hyps
            count-list.simps(2)
            dual-order.trans
            le-add-same-cancel1
            zero-le-one)
      have $(\xi \# \Phi) \ominus \Lambda \rightleftharpoons$ (*removeAll* φ $(\xi \# \Phi)$) $\ominus$ (*removeAll* φ Λ)
      proof *cases*
        assume $\xi = \varphi$
        hence *count-list* Φ $\varphi <$ *count-list* Λ φ
          using ‹*count-list* $(\xi \# \Phi)$ $\varphi \leq$ *count-list* Λ φ›
          by *auto*
        hence *count-list* Φ $\varphi \leq$ *count-list* (*remove1* φ Λ) φ
          by (*induct* Λ, *simp*, *auto*)
        hence $\Phi \ominus$ (*remove1* φ Λ)
            $\rightleftharpoons$ *removeAll* φ $\Phi \ominus$ *removeAll* φ (*remove1* φ Λ)
          using *Cons.hyps* **by** *blast*
        hence $\Phi \ominus$ (*remove1* φ Λ) $\rightleftharpoons$ *removeAll* φ $\Phi \ominus$ *removeAll* φ Λ
          by (*simp add*: *filter-remove1 removeAll-filter-not-eq*)
        moreover have $\varphi \in$ *set* Λ **and** $\varphi \in$ *set* $(\varphi \# \Phi)$
          using ‹$\xi = \varphi$›
            ‹*count-list* $(\xi \# \Phi)$ $\varphi \leq$ *count-list* Λ φ›
            gr-implies-not0
          by *fastforce+*
        hence $(\varphi \# \Phi) \ominus \Lambda \rightleftharpoons$ (*remove1* φ $(\varphi \# \Phi)$) $\ominus$ (*remove1* φ Λ)
          by (*meson list-subtract-remove1-perm*)
        hence $(\varphi \# \Phi) \ominus \Lambda \rightleftharpoons \Phi \ominus$ (*remove1* φ Λ) **by** *simp*
        ultimately show *?thesis* **using** ‹$\xi = \varphi$› **by** *auto*

```
      next
        assume ξ ≠ φ
        show ?thesis
        proof cases
          assume ξ ∈ set Λ
          hence (ξ # Φ) ⊖ Λ ⇌ Φ ⊖ remove1 ξ Λ
            by (simp add: list-subtract-cons-remove1-perm)
          moreover have count-list Λ φ = count-list (remove1 ξ Λ) φ
            using ‹ξ ≠ φ› ‹ξ ∈ set Λ› perm-count-list perm-remove
            by force
          hence count-list Φ φ ≤ count-list (remove1 ξ Λ) φ
            using ‹ξ ≠ φ› ‹count-list (ξ # Φ) φ ≤ count-list Λ φ› by auto
          hence Φ ⊖ remove1 ξ Λ
                  ⇌ (removeAll φ Φ) ⊖ (removeAll φ (remove1 ξ Λ))
            using Cons.hyps by blast
          moreover
          have (removeAll φ Φ) ⊖ (removeAll φ (remove1 ξ Λ)) ⇌
                 (removeAll φ Φ) ⊖ (remove1 ξ (removeAll φ Λ))
            by (induct Λ,
                 simp,
                 simp add: filter-remove1 removeAll-filter-not-eq)
          hence (removeAll φ Φ) ⊖ (removeAll φ (remove1 ξ Λ)) ⇌
                  (removeAll φ (ξ # Φ)) ⊖ (removeAll φ Λ)
            by (simp add: ‹ξ ∈ set Λ›
                          filter-remove1
                          list-subtract-cons-remove1-perm
                          perm-sym
                          removeAll-filter-not-eq)
          ultimately show ?thesis by blast
        next
          assume ξ ∉ set Λ
          hence (ξ # Φ) ⊖ Λ ⇌ ξ # (Φ ⊖ Λ)
            by (simp add: list-subtract-cons-absorb perm-sym)
          hence (ξ # Φ) ⊖ Λ ⇌ ξ # ((removeAll φ Φ) ⊖ (removeAll φ Λ))
            using ‹Φ ⊖ Λ ⇌ removeAll φ Φ ⊖ removeAll φ Λ› by blast
          hence (ξ # Φ) ⊖ Λ ⇌ (ξ # (removeAll φ Φ)) ⊖ (removeAll φ Λ)
            by (simp add: ‹ξ ∉ set Λ› list-subtract-cons)
          thus ?thesis using ‹ξ ≠ φ› by auto
        qed
      qed
    }
    then show ?case by auto
  qed
  with assms show ?thesis by blast
qed

lemma list-subtract-permute:
  assumes Φ ⇌ Ψ
  shows Φ ⊖ Λ ⇌ Ψ ⊖ Λ
```

proof −
  from ‹$\Phi \rightleftharpoons \Psi$› **have** *mset* Φ = *mset* Ψ
    by (*simp add*: *mset-eq-perm*)
  hence *mset* $(\Phi \ominus \Lambda)$ = *mset* $(\Psi \ominus \Lambda)$
    by *simp*
  thus *?thesis*
    using *mset-eq-perm* **by** *blast*
qed

lemma *append-perm-list-subtract-intro*:
  assumes $A \rightleftharpoons B$ @ C
  shows $A \ominus C \rightleftharpoons B$
proof −
  from ‹$A \rightleftharpoons B$ @ C› **have** *mset* A = *mset* (B @ C)
    using *mset-eq-perm* **by** *blast*
  hence *mset* $(A \ominus C)$ = *mset* B
    by *simp*
  thus *?thesis* **using** *mset-eq-perm* **by** *blast*
qed

lemma *list-subtract-concat*:
  assumes $\Psi \in$ *set* $\mathcal{L}$
  shows *concat* $(\mathcal{L} \ominus [\Psi]) \rightleftharpoons$ (*concat* $\mathcal{L}$) $\ominus \Psi$
  using *assms*
  by (*simp*,
      meson
        append-perm-list-subtract-intro
        concat-remove1
        perm.trans
        perm-append-swap
        perm-sym)

lemma (**in** *comm-monoid-add*) *listSubstract-multisubset-list-summation*:
  assumes *mset* $\Psi \subseteq\#$ *mset* Φ
  shows $(\sum \psi \leftarrow \Psi.\ f\ \psi) + (\sum \varphi' \leftarrow (\Phi \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow \Phi.\ f\ \varphi')$
proof −
  have $\forall\ \Phi.$ *mset* $\Psi \subseteq\#$ *mset* Φ
        $\longrightarrow (\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow \Phi.\ f\ \varphi')$
  proof(*induct* Ψ)
    case *Nil*
    then show *?case*
      by *simp*
  next
    case (*Cons* ψ Ψ)
    {
      fix Φ
      assume *hypothesis*: *mset* $(\psi\ \#\ \Psi) \subseteq\#$ *mset* Φ
      hence *mset* $\Psi \subseteq\#$ *mset* (*remove1* ψ Φ)
        by (*metis append-Cons mset-le-perm-append perm-remove-perm remove-hd*)

hence
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi'\leftarrow((\mathit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi')$
$= (\sum \varphi'\leftarrow (\mathit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
using *Cons.hyps* **by** *blast*
moreover have $(\mathit{remove1}\ \psi\ \Phi) \ominus \Psi \rightleftharpoons \Phi \ominus (\psi \# \Psi)$
by (*meson list-subtract-remove1-cons-perm perm-sym*)
hence $(\sum \varphi'\leftarrow((\mathit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi') = (\sum \varphi'\leftarrow(\Phi \ominus (\psi \# \Psi)).\ f\ \varphi')$
using *perm-list-summation* **by** *blast*
ultimately have
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi'\leftarrow(\Phi \ominus (\psi \# \Psi)).\ f\ \varphi')$
$= (\sum \varphi'\leftarrow (\mathit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
by *simp*
hence
$(\sum \psi' \leftarrow (\psi \# \Psi).\ f\ \psi') + (\sum \varphi'\leftarrow(\Phi \ominus (\psi \# \Psi)).\ f\ \varphi')$
$= (\sum \varphi'\leftarrow (\psi \# (\mathit{remove1}\ \psi\ \Phi)).\ f\ \varphi')$
by (*simp add*: *add.assoc*)
moreover have $\psi \in \mathit{set}\ \Phi$
by (*metis*
append-Cons
hypothesis
list.set-intros(*1*)
mset-le-perm-append
perm-set-eq)
hence $(\psi \# (\mathit{remove1}\ \psi\ \Phi)) \rightleftharpoons \Phi$
by (*simp add*: *perm-remove perm-sym*)
hence $(\sum \varphi' \leftarrow (\psi \# (\mathit{remove1}\ \psi\ \Phi)).\ f\ \varphi') = (\sum \varphi'\leftarrow\Phi.\ f\ \varphi')$
using *perm-list-summation* **by** *blast*
ultimately have
$(\sum \psi' \leftarrow (\psi \# \Psi).\ f\ \psi') + (\sum \varphi'\leftarrow(\Phi \ominus (\psi \# \Psi)).\ f\ \varphi')$
$= (\sum \varphi'\leftarrow\Phi.\ f\ \varphi')$
by *simp*
}
then show *?case*
by *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma *list-subtract-set-difference-lower-bound*:
$\mathit{set}\ \Gamma - \mathit{set}\ \Phi \subseteq \mathit{set}\ (\Gamma \ominus \Phi)$
using *subset-Diff-insert*
by (*induct* Φ, *simp*, *fastforce*)

lemma *list-subtract-set-trivial-upper-bound*:
$\mathit{set}\ (\Gamma \ominus \Phi) \subseteq \mathit{set}\ \Gamma$
by (*induct* Φ,
simp,
simp,
meson

```
        dual-order.trans
        set-remove1-subset)

lemma list-subtract-msub-eq:
  assumes mset Φ ⊆# mset Γ
      and length (Γ ⊖ Φ) = m
    shows length Γ = m + length Φ
  using assms
proof –
  have ∀ Γ. mset Φ ⊆# mset Γ
          ⟶ length (Γ ⊖ Φ) = m −−> length Γ = m + length Φ
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons φ Φ)
    {
      fix Γ :: 'a list
      assume mset (φ # Φ) ⊆# mset Γ
             length (Γ ⊖ (φ # Φ)) = m
      moreover from this have
        mset Φ ⊆# mset (remove1 φ Γ)
        mset (Γ ⊖ (φ # Φ)) = mset ((remove1 φ Γ) ⊖ Φ)
        by (metis
              append-Cons
              mset-le-perm-append
              perm-remove-perm
              remove-hd,
            simp)
      ultimately have length (remove1 φ Γ) = m + length Φ
        using Cons.hyps
        by (metis mset-eq-length)
      hence length (φ # (remove1 φ Γ)) = m + length (φ # Φ)
        by simp
      moreover have φ ∈ set Γ
        by (metis
              ‹mset (Γ ⊖ (φ # Φ)) = mset (remove1 φ Γ ⊖ Φ)›
              ‹mset (φ # Φ) ⊆# mset Γ›
              ‹mset Φ ⊆# mset (remove1 φ Γ)›
              add-diff-cancel-left'
              add-right-cancel
              eq-iff
              impossible-Cons
              list-subtract-mset-homomorphism
              mset-subset-eq-exists-conv
              remove1-idem size-mset)
      hence length (φ # (remove1 φ Γ)) = length Γ
        by (metis
              One-nat-def
```

Suc-pred
length-Cons
length-pos-if-in-set
length-remove1)
ultimately have *length* $\Gamma = m +$ *length* $(\varphi \# \Phi)$ **by** *simp*
}
thus *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma *list-subtract-not-member*:
assumes $b \notin$ *set* A
shows $A \ominus B = A \ominus$ (*remove1* b B)
using *assms*
by (*induct* B,
simp,
simp,
metis
add-mset-add-single
diff-subset-eq-self
insert-DiffM2
insert-subset-eq-iff
list-subtract-mset-homomorphism
remove1-idem
set-mset-mset)

lemma *list-subtract-monotonic*:
assumes *mset* $A \subseteq\#$ *mset* B
shows *mset* $(A \ominus C) \subseteq\#$ *mset* $(B \ominus C)$
by (*simp*,
meson
assms
subset-eq-diff-conv
subset-mset.dual-order.refl
subset-mset.order-trans)

lemma *map-list-subtract-mset-containment*:
mset ((*map* f A) $\ominus$ (*map* f B)) $\subseteq\#$ *mset* (*map* f $(A \ominus B)$)
by (*induct* B, *simp*, *simp*,
metis
diff-subset-eq-self
diff-zero
image-mset-add-mset
image-mset-subseteq-mono
image-mset-union
subset-eq-diff-conv
subset-eq-diff-conv)

```
lemma map-list-subtract-mset-equivalence:
  assumes mset B ⊆# mset A
  shows mset ((map f A) ⊖ (map f B)) = mset (map f (A ⊖ B))
  using assms
  by (induct B, simp, simp add: image-mset-Diff)

lemma msub-list-subtract-elem-cons-msub:
  assumes mset Ξ ⊆# mset Γ
      and ψ ∈ set (Γ ⊖ Ξ)
    shows mset (ψ # Ξ) ⊆# mset Γ
proof −
  have ∀ Γ. mset Ξ ⊆# mset Γ
           ⟶ ψ ∈ set (Γ ⊖ Ξ) −−> mset (ψ # Ξ) ⊆# mset Γ
  proof(induct Ξ)
    case Nil
    then show ?case by simp
  next
    case (Cons ξ Ξ)
    {
      fix Γ
      assume
        mset (ξ # Ξ) ⊆# mset Γ
        ψ ∈ set (Γ ⊖ (ξ # Ξ))
      hence
         ξ ∈ set Γ
         mset Ξ ⊆# mset (remove1 ξ Γ)
         ψ ∈ set ((remove1 ξ Γ) ⊖ Ξ)
        by (simp,
            metis
              ex-mset
              list.set-intros(1)
              mset.simps(2)
              mset-eq-setD
              subset-mset.le-iff-add
              union-mset-add-mset-left,
            metis
              list-subtract.simps(1)
              list-subtract.simps(2)
              list-subtract-monotonic
              remove-hd,
            simp,
            metis
              list-subtract-remove1-cons-perm
              perm-set-eq)
      with Cons.hyps have
        mset Γ = mset (ξ # (remove1 ξ Γ))
        mset (ψ # Ξ) ⊆# mset (remove1 ξ Γ)
        by (simp, blast)
      hence mset (ψ # ξ # Ξ) ⊆# mset Γ
```

```
        by (simp,
            metis
              add-mset-commute
              mset-subset-eq-add-mset-cancel)
    }
    then show ?case by auto
  qed
  thus ?thesis using assms by blast
qed
```

1.4.7 Tuple Lists

```
lemma remove1-pairs-list-projections-fst:
  assumes (γ,σ) ∈# mset Φ
  shows mset (map fst (remove1 (γ, σ) Φ)) = mset (map fst Φ) − {# γ #}
using assms
proof (induct Φ)
  case Nil
  then show ?case by simp
next
  case (Cons φ Φ)
  assume (γ, σ) ∈# mset (φ # Φ)
  show ?case
  proof (cases φ = (γ, σ))
    assume φ = (γ, σ)
    then show ?thesis by simp
  next
    assume φ ≠ (γ, σ)
    then have add-mset φ (mset Φ − {#(γ, σ)#})
            = add-mset φ (mset Φ) − {#(γ, σ)#}
        by force
    then have add-mset (fst φ) (image-mset fst (mset Φ − {#(γ, σ)#}))
                = add-mset (fst φ) (image-mset fst (mset Φ)) − {#γ#}
      by (metis (no-types) Cons.prems
                          add-mset-remove-trivial
                          fst-conv
                          image-mset-add-mset
                          insert-DiffM mset.simps(2))
    with ‹φ ≠ (γ, σ)› show ?thesis
      by simp
  qed
qed

lemma remove1-pairs-list-projections-snd:
  assumes (γ,σ) ∈# mset Φ
  shows mset (map snd (remove1 (γ, σ) Φ)) = mset (map snd Φ) − {# σ #}
using assms
proof (induct Φ)
  case Nil
```

```
  then show ?case by simp
next
  case (Cons φ Φ)
  assume (γ, σ) ∈# mset (φ # Φ)
  show ?case
  proof (cases φ = (γ, σ))
    assume φ = (γ, σ)
    then show ?thesis by simp
  next
    assume φ ≠ (γ, σ)
    then have add-mset (snd φ) (image-mset snd (mset Φ − {#(γ, σ)#}))
            = image-mset snd (mset (φ # Φ) − {#(γ, σ)#})
      by auto
    moreover have add-mset (snd φ) (image-mset snd (mset Φ))
                = add-mset σ (image-mset snd (mset (φ # Φ) − {#(γ, σ)#}))
      by (metis (no-types)
            Cons.prems
            image-mset-add-mset
            insert-DiffM
            mset.simps(2)
            snd-conv)
    ultimately
    have add-mset (snd φ) (image-mset snd (mset Φ − {#(γ, σ)#}))
              = add-mset (snd φ) (image-mset snd (mset Φ)) − {#σ#}
      by simp
    with ‹φ ≠ (γ, σ)› show ?thesis
      by simp
  qed
qed

lemma triple-list-exists:
  assumes mset (map snd Ψ) ⊆# mset Σ
      and mset Σ ⊆# mset (map snd Δ)
    shows ∃ Ω. map (λ (ψ, σ, -). (ψ, σ)) Ω = Ψ ∧
                mset (map (λ (-, σ, γ). (γ, σ)) Ω) ⊆# mset Δ
  using assms(1)
proof (induct Ψ)
  case Nil
  then show ?case by fastforce
next
  case (Cons ψ Ψ)
  from Cons obtain Ω where Ω:
    map (λ (ψ, σ, -). (ψ, σ)) Ω = Ψ
    mset (map (λ (-, σ, γ). (γ, σ)) Ω) ⊆# mset Δ
    by (metis
          (no-types, lifting)
          diff-subset-eq-self
          list.set-intros(1)
          remove1-pairs-list-projections-snd
```

remove-hd
set-mset-mset
subset-mset.dual-order.trans
surjective-pairing)
let $?\Delta_\Omega = map\ (\lambda\ (\text{-}, \sigma, \gamma).\ (\gamma, \sigma))\ \Omega$
let $?\psi = fst\ \psi$
let $?\sigma = snd\ \psi$
from *Cons.prems* **have** $add\text{-}mset\ ?\sigma\ (image\text{-}mset\ snd\ (mset\ \Psi)) \subseteq\#\ mset\ \Sigma$
by *simp*
then have $mset\ \Sigma - \{\#?\sigma\#\} - image\text{-}mset\ snd\ (mset\ \Psi)$
$\neq mset\ \Sigma - image\text{-}mset\ snd\ (mset\ \Psi)$
by (*metis*
(*no-types*)
insert-subset-eq-iff
mset-subset-eq-insertD
multi-drop-mem-not-eq
subset-mset.diff-add
subset-mset-def)
hence $?\sigma \in\#\ mset\ \Sigma - mset\ (map\ snd\ \Psi)$
using *diff-single-trivial* **by** *fastforce*
have $mset\ (map\ snd\ (\psi\ \#\ \Psi)) \subseteq\#\ mset\ (map\ snd\ \Delta)$
by (*meson*
Cons.prems
‹$mset\ \Sigma \subseteq\#\ mset\ (map\ snd\ \Delta)$›
subset-mset.dual-order.trans)
then have
$mset\ (map\ snd\ \Delta) - mset\ (map\ snd\ (\psi\ \#\ \Psi)) + (\{\#\} + \{\#snd\ \psi\#\})$
$= mset\ (map\ snd\ \Delta) + (\{\#\} + \{\#snd\ \psi\#\})$
$- add\text{-}mset\ (snd\ \psi)\ (mset\ (map\ snd\ \Psi))$
by (*metis*
(*no-types*)
list.simps(*9*)
mset.simps(*2*)
mset-subset-eq-multiset-union-diff-commute)
then have
$mset\ (map\ snd\ \Delta) - mset\ (map\ snd\ (\psi\ \#\ \Psi)) + (\{\#\} + \{\#snd\ \psi\#\})$
$= mset\ (map\ snd\ \Delta) - mset\ (map\ snd\ \Psi)$
by *auto*
hence $?\sigma \in\#\ mset\ (map\ snd\ \Delta) - mset\ (map\ snd\ \Psi)$
using *add-mset-remove-trivial-eq* **by** *fastforce*
moreover have $snd \circ (\lambda\ (\psi, \sigma, \text{-}).\ (\psi, \sigma)) = snd \circ (\lambda\ (\text{-}, \sigma, \gamma).\ (\gamma, \sigma))$
by *auto*
hence $map\ snd\ (?\Delta_\Omega) = map\ snd\ (map\ (\lambda\ (\psi, \sigma, \text{-}).\ (\psi, \sigma))\ \Omega)$
by *fastforce*
hence $map\ snd\ (?\Delta_\Omega) = map\ snd\ \Psi$
using $\Omega(1)$ **by** *simp*
ultimately have $?\sigma \in\#\ mset\ (map\ snd\ \Delta) - mset\ (map\ snd\ ?\Delta_\Omega)$
by *simp*
hence $?\sigma \in\#\ image\text{-}mset\ snd\ (mset\ \Delta - mset\ ?\Delta_\Omega)$

using $\Omega(2)$ **by** (*metis image-mset-Diff mset-map*)
hence $?\sigma \in$ *snd* ‘ *set-mset* (*mset* Δ − *mset* $?\Delta_\Omega$)
by (*metis in-image-mset*)
from *this* **obtain** ϱ **where** ϱ:
snd $\varrho = ?\sigma$ $\varrho \in\#$ *mset* Δ − *mset* $?\Delta_\Omega$
using *imageE* **by** *auto*
from *this* **obtain** γ **where**
$(\gamma, ?\sigma) = \varrho$
by (*metis prod.collapse*)
with $\varrho(2)$ **have** γ: $(\gamma, ?\sigma) \in\#$ *mset* Δ − *mset* $?\Delta_\Omega$ **by** *auto*
let $?\Omega = (?\psi, ?\sigma, \gamma)$ # Ω
have *map* (λ (ψ, σ, -). (ψ, σ)) $?\Omega = \psi$ # Ψ
using $\Omega(1)$ **by** *simp*
moreover
have *A*: (γ, *snd* ψ) = (*case* (*snd* ψ, γ) *of* (a, c) $\Rightarrow$ (c, a))
by *auto*
have *B*: *mset* (*map* (λ(b, a, c). (c, a)) Ω)
+ {# *case* (*snd* ψ, γ) *of* (a, c) $\Rightarrow$ (c, a) #}
= *mset* (*map* (λ(b, a, c). (c, a)) ((*fst* ψ, *snd* ψ, γ) # Ω))
by *simp*
obtain *mm*
:: ($'c \times 'a$) *multiset*
$\Rightarrow$ ($'c \times 'a$) *multiset*
$\Rightarrow$ ($'c \times 'a$) *multiset*
where $\forall$ *x0 x1*. ($\exists$ *v2*. *x0* = *x1* + *v2*) = (*x0* = *x1* + *mm x0 x1*)
by *moura*
then have *mset* Δ = *mset* (*map* (λ(b, a, c). (c, a)) Ω)
+ *mm* (*mset* Δ) (*mset* (*map* (λ(b, a, c). (c, a)) Ω))
by (*metis* $\Omega(2)$ *subset-mset.le-iff-add*)
then have *mset* (*map* (λ (-, σ, γ). (γ, σ)) $?\Omega$) $\subseteq\#$ *mset* Δ
using *A B* **by**
(*metis*
γ
add-diff-cancel-left'
single-subset-iff
subset-mset.add-le-cancel-left)
ultimately show *?case* **by** *meson*
qed

1.4.8 List Intersection

primrec *list-intersect* :: *'a list* => *'a list* => *'a list* (**infixl** $\cap$ *60*)
where
- $\cap$ [] = []
| *xs* $\cap$ (*y* # *ys*) =
(*if* (*y* $\in$ *set xs*)
then (*y* # (*remove1 y xs* $\cap$ *ys*))
else (*xs* $\cap$ *ys*))

lemma *list-intersect-mset-homomorphism* [*simp*]:
  mset (Φ ∩ Ψ) = *mset* Φ ∩# *mset* Ψ
proof −
  have ∀ Φ. *mset* (Φ ∩ Ψ) = *mset* Φ ∩# *mset* Ψ
  proof (*induct* Ψ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ψ Ψ)
    {
      fix Φ
      have *mset* (Φ ∩ ψ # Ψ) = *mset* Φ ∩# *mset* (ψ # Ψ)
        using *Cons.hyps*
        by (*cases* ψ ∈ *set* Φ,
            simp add: *inter-add-right2*,
            simp add: *inter-add-right1*)
    }
    then show *?case* **by** *blast*
  qed
  thus *?thesis* **by** *simp*
qed

lemma *list-intersect-left-empty* [*simp*]: [] ∩ Φ = [] **by** (*induct* Φ, *simp*+)

lemma *list-diff-intersect-comp*:
  mset Φ = *mset* (Φ ⊖ Ψ) + *mset* (Φ ∩ Ψ)
  by (*simp add*: *multiset-inter-def*)

lemma *list-intersect-left-project*: *mset* (Φ ∩ Ψ) ⊆# *mset* Φ
  by *simp*

lemma *list-intersect-right-project*: *mset* (Φ ∩ Ψ) ⊆# *mset* Ψ
  by *simp*

end

1.5 Classical Logic Connectives

Here the usual connectives for logic are given.

theory *Classical-Connectives*
  imports
    Classical-Logic-Completeness
    ../../Utilities/List-Utilities
begin

It elegant to use axiom classes for each connective and have the *classical-logic* class extend those classes. However, this would have complicated the completeness proof provided in §1.3.4. Instead, typical definitions of logical symbols are provided.

sledgehammer-params [*smt-proofs* = *false*]

1.5.1 Verum

definition (**in** *classical-logic*) *verum* :: $'a$ ($\top$)
  where
    $\top = \bot \to \bot$

lemma (**in** *classical-logic*) *verum-tautology* [*simp*]: $\vdash \top$
  by (*metis list-implication.simps*(*1*) *list-implication-axiom-k verum-def*)

lemma *verum-semantics* [*simp*]:
  $\mathfrak{M} \models_{prop} \top$
  unfolding *verum-def* **by** *simp*

lemma (**in** *classical-logic*) *verum-embedding* [*simp*]:
  $(\!|\ \top\ |\!) = \top$
  by (*simp add*: *classical-logic-class.verum-def verum-def*)

1.5.2 Conjunction

definition (**in** *classical-logic*)
  conjunction :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\sqcap$ *67*)
  where
    $\varphi \sqcap \psi = (\varphi \to \psi \to \bot) \to \bot$

primrec (**in** *classical-logic*)
  arbitrary-conjunction :: $'a$ *list* $\Rightarrow 'a$ ($\bigsqcap$)
  where
    $\bigsqcap\ [] = \top$
  $|\ \bigsqcap\ (\varphi \# \Phi) = \varphi \sqcap \bigsqcap\ \Phi$

lemma (**in** *classical-logic*) *conjunction-introduction*:
  $\vdash \varphi \to \psi \to (\varphi \sqcap \psi)$
  by (*metis*
        modus-ponens
        conjunction-def
        list-flip-implication1
        list-implication.simps(*1*)
        list-implication.simps(*2*))

lemma (**in** *classical-logic*) *conjunction-left-elimination*:
  $\vdash (\varphi \sqcap \psi) \to \varphi$
  by (*metis* (*full-types*)
        Peirces-law
        pseudo-scotus
        conjunction-def
        list-deduction-base-theory

list-deduction-modus-ponens
list-deduction-theorem
list-deduction-weaken)

lemma (**in** *classical-logic*) *conjunction-right-elimination*:
$\vdash (\varphi \sqcap \psi) \to \psi$
by (*metis* (*full-types*)
axiom-k
Contraposition
modus-ponens
conjunction-def
flip-hypothetical-syllogism
flip-implication)

lemma (**in** *classical-logic*) *conjunction-embedding* [*simp*]:
$(\!| \varphi \sqcap \psi |\!) = (\!| \varphi |\!) \sqcap (\!| \psi |\!)$
unfolding *conjunction-def classical-logic-class.conjunction-def*
by *simp*

lemma *conjunction-semantics* [*simp*]:
$\mathfrak{M} \models_{prop} \varphi \sqcap \psi = (\mathfrak{M} \models_{prop} \varphi \wedge \mathfrak{M} \models_{prop} \psi)$
unfolding *conjunction-def* **by** *simp*

1.5.3 Biconditional

definition (**in** *classical-logic*) *biconditional* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\leftrightarrow$ *75*)
where
$\varphi \leftrightarrow \psi = (\varphi \to \psi) \sqcap (\psi \to \varphi)$

lemma (**in** *classical-logic*) *biconditional-introduction*:
$\vdash (\varphi \to \psi) \to (\psi \to \varphi) \to (\varphi \leftrightarrow \psi)$
by (*simp add*: *biconditional-def conjunction-introduction*)

lemma (**in** *classical-logic*) *biconditional-left-elimination*:
$\vdash (\varphi \leftrightarrow \psi) \to \varphi \to \psi$
by (*simp add*: *biconditional-def conjunction-left-elimination*)

lemma (**in** *classical-logic*) *biconditional-right-elimination*:
$\vdash (\varphi \leftrightarrow \psi) \to \psi \to \varphi$
by (*simp add*: *biconditional-def conjunction-right-elimination*)

lemma (**in** *classical-logic*) *biconditional-embedding* [*simp*]:
$(\!| \varphi \leftrightarrow \psi |\!) = (\!| \varphi |\!) \leftrightarrow (\!| \psi |\!)$
unfolding *biconditional-def classical-logic-class.biconditional-def*
by *simp*

lemma *biconditional-semantics* [*simp*]:
$\mathfrak{M} \models_{prop} \varphi \leftrightarrow \psi = (\mathfrak{M} \models_{prop} \varphi \longleftrightarrow \mathfrak{M} \models_{prop} \psi)$
unfolding *biconditional-def*

by (*simp*, *blast*)

1.5.4 Negation

definition (**in** *classical-logic*) *negation* :: $'a \Rightarrow 'a$ $(\sim)$
where
$\sim \varphi = \varphi \rightarrow \bot$

lemma (**in** *classical-logic*) *negation-introduction*:
$\vdash (\varphi \rightarrow \bot) \rightarrow \sim \varphi$
unfolding *negation-def*
by (*metis axiom-k modus-ponens implication-absorption*)

lemma (**in** *classical-logic*) *negation-elimination*:
$\vdash \sim \varphi \rightarrow (\varphi \rightarrow \bot)$
unfolding *negation-def*
by (*metis axiom-k modus-ponens implication-absorption*)

lemma (**in** *classical-logic*) *negation-embedding* [*simp*]:
$(\!| \sim \varphi |\!) = \sim (\!| \varphi |\!)$
by (*simp add*:
classical-logic-class.negation-def
negation-def)

lemma *negation-semantics* [*simp*]:
$\mathfrak{M} \models_{prop} \sim \varphi = (\neg\, \mathfrak{M} \models_{prop} \varphi)$
unfolding *negation-def*
by *simp*

1.5.5 Disjunction

definition (**in** *classical-logic*) *disjunction* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\sqcup$ *67*)
where
$\varphi \sqcup \psi = (\varphi \rightarrow \bot) \rightarrow \psi$

primrec (**in** *classical-logic*) *arbitrary-disjunction* :: $'a$ *list* $\Rightarrow 'a$ $(\bigsqcup)$
where
$\bigsqcup\ [] = \bot$
$|\ \bigsqcup\ (\varphi \# \Phi) = \varphi \sqcup \bigsqcup\ \Phi$

lemma (**in** *classical-logic*) *disjunction-elimination*:
$\vdash (\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi) \rightarrow (\varphi \sqcup \psi) \rightarrow \chi$
proof −
let $?\Gamma = [\varphi \rightarrow \chi, \psi \rightarrow \chi, \varphi \sqcup \psi]$
have $?\Gamma :\vdash (\varphi \rightarrow \bot) \rightarrow \chi$
unfolding *disjunction-def*
by (*metis hypothetical-syllogism*
list-deduction-def
list-implication.simps(*1*)
list-implication.simps(*2*)

set-deduction-base-theory
set-deduction-theorem
set-deduction-weaken)
hence $?\Gamma :\vdash \chi$
using *excluded-middle-elimination*
list-deduction-modus-ponens
list-deduction-theorem
list-deduction-weaken
by *blast*
thus *?thesis*
unfolding *list-deduction-def*
by *simp*
qed

lemma (**in** *classical-logic*) *disjunction-left-introduction*:
$\vdash \varphi \to (\varphi \sqcup \psi)$
unfolding *disjunction-def*
by (*metis modus-ponens*
pseudo-scotus
flip-implication)

lemma (**in** *classical-logic*) *disjunction-right-introduction*:
$\vdash \psi \to (\varphi \sqcup \psi)$
unfolding *disjunction-def*
using *axiom-k*
by *simp*

lemma (**in** *classical-logic*) *disjunction-embedding* [*simp*]:
$(\!|\ \varphi \sqcup \psi\ |\!) = (\!|\ \varphi\ |\!) \sqcup (\!|\ \psi\ |\!)$
unfolding *disjunction-def classical-logic-class.disjunction-def*
by *simp*

lemma *disjunction-semantics* [*simp*]:
$\mathfrak{M} \models_{prop} \varphi \sqcup \psi = (\mathfrak{M} \models_{prop} \varphi \vee \mathfrak{M} \models_{prop} \psi)$
unfolding *disjunction-def*
by (*simp*, *blast*)

1.5.6 Mutual Exclusion

primrec (**in** *classical-logic*) *exclusive* :: $'a\ list \Rightarrow 'a$ ($\coprod$)
where
$\coprod\ [] = \top$
$\mid \coprod\ (\varphi \# \Phi) = \sim (\varphi \sqcap \bigsqcup\ \Phi) \sqcap \coprod\ \Phi$

1.5.7 Subtraction

definition (**in** *classical-logic*)
subtraction :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\setminus$ *69*)
where
$\varphi \setminus \psi = \varphi \sqcap \sim \psi$

lemma (**in** *classical-logic*) *subtraction-embedding* [*simp*]:
  $(\!|\ \varphi \setminus \psi\ |\!) = (\!|\ \varphi\ |\!) \setminus (\!|\ \psi\ |\!)$
  unfolding *subtraction-def classical-logic-class.subtraction-def*
  by *simp*

1.5.8 Negated Lists

definition (**in** *classical-logic*) *map-negation* :: *'a list* ⇒ *'a list* (**∼**)
  where [*simp*]: $\boldsymbol{\sim}\ \Phi \equiv$ *map* $\sim \Phi$

lemma (**in** *classical-logic*) *map-negation-list-implication*:
  $\vdash ((\boldsymbol{\sim}\ \Phi) :\rightarrow (\sim \varphi)) \leftrightarrow (\varphi \rightarrow \bigsqcup\ \Phi)$
proof (*induct* Φ)
  case *Nil*
  then show *?case*
    unfolding
      biconditional-def
      map-negation-def
      negation-def
    using
      conjunction-introduction
      modus-ponens
      trivial-implication
    by (*simp*, *blast*)
next
  case (*Cons* ψ Φ)
  have $\vdash (\boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi \leftrightarrow (\varphi \rightarrow \bigsqcup\ \Phi))$
      $\rightarrow (\sim \psi \rightarrow \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi) \leftrightarrow (\varphi \rightarrow (\psi \sqcup \bigsqcup\ \Phi))$
  proof −
    have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle \leftrightarrow (\langle\varphi\rangle \rightarrow \langle\bigsqcup\ \Phi\rangle)) \rightarrow$
        $(\sim \langle\psi\rangle \rightarrow \langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle) \leftrightarrow (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle))$
      by *fastforce*
    hence $\vdash (\!|\ (\langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle \leftrightarrow (\langle\varphi\rangle \rightarrow \langle\bigsqcup\ \Phi\rangle)) \rightarrow$
        $(\sim \langle\psi\rangle \rightarrow \langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle) \leftrightarrow (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle))\ |\!)$
      using *propositional-semantics* **by** *blast*
    thus *?thesis*
      by *simp*
  qed
  with *Cons* **show** *?case*
    by (*metis*
        map-negation-def
        list.simps(*9*)
        arbitrary-disjunction.simps(*2*)
        modus-ponens
        list-implication.simps(*2*))
qed

1.5.9 Common Identities

1.5.10 Biconditional Equivalence Relation

lemma (**in** *classical-logic*) *biconditional-reflection*:
  $\vdash \varphi \leftrightarrow \varphi$
  by (*meson*
      axiom-k
      modus-ponens
      biconditional-introduction
      implication-absorption)

lemma (**in** *classical-logic*) *biconditional-symmetry*:
  $\vdash (\varphi \leftrightarrow \psi) \leftrightarrow (\psi \leftrightarrow \varphi)$
  by (*metis* (*full-types*) *modus-ponens*
      biconditional-def
      conjunction-def
      flip-hypothetical-syllogism
      flip-implication)

lemma (**in** *classical-logic*) *biconditional-symmetry-rule*:
  $\vdash \varphi \leftrightarrow \psi \Longrightarrow \vdash \psi \leftrightarrow \varphi$
  by (*meson modus-ponens*
      biconditional-introduction
      biconditional-left-elimination
      biconditional-right-elimination)

lemma (**in** *classical-logic*) *biconditional-transitivity*:
    $\vdash (\varphi \leftrightarrow \psi) \rightarrow (\psi \leftrightarrow \chi) \rightarrow (\varphi \leftrightarrow \chi)$
proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \leftrightarrow \langle\psi\rangle) \rightarrow (\langle\psi\rangle \leftrightarrow \langle\chi\rangle) \rightarrow (\langle\varphi\rangle \leftrightarrow \langle\chi\rangle)$
    by *simp*
  hence $\vdash (\!|\ ((\langle\varphi\rangle \leftrightarrow \langle\psi\rangle) \rightarrow (\langle\psi\rangle \leftrightarrow \langle\chi\rangle) \rightarrow (\langle\varphi\rangle \leftrightarrow \langle\chi\rangle)\ |\!)$
    using *propositional-semantics* **by** *blast*
 thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *biconditional-transitivity-rule*:
  $\vdash \varphi \leftrightarrow \psi \Longrightarrow \vdash \psi \leftrightarrow \chi \Longrightarrow \vdash \varphi \leftrightarrow \chi$
  using *modus-ponens biconditional-transitivity* **by** *blast*

1.5.11 Biconditional Weakening

lemma (**in** *classical-logic*) *biconditional-weaken*:
  assumes $\Gamma \Vdash \varphi \leftrightarrow \psi$
  shows $\Gamma \Vdash \varphi = \Gamma \Vdash \psi$
  by (*metis assms*
      biconditional-left-elimination
      biconditional-right-elimination
      set-deduction-modus-ponens

set-deduction-weaken)

lemma (**in** *classical-logic*) *list-biconditional-weaken*:
assumes $\Gamma :\vdash \varphi \leftrightarrow \psi$
shows $\Gamma :\vdash \varphi = \Gamma :\vdash \psi$
by (*metis assms*
biconditional-left-elimination
biconditional-right-elimination
list-deduction-modus-ponens
list-deduction-weaken)

lemma (**in** *classical-logic*) *weak-biconditional-weaken*:
assumes $\vdash \varphi \leftrightarrow \psi$
shows $\vdash \varphi = \vdash \psi$
by (*metis assms*
biconditional-left-elimination
biconditional-right-elimination
modus-ponens)

1.5.12 Conjunction Identities

lemma (**in** *classical-logic*) *conjunction-negation-identity*:
$\vdash \sim (\varphi \sqcap \psi) \leftrightarrow (\varphi \rightarrow \psi \rightarrow \bot)$
by (*metis Contraposition*
double-negation-converse
modus-ponens
biconditional-introduction
conjunction-def
negation-def)

lemma (**in** *classical-logic*) *conjunction-set-deduction-equivalence* [*simp*]:
$\Gamma \Vdash \varphi \sqcap \psi = (\Gamma \Vdash \varphi \wedge \Gamma \Vdash \psi)$
by (*metis set-deduction-weaken* [**where** Γ=Γ]
set-deduction-modus-ponens [**where** Γ=Γ]
conjunction-introduction
conjunction-left-elimination
conjunction-right-elimination)

lemma (**in** *classical-logic*) *conjunction-list-deduction-equivalence* [*simp*]:
$\Gamma :\vdash \varphi \sqcap \psi = (\Gamma :\vdash \varphi \wedge \Gamma :\vdash \psi)$
by (*metis list-deduction-weaken* [**where** Γ=Γ]
list-deduction-modus-ponens [**where** Γ=Γ]
conjunction-introduction
conjunction-left-elimination
conjunction-right-elimination)

lemma (**in** *classical-logic*) *weak-conjunction-deduction-equivalence* [*simp*]:
$\vdash \varphi \sqcap \psi = (\vdash \varphi \wedge \vdash \psi)$
by (*metis conjunction-set-deduction-equivalence set-deduction-base-theory*)

lemma (**in** *classical-logic*) *conjunction-set-deduction-arbitrary-equivalence* [*simp*]:
  $\Gamma \Vdash \bigsqcap \Phi = (\forall \varphi \in set\ \Phi.\ \Gamma \Vdash \varphi)$
  by (*induct* Φ, *simp add*: *set-deduction-weaken*, *simp*)

lemma (**in** *classical-logic*) *conjunction-list-deduction-arbitrary-equivalence* [*simp*]:
  $\Gamma :\vdash \bigsqcap \Phi = (\forall \varphi \in set\ \Phi.\ \Gamma :\vdash \varphi)$
  by (*induct* Φ, *simp add*: *list-deduction-weaken*, *simp*)

lemma (**in** *classical-logic*) *weak-conjunction-deduction-arbitrary-equivalence* [*simp*]:
  $\vdash \bigsqcap \Phi = (\forall \varphi \in set\ \Phi.\ \vdash \varphi)$
  by (*induct* Φ, *simp*+)

lemma (**in** *classical-logic*) *conjunction-commutativity*:
  $\vdash (\psi \sqcap \varphi) \leftrightarrow (\varphi \sqcap \psi)$
  by (*metis*
    (*full-types*)
    modus-ponens
    biconditional-introduction
    conjunction-def
    flip-hypothetical-syllogism
    flip-implication)

lemma (**in** *classical-logic*) *conjunction-associativity*:
  $\vdash ((\varphi \sqcap \psi) \sqcap \chi) \leftrightarrow (\varphi \sqcap (\psi \sqcap \chi))$
proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcap \langle\chi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcap \langle\chi\rangle))$
    by *simp*
  hence $\vdash (\!|\ ((\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcap \langle\chi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcap \langle\chi\rangle))\ |\!)$
    using *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *arbitrary-conjunction-antitone*:
  $set\ \Phi \subseteq set\ \Psi \Longrightarrow \vdash \bigsqcap \Psi \rightarrow \bigsqcap \Phi$
proof −
  have $\forall\ \Phi.\ set\ \Phi \subseteq set\ \Psi \longrightarrow \vdash \bigsqcap \Psi \rightarrow \bigsqcap \Phi$
  proof (*induct* Ψ)
    case *Nil*
    then show *?case*
      by (*simp add*: *pseudo-scotus verum-def*)
  next
    case (*Cons* ψ Ψ)
    {
      fix Φ
      assume $set\ \Phi \subseteq set\ (\psi\ \#\ \Psi)$
      have $\vdash \bigsqcap (\psi\ \#\ \Psi) \rightarrow \bigsqcap \Phi$
      proof (*cases* $\psi \in set\ \Phi$)
        assume $\psi \in set\ \Phi$

have $\forall\ \varphi \in set\ \Phi.\ \vdash \bigsqcap\ \Phi \leftrightarrow (\varphi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))$
proof $(induct\ \Phi)$
  case *Nil*
  then show *?case* **by** *simp*
next
  case $(Cons\ \chi\ \Phi)$
  {
    fix φ
    assume $\varphi \in set\ (\chi\ \#\ \Phi)$
    have $\vdash \bigsqcap\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcap \bigsqcap\ (removeAll\ \varphi\ (\chi\ \#\ \Phi)))$
    proof *cases*
      assume $\varphi \in set\ \Phi$
      hence $\vdash \bigsqcap\ \Phi \leftrightarrow (\varphi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))$
        using *Cons.hyps* ‹$\varphi \in set\ \Phi$›
        by *auto*
      moreover
      have $\vdash (\bigsqcap\ \Phi \leftrightarrow (\varphi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))) \rightarrow$
            $(\chi \sqcap \bigsqcap\ \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))$
      proof −
        have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop}$
              $(\langle \bigsqcap\ \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle \bigsqcap\ (removeAll\ \varphi\ \Phi)\rangle))$
                $\rightarrow (\langle\chi\rangle \sqcap \langle \bigsqcap\ \Phi\rangle)$
                    $\leftrightarrow (\langle\varphi\rangle \sqcap \langle\chi\rangle \sqcap \langle \bigsqcap\ (removeAll\ \varphi\ \Phi)\rangle)$
          by *auto*
        hence $\vdash (\!|\ (\langle \bigsqcap\ \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle \bigsqcap\ (removeAll\ \varphi\ \Phi)\rangle))$
                  $\rightarrow (\langle\chi\rangle \sqcap \langle \bigsqcap\ \Phi\rangle)$
                      $\leftrightarrow (\langle\varphi\rangle \sqcap \langle\chi\rangle \sqcap \langle \bigsqcap\ (removeAll\ \varphi\ \Phi)\rangle)\ |\!)$
          using *propositional-semantics* **by** *blast*
        thus *?thesis* **by** *simp*
      qed
      ultimately have $\vdash \bigsqcap\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))$
        using *modus-ponens* **by** *auto*
      show *?thesis*
      proof *cases*
        assume $\varphi = \chi$
        moreover
        {
          fix φ
          have $\vdash (\chi \sqcap \varphi) \rightarrow (\chi \sqcap \chi \sqcap \varphi)$
            unfolding *conjunction-def*
            by (*meson*
                axiom-s
                double-negation
                modus-ponens
                flip-hypothetical-syllogism
                flip-implication)
        } **note** *tautology* = *this*
        from ‹$\vdash \bigsqcap\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap\ (removeAll\ \varphi\ \Phi))$›
            ‹$\varphi = \chi$›

have $\vdash (\chi \sqcap \bigsqcap\ (\mathit{removeAll}\ \chi\ \Phi)) \to (\chi \sqcap \bigsqcap\ \Phi)$
  unfolding *biconditional-def*
  by (*simp*, *metis tautology hypothetical-syllogism modus-ponens*)
moreover
from $\langle \vdash \bigsqcap\ (\chi \# \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap\ (\mathit{removeAll}\ \varphi\ \Phi))\rangle$
    $\langle \varphi = \chi \rangle$
have $\vdash (\chi \sqcap \bigsqcap\ \Phi) \to (\chi \sqcap \bigsqcap\ (\mathit{removeAll}\ \chi\ \Phi))$
  unfolding *biconditional-def*
  by (*simp*,
      metis conjunction-right-elimination
            hypothetical-syllogism
            modus-ponens)
ultimately show *?thesis*
  unfolding *biconditional-def*
  by *simp*
next
  assume $\varphi \neq \chi$
  then show *?thesis*
    using $\langle \vdash \bigsqcap\ (\chi \# \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap\ (\mathit{removeAll}\ \varphi\ \Phi))\rangle$
    by *simp*
qed
next
  assume $\varphi \notin \mathit{set}\ \Phi$
  hence $\varphi = \chi\ \chi \notin \mathit{set}\ \Phi$
    using $\langle \varphi \in \mathit{set}\ (\chi \# \Phi)\rangle$ by *auto*
  then show *?thesis*
    using *biconditional-reflection*
    by *simp*
qed
}
thus *?case* by *blast*
qed
hence $\vdash (\psi \sqcap \bigsqcap\ (\mathit{removeAll}\ \psi\ \Phi)) \to \bigsqcap\ \Phi$
  using *modus-ponens biconditional-right-elimination* $\langle \psi \in \mathit{set}\ \Phi\rangle$
  by *blast*
moreover
from $\langle \psi \in \mathit{set}\ \Phi\rangle$ $\langle \mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi \# \Psi)\rangle$ *Cons.hyps*
have $\vdash \bigsqcap\ \Psi \to \bigsqcap\ (\mathit{removeAll}\ \psi\ \Phi)$
  by (*simp add*: *subset-insert-iff insert-absorb*)
hence $\vdash (\psi \sqcap \bigsqcap\ \Psi) \to (\psi \sqcap \bigsqcap\ (\mathit{removeAll}\ \psi\ \Phi))$
  unfolding *conjunction-def*
  using
    modus-ponens
    hypothetical-syllogism
    flip-hypothetical-syllogism
  by *meson*
ultimately have $\vdash (\psi \sqcap \bigsqcap\ \Psi) \to \bigsqcap\ \Phi$
  using *modus-ponens hypothetical-syllogism*
  by *blast*

thus *?thesis*
  by *simp*
next
  assume $\psi \notin$ *set* Φ
  hence $\vdash \bigsqcap\ \Psi \rightarrow \bigsqcap\ \Phi$
    using *Cons.hyps* ‹*set* $\Phi \subseteq$ *set* $(\psi\ \#\ \Psi)$›
    by *auto*
  hence $\vdash (\psi \sqcap \bigsqcap\ \Psi) \rightarrow \bigsqcap\ \Phi$
    unfolding *conjunction-def*
    by (*metis*
      modus-ponens
      conjunction-def
      conjunction-right-elimination
      hypothetical-syllogism)
  thus *?thesis*
    by *simp*
qed
}
thus *?case* **by** *blast*
qed
thus *set* $\Phi \subseteq$ *set* $\Psi \Longrightarrow \vdash \bigsqcap\ \Psi \rightarrow \bigsqcap\ \Phi$ **by** *blast*
qed

lemma (**in** *classical-logic*) *arbitrary-conjunction-remdups*:
$\vdash (\bigsqcap\ \Phi) \leftrightarrow \bigsqcap\ (remdups\ \Phi)$
by (*simp add*: *arbitrary-conjunction-antitone biconditional-def*)

lemma (**in** *classical-logic*) *curry-uncurry*:
$\vdash (\varphi \rightarrow \psi \rightarrow \chi) \leftrightarrow ((\varphi \sqcap \psi) \rightarrow \chi)$
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \rightarrow \langle\psi\rangle \rightarrow \langle\chi\rangle) \leftrightarrow (((\langle\varphi\rangle \sqcap \langle\psi\rangle) \rightarrow \langle\chi\rangle)$
  by *auto*
hence $\vdash (\!|\ ((\langle\varphi\rangle \rightarrow \langle\psi\rangle \rightarrow \langle\chi\rangle) \leftrightarrow (((\langle\varphi\rangle \sqcap \langle\psi\rangle) \rightarrow \langle\chi\rangle)\ |\!)$
  using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *list-curry-uncurry*:
$\vdash (\Phi :\rightarrow \chi) \leftrightarrow (\bigsqcap\ \Phi \rightarrow \chi)$
proof (*induct* Φ)
case *Nil*
have $\vdash \chi \leftrightarrow (\top \rightarrow \chi)$
  unfolding *biconditional-def*
    conjunction-def
    verum-def
  using
    axiom-k
    ex-falso-quodlibet
    modus-ponens

conjunction-def
excluded-middle-elimination
set-deduction-base-theory
conjunction-set-deduction-equivalence
by *metis*
with *Nil* **show** *?case*
by *simp*
next
case (*Cons* φ Φ)
have $\vdash$ $((\varphi \ \# \ \Phi) :\to \chi) \leftrightarrow (\varphi \to (\Phi :\to \chi))$
by (*simp add*: *biconditional-reflection*)
with *Cons* **have** $\vdash$ $((\varphi \ \# \ \Phi) :\to \chi) \leftrightarrow (\varphi \to \bigsqcap \ \Phi \to \chi)$
by (*metis modus-ponens*
biconditional-def
hypothetical-syllogism
list-implication.simps(*2*)
weak-conjunction-deduction-equivalence)
with *curry-uncurry* [**where** *?*φ=φ **and** *?*ψ=$\bigsqcap$ Φ **and** *?*χ=χ]
show *?case*
unfolding *biconditional-def*
by (*simp*, *metis modus-ponens hypothetical-syllogism*)
qed

1.5.13 Disjunction Identities

lemma (**in** *classical-logic*) *bivalence*:
$\vdash \sim \varphi \sqcup \varphi$
by (*simp add*: *double-negation disjunction-def negation-def*)

lemma (**in** *classical-logic*) *implication-equivalence*:
$\vdash (\sim \varphi \sqcup \psi) \leftrightarrow (\varphi \to \psi)$
by (*metis double-negation-converse*
modus-ponens
biconditional-introduction
bivalence
disjunction-def
flip-hypothetical-syllogism
negation-def)

lemma (**in** *classical-logic*) *disjunction-commutativity*:
$\vdash (\psi \sqcup \varphi) \leftrightarrow (\varphi \sqcup \psi)$
by (*meson modus-ponens*
biconditional-introduction
disjunction-elimination
disjunction-left-introduction
disjunction-right-introduction)

lemma (**in** *classical-logic*) *disjunction-associativity*:
$\vdash ((\varphi \sqcup \psi) \sqcup \chi) \leftrightarrow (\varphi \sqcup (\psi \sqcup \chi))$

proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcup \langle\chi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle))$
    by *simp*
  hence $\vdash (\!|\ (((\langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcup \langle\chi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle))\ |\!)$
    using *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *arbitrary-disjunction-monotone*:
  $set\ \Phi \subseteq set\ \Psi \Longrightarrow \vdash \bigsqcup\ \Phi \to \bigsqcup\ \Psi$
proof −
  have $\forall\ \Phi.\ set\ \Phi \subseteq set\ \Psi \longrightarrow \vdash \bigsqcup\ \Phi \to \bigsqcup\ \Psi$
  proof (*induct* Ψ)
    case *Nil*
    then show *?case* **using** *verum-def verum-tautology* **by** *auto*
  next
    case (*Cons* ψ Ψ)
    {
      fix Φ
      assume $set\ \Phi \subseteq set\ (\psi\ \#\ \Psi)$
      have $\vdash \bigsqcup\ \Phi \to \bigsqcup\ (\psi\ \#\ \Psi)$
      proof *cases*
        assume $\psi \in set\ \Phi$
        have $\forall\ \varphi \in set\ \Phi.\ \vdash \bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$
        proof (*induct* Φ)
          case *Nil*
          then show *?case* **by** *simp*
        next
          case (*Cons* χ Φ)
          {
            fix φ
            assume $\varphi \in set\ (\chi\ \#\ \Phi)$
            have $\vdash \bigsqcup\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcup \bigsqcup\ (removeAll\ \varphi\ (\chi\ \#\ \Phi)))$
            proof *cases*
              assume $\varphi \in set\ \Phi$
              hence $\vdash \bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$
                using *Cons.hyps* ‹$\varphi \in set\ \Phi$›
                by *auto*
              moreover
              have $\vdash (\bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))) \to$
                  $(\chi \sqcup \bigsqcup\ \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$
              proof −
                have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop}$
                    $(\langle\bigsqcup\ \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcup \langle\bigsqcup\ (removeAll\ \varphi\ \Phi)\rangle))$
                    $\to (\langle\chi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle)$
                      $\leftrightarrow (\langle\varphi\rangle \sqcup \langle\chi\rangle \sqcup \langle\bigsqcup\ (removeAll\ \varphi\ \Phi)\rangle)$
                  by *auto*
                hence $\vdash (\!|\ (\langle\bigsqcup\ \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcup \langle\bigsqcup\ (removeAll\ \varphi\ \Phi)\rangle))$
                    $\to (\langle\chi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle)$

$\leftrightarrow (\langle\varphi\rangle \sqcup \langle\chi\rangle \sqcup \langle\bigsqcup\ (removeAll\ \varphi\ \Phi)\rangle)\ \|)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately have $\vdash \bigsqcup\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$
using *modus-ponens* **by** *auto*
show *?thesis*
proof *cases*
assume $\varphi = \chi$
then show *?thesis*
using ‹$\vdash \bigsqcup\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$›
unfolding *biconditional-def*
by (*simp add*: *disjunction-def*,
meson
axiom-k
modus-ponens
flip-hypothetical-syllogism
implication-absorption)
next
assume $\varphi \neq \chi$
then show *?thesis*
using ‹$\vdash \bigsqcup\ (\chi\ \#\ \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \bigsqcup\ (removeAll\ \varphi\ \Phi))$›
by *simp*
qed
next
assume $\varphi \notin set\ \Phi$
hence $\varphi = \chi\ \chi \notin set\ \Phi$
using ‹$\varphi \in set\ (\chi\ \#\ \Phi)$› **by** *auto*
then show *?thesis*
using *biconditional-reflection*
by *simp*
qed
}
thus *?case* **by** *blast*
qed
hence $\vdash \bigsqcup\ \Phi \to (\psi \sqcup \bigsqcup\ (removeAll\ \psi\ \Phi))$
using *modus-ponens biconditional-left-elimination* ‹$\psi \in set\ \Phi$›
by *blast*
moreover
from ‹$\psi \in set\ \Phi$› ‹$set\ \Phi \subseteq set\ (\psi\ \#\ \Psi)$› *Cons.hyps*
have $\vdash \bigsqcup\ (removeAll\ \psi\ \Phi) \to \bigsqcup\ \Psi$
by (*simp add*: *subset-insert-iff insert-absorb*)
hence $\vdash (\psi \sqcup \bigsqcup\ (removeAll\ \psi\ \Phi)) \to \bigsqcup\ (\psi\ \#\ \Psi)$
using
modus-ponens
disjunction-def
hypothetical-syllogism
by *fastforce*
ultimately show *?thesis*

```
        by (simp, metis modus-ponens hypothetical-syllogism)
      next
        assume ψ ∉ set Φ
        hence ⊢ ⨆ Φ → ⨆ Ψ
          using Cons.hyps ‹set Φ ⊆ set (ψ # Ψ)›
          by auto
        then show ?thesis
          by (metis
                arbitrary-disjunction.simps(2)
                disjunction-def
                list-deduction-def
                list-deduction-theorem
                list-deduction-weaken
                list-implication.simps(1)
                list-implication.simps(2))
      qed
    }
    then show ?case by blast
  qed
  thus set Φ ⊆ set Ψ ⟹ ⊢ ⨆ Φ → ⨆ Ψ by blast
qed

lemma (in classical-logic) arbitrary-disjunction-remdups:
  ⊢ (⨆ Φ) ↔ ⨆ (remdups Φ)
  by (simp add: arbitrary-disjunction-monotone biconditional-def)

lemma (in classical-logic) arbitrary-disjunction-exclusion-MCS:
  assumes MCS Ω
  shows ⨆ Ψ ∉ Ω ≡ ∀ ψ ∈ set Ψ. ψ ∉ Ω
proof (induct Ψ)
  case Nil
  then show ?case
    using
      assms
      formula-consistent-def
      formula-maximally-consistent-set-def-def
      maximally-consistent-set-def
      set-deduction-reflection
    by (simp, blast)
next
  case (Cons ψ Ψ)
  have ⨆ (ψ # Ψ) ∉ Ω = (ψ ∉ Ω ∧ ⨆ Ψ ∉ Ω)
    by (simp add: disjunction-def,
        meson
          assms
          formula-consistent-def
          formula-maximally-consistent-set-def-def
          formula-maximally-consistent-set-def-implication
          maximally-consistent-set-def
```

```
          set-deduction-reflection)
  thus ?case using Cons.hyps by simp
qed

lemma (in classical-logic) contra-list-curry-uncurry:
  ⊢ (Φ :→ χ) ↔ (∼ χ → ⨆ (∼ Φ))
proof (induct Φ)
  case Nil
  then show ?case
    by (simp,
          metis
            biconditional-introduction
            bivalence
            disjunction-def
            double-negation-converse
            modus-ponens
            negation-def)
next
  case (Cons φ Φ)
  hence ⊢ (⨅ Φ → χ) ↔ (∼ χ → ⨆ (∼ Φ))
    by (metis
          biconditional-symmetry-rule
          biconditional-transitivity-rule
          list-curry-uncurry)
  have ⊢ (⨅ (φ # Φ) → χ) ↔ (∼ χ → ⨆ (∼ (φ # Φ)))
  proof −
    have ⊢ (⨅ Φ → χ) ↔ (∼ χ → ⨆ (∼ Φ))
            → ((φ ⊓ ⨅ Φ) → χ) ↔ (∼ χ → (∼ φ ⊔ ⨆ (∼ Φ)))
    proof −
      have
       ∀ 𝔐. 𝔐 ⊨prop
         (⟨⨅ Φ⟩ → ⟨χ⟩) ↔ (∼ ⟨χ⟩ → ⟨⨆ (∼ Φ)⟩)
            → ((⟨φ⟩ ⊓ ⟨⨅ Φ⟩) → ⟨χ⟩) ↔ (∼ ⟨χ⟩ → (∼ ⟨φ⟩ ⊔ ⟨⨆ (∼ Φ)⟩))
        by auto
      hence
       ⊢ ⦇ (⟨⨅ Φ⟩ → ⟨χ⟩) ↔ (∼ ⟨χ⟩ → ⟨⨆ (∼ Φ)⟩)
            → ((⟨φ⟩ ⊓ ⟨⨅ Φ⟩) → ⟨χ⟩) ↔ (∼ ⟨χ⟩ → (∼ ⟨φ⟩ ⊔ ⟨⨆ (∼ Φ)⟩)) ⦈
        using propositional-semantics by blast
      thus ?thesis by simp
    qed
    thus ?thesis
      using ‹⊢ (⨅ Φ → χ) ↔ (∼ χ → ⨆ (∼ Φ))› modus-ponens by auto
  qed
  then show ?case
    using biconditional-transitivity-rule list-curry-uncurry by blast
qed
```

1.5.14 Monotony of Conjunction and Disjunction

lemma (**in** *classical-logic*) *conjunction-monotonic-identity*:
 $\vdash (\varphi \to \psi) \to (\varphi \sqcap \chi) \to (\psi \sqcap \chi)$
 unfolding *conjunction-def*
 using *modus-ponens*
  flip-hypothetical-syllogism
 by *blast*

lemma (**in** *classical-logic*) *conjunction-monotonic*:
 assumes $\vdash \varphi \to \psi$
 shows $\vdash (\varphi \sqcap \chi) \to (\psi \sqcap \chi)$
 using *assms*
  modus-ponens
  conjunction-monotonic-identity
 by *blast*

lemma (**in** *classical-logic*) *disjunction-monotonic-identity*:
 $\vdash (\varphi \to \psi) \to (\varphi \sqcup \chi) \to (\psi \sqcup \chi)$
 unfolding *disjunction-def*
 using *modus-ponens*
  flip-hypothetical-syllogism
 by *blast*

lemma (**in** *classical-logic*) *disjunction-monotonic*:
 assumes $\vdash \varphi \to \psi$
 shows $\vdash (\varphi \sqcup \chi) \to (\psi \sqcup \chi)$
 using *assms*
  modus-ponens
  disjunction-monotonic-identity
 by *blast*

1.5.15 Distribution Identities

lemma (**in** *classical-logic*) *conjunction-distribution*:
 $\vdash ((\psi \sqcup \chi) \sqcap \varphi) \leftrightarrow ((\psi \sqcap \varphi) \sqcup (\chi \sqcap \varphi))$
proof −
 have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\psi\rangle \sqcup \langle\chi\rangle) \sqcap \langle\varphi\rangle) \leftrightarrow (((\langle\psi\rangle \sqcap \langle\varphi\rangle) \sqcup (\langle\chi\rangle \sqcap \langle\varphi\rangle)))$
  by *auto*
 hence $\vdash (\!|\ (((\langle\psi\rangle \sqcup \langle\chi\rangle) \sqcap \langle\varphi\rangle) \leftrightarrow (((\langle\psi\rangle \sqcap \langle\varphi\rangle) \sqcup (\langle\chi\rangle \sqcap \langle\varphi\rangle)))\ |\!)$
  using *propositional-semantics* **by** *blast*
 thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *subtraction-distribution*:
 $\vdash ((\psi \sqcup \chi) \setminus \varphi) \leftrightarrow ((\psi \setminus \varphi) \sqcup (\chi \setminus \varphi))$
 by (*simp add*: *conjunction-distribution subtraction-def*)

lemma (**in** *classical-logic*) *conjunction-arbitrary-distribution*:
 $\vdash (\bigsqcup\ \Psi \sqcap \varphi) \leftrightarrow \bigsqcup\ [\psi \sqcap \varphi.\ \psi \leftarrow \Psi]$

proof (*induct* Ψ)
  case *Nil*
  then show *?case*
    by (*simp add*: *ex-falso-quodlibet*
                  biconditional-def
                  conjunction-left-elimination)
next
  case (*Cons* ψ Ψ)
  have $\vdash (\bigsqcup\ (\psi \# \Psi) \sqcap \varphi) \leftrightarrow ((\psi \sqcap \varphi) \sqcup ((\bigsqcup\ \Psi) \sqcap \varphi))$
    using *conjunction-distribution* **by** *auto*
  moreover
  from *Cons* **have**
    $\vdash ((\psi \sqcap \varphi) \sqcup ((\bigsqcup\ \Psi) \sqcap \varphi)) \leftrightarrow ((\psi \sqcap \varphi) \sqcup (\bigsqcup\ [\psi \sqcap \varphi.\ \psi \leftarrow \Psi]))$
    unfolding *disjunction-def biconditional-def*
    by (*simp*, *meson modus-ponens hypothetical-syllogism*)
  ultimately show *?case*
    by (*simp*, *metis biconditional-transitivity-rule*)
qed

lemma (**in** *classical-logic*) *subtraction-arbitrary-distribution*:
  $\vdash (\bigsqcup\ \Psi \setminus \varphi) \leftrightarrow \bigsqcup\ [\psi \setminus \varphi.\ \psi \leftarrow \Psi]$
  by (*simp add*: *conjunction-arbitrary-distribution subtraction-def*)

lemma (**in** *classical-logic*) *disjunction-distribution*:
  $\vdash (\varphi \sqcup (\psi \sqcap \chi)) \leftrightarrow ((\varphi \sqcup \psi) \sqcap (\varphi \sqcup \chi))$
proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcap \langle\chi\rangle)) \leftrightarrow ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcup \langle\chi\rangle))$
      by *auto*
  hence $\vdash (\!|\ (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcap \langle\chi\rangle)) \leftrightarrow ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcup \langle\chi\rangle))\ |\!)$
    using *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *implication-distribution*:
  $\vdash (\varphi \rightarrow (\psi \sqcap \chi)) \leftrightarrow ((\varphi \rightarrow \psi) \sqcap (\varphi \rightarrow \chi))$
proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcap \langle\chi\rangle)) \leftrightarrow ((\langle\varphi\rangle \rightarrow \langle\psi\rangle) \sqcap (\langle\varphi\rangle \rightarrow \langle\chi\rangle))$
      by *auto*
  hence $\vdash (\!|\ (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcap \langle\chi\rangle)) \leftrightarrow ((\langle\varphi\rangle \rightarrow \langle\psi\rangle) \sqcap (\langle\varphi\rangle \rightarrow \langle\chi\rangle))\ |\!)$
    using *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *list-implication-distribution*:
  $\vdash (\Phi :\rightarrow (\psi \sqcap \chi)) \leftrightarrow ((\Phi :\rightarrow \psi) \sqcap (\Phi :\rightarrow \chi))$
proof (*induct* Φ)
  case *Nil*
  then show *?case*
    by (*simp add*: *biconditional-reflection*)

next
  case (*Cons* φ Φ)
  hence $\vdash (\varphi \# \Phi) :\rightarrow (\psi \sqcap \chi) \leftrightarrow (\varphi \rightarrow (\Phi :\rightarrow \psi \sqcap \Phi :\rightarrow \chi))$
    by (*metis*
        modus-ponens
        biconditional-def
        hypothetical-syllogism
        list-implication.simps(*2*)
        weak-conjunction-deduction-equivalence)
  moreover
  have $\vdash (\varphi \rightarrow (\Phi :\rightarrow \psi \sqcap \Phi :\rightarrow \chi)) \leftrightarrow (((\varphi \# \Phi) :\rightarrow \psi) \sqcap ((\varphi \# \Phi) :\rightarrow \chi))$
    using *implication-distribution* **by** *auto*
  ultimately show *?case*
    by (*simp*, *metis biconditional-transitivity-rule*)
qed

lemma (**in** *classical-logic*) *biconditional-conjunction-weaken*:
  $\vdash (\alpha \leftrightarrow \beta) \rightarrow ((\gamma \sqcap \alpha) \leftrightarrow (\gamma \sqcap \beta))$
proof −
  have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\alpha\rangle \leftrightarrow \langle\beta\rangle) \rightarrow ((\langle\gamma\rangle \sqcap \langle\alpha\rangle) \leftrightarrow (\langle\gamma\rangle \sqcap \langle\beta\rangle))$
      by *auto*
  hence $\vdash (\!|\ (\langle\alpha\rangle \leftrightarrow \langle\beta\rangle) \rightarrow ((\langle\gamma\rangle \sqcap \langle\alpha\rangle) \leftrightarrow (\langle\gamma\rangle \sqcap \langle\beta\rangle))\ |\!)$
    using *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *biconditional-conjunction-weaken-rule*:
  $\vdash (\alpha \leftrightarrow \beta) \Longrightarrow \vdash (\gamma \sqcap \alpha) \leftrightarrow (\gamma \sqcap \beta)$
  using *modus-ponens biconditional-conjunction-weaken* **by** *blast*

lemma (**in** *classical-logic*) *disjunction-arbitrary-distribution*:
  $\vdash (\varphi \sqcup \prod \Psi) \leftrightarrow \prod [\varphi \sqcup \psi.\ \psi \leftarrow \Psi]$
proof (*induct* Ψ)
  case *Nil*
  then show *?case*
    unfolding *disjunction-def biconditional-def*
    using *axiom-k modus-ponens verum-tautology*
    by (*simp*, *blast*)
next
  case (*Cons* ψ Ψ)
  have $\vdash (\varphi \sqcup \prod (\psi \# \Psi)) \leftrightarrow ((\varphi \sqcup \psi) \sqcap (\varphi \sqcup \prod \Psi))$
    by (*simp add*: *disjunction-distribution*)
  moreover
  from *biconditional-conjunction-weaken-rule*
      Cons
  have $\vdash ((\varphi \sqcup \psi) \sqcap \varphi \sqcup \prod \Psi) \leftrightarrow \prod (map\ (\lambda\ \chi\ .\ \varphi \sqcup \chi)\ (\psi \# \Psi))$
    by *simp*
  ultimately show *?case*
    by (*metis biconditional-transitivity-rule*)

qed

lemma (**in** *classical-logic*) *list-implication-arbitrary-distribution*:
$\vdash (\Phi :\rightarrow \bigsqcap \Psi) \leftrightarrow \bigsqcap [\Phi :\rightarrow \psi.\ \psi \leftarrow \Psi]$
proof (*induct* Ψ)
case *Nil*
then show *?case*
by (*simp add*: *biconditional-def*,
meson
axiom-k
modus-ponens
list-implication-axiom-k
verum-tautology)
next
case (*Cons* ψ Ψ)
have $\vdash \Phi :\rightarrow \bigsqcap (\psi \# \Psi) \leftrightarrow (\Phi :\rightarrow \psi \sqcap \Phi :\rightarrow \bigsqcap \Psi)$
using *list-implication-distribution*
by *fastforce*
moreover
from *biconditional-conjunction-weaken-rule*
Cons
have $\vdash (\Phi :\rightarrow \psi \sqcap \Phi :\rightarrow \bigsqcap \Psi) \leftrightarrow \bigsqcap [\Phi :\rightarrow \psi.\ \psi \leftarrow (\psi \# \Psi)]$
by *simp*
ultimately show *?case*
by (*metis biconditional-transitivity-rule*)
qed

lemma (**in** *classical-logic*) *implication-arbitrary-distribution*:
$\vdash (\varphi \rightarrow \bigsqcap \Psi) \leftrightarrow \bigsqcap [\varphi \rightarrow \psi.\ \psi \leftarrow \Psi]$
using *list-implication-arbitrary-distribution* [**where** *?*Φ=[φ]]
by *simp*

1.5.16 Negation

lemma (**in** *classical-logic*) *double-negation-biconditional*:
$\vdash \sim (\sim \varphi) \leftrightarrow \varphi$
unfolding *biconditional-def negation-def*
by (*simp add*: *double-negation double-negation-converse*)

lemma (**in** *classical-logic*) *double-negation-elimination* [*simp*]:
$\Gamma \Vdash \sim (\sim \varphi) = \Gamma \Vdash \varphi$
using
set-deduction-weaken
biconditional-weaken
double-negation-biconditional
by *metis*

lemma (**in** *classical-logic*) *alt-double-negation-elimination* [*simp*]:
$\Gamma \Vdash (\varphi \rightarrow \bot) \rightarrow \bot \equiv \Gamma \Vdash \varphi$

using *double-negation-elimination*
unfolding *negation-def*
by *auto*

lemma (**in** *classical-logic*) *base-double-negation-elimination* [*simp*]:
$\vdash \sim (\sim \varphi) = \vdash \varphi$
by (*metis double-negation-elimination set-deduction-base-theory*)

lemma (**in** *classical-logic*) *alt-base-double-negation-elimination* [*simp*]:
$\vdash (\varphi \rightarrow \bot) \rightarrow \bot \equiv \vdash \varphi$
using *base-double-negation-elimination*
unfolding *negation-def*
by *auto*

1.5.17 Mutual Exclusion Identities

lemma (**in** *classical-logic*) *exclusion-contrapositive-equivalence*:
$\vdash (\varphi \rightarrow \gamma) \leftrightarrow \sim (\varphi \sqcap \sim \gamma)$
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \rightarrow \langle\gamma\rangle) \leftrightarrow \sim (\langle\varphi\rangle \sqcap \sim \langle\gamma\rangle)$
by *auto*
hence $\vdash (\!|\ (\langle\varphi\rangle \rightarrow \langle\gamma\rangle) \leftrightarrow \sim (\langle\varphi\rangle \sqcap \sim \langle\gamma\rangle)\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *disjuction-exclusion-equivalence*:
$\Gamma \Vdash \sim (\psi \sqcap \bigsqcup\ \Phi) \equiv \forall\ \varphi \in set\ \Phi.\ \Gamma \Vdash \sim (\psi \sqcap \varphi)$
proof (*induct* Φ)
case *Nil*
then show *?case*
by (*simp add*:
conjunction-right-elimination
negation-def
set-deduction-weaken)
next
case (*Cons* φ Φ)
have $\vdash \sim (\psi \sqcap \bigsqcup\ (\varphi \# \Phi)) \leftrightarrow \sim (\psi \sqcap (\varphi \sqcup \bigsqcup\ \Phi))$
by (*simp add*: *biconditional-reflection*)
moreover have $\vdash \sim (\psi \sqcap (\varphi \sqcup \bigsqcup\ \Phi)) \leftrightarrow (\sim (\psi \sqcap \varphi) \sqcap \sim (\psi \sqcap \bigsqcup\ \Phi))$
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \sim (\langle\psi\rangle \sqcap (\langle\varphi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle))$
$\leftrightarrow (\sim (\langle\psi\rangle \sqcap \langle\varphi\rangle) \sqcap \sim (\langle\psi\rangle \sqcap \langle\bigsqcup\ \Phi\rangle))$
by *auto*
hence $\vdash (\!|\ \sim (\langle\psi\rangle \sqcap (\langle\varphi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle))$
$\leftrightarrow (\sim (\langle\psi\rangle \sqcap \langle\varphi\rangle) \sqcap \sim (\langle\psi\rangle \sqcap \langle\bigsqcup\ \Phi\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

```
  ultimately
  have ⊢ ∼ (ψ ⊓ ⨆ (φ # Φ)) ↔ (∼ (ψ ⊓ φ) ⊓ ∼ (ψ ⊓ ⨆ Φ))
    by simp
  hence Γ ⊩ ∼ (ψ ⊓ ⨆ (φ # Φ)) = (Γ ⊩ ∼ (ψ ⊓ φ)
          ∧ (∀φ∈set Φ. Γ ⊩ ∼ (ψ ⊓ φ)))
    using set-deduction-weaken [where Γ=Γ]
          conjunction-set-deduction-equivalence [where Γ=Γ]
          Cons.hyps
          biconditional-def
          set-deduction-modus-ponens
    by metis
  thus Γ ⊩ ∼ (ψ ⊓ ⨆ (φ # Φ)) = (∀φ∈set (φ # Φ). Γ ⊩ ∼ (ψ ⊓ φ))
    by simp
qed

lemma (in classical-logic) exclusive-elimination1:
  assumes Γ ⊩ ∐ Φ
  shows ∀ φ ∈ set Φ. ∀ ψ ∈ set Φ. (φ ≠ ψ) ⟶ Γ ⊩ ∼ (φ ⊓ ψ)
  using assms
proof (induct Φ)
  case Nil
  thus ?case by auto
next
  case (Cons χ Φ)
  assume Γ ⊩ ∐ (χ # Φ)
  hence Γ ⊩ ∐ Φ by simp
  hence ∀φ∈set Φ. ∀ψ∈set Φ. φ ≠ ψ ⟶ Γ ⊩ ∼ (φ ⊓ ψ)
    using Cons.hyps by blast
  moreover have Γ ⊩ ∼ (χ ⊓ ⨆ Φ)
    using ‹Γ ⊩ ∐ (χ # Φ)› conjunction-set-deduction-equivalence by auto
  hence ∀ φ ∈ set Φ. Γ ⊩ ∼ (χ ⊓ φ)
    using disjuction-exclusion-equivalence by auto
  moreover {
    fix φ
    have ⊢ ∼ (χ ⊓ φ) → ∼ (φ ⊓ χ)
      unfolding negation-def
                conjunction-def
      using modus-ponens flip-hypothetical-syllogism flip-implication by blast
  }
  with ‹∀ φ ∈ set Φ. Γ ⊩ ∼ (χ ⊓ φ)› have ∀ φ ∈ set Φ. Γ ⊩ ∼ (φ ⊓ χ)
    using set-deduction-weaken [where Γ=Γ]
          set-deduction-modus-ponens [where Γ=Γ]
    by blast
  ultimately
  show ∀φ ∈ set (χ # Φ). ∀ψ ∈ set (χ # Φ). φ ≠ ψ ⟶ Γ ⊩ ∼ (φ ⊓ ψ)
    by simp
qed

lemma (in classical-logic) exclusive-elimination2:
```

```
  assumes Γ ⊩ ∐ Φ
  shows ∀ φ ∈ duplicates Φ. Γ ⊩ ∼ φ
  using assms
proof (induct Φ)
  case Nil
  then show ?case by simp
next
  case (Cons φ Φ)
  assume Γ ⊩ ∐ (φ # Φ)
  hence Γ ⊩ ∐ Φ by simp
  hence ∀ φ∈duplicates Φ. Γ ⊩ ∼ φ using Cons.hyps by auto
  show ?case
  proof cases
    assume φ ∈ set Φ
    moreover {
      fix φ ψ χ
      have ⊢ ∼ (φ ⊓ (ψ ⊔ χ)) ↔ (∼ (φ ⊓ ψ) ⊓ ∼ (φ ⊓ χ))
      proof −
        have ∀ 𝔐. 𝔐 ⊨prop ∼ (⟨φ⟩ ⊓ (⟨ψ⟩ ⊔ ⟨χ⟩))
                        ↔ (∼ (⟨φ⟩ ⊓ ⟨ψ⟩) ⊓ ∼ (⟨φ⟩ ⊓ ⟨χ⟩))
          by auto
        hence ⊢ ⦇ ∼ (⟨φ⟩ ⊓ (⟨ψ⟩ ⊔ ⟨χ⟩)) ↔ (∼ (⟨φ⟩ ⊓ ⟨ψ⟩) ⊓ ∼ (⟨φ⟩ ⊓ ⟨χ⟩)) ⦈
          using propositional-semantics by blast
        thus ?thesis by simp
      qed
      hence Γ ⊩ ∼ (φ ⊓ (ψ ⊔ χ)) ≡ Γ ⊩ ∼ (φ ⊓ ψ) ⊓ ∼ (φ ⊓ χ)
        using set-deduction-weaken
            biconditional-weaken by presburger
    }
    moreover
    have ⊢ ∼ (φ ⊓ φ) ↔ ∼ φ
    proof −
      have ∀ 𝔐. 𝔐 ⊨prop ∼ (⟨φ⟩ ⊓ ⟨φ⟩) ↔ ∼ ⟨φ⟩
        by auto
      hence ⊢ ⦇ ∼ (⟨φ⟩ ⊓ ⟨φ⟩) ↔ ∼ ⟨φ⟩ ⦈
        using propositional-semantics by blast
      thus ?thesis by simp
    qed
    hence Γ ⊩ ∼ (φ ⊓ φ) ≡ Γ ⊩ ∼ φ
      using set-deduction-weaken
          biconditional-weaken by presburger
    moreover have Γ ⊩ ∼ (φ ⊓ ⨆ Φ) using ‹Γ ⊩ ∐ (φ # Φ)› by simp
    ultimately have Γ ⊩ ∼ φ by (induct Φ, simp, simp, blast)
    thus ?thesis using ‹φ ∈ set Φ› ‹∀ φ∈duplicates Φ. Γ ⊩ ∼ φ› by simp
  next
    assume φ ∉ set Φ
    hence duplicates (φ # Φ) = duplicates Φ by simp
    then show ?thesis using ‹∀ φ∈duplicates Φ. Γ ⊩ ∼ φ›
      by auto
```

```
  qed
qed

lemma (in classical-logic) exclusive-equivalence:
  Γ ⊩ ∐ Φ =
     ((∀φ∈duplicates Φ. Γ ⊩ ∼ φ) ∧
       (∀ φ ∈ set Φ. ∀ ψ ∈ set Φ. (φ ≠ ψ) ⟶ Γ ⊩ ∼ (φ ⊓ ψ)))
proof −
  {
    assume ∀φ∈duplicates Φ. Γ ⊩ ∼ φ
           ∀ φ ∈ set Φ. ∀ ψ ∈ set Φ. (φ ≠ ψ) ⟶ Γ ⊩ ∼ (φ ⊓ ψ)
    hence Γ ⊩ ∐ Φ
    proof (induct Φ)
      case Nil
      then show ?case
        by (simp add: set-deduction-weaken)
    next
      case (Cons φ Φ)
      assume A: ∀φ∈duplicates (φ # Φ). Γ ⊩ ∼ φ
         and B: ∀χ∈set (φ # Φ). ∀ψ∈set (φ # Φ). χ ≠ ψ ⟶ Γ ⊩ ∼ (χ ⊓ ψ)
      hence C: Γ ⊩ ∐ Φ using Cons.hyps by simp
      then show ?case
      proof cases
        assume φ ∈ duplicates (φ # Φ)
        moreover from this have Γ ⊩ ∼ φ using A by auto
        moreover have duplicates Φ ⊆ set Φ by (induct Φ, simp, auto)
        ultimately have φ ∈ set Φ by (metis duplicates.simps(2) subsetCE)
        hence ⊢ ∼ φ ↔ ∼(φ ⊓ ⨆ Φ)
        proof (induct Φ)
          case Nil
          then show ?case by simp
        next
          case (Cons ψ Φ)
          assume φ ∈ set (ψ # Φ)
          then show ⊢ ∼ φ ↔ ∼ (φ ⊓ ⨆ (ψ # Φ))
          proof −
            {
              assume φ = ψ
              hence ?thesis
              proof −
                have ∀ 𝔐. 𝔐 ⊨prop ∼ ⟨φ⟩ ↔ ∼ (⟨φ⟩ ⊓ (⟨ψ⟩ ⊔ ⟨⨆ Φ⟩))
                  using ‹φ = ψ› by auto
                hence ⊢ ⦇ ∼ ⟨φ⟩ ↔ ∼ (⟨φ⟩ ⊓ (⟨ψ⟩ ⊔ ⟨⨆ Φ⟩)) ⦈
                  using propositional-semantics by blast
                thus ?thesis by simp
              qed
            }
            moreover
            {
```

assume $\varphi \neq \psi$
hence $\varphi \in \mathit{set}\ \Phi$ **using** ‹$\varphi \in \mathit{set}\ (\psi \# \Phi)$› **by** *auto*
hence $\vdash \sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi)$ **using** *Cons.hyps* **by** *auto*
moreover have $\vdash (\sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi))$
$\rightarrow (\sim \varphi \leftrightarrow \sim (\varphi \sqcap (\psi \sqcup \bigsqcup \Phi)))$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle)) \rightarrow$
$(\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle)))$
by *auto*
hence $\vdash (\!|\ (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle))$
$\rightarrow (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle)))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately have *?thesis* **using** *modus-ponens* **by** *simp*
}
ultimately show *?thesis* **by** *auto*
qed
qed
with ‹$\Gamma \Vdash \sim \varphi$› **have** $\Gamma \Vdash \sim(\varphi \sqcap \bigsqcup \Phi)$
using *biconditional-weaken set-deduction-weaken* **by** *blast*
with ‹$\Gamma \Vdash \coprod \Phi$› **show** *?thesis* **by** *simp*
next
assume $\varphi \notin \mathit{duplicates}\ (\varphi \# \Phi)$
hence $\varphi \notin \mathit{set}\ \Phi$ **by** *auto*
with B **have** $\forall \psi{\in}\mathit{set}\ \Phi.\ \Gamma \Vdash \sim (\varphi \sqcap \psi)$ **by** (*simp*, *metis*)
hence $\Gamma \Vdash \sim (\varphi \sqcap \bigsqcup \Phi)$
by (*simp add*: *disjuction-exclusion-equivalence*)
with ‹$\Gamma \Vdash \coprod \Phi$› **show** *?thesis* **by** *simp*
qed
qed
}
thus *?thesis*
by (*metis exclusive-elimination1 exclusive-elimination2*)
qed

1.5.18 Miscellaneous Disjunctive Normal Form Identities

lemma (**in** *classical-logic*) *conj-dnf-distribute*:
$\vdash \bigsqcup\ (\mathit{map}\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s))\ \Lambda) \leftrightarrow (\varphi \sqcap \bigsqcup\ (\mathit{map} \bigsqcap\ \Lambda))$
proof(*induct* Λ)
case *Nil*
have $\vdash \bot \leftrightarrow (\varphi \sqcap \bot)$
proof −
let $?\varphi = \bot \leftrightarrow (\langle\varphi\rangle \sqcap \bot)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

then show *?case* **by** *simp*
next
case (*Cons* Ψ Λ)
assume $\vdash \bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) Λ) $\leftrightarrow (\varphi \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
(**is** $\vdash$ *?A* $\leftrightarrow (\varphi \sqcap$ *?B*))
moreover
have $\vdash$ (*?A* $\leftrightarrow (\varphi \sqcap$ *?B*)) $\rightarrow$ ((($\varphi \sqcap \bigsqcap \Psi) \sqcup$ *?A*) $\leftrightarrow (\varphi \sqcap \bigsqcap \Psi \sqcup$ *?B*))
proof −
let *?φ* = (⟨*?A*⟩ $\leftrightarrow$ (⟨φ⟩ $\sqcap$ ⟨*?B*⟩)) $\rightarrow$
((((⟨φ⟩ $\sqcap$ ⟨$\bigsqcap$ Ψ⟩) $\sqcup$ ⟨*?A*⟩) $\leftrightarrow$ (⟨φ⟩ $\sqcap$ ⟨$\bigsqcap$ Ψ⟩ $\sqcup$ ⟨*?B*⟩)))
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis*
by *simp*
qed
ultimately have $\vdash$ (($\varphi \sqcap \bigsqcap \Psi) \sqcup$ *?A*) $\leftrightarrow (\varphi \sqcap \bigsqcap \Psi \sqcup$ *?B*)
using *modus-ponens*
by *blast*
moreover
have *map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) Λ = *map* ($\lambda\Psi.\ \varphi \sqcap \bigsqcap \Psi$) Λ **by** *simp*
ultimately show *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *append-dnf-distribute*:
$\vdash \bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \Psi.\ \Phi @ \Psi)$) Λ) $\leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
proof(*induct* Φ)
case *Nil*
have $\vdash \bigsqcup$ (*map* $\bigsqcap$ Λ) $\leftrightarrow (\top \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
(**is** $\vdash$ *?A* $\leftrightarrow (\top \sqcap$ *?A*))
proof −
let *?φ* = ⟨*?A*⟩ $\leftrightarrow$ (($\bot \rightarrow \bot$) $\sqcap$ ⟨*?A*⟩)
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *simp*
hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis*
unfolding *verum-def*
by *simp*
qed
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
have $\vdash \bigsqcup$ (*map* ($\bigsqcap \circ (@)\ \Phi$) Λ) $\leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
= $\vdash \bigsqcup$ (*map* $\bigsqcap$ (*map* ((@) Φ) Λ)) $\leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
by *simp*
with *Cons* **have**
$\vdash \bigsqcup$ (*map* $\bigsqcap$ (*map* ($\lambda\ \Psi.\ \Phi @ \Psi$) Λ)) $\leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcup$ (*map* $\bigsqcap$ Λ))
(**is** $\vdash \bigsqcup$ (*map* $\bigsqcap$ *?A*) $\leftrightarrow$ (*?B* $\sqcap$ *?C*))
by *meson*
moreover have $\vdash \bigsqcup$ (*map* $\bigsqcap$ *?A*) $\leftrightarrow$ (*?B* $\sqcap$ *?C*)
$\rightarrow$ ($\bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) *?A*) $\leftrightarrow (\varphi \sqcap \bigsqcup$ (*map* $\bigsqcap$ *?A*)))

$\rightarrow \bigsqcup$ (map ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) $?A$) $\leftrightarrow$ (($\varphi \sqcap ?B$) $\sqcap ?C$)

proof –

let $?\varphi = \langle\bigsqcup$ ($map \bigsqcap\ ?A$)$\rangle \leftrightarrow (\langle ?B\rangle \sqcap \langle ?C\rangle)$

$\rightarrow (\langle\bigsqcup$ (map ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) $?A$)$\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcup$ ($map \bigsqcap\ ?A$)$\rangle$))

$\rightarrow \langle\bigsqcup$ (map ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) $?A$)$\rangle \leftrightarrow ((\langle\varphi\rangle \sqcap \langle ?B\rangle) \sqcap \langle ?C\rangle)$

have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *simp*

hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*

thus *?thesis*

by *simp*

qed

ultimately have $\vdash \bigsqcup$ (map ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) $?A$) $\leftrightarrow$ (($\varphi \sqcap ?B$) $\sqcap ?C$)

using *modus-ponens conj-dnf-distribute*

by *blast*

moreover

have $\bigsqcap \circ (@)\ (\varphi \# \Phi) = \bigsqcap \circ (\#)\ \varphi \circ (@)\ \Phi$ **by** *auto*

hence

$\vdash \bigsqcup$ (map ($\bigsqcap \circ (@)\ (\varphi \# \Phi)$) Λ) $\leftrightarrow$ ($\bigsqcap\ (\varphi \# \Phi) \sqcap ?C$)

$= \vdash \bigsqcup$ (map ($\bigsqcap \circ (\#)\ \varphi$) $?A$) $\leftrightarrow$ (($\varphi \sqcap ?B$) $\sqcap ?C$)

by *simp*

ultimately show *?case* **by** *meson*

qed

end

Chapter 2

Probability Logic

theory *Probability-Logic*
  imports
    ../../Logic/Classical/Classical-Connectives
    HOL.Real
    HOL−Library.Countable
begin

sledgehammer-params [*smt-proofs* = *false*]

2.1 Definition of Probability Logic

Probability logic is defined in terms of an operator over classical logic obeying certain postulates. Scholars often credit George Boole for first conceiving this kind of formulation [4]. Theodore Hailperin in particular has written extensively on this subject [14, 15, 16].

The presentation below roughly follows Kolmogorov's axiomatization [18]. A key difference is that we only require *finite additivity*, rather than *countable additivity*. Finite additivity is also defined in terms of $(\rightarrow)$. This formulation is required so that probability logic may be extended to Boolean algebra in §2.4.2.

class *probability-logic* = *classical-logic* +
  fixes $Pr :: 'a \Rightarrow real$
  assumes *probability-non-negative*: $Pr\ \varphi \geq 0$
  assumes *probability-unity*: $\vdash \varphi \Longrightarrow Pr\ \varphi = 1$
  assumes *probability-implicational-additivity*:
    $\vdash \varphi \rightarrow \psi \rightarrow \bot \Longrightarrow Pr\ ((\varphi \rightarrow \bot) \rightarrow \psi) = Pr\ \varphi + Pr\ \psi$

A similar axiomatization may be credited to Rescher [20, pg. 185]. However, our formulation has fewer axioms. While Rescher assumes $\vdash \varphi \leftrightarrow \psi \Longrightarrow Pr\ \varphi = Pr\ \psi$, we provide it as a lemma in §2.1.3.

2.1.1 Why Finite Additivity?

In this section we touch on why we have chosen to employ finite additivity in our axiomatization of *probability-logic* and deviate from conventional probability theory.

Conventional probability obeys an axiom known as *countable additivity*. Traditionally it states if $?S$ is a countable set of sets which are pairwise disjoint, then the limit $\sum\ s \in\ ?S.\ Pr\ s$ exists and $Pr\ (\bigcup\ ?S) = (\sum\ s \in\ ?S.\ Pr\ s)$. This is more powerful than our finite additivity axiom $\vdash \varphi \to \psi \to \bot \Longrightarrow Pr\ ((\varphi \to \bot) \to \psi) = Pr\ \varphi + Pr\ \psi$.

However, we argue that demanding countable additivity is not practical. It prohibits data structures we would naturally use in programs exploiting the Dutch book theorem from Chapter 3.

Historically, the statisticians Bruno di Finetti and Leonard Savage gave the most well known critiques. In [9] di Finetti shows various properties which are true for countably additive probability measures may not hold for finitely additive measures. Savage [21], on the other hand, develops probability based on choices prizes in lotteries.

We instead argue that if we demand countable additivity, then certain properties of real world financial software would no longer be formally verifiable as we demonstrate here. In particular, it prohibits data structures we would naturally use in programs exploiting the Dutch book theorem from Chapter 3. Our argument is derivative of one given by Giangiacomo Gerla [13, Section 3].

By taking equivalence classes modulo $\lambda\varphi\ \psi.\ \vdash \varphi \leftrightarrow \psi$, any classical logic instance gives rise to a Boolean algebra known as a *Lindenbaum Algebra*. In the case of $'a$ *classical-propositional-formula* this Boolean algebra algebra is both countable and *atomless*. A theorem of Horn and Tarski [17, Theorem 3.2] asserts there can be no countable additive Pr for a countable atomless Boolean algebra.

A software trading system could reasonably use data structures just like $'nat$ *classical-propositional-formula* when analyzing fixed odds gambling markets. We go into detail regarding this in §3.1. Both *Haskell* and *Rust* allow for declaring data types like $'a$ *classical-propositional-formula*. These languages share a heritage from the ML family of languages just like Isabelle/HOL.

Hence, if we insist on countably additivity then the Dutch Book theorem presented in §3.2.2 cannot be obtained for certain programs we may reasonably wish to write. Not only is our formulation in *probability-logic* suitable for obtaining the Dutch book theorem, so demanding countable additivity is unreasonable.

The above argument is not intended as a blanket refutation of conventional probability theory. It is simply an impossibility result with respect to our Dutch book theorem. Plenty of classic results in probability rely on countable additivity. A nice example recently formalized in Isabelle/HOL is Bouffon's needle [10].

2.1.2 Basic Properties of Probability Logic

lemma (**in** *probability-logic*) *probability-additivity*:
  assumes $\vdash \sim (\varphi \sqcap \psi)$
  shows $Pr\ (\varphi \sqcup \psi) = Pr\ \varphi + Pr\ \psi$
  using
    assms
  unfolding
    conjunction-def
    disjunction-def
    negation-def
  by (*simp add*: *probability-implicational-additivity*)

lemma (**in** *probability-logic*) *probability-alternate-additivity*:
  assumes $\vdash \varphi \to \psi \to \bot$
  shows $Pr\ (\varphi \sqcup \psi) = Pr\ \varphi + Pr\ \psi$
  using *assms*
  by (*metis*
      probability-additivity
      double-negation-converse
      modus-ponens
      conjunction-def
      negation-def)

lemma (**in** *probability-logic*) *complementation*:
  $Pr\ (\sim \varphi) = 1 - Pr\ \varphi$
  by (*metis*
      probability-alternate-additivity
      probability-unity
      bivalence
      negation-elimination
      add.commute
      add-diff-cancel-left')

lemma (**in** *probability-logic*) *unity-upper-bound*:
  $Pr\ \varphi \leq 1$
  by (*metis*
      (*no-types*)
      diff-ge-0-iff-ge
      probability-non-negative
      complementation)

2.1.3 Alternate Definition of Probability Logic

There is an alternate axiomatization of logical probability, due to Brian Gaines [11, pg. 159, postulates P7, P8, and P8] and independently formulated by Brian Weatherson [26]. As Weatherson notes, this axiomatization is suited to formulating *intuitionistic* probability logic. In the case where the underlying logic is classical this is simply equivalent to the traditional axiomatization in §2.1.

class *gaines-weatherson-probability* = *classical-logic* +
  fixes *Pr* :: $'a \Rightarrow real$
  assumes *gaines-weatherson-thesis*:
    $Pr\ \top = 1$
  assumes *gaines-weatherson-antithesis*:
    $Pr\ \bot = 0$
  assumes *gaines-weatherson-monotonicity*:
    $\vdash \varphi \to \psi \Longrightarrow Pr\ \varphi \leq Pr\ \psi$
  assumes *gaines-weatherson-sum-rule*:
    $Pr\ \varphi + Pr\ \psi = Pr\ (\varphi \sqcap \psi) + Pr\ (\varphi \sqcup \psi)$

sublocale *gaines-weatherson-probability* ⊆ *probability-logic*
proof
  fix φ
  have $\vdash \bot \to \varphi$
    by (*simp add*: *ex-falso-quodlibet*)
  thus $0 \leq Pr\ \varphi$
    using
      gaines-weatherson-antithesis
      gaines-weatherson-monotonicity
    by *fastforce*
next
  fix φ
  assume $\vdash \varphi$
  thus $Pr\ \varphi = 1$
    by (*metis*
        gaines-weatherson-thesis
        gaines-weatherson-monotonicity
        eq-iff
        axiom-k
        ex-falso-quodlibet
        modus-ponens
        verum-def)
next
  fix $\varphi\ \psi$
  assume $\vdash \varphi \to \psi \to \bot$
  hence $\vdash \sim (\varphi \sqcap \psi)$
    by (*simp add*: *conjunction-def negation-def*)
  thus $Pr\ ((\varphi \to \bot) \to \psi) = Pr\ \varphi + Pr\ \psi$
    by (*metis*

add.commute
add.right-neutral
eq-iff
disjunction-def
ex-falso-quodlibet
negation-def
gaines-weatherson-antithesis
gaines-weatherson-monotonicity
gaines-weatherson-sum-rule)
qed

lemma (**in** *probability-logic*) *monotonicity*:
$\vdash \varphi \to \psi \Longrightarrow Pr\ \varphi \leq Pr\ \psi$
proof −
assume $\vdash \varphi \to \psi$
hence $\vdash \sim (\varphi \sqcap \sim \psi)$
unfolding *negation-def conjunction-def*
by (*metis*
conjunction-def
exclusion-contrapositive-equivalence
negation-def
weak-biconditional-weaken)
hence $Pr\ (\varphi \sqcup \sim \psi) = Pr\ \varphi + Pr\ (\sim \psi)$
by (*simp add*: *probability-additivity*)
hence $Pr\ \varphi + Pr\ (\sim \psi) \leq 1$
by (*metis unity-upper-bound*)
hence $Pr\ \varphi + 1 - Pr\ \psi \leq 1$
by (*simp add*: *complementation*)
thus *?thesis* **by** *linarith*
qed

lemma (**in** *probability-logic*) *biconditional-equivalence*:
$\vdash \varphi \leftrightarrow \psi \Longrightarrow Pr\ \varphi = Pr\ \psi$
by (*meson*
eq-iff
modus-ponens
biconditional-left-elimination
biconditional-right-elimination
monotonicity)

lemma (**in** *probability-logic*) *sum-rule*:
$Pr\ (\varphi \sqcup \psi) + Pr\ (\varphi \sqcap \psi) = Pr\ \varphi + Pr\ \psi$
proof −
have $\vdash (\varphi \sqcup \psi) \leftrightarrow (\varphi \sqcup \psi \setminus (\varphi \sqcap \psi))$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \sqcup \langle\psi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcup \langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle))$
unfolding
classical-logic-class.subtraction-def
classical-logic-class.negation-def

classical-logic-class.biconditional-def
classical-logic-class.conjunction-def
classical-logic-class.disjunction-def
by *simp*
hence $\vdash (\!| (\langle\varphi\rangle \sqcup \langle\psi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcup \langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle)) |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
moreover have $\vdash \varphi \to (\psi \setminus (\varphi \sqcap \psi)) \to \bot$
proof $-$
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \langle\varphi\rangle \to (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle)) \to \bot$
unfolding
classical-logic-class.subtraction-def
classical-logic-class.negation-def
classical-logic-class.biconditional-def
classical-logic-class.conjunction-def
classical-logic-class.disjunction-def
by *simp*
hence $\vdash (\!| \langle\varphi\rangle \to (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle)) \to \bot |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence $Pr\ (\varphi \sqcup \psi) = Pr\ \varphi + Pr\ (\psi \setminus (\varphi \sqcap \psi))$
using
probability-alternate-additivity
biconditional-equivalence
calculation
by *auto*
moreover have $\vdash \psi \leftrightarrow (\psi \setminus (\varphi \sqcap \psi) \sqcup (\varphi \sqcap \psi))$
proof $-$
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \langle\psi\rangle \leftrightarrow (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcup (\langle\varphi\rangle \sqcap \langle\psi\rangle))$
unfolding
classical-logic-class.subtraction-def
classical-logic-class.negation-def
classical-logic-class.biconditional-def
classical-logic-class.conjunction-def
classical-logic-class.disjunction-def
by *auto*
hence $\vdash (\!| \langle\psi\rangle \leftrightarrow (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcup (\langle\varphi\rangle \sqcap \langle\psi\rangle)) |\!)$
using *propositional-semantics* **by**
blast
thus *?thesis* **by** *simp*
qed
moreover have $\vdash (\psi \setminus (\varphi \sqcap \psi)) \to (\varphi \sqcap \psi) \to \bot$
unfolding
subtraction-def
negation-def
conjunction-def
using

conjunction-def
conjunction-right-elimination
by *auto*
hence $Pr\ \psi = Pr\ (\psi \setminus (\varphi \sqcap \psi)) + Pr\ (\varphi \sqcap \psi)$
using
probability-alternate-additivity
biconditional-equivalence
calculation
by *auto*
ultimately show *?thesis*
by *simp*
qed

sublocale *probability-logic* $\subseteq$ *gaines-weatherson-probability*
proof
show $Pr\ \top = 1$
by (*simp add*: *probability-unity*)
next
show $Pr\ \bot = 0$
by (*metis*
add-cancel-left-right
probability-additivity
ex-falso-quodlibet
probability-unity
bivalence
conjunction-right-elimination
negation-def)
next
fix $\varphi\ \psi$
assume $\vdash \varphi \to \psi$
thus $Pr\ \varphi \leq Pr\ \psi$
using *monotonicity*
by *auto*
next
fix $\varphi\ \psi$
show $Pr\ \varphi + Pr\ \psi = Pr\ (\varphi \sqcap \psi) + Pr\ (\varphi \sqcup \psi)$
by (*metis sum-rule add.commute*)
qed

sublocale *probability-logic* $\subseteq$ *consistent-classical-logic*
proof
show $\neg \vdash \bot$ **using** *probability-unity gaines-weatherson-antithesis* **by** *auto*
qed

lemma (**in** *probability-logic*) *subtraction-identity*:
$Pr\ (\varphi \setminus \psi) = Pr\ \varphi - Pr\ (\varphi \sqcap \psi)$
proof −
have $\vdash \varphi \leftrightarrow ((\varphi \setminus \psi) \sqcup (\varphi \sqcap \psi))$
proof −

have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \langle\varphi\rangle \leftrightarrow ((\langle\varphi\rangle \setminus \langle\psi\rangle) \sqcup (\langle\varphi\rangle \sqcap \langle\psi\rangle))$
unfolding
classical-logic-class.subtraction-def
classical-logic-class.negation-def
classical-logic-class.biconditional-def
classical-logic-class.conjunction-def
classical-logic-class.disjunction-def
by (*simp*, *blast*)
hence $\vdash (\!|\ \langle\varphi\rangle \leftrightarrow ((\langle\varphi\rangle \setminus \langle\psi\rangle) \sqcup (\langle\varphi\rangle \sqcap \langle\psi\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence $Pr\ \varphi\ = Pr\ ((\varphi \setminus \psi) \sqcup (\varphi \sqcap \psi))$
using *biconditional-equivalence*
by *simp*
moreover have $\vdash \sim((\varphi \setminus \psi) \sqcap (\varphi \sqcap \psi))$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \sim((\langle\varphi\rangle \setminus \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcap \langle\psi\rangle))$
unfolding
classical-logic-class.subtraction-def
classical-logic-class.negation-def
classical-logic-class.conjunction-def
classical-logic-class.disjunction-def
by *simp*
hence $\vdash (\!|\ \sim((\langle\varphi\rangle \setminus \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcap \langle\psi\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately show *?thesis*
using *probability-additivity*
by *auto*
qed

2.1.4 Basic Probability Logic Inequality Results

lemma (**in** *probability-logic*) *disjunction-sum-inequality*:
$Pr\ (\varphi \sqcup \psi) \leq Pr\ \varphi + Pr\ \psi$
proof −
have $Pr\ (\varphi \sqcup \psi) + Pr\ (\varphi \sqcap \psi) = Pr\ \varphi + Pr\ \psi$
$0 \leq Pr\ (\varphi \sqcap \psi)$
by (*simp add*: *sum-rule*, *simp add*: *probability-non-negative*)
thus *?thesis* **by** *linarith*
qed

lemma (**in** *probability-logic*)
arbitrary-disjunction-list-summation-inequality:
$Pr\ (\bigsqcup\ \Phi) \leq (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi)$
proof (*induct* Φ)
case *Nil*

then show *?case* **by** (*simp add*: *gaines-weatherson-antithesis*)
next
case (*Cons* φ Φ)
have Pr ($\bigsqcup$ (φ # Φ)) $\leq$ Pr φ + Pr ($\bigsqcup$ Φ)
using *disjunction-sum-inequality*
by *simp*
with *Cons* **have** Pr ($\bigsqcup$ (φ # Φ)) $\leq$ Pr φ + ($\sum \varphi \leftarrow \Phi.$ Pr φ) **by** *linarith*
then show *?case* **by** *simp*
qed

lemma (**in** *probability-logic*) *implication-list-summation-inequality*:
assumes $\vdash \varphi \rightarrow \bigsqcup \Psi$
shows Pr $\varphi \leq$ ($\sum \psi \leftarrow \Psi.$ Pr ψ)
using
assms
arbitrary-disjunction-list-summation-inequality
monotonicity
order-trans
by *blast*

lemma (**in** *probability-logic*) *arbitrary-disjunction-set-summation-inequality*:
Pr ($\bigsqcup$ Φ) $\leq$ ($\sum \varphi \in$ *set* Φ. Pr φ)
by (*metis*
arbitrary-disjunction-list-summation-inequality
arbitrary-disjunction-remdups
biconditional-equivalence
sum.set-conv-list)

lemma (**in** *probability-logic*) *implication-set-summation-inequality*:
assumes $\vdash \varphi \rightarrow \bigsqcup \Psi$
shows Pr $\varphi \leq$ ($\sum \psi \in$ *set* Ψ. Pr ψ)
using
assms
arbitrary-disjunction-set-summation-inequality
monotonicity
order-trans
by *blast*

2.1.5 Dirac Measures

Before presenting *Dirac measures* in probability logic, we first give the set of all functions satisfying probability logic.

definition (**in** *classical-logic*) *probabilities* :: ($'a \Rightarrow$ *real*) *set*
where *probabilities* =
{Pr. *class.probability-logic* (λ φ. $\vdash \varphi$) ($\rightarrow$) $\perp$ Pr }

Traditionally, a Dirac measure is a function δ_x where δ_x S = *1* if $x \in S$ and δ_x S = *0* otherwise. This means that Dirac measures correspond to special ultrafilters on their underlying σ-algebra which are closed under countable unions.

Probability logic, as discussed in §2.1.1, may not have countable joins in its underlying logic. In the setting of probability logic, Dirac measures are simple probability functions that are either 0 or 1.

definition (**in** *classical-logic*) *dirac-measures* :: (*'a* ⇒ *real*) *set*
  where *dirac-measures* =
    { *Pr*. *class.probability-logic* (λ φ. ⊢ φ) (→) ⊥ *Pr*
      ∧ (∀ *x*. *Pr x* = *0* ∨ *Pr x* = *1*) }

lemma (**in** *classical-logic*) *dirac-measures-subset*:
  dirac-measures ⊆ *probabilities*
  unfolding
    probabilities-def
    dirac-measures-def
  by *fastforce*

Maximally consistent sets correspond to Dirac measures. One direction of this correspondence is established below.

lemma (**in** *classical-logic*) *MCS-dirac-measure*:
  assumes *MCS* Ω
    shows (λ χ. *if* χ∈Ω *then* (*1* :: *real*) *else* *0*) ∈ *dirac-measures*
      (**is** *?Pr* ∈ *dirac-measures*)
proof −
  have *class.probability-logic* (λ φ. ⊢ φ) (→) ⊥ *?Pr*
  proof (*standard*, *simp*,
        meson
          assms
          formula-maximally-consistent-set-def-reflection
          maximally-consistent-set-def
          set-deduction-weaken)
    fix φ ψ
    assume ⊢ φ → ψ → ⊥
    hence φ ⊓ ψ ∉ Ω
      by (*metis*
            assms
            formula-consistent-def
            formula-maximally-consistent-set-def-def
            maximally-consistent-set-def
            conjunction-def
            set-deduction-modus-ponens
            set-deduction-reflection
            set-deduction-weaken)
    hence φ ∉ Ω ∨ ψ ∉ Ω
      using
        assms
        formula-maximally-consistent-set-def-reflection
        maximally-consistent-set-def
        conjunction-set-deduction-equivalence

```
      by meson

    have φ ⊔ ψ ∈ Ω = (φ ∈ Ω ∨ ψ ∈ Ω)
      by (metis
            ‹φ ⊓ ψ ∉ Ω›
            assms
            formula-maximally-consistent-set-def-implication
            maximally-consistent-set-def
            conjunction-def
            disjunction-def)
    have ?Pr (φ ⊔ ψ) = ?Pr φ + ?Pr ψ
    proof (cases φ ⊔ ψ ∈ Ω)
      case True
      hence ◇: 1 = ?Pr (φ ⊔ ψ) by simp
      show ?thesis
      proof (cases φ ∈ Ω)
        case True
        hence ψ ∉ Ω
          using ‹φ ∉ Ω ∨ ψ ∉ Ω›
          by blast
        have ?Pr (φ ⊔ ψ) = (1::real) using ◇ by simp
        also have ... = 1 + (0::real) by linarith
        also have ... = ?Pr φ + ?Pr ψ
          using ‹ψ ∉ Ω› ‹φ ∈ Ω› by simp
        finally show ?thesis .
      next
        case False
        hence ψ ∈ Ω
          using ‹φ ⊔ ψ ∈ Ω› ‹(φ ⊔ ψ ∈ Ω) = (φ ∈ Ω ∨ ψ ∈ Ω)›
          by blast
        have ?Pr (φ ⊔ ψ) = (1::real) using ◇ by simp
        also have ... = (0::real) + 1 by linarith
        also have ... = ?Pr φ + ?Pr ψ
          using ‹ψ ∈ Ω› ‹φ ∉ Ω› by simp
        finally show ?thesis .
      qed
    next
      case False
      moreover from this have φ ∉ Ω ψ ∉ Ω
        using ‹(φ ⊔ ψ ∈ Ω) = (φ ∈ Ω ∨ ψ ∈ Ω)› by blast+
      ultimately show ?thesis by simp
    qed
    thus ?Pr ((φ → ⊥) → ψ) = ?Pr φ + ?Pr ψ
      unfolding disjunction-def .
  qed
  thus ?thesis
    unfolding dirac-measures-def
    by simp
qed
```

end

2.2 Suppes' Theorem

theory *Suppes-Theorem*
  imports *Probability-Logic*
begin

An elementary completeness theorem for inequalities for probability logic to be investigated is due to Patrick Suppes [22].

The completeness theorem presented in §2.7 can be understood as a vast generalization of this theorem. A consequence of this theorem is an elementary form of *collapse*, which asserts that inequalities for probabilities are logically equivalent to the more restricted class of *Dirac measures* as defined in §2.1.5. Collapse theorems are further investigated in §2.4.3 and §2.7.1.

sledgehammer-params [*smt-proofs* = *false*]

2.2.1 Suppes' List Theorem

We first establish Suppes' theorem for lists of propositions. This is done by establishing our first completeness theorem using *Dirac measures*.

First, we use the result from §2.1.4 that shows $\vdash \varphi \to \bigsqcup \Psi$ implies $Pr\ \varphi \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi)$. This can be understood as a *soundness* result.

To show completeness, assume $\neg \vdash \varphi \to \bigsqcup \Psi$. From this obtain a maximally consistent Ω, as per §1.2.3, such that $\varphi \to \bigsqcup \Psi \notin \Omega$. We then define $\delta\ \varphi = (\varphi \in \Omega)$ and show δ is a *Dirac measure*, such that $\delta\ \varphi > (\sum \psi \leftarrow \Psi.\ \delta\ \psi)$. This basic approach will be elaborated on in subsequent completeness theorems.

lemma (**in** *classical-logic*) *dirac-list-summation-completeness*:
  $(\forall\ \delta \in \textit{dirac-measures}.\ \delta\ \varphi \leq (\sum \psi \leftarrow \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup \Psi$
proof –
  {
    fix δ :: $'a \Rightarrow \textit{real}$
    assume $\delta \in \textit{dirac-measures}$
    from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\to)\ \bot\ \delta$
      unfolding *dirac-measures-def*
      by *auto*
    assume $\vdash \varphi \to \bigsqcup \Psi$
    hence $\delta\ \varphi \leq (\sum \psi \leftarrow \Psi.\ \delta\ \psi)$
      using *implication-list-summation-inequality*
      by *auto*
  }

```
  moreover {
    assume ¬ ⊢ φ → ⨆ Ψ
    from this obtain Ω where Ω:
      MCS Ω
      φ ∈ Ω
      ⨆ Ψ ∉ Ω
      by (meson
            insert-subset
            formula-consistent-def
            formula-maximal-consistency
            formula-maximally-consistent-extension
            formula-maximally-consistent-set-def-def
            set-deduction-base-theory
            set-deduction-reflection
            set-deduction-theorem)
    hence∀ ψ ∈ set Ψ. ψ ∉ Ω
      using arbitrary-disjunction-exclusion-MCS by blast
    define δ where δ = (λ χ . if χ∈Ω then (1 :: real) else 0)
    from ‹∀ ψ ∈ set Ψ. ψ ∉ Ω› have (∑ψ←Ψ. δ ψ) = 0
      unfolding δ-def
      by (induct Ψ, simp, simp)
    hence ¬ δ φ ≤ (∑ψ←Ψ. δ ψ)
      unfolding δ-def
      by (simp add: Ω(2))
    hence
      ∃ δ ∈ dirac-measures. ¬ (δ φ ≤ (∑ψ←Ψ. δ ψ))
      unfolding δ-def
      using Ω(1) MCS-dirac-measure by auto
  }
  ultimately show ?thesis by blast
qed

theorem (in classical-logic) list-summation-completeness:
  (∀ Pr ∈ probabilities. Pr φ ≤ (∑ψ←Ψ. Pr ψ)) = ⊢ φ → ⨆ Ψ
  (is ?lhs = ?rhs)
proof
  assume ?lhs
  hence ∀ δ ∈ dirac-measures. δ φ ≤ (∑ψ←Ψ. δ ψ)
    unfolding dirac-measures-def probabilities-def
    by blast
  thus ?rhs
    using dirac-list-summation-completeness by blast
next
  assume ?rhs
  show ?lhs
  proof
    fix Pr :: 'a ⇒ real
    assume Pr ∈ probabilities
    from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ Pr
```

unfolding *probabilities-def*
by *auto*
show $Pr\ \varphi \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi)$
using ‹*?rhs*› *implication-list-summation-inequality*
by *simp*
qed
qed

The theorem below is a special case of the full *collapse* theorem given in §2.7.1. The collapse theorem asserts that to prove an inequalities for all probabilities in probability logic, you only need to consider the case of functions which take on values of 0 or 1. The full collapse theorem generalizes to inequalities of the form $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + c \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi)$.

lemma (**in** *classical-logic*) *suppes-collapse*:
$(\forall\ Pr \in probabilities.\ Pr\ \varphi \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi))$
$= (\forall\ \delta \in dirac\text{-}measures.\ \delta\ \varphi \leq (\sum \psi \leftarrow \Psi.\ \delta\ \psi))$
by (*simp add*:
dirac-list-summation-completeness
list-summation-completeness)

lemma (**in** *classical-logic*) *probability-member-neg*:
fixes *Pr*
assumes $Pr \in probabilities$
shows $Pr\ (\sim \varphi) = 1 - Pr\ \varphi$
proof −
from *assms* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *probabilities-def*
by *auto*
show *?thesis*
by (*simp add*: *complementation*)
qed

Suppes' theorem has a philosophical interpretation. It asserts that if $\Psi :\vdash \varphi$, then our *uncertainty* in φ is bounded above by our uncertainty in Ψ. Here the uncertainty in the proposition φ is $1 - Pr\ \varphi$. Our uncertainty in Ψ, on the other hand, is $\sum \psi \leftarrow \Psi.\ 1 - Pr\ \psi$.

theorem (**in** *classical-logic*) *suppes-list-theorem*:
$\Psi :\vdash \varphi = (\forall\ Pr \in probabilities.\ (\sum \psi \leftarrow \Psi.\ 1 - Pr\ \psi) \geq 1 - Pr\ \varphi)$
proof −
have
$\Psi :\vdash \varphi = (\forall\ Pr \in probabilities.\ (\sum \psi \leftarrow \mathbf{\sim}\ \Psi.\ Pr\ \psi) \geq Pr\ (\sim \varphi))$
using
list-summation-completeness
weak-biconditional-weaken
contra-list-curry-uncurry
list-deduction-def
by *blast*
moreover have

$\forall\ Pr \in probabilities.\ (\sum \psi \leftarrow (\sim \Psi).\ Pr\ \psi) = (\sum \psi \leftarrow \Psi.\ Pr\ (\sim \psi))$
  by (*induct* Ψ, *auto*)
ultimately show *?thesis*
  using *probability-member-neg*
  by (*induct* Ψ, *simp+*)
qed

2.2.2 Suppes' Set Theorem

Suppes theorem also obtains for *sets*.

lemma (**in** *classical-logic*) *dirac-set-summation-completeness*:
$(\forall\ \delta \in dirac\text{-}measures.\ \delta\ \varphi \leq (\sum \psi \in\ set\ \Psi.\ \delta\ \psi)) = \vdash \varphi \rightarrow \bigsqcup\ \Psi$
by (*metis*
dirac-list-summation-completeness
modus-ponens
arbitrary-disjunction-remdups
biconditional-left-elimination
biconditional-right-elimination
hypothetical-syllogism
sum.set-conv-list)

theorem (**in** *classical-logic*) *set-summation-completeness*:
$(\forall\ \delta \in probabilities.\ \delta\ \varphi \leq (\sum \psi \in\ set\ \Psi.\ \delta\ \psi)) = \vdash \varphi \rightarrow \bigsqcup\ \Psi$
by (*metis*
dirac-list-summation-completeness
dirac-set-summation-completeness
list-summation-completeness
sum.set-conv-list)

lemma (**in** *classical-logic*) *suppes-set-collapse*:
$(\forall\ Pr \in probabilities.\ Pr\ \varphi \leq (\sum \psi \in set\ \Psi.\ Pr\ \psi))$
$= (\forall\ \delta \in dirac\text{-}measures.\ \delta\ \varphi \leq (\sum \psi \in set\ \Psi.\ \delta\ \psi))$
by (*simp add*:
dirac-set-summation-completeness
set-summation-completeness)

In our formulation of logic, there is not reason that $\sim a = \sim b$ while $a \neq b$. As a consequence the Suppes theorem for sets presented below is different than the one given in §2.2.1.

theorem (**in** *classical-logic*) *suppes-set-theorem*:
$\Psi :\vdash \varphi$
$= (\forall\ Pr \in probabilities.\ (\sum \psi \in set\ (\sim \Psi).\ Pr\ \psi) \geq 1 - Pr\ \varphi)$
proof −
have $\Psi :\vdash \varphi$
$= (\forall\ Pr \in probabilities.\ (\sum \psi \in set\ (\sim \Psi).\ Pr\ \psi) \geq Pr\ (\sim \varphi))$
using
contra-list-curry-uncurry
list-deduction-def

```
      set-summation-completeness
      weak-biconditional-weaken
    by blast
  thus ?thesis
    using probability-member-neg
    by (induct Ψ, auto)
qed
```

2.2.3 Converse Suppes' Theorem

A formulation of the converse of Suppes' theorem obtains for lists/sets of *logically disjoint* propositions.

```
lemma (in probability-logic) exclusive-sum-list-identity:
  assumes ⊢ ∐ Φ
  shows Pr (⨆ Φ) = (∑φ←Φ. Pr φ)
  using assms
proof (induct Φ)
  case Nil
  then show ?case
    by (simp add: gaines-weatherson-antithesis)
next
  case (Cons φ Φ)
  assume ⊢ ∐ (φ # Φ)
  hence ⊢ ∼ (φ ⊓ ⨆ Φ) ⊢ ∐ Φ by simp+
  hence Pr (⨆(φ # Φ)) = Pr φ + Pr (⨆ Φ)
        Pr (⨆ Φ) = (∑φ←Φ. Pr φ)
    using Cons.hyps probability-additivity by auto
  hence Pr (⨆(φ # Φ)) = Pr φ + (∑φ←Φ. Pr φ) by auto
  thus ?case by simp
qed

lemma sum-list-monotone:
  fixes f :: 'a ⇒ real
  assumes ∀ x. f x ≥ 0
     and set Φ ⊆ set Ψ
     and distinct Φ
   shows (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
  using assms
proof −
  assume ∀ x. f x ≥ 0
  have ∀Φ. set Φ ⊆ set Ψ
           ⟶ distinct Φ
           ⟶ (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
  proof (induct Ψ)
    case Nil
    then show ?case by simp
  next
    case (Cons ψ Ψ)
    {
```

```
      fix Φ
      assume set Φ ⊆ set (ψ # Ψ)
         and distinct Φ
      have (∑φ←Φ. f φ) ≤ (∑ψ'←(ψ # Ψ). f ψ')
      proof −
        {
          assume ψ ∉ set Φ
          with ‹set Φ ⊆ set (ψ # Ψ)› have set Φ ⊆ set Ψ by auto
          hence (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
            using Cons.hyps ‹distinct Φ› by auto
          moreover have f ψ ≥ 0 using ‹∀ x. f x ≥ 0› by metis
          ultimately have ?thesis by simp
        }
        moreover
        {
          assume ψ ∈ set Φ
          hence set Φ = insert ψ (set (removeAll ψ Φ))
            by auto
          with ‹set Φ ⊆ set (ψ # Ψ)› have set (removeAll ψ Φ) ⊆ set Ψ
            by (metis
                  insert-subset
                  list.simps(15)
                  set-removeAll
                  subset-insert-iff)
          moreover from ‹distinct Φ› have distinct (removeAll ψ Φ)
            by (meson distinct-removeAll)
          ultimately have (∑φ←(removeAll ψ Φ). f φ) ≤ (∑ψ←Ψ. f ψ)
            using Cons.hyps
            by simp
          moreover from ‹ψ ∈ set Φ› ‹distinct Φ›
          have (∑φ←Φ. f φ) = f ψ + (∑φ←(removeAll ψ Φ). f φ)
            using distinct-remove1-removeAll sum-list-map-remove1
            by fastforce
          ultimately have ?thesis using ‹∀ x. f x ≥ 0›
            by simp
        }
        ultimately show ?thesis by blast
      qed
    }
    thus ?case by blast
  qed
  moreover assume set Φ ⊆ set Ψ and distinct Φ
  ultimately show ?thesis by blast
qed

lemma count-remove-all-sum-list:
  fixes f :: 'a ⇒ real
  shows real (count-list xs x) * f x + (∑x'←(removeAll x xs). f x')
          = (∑x←xs. f x)
```

by (*induct xs*, *simp*, *simp*,
metis
(*no-types*, *hide-lams*)
semiring-normalization-rules(*3*)
add.commute
add.left-commute)

lemma (**in** *classical-logic*) *dirac-exclusive-implication-completeness*:
$(\forall\ \delta \in \textit{dirac-measures}.\ (\sum \varphi{\leftarrow}\Phi.\ \delta\ \varphi) \leq \delta\ \psi) = (\vdash \coprod\ \Phi \wedge\ \vdash \bigsqcup\ \Phi \rightarrow \psi)$
proof −
{
fix δ
assume $\delta \in$ *dirac-measures*
from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ \delta$
unfolding *dirac-measures-def*
by *simp*
assume $\vdash \coprod\ \Phi \vdash \bigsqcup\ \Phi \rightarrow \psi$
hence $(\sum \varphi{\leftarrow}\Phi.\ \delta\ \varphi) \leq \delta\ \psi$
using *exclusive-sum-list-identity monotonicity* **by** *fastforce*
}
moreover
{
assume $\neg \vdash \coprod\ \Phi$
hence $(\exists\ \varphi \in \textit{set}\ \Phi.\ \exists\ \psi \in \textit{set}\ \Phi.$
$\varphi \neq \psi \wedge \neg \vdash \sim (\varphi \sqcap \psi)) \vee (\exists\ \varphi \in \textit{duplicates}\ \Phi.\ \neg \vdash \sim \varphi)$
using *exclusive-equivalence set-deduction-base-theory* **by** *blast*
hence $\neg\ (\forall\ \delta \in \textit{dirac-measures}.\ (\sum \varphi{\leftarrow}\Phi.\ \delta\ \varphi) \leq \delta\ \psi)$
proof (*elim disjE*)
assume $\exists\ \varphi \in \textit{set}\ \Phi.\ \exists\ \chi \in \textit{set}\ \Phi.\ \varphi \neq \chi \wedge \neg \vdash \sim (\varphi \sqcap \chi)$
from *this* **obtain** φ **and** χ
where $\varphi\chi$*-properties*:
$\varphi \in \textit{set}\ \Phi$
$\chi \in \textit{set}\ \Phi$
$\varphi \neq \chi$
$\neg \vdash \sim (\varphi \sqcap \chi)$
by *blast*
from *this* **obtain** Ω **where** Ω: *MCS* $\Omega \sim (\varphi \sqcap \chi) \notin \Omega$
by (*meson*
insert-subset
formula-consistent-def
formula-maximal-consistency
formula-maximally-consistent-extension
formula-maximally-consistent-set-def-def
set-deduction-base-theory
set-deduction-reflection
set-deduction-theorem)
let $?\delta = \lambda\ \chi.\ \textit{if}\ \chi{\in}\Omega\ \textit{then}\ (1 :: \textit{real})\ \textit{else}\ 0$
from Ω **have** $\varphi \in \Omega\ \chi \in \Omega$
by (*metis*

```
          formula-maximally-consistent-set-def-implication
          maximally-consistent-set-def
          conjunction-def
          negation-def)+
  with φχ-properties have
      (∑ φ←[φ, χ]. ?δ φ) = 2
      set [φ, χ] ⊆ set Φ
      distinct [φ, χ]
      ∀ φ. ?δ φ ≥ 0
    by simp+
  hence (∑ φ←Φ. ?δ φ) ≥ 2 using sum-list-monotone by metis
  hence ¬ (∑ φ←Φ. ?δ φ) ≤ ?δ (ψ) by auto
  thus ?thesis
    using Ω(1) MCS-dirac-measure
    by auto
next
  assume ∃ φ ∈ duplicates Φ. ¬ ⊢ ∼ φ
  from this obtain φ where φ: φ ∈ duplicates Φ ¬ ⊢ ∼ φ
    using
      exclusive-equivalence [where Γ={}]
      set-deduction-base-theory
    by blast
  from φ obtain Ω where Ω: MCS Ω ∼ φ ∉ Ω
    by (meson
          insert-subset
          formula-consistent-def
          formula-maximal-consistency
          formula-maximally-consistent-extension
          formula-maximally-consistent-set-def-def
          set-deduction-base-theory
          set-deduction-reflection
          set-deduction-theorem)
  hence φ ∈ Ω
    using negation-def by auto
  let ?δ = λ χ. if χ∈Ω then (1 :: real) else 0
  from φ have count-list Φ φ ≥ 2
    using duplicates-alt-def [where xs=Φ]
    by blast
  hence real (count-list Φ φ) ∗ ?δ φ ≥ 2 using ‹φ ∈ Ω› by simp
  moreover
  {
    fix Ψ
    have (∑ φ←Ψ. ?δ φ) ≥ 0 by (induct Ψ, simp, simp)
  }
  moreover have (0::real)
                    ≤ (∑ a←removeAll φ Φ. if a ∈ Ω then 1 else 0)
    using ‹⋀Ψ. 0 ≤ (∑ φ←Ψ. if φ ∈ Ω then 1 else 0)›
    by presburger
  ultimately have real (count-list Φ φ) ∗ ?δ φ
```

```
              + (∑ φ ← (removeAll φ Φ). ?δ φ) ≥ 2
      using ‹2 ≤ real (count-list Φ φ) * (if φ ∈ Ω then 1 else 0)›
      by linarith
    hence (∑φ←Φ. ?δ φ) ≥ 2 by (metis count-remove-all-sum-list)
    hence ¬ (∑φ←Φ. ?δ φ) ≤ ?δ (ψ) by auto
    thus ?thesis
      using Ω(1) MCS-dirac-measure
      by auto
  qed
}
moreover
{
  assume ¬ ⊢ ⨆ Φ → ψ
  from this obtain Ω φ
    where
      Ω: MCS Ω
      and ψ: ψ ∉ Ω
      and φ: φ ∈ set Φ φ ∈ Ω
    by (meson
          insert-subset
          formula-consistent-def
          formula-maximal-consistency
          formula-maximally-consistent-extension
          formula-maximally-consistent-set-def-def
          arbitrary-disjunction-exclusion-MCS
          set-deduction-base-theory
          set-deduction-reflection
          set-deduction-theorem)
  let ?δ = λ χ. if χ∈Ω then (1 :: real) else 0
  from φ have (∑φ←Φ. ?δ φ) ≥ 1
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons φ' Φ)
    obtain f :: real list ⇒ real where f:
      ∀rs. f rs ∈ set rs ∧ ¬ 0 ≤ f rs ∨ 0 ≤ sum-list rs
      using sum-list-nonneg by moura
    moreover have f (map ?δ Φ) ∉ set (map ?δ Φ) ∨ 0 ≤ f (map ?δ Φ)
      by fastforce
    ultimately show ?case
      by (simp, metis Cons.hyps Cons.prems(1) φ(2) set-ConsD)
  qed
  hence ¬ (∑φ←Φ. ?δ φ) ≤ ?δ (ψ) using ψ by auto
  hence ¬ (∀ δ ∈ dirac-measures. (∑φ←Φ. δ φ) ≤ δ ψ)
    using Ω(1) MCS-dirac-measure
    by auto
}
ultimately show ?thesis by blast
```

qed

theorem (**in** *classical-logic*) *exclusive-implication-completeness*:
  ($\forall$ *Pr* $\in$ *probabilities*. ($\sum \varphi \leftarrow \Phi$. *Pr* φ) $\leq$ *Pr* ψ) = ($\vdash \coprod \Phi \wedge \vdash \bigsqcup \Phi \rightarrow \psi$)
  (**is** *?lhs* = *?rhs*)
proof
  assume *?lhs*
  thus *?rhs*
    by (*meson*
        dirac-exclusive-implication-completeness
        dirac-measures-subset
        subset-eq)
next
  assume *?rhs*
  show *?lhs*
  proof
    fix *Pr* :: $'a \Rightarrow$ *real*
    assume *Pr* $\in$ *probabilities*
    from *this* **interpret** *probability-logic* (λ φ. $\vdash \varphi$) ($\rightarrow$) $\bot$ *Pr*
      unfolding *probabilities-def*
      by *simp*
    show ($\sum \varphi \leftarrow \Phi$. *Pr* φ) $\leq$ *Pr* ψ
      using
        ‹*?rhs*›
        exclusive-sum-list-identity
        monotonicity
      by *fastforce*
  qed
qed

lemma (**in** *classical-logic*) *dirac-inequality-completeness*:
  ($\forall$ $\delta \in$ *dirac-measures*. δ $\varphi \leq \delta$ ψ) = $\vdash \varphi \rightarrow \psi$
proof –
  have $\vdash \coprod$ [φ]
    by (*simp add*: *conjunction-right-elimination negation-def*)
  hence ($\vdash \coprod$ [φ] $\wedge$ $\vdash \bigsqcup$ [φ] $\rightarrow \psi$) = $\vdash \varphi \rightarrow \psi$
    by (*metis*
        arbitrary-disjunction.simps(*1*)
        arbitrary-disjunction.simps(*2*)
        disjunction-def implication-equivalence
        negation-def
        weak-biconditional-weaken)
  thus *?thesis*
    using *dirac-exclusive-implication-completeness* [**where** Φ=[φ]]
    by *auto*
qed

2.2.4 Implication Inequality Completeness

The following theorem establishes the converse of $\vdash \varphi \to \psi \Longrightarrow Pr\ \varphi \leq Pr\ \psi$, which was proved in §2.1.3.

theorem (**in** *classical-logic*) *implication-inequality-completeness*:
  $(\forall\ Pr \in probabilities.\ Pr\ \varphi \leq Pr\ \psi) = \vdash \varphi \to \psi$
proof −
  have $\vdash \coprod\ [\varphi]$
    by (*simp add*: *conjunction-right-elimination negation-def*)
  hence $(\vdash \coprod\ [\varphi] \wedge\ \vdash \bigsqcup\ [\varphi] \to \psi) = \vdash \varphi \to \psi$
    by (*metis*
        arbitrary-disjunction.simps(*1*)
        arbitrary-disjunction.simps(*2*)
        disjunction-def implication-equivalence
        negation-def
        weak-biconditional-weaken)
  thus *?thesis*
    using *exclusive-implication-completeness* [**where** Φ=[φ]]
    by *simp*
qed

2.2.5 Characterizing Logical Exclusiveness In Probability Logic

Finally, we can say that $Pr\ (\bigsqcup\ \Phi) = (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi)$ if and only if the propositions in Φ are mutually exclusive (i.e. $\vdash \coprod\ \Phi$). This result also obtains for sets.

lemma (**in** *classical-logic*) *dirac-exclusive-list-summation-completeness*:
  $(\forall\ \delta \in dirac\text{-}measures.\ \delta\ (\bigsqcup\ \Phi) = (\sum \varphi \leftarrow \Phi.\ \delta\ \varphi)) = \vdash \coprod\ \Phi$
  by (*metis*
      antisym-conv
      dirac-exclusive-implication-completeness
      dirac-list-summation-completeness
      trivial-implication)

theorem (**in** *classical-logic*) *Exclusive-list-summation-completeness*:
  $(\forall\ Pr \in probabilities.\ Pr\ (\bigsqcup\ \Phi) = (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi)) = \vdash \coprod\ \Phi$
  by (*metis*
      antisym-conv
      exclusive-implication-completeness
      list-summation-completeness
      trivial-implication)

lemma (**in** *classical-logic*) *dirac-exclusive-set-summation-completeness*:
  $(\forall\ \delta \in dirac\text{-}measures.\ \delta\ (\bigsqcup\ \Phi) = (\sum \varphi \in set\ \Phi.\ \delta\ \varphi))$
    $= \vdash \coprod\ (remdups\ \Phi)$
  by (*metis*
      (*mono-tags*, *hide-lams*)
      eq-iff

dirac-exclusive-implication-completeness
dirac-set-summation-completeness
trivial-implication
set-remdups
sum.set-conv-list)

theorem (**in** *classical-logic*) *Exclusive-set-summation-completeness*:
$(\forall\ Pr \in probabilities.$
$Pr\ (\bigsqcup\ \Phi) = (\sum \varphi \in set\ \Phi.\ Pr\ \varphi)) = \vdash \coprod\ (remdups\ \Phi)$
by (*metis*
(*mono-tags*, *hide-lams*)
eq-iff
exclusive-implication-completeness
set-summation-completeness
trivial-implication
set-remdups
sum.set-conv-list)

lemma (**in** *probability-logic*) *exclusive-list-set-inequality*:
assumes $\vdash \coprod\ \Phi$
shows $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) = (\sum \varphi{\in}set\ \Phi.\ Pr\ \varphi)$
proof −
have *distinct* (*remdups* Φ) **using** *distinct-remdups* **by** *auto*
hence *duplicates* (*remdups* Φ) = {}
by (*induct* Φ, *simp*+)
moreover have *set* (*remdups* Φ) = *set* Φ
by (*induct* Φ, *simp*, *simp add*: *insert-absorb*)
moreover have $(\forall \varphi \in duplicates\ \Phi.\ \vdash \sim \varphi)$
$\wedge\ (\forall\ \varphi \in set\ \Phi.\ \forall\ \psi \in set\ \Phi.\ (\varphi \neq \psi) \longrightarrow \vdash \sim (\varphi \sqcap \psi))$
using
assms
exclusive-elimination1
exclusive-elimination2
set-deduction-base-theory
by *blast*
ultimately have
$(\forall \varphi{\in}duplicates\ (remdups\ \Phi).\ \vdash \sim \varphi)$
$\wedge\ (\forall\ \varphi \in set\ (remdups\ \Phi).\ \forall\ \psi \in set\ (remdups\ \Phi).$
$(\varphi \neq \psi) \longrightarrow \vdash \sim (\varphi \sqcap \psi))$
by *auto*
hence $\vdash \coprod\ (remdups\ \Phi)$
by (*meson exclusive-equivalence set-deduction-base-theory*)
hence $(\sum \varphi{\in}set\ \Phi.\ Pr\ \varphi) = Pr\ (\bigsqcup\ \Phi)$
by (*metis*
arbitrary-disjunction-remdups
biconditional-equivalence
exclusive-sum-list-identity
sum.set-conv-list)
moreover have $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) = Pr\ (\bigsqcup\ \Phi)$

by (*simp add*: *assms exclusive-sum-list-identity*)
ultimately show *?thesis* **by** *metis*
qed

end

2.3 Finite Boolean Algebra

theory *Finite-Boolean-Algebra*
imports
HOL−Library.Finite-Lattice
HOL−Library.Lattice-Syntax
HOL.Transcendental
begin

sledgehammer-params [*smt-proofs* = *false*]

This section presents finite Boolean algebras and *Birkoff's theorem* [1]. Birkoff's theorem states that any finite Boolean algebra is isomorphic to a powerset, with the usual set-based boolean algebra operations.

In this section and §2.4 build up to a finitary formulation of the *collapse theorem* to be proved in §2.7.1.

2.3.1 Finite Boolean Algebra Axiomatization

The class of finite boolean algebras is simply an extension of *boolean-algebra*. In particular, we assume *finite UNIV* as per *finite*. We also extend the language with infina and suprema (i.e. $\bigsqcap A$ and $\bigsqcup A$ respectively). The new axioms are $\bigsqcap A = \mathit{fold}\ (\sqcap)\ \top\ A$ and its dual $\bigsqcup A = \mathit{fold}\ (\sqcup)\ \bot\ A$.

class *finite-boolean-algebra* = *boolean-algebra* + *finite* + *Inf* + *Sup* +
assumes *Inf-def*: $\bigsqcap A = \mathit{Finite\text{-}Set.fold}\ (\sqcap)\ \top\ A$
assumes *Sup-def*: $\bigsqcup A = \mathit{Finite\text{-}Set.fold}\ (\sqcup)\ \bot\ A$
begin

Finite Boolean algebras are a subclass of complete finite distributive lattices.

subclass *finite-distrib-lattice-complete*
using
Inf-fin.coboundedI
Sup-fin.coboundedI
finite-UNIV
le-bot
top-unique
Inf-def
Sup-def
by (*unfold-locales*, *blast*, *fastforce*, *auto*)
end

2.3.2 Join Prime Elements

The proof of Birkoff's theorem presented here follows Davey and Priestley [8]. The key to their proof is to show that the elements of a finite Boolean algebra have a 1-1 correspondence with sets of *join prime* elements of the Boolean algebra.

Join prime elements are defined as follows.

definition (**in** *bounded-lattice-bot*) *join-prime* :: $'a \Rightarrow bool$ **where**
  join-prime $x \equiv x \neq \bot \wedge (\forall\ y\ z\ .\ x \leq y \sqcup z \longrightarrow x \leq y \vee x \leq z)$

Join prime elements are also sometimes known as *atoms*. They are the smallest elements of the Boolean algebra distinct from $\bot$.

lemma (**in** *boolean-algebra*) *join-prime-def'*:
  join-prime $x = (x \neq \bot \wedge (\forall\ y.\ y \leq x \longrightarrow y = \bot \vee y = x))$
proof
  assume *join-prime* x
  hence $x \neq \bot$
    using *join-prime-def* **by** *blast*
  moreover
  {
    fix y
    assume $y \leq x$ $y \neq x$
    hence $x = x \sqcup y$
      using *sup.orderE* **by** *blast*
    also have $\ldots = (x \sqcup y) \sqcap (y \sqcup -y)$
      by *simp*
    finally have $x = (x \sqcap -y) \sqcup y$
      by (*simp add*: *sup-inf-distrib2*)
    hence $x \leq -y$
      using ‹*join-prime* x› ‹$y \neq x$› ‹$y \leq x$› *eq-iff*
      unfolding *join-prime-def*
      by *force*
    hence $y \leq y \sqcap -y$
      by (*metis*
          ‹$x = x \sqcup y$›
          inf.orderE
          inf-compl-bot-right
          inf-sup-absorb
          order-refl
          sup.commute)
    hence $y = \bot$
      using *sup-absorb2* **by** *fastforce*
  }
  ultimately show $x \neq \bot \wedge (\forall\ y.\ y \leq x \longrightarrow y = \bot \vee y = x)$ **by** *auto*
next
  assume *atomic*: $x \neq \bot \wedge (\forall\ y.\ y \leq x \longrightarrow y = \bot \vee y = x)$
  hence $x \neq \bot$ **by** *auto*
  moreover

{
  fix $y\ z$
  assume $x \leq y \sqcup z$
  hence $x = (x \sqcap y) \sqcup (x \sqcap z)$
    using *inf.absorb1 inf-sup-distrib1* **by** *fastforce*
  moreover
  have $x \leq y \vee (x \sqcap y) = \bot$
    $x \leq z \vee (x \sqcap z) = \bot$
    using *atomic inf.cobounded1 inf.cobounded2* **by** *fastforce+*
  ultimately have $x \leq y \vee x \leq z$
    using *atomic* **by** *auto*
}
ultimately show *join-prime x*
  unfolding *join-prime-def*
  by *auto*
qed

All join prime elements are disjoint.

lemma (**in** *boolean-algebra*) *join-prime-disjoint*:
  assumes *join-prime* α
    and *join-prime* β
  shows $(\alpha = \beta) \longleftrightarrow (\alpha \sqcap \beta \neq \bot)$
proof
  assume $\alpha = \beta$
  hence $\alpha \sqcap \beta = \alpha$
    by *simp*
  thus $\alpha \sqcap \beta \neq \bot$
    using ⟨*join-prime* α⟩
    unfolding *join-prime-def*
    by *auto*
next
  assume $\alpha \sqcap \beta \neq \bot$
  show $\alpha = \beta$
  proof (*rule ccontr*)
    assume $\alpha \neq \beta$
    hence $\neg (\alpha \leq \beta)$
      using ⟨*join-prime* α⟩
        ⟨*join-prime* β⟩
      unfolding *join-prime-def'*
      by *blast*
    hence $\alpha \leq - \beta$
      using *assms(1) join-prime-def* **by** *force*
    hence $\alpha \sqcap \beta = \bot$
      by (*metis inf.commute inf.orderE inf-compl-bot-right*)
    thus *False*
      using ⟨$\alpha \sqcap \beta \neq \bot$⟩
      by *blast*
  qed
qed

definition (**in** *bounded-lattice-bot*) *join-primes* ($\mathcal{J}$) **where**
$\mathcal{J} \equiv \{a \ . \ \textit{join-prime } a\}$

2.3.3 Birkoff's Theorem

Birkoff's theorem states that every non-$\bot$ element of a finite Boolean algebra can be represented by the join prime elements beneath it. It goes on to assert that this representation is a Boolean algebra isomorphism.

In this section we merely demonstrate the representation aspect of Birkoff's theorem. In §2.3.4 we show this representation is a Boolean algebra isomorphism.

The fist step to representing elements is to show that there *exist* join prime elements beneath them. This is done by showing if there is no join prime element, we can make a descending chain with more elements than the finite Boolean algebra we are considering.

fun (**in** *order*) *descending-chain-list* :: *'a list* $\Rightarrow$ *bool* **where**
descending-chain-list [] = *True*
| *descending-chain-list* $[x]$ = *True*
| *descending-chain-list* $(x \# x' \# xs)$
$= (x < x' \wedge \textit{descending-chain-list } (x' \# xs))$

lemma (**in** *order*) *descending-chain-list-tail*:
assumes *descending-chain-list* $(s \# S)$
shows *descending-chain-list* S
using *assms*
by (*induct* S, *auto*)

lemma (**in** *order*) *descending-chain-list-drop-penultimate*:
assumes *descending-chain-list* $(s \# s' \# S)$
shows *descending-chain-list* $(s \# S)$
using *assms*
by (*induct* S, *simp*, *auto*)

lemma (**in** *order*) *descending-chain-list-less-than-others*:
assumes *descending-chain-list* $(s \# S)$
shows $\forall s' \in \textit{set } S. \ s < s'$
using *assms*
by (*induct* S, *auto*, *simp add*: *descending-chain-list-drop-penultimate*)

lemma (**in** *order*) *descending-chain-list-distinct*:
assumes *descending-chain-list* S
shows *distinct* S
using *assms*
by (*induct* S,
simp,

meson
  descending-chain-list-less-than-others
  descending-chain-list-tail
  distinct.simps(*2*)
  less-irrefl)

lemma (**in** *finite-boolean-algebra*) *join-prime-lower-bound-exists*:
  assumes $x \neq \bot$
  shows $\exists\ y \in \mathcal{J}.\ y \leq x$
proof (*rule ccontr*)
  assume $\neg\ (\exists\, y \in \mathcal{J}.\ y \leq x)$
  hence *fresh*: $\forall\ y \leq x.\ y \neq \bot \longrightarrow (\exists\, z < y.\ z \neq \bot)$
    unfolding *join-primes-def*
        join-prime-def'
    using *dual-order.not-eq-order-implies-strict*
    by *fastforce*
  {
    fix $n :: nat$
    have $\exists\ S\ .\ \textit{descending-chain-list}\ S$
        $\wedge\ \textit{length}\ S = n$
        $\wedge\ (\forall\, s \in \textit{set}\ S.\ s \neq \bot \wedge s \leq x)$
    proof (*induct n*)
      case *0*
      have *descending-chain-list* []
          $\wedge$ *length* [] = *0*
          $\wedge$ ($\forall\, s \in$ *set* []. $s \neq \bot \wedge s \leq x$)
        by *auto*
      then show *?case* **by** *simp*
    next
      case (*Suc n*)
      then show *?case* **proof** (*cases* $n = 0$)
        case *True*
        hence *descending-chain-list* [x]
            $\wedge$ *length* [x] = *Suc n*
            $\wedge$ ($\forall\, s \in$ *set* [x]. $s \neq \bot \wedge s \leq x$)
          using ‹$x \neq \bot$›
          by *simp*
        then show *?thesis*
          by *blast*
      next
        case *False*
        from *this* **obtain** *s S* **where**
          descending-chain-list (s # S)
          length (s # S) = n
          $\forall\, s \in$ *set* (s # S). $s \neq \bot \wedge s \leq x$
          using *Suc.hyps length-0-conv descending-chain-list.elims*(*2*)
          by *metis*
        note A = *this*
        from *this* **obtain** s' **where**

$s' < s$
$s' \neq \bot$
using *fresh list.set-intros*(*1*)
by *metis*
note $B = this$
let $?S' = s' \# s \# S$
from A **and** B **have**
descending-chain-list $?S'$
length $?S' = Suc\ n$
$\forall s \in set\ ?S'.\ s \neq \bot \wedge s \leq x$
by *auto*
then show *?thesis* **by** *blast*
qed
qed
}
from *this* **obtain** S :: *'a list* **where**
descending-chain-list S
length $S = 1 + (card\ (UNIV::'a\ set))$
by *auto*
hence $card\ (set\ S) = 1 + (card\ (UNIV::'a\ set))$
using *descending-chain-list-distinct*
distinct-card
by *fastforce*
hence $\neg\ card\ (set\ S) \leq card\ (UNIV::'a\ set)$
by *presburger*
thus *False*
using *card-mono finite-UNIV* **by** *blast*
qed

Having shown that there exists a join prime element beneath every non-$\bot$ element, we show that elements are exactly the suprema of the join prime elements beneath them.

definition (**in** *bounded-lattice-bot*)
join-prime-embedding :: $'a \Rightarrow 'a\ set$ ($\{\!|\ \text{-}\ |\!\}$ [*50*]) **where**
$\{\!|\ x\ |\!\} \equiv \{a \in \mathcal{J}.\ a \leq x\}$

theorem (**in** *finite-boolean-algebra*) *sup-join-prime-embedding-ident*:
$x = \bigsqcup\ \{\!|\ x\ |\!\}$
proof −
have $\forall\ a \in \{\!|\ x\ |\!\}.\ a \leq x$ **unfolding** *join-prime-embedding-def* **by** *auto*
hence $\bigsqcup\ \{\!|\ x\ |\!\} \leq x$
by (*simp add*: *Sup-least*)
moreover
{
fix y
assume $\bigsqcup\ \{\!|\ x\ |\!\} \leq y$
have $x \leq y$
proof (*rule ccontr*)
assume $\neg\ x \leq y$

hence $\bot < x \sqcap -y$
by (*metis bot-less*
compl-sup-top
inf-top-right
le-iff-sup
sup.commute
sup-bot-right
sup-inf-distrib1)
from *this* **obtain** a **where**
$a \in \mathcal{J}$
$a \leq x \sqcap -y$
using *join-prime-lower-bound-exists* [*of* $x \sqcap -y$]
by *blast*
hence $a \in \{\!\!\mid x \mid\!\!\}$
by (*simp add*: *join-prime-embedding-def*)
hence $a \leq y$
using ‹$\bigsqcup \{\!\!\mid x \mid\!\!\} \leq y$›
Sup-upper
order.trans
by *blast*
hence $a \leq y \sqcap -y$
using ‹$a \leq x \sqcap - y$›
inf.boundedE
inf-greatest
by *blast*
hence $a = \bot$
by (*simp add*: *le-bot*)
thus *False*
using ‹$a \in \mathcal{J}$›
unfolding *join-primes-def*
join-prime-def
by *fast*
qed
}
ultimately show *?thesis*
by (*simp add*: *antisym*)
qed

Just as $x = \bigsqcup \{\!\!\mid x \mid\!\!\}$, the reverse is also true; $\lambda\ x.\ \{\!\!\mid x \mid\!\!\}$ and $\lambda\ S.\ \bigsqcup\ S$ are inverses where $S \in \mathcal{J}$.

lemma (**in** *finite-boolean-algebra*) *join-prime-embedding-sup-ident*:
assumes $S \subseteq \mathcal{J}$
shows $S = \{\!\!\mid \bigsqcup\ S \mid\!\!\}$
proof −
have $\forall\ s \in S.\ s \in \mathcal{J} \wedge s \leq \bigsqcup\ S$
using ‹$S \subseteq \mathcal{J}$› *Sup-upper* **by** *auto*
hence $S \subseteq \{\!\!\mid \bigsqcup\ S \mid\!\!\}$
unfolding *join-prime-embedding-def*
by *blast*

moreover
{
  fix x
  assume $x \in \mathcal{J}$
  $x \leq \bigsqcup S$
  have $\exists\ s \in S.\ x \leq s$
  proof (*rule ccontr*)
    assume $\neg\ (\exists\ s \in S.\ x \leq s)$
    hence $\forall\ s \in S.\ x \sqcap s \neq x$
      using *inf.order-iff*
      by *auto*
    moreover
    have $\forall\ s \in S.\ x \sqcap s \leq x$
      by *simp*
    hence $\forall\ s \in S.\ x \sqcap s = \bot \lor x \sqcap s = x$
      using $\langle x \in \mathcal{J} \rangle$
      unfolding *join-primes-def*
      join-prime-def'
      by *blast*
    ultimately have $\forall\ s \in S.\ x \sqcap s = \bot$ **by** *blast*
    hence $x \sqcap \bigsqcup S = \bot$
      by (*simp add*: *inf-Sup*)
    hence $x = \bot$
      using $\langle x \leq \bigsqcup S \rangle$ *inf.order-iff* **by** *blast*
    thus *False*
      using $\langle x \in \mathcal{J} \rangle$
      unfolding *join-primes-def*
      join-prime-def'
      by *auto*
  qed
  hence $\exists\ s \in S.\ x = s$
    using $\langle x \in \mathcal{J} \rangle$
    $\langle S \subseteq \mathcal{J} \rangle$
    unfolding *join-primes-def*
    join-prime-def'
    by *auto*
  hence $x \in S$ **by** *auto*
}
hence $\lbrace\!| \bigsqcup S |\!\rbrace \subseteq S$
  unfolding *join-prime-embedding-def*
  by *blast*
ultimately show *?thesis* **by** *auto*
qed

Given that $\lambda\ x.\ \lbrace\!|\ x\ |\!\rbrace$ has a left and right inverse, we can show it is a *bijection*. Every finite Boolean algebra is isomorphic to the powerset of its join prime elements.

The bijection below is recognizable as a form of *Birkoff's Theorem*.

theorem (**in** *finite-boolean-algebra*) *birkoffs-theorem*:
  bij-betw (λ *x*. ⦃ *x* ⦄) *UNIV* (*Pow* $\mathcal{J}$)
  unfolding *bij-betw-def*
proof
  {
    fix *x y*
    assume ⦃ *x* ⦄ = ⦃ *y* ⦄
    hence $\bigsqcup$ ⦃ *x* ⦄ = $\bigsqcup$ ⦃ *y* ⦄
      by *simp*
    hence *x* = *y*
      using *sup-join-prime-embedding-ident*
      by *auto*
  }
  thus *inj* (λ *x*. ⦃ *x* ⦄)
    unfolding *inj-def*
    by *auto*
next
  show *range* (λ *x*. ⦃ *x* ⦄) = *Pow* $\mathcal{J}$
  proof (*intro equalityI subsetI*)
    fix *S*
    assume *S* $\in$ *range* (λ *x*. ⦃ *x* ⦄)
    thus *S* $\in$ *Pow* $\mathcal{J}$
      unfolding *join-prime-embedding-def*
            Pow-def
      by *auto*
  next
    fix *S*
    assume *S* $\in$ *Pow* $\mathcal{J}$
    hence $\exists$ *x*. ⦃ *x* ⦄ = *S*
      using *join-prime-embedding-sup-ident*
      by *blast*
    thus *S* $\in$ *range* (λ *x*. ⦃ *x* ⦄)
      by *blast*
  qed
qed

2.3.4 Boolean Algebra Isomorphism

The form of Birkoff's theorem presented in §2.3.3 simply gave a bijection between a finite Boolean algebra and the powerset of its join prime elements. This relationship can be extended to a full-blown *Boolean algebra isomorphism*. In particular we have the following properties:

- $\bot$ and $\top$ are preserved; in particular ⦃ $\bot$ ⦄ = {} and ⦃ $\top$ ⦄ = $\mathcal{J}$.
- λ x . ⦃ x ⦄ is a lower complete semi-lattice homomorphism, mapping ⦃ $\bigsqcup$ X ⦄ = ($\bigcup$ $x \in X$. ⦃ x ⦄).
- In addition to preserving arbitrary joins, λ x . ⦃ x ⦄ is a lattice ho-

momorphism, since it also preserves finitary meets with $\{\!\!\mid x \sqcap y \mid\!\!\} = \{\!\!\mid x \mid\!\!\} \cap \{\!\!\mid y \mid\!\!\}$.

- Complementation corresponds to relative set complementation via $\{\!\!\mid - x \mid\!\!\} = \mathcal{J} - \{\!\!\mid x \mid\!\!\}$.

- And finally order is preserved: $x \leq y = (\{\!\!\mid x \mid\!\!\} \subseteq \{\!\!\mid y \mid\!\!\})$

lemma (**in** *finite-boolean-algebra*) *join-primes-bot*:
  $\{\!\!\mid \bot \mid\!\!\} = \{\}$
  unfolding
    join-prime-embedding-def
    join-primes-def
    join-prime-def
  by (*simp add*: *bot-unique*)

lemma (**in** *finite-boolean-algebra*) *join-primes-top*:
  $\{\!\!\mid \top \mid\!\!\} = \mathcal{J}$
  unfolding
    join-prime-embedding-def
  by *auto*

lemma (**in** *finite-boolean-algebra*) *join-primes-join-homomorphism*:
  $\{\!\!\mid x \sqcup y \mid\!\!\} = \{\!\!\mid x \mid\!\!\} \cup \{\!\!\mid y \mid\!\!\}$
proof
  show $\{\!\!\mid x \sqcup y \mid\!\!\} \subseteq \{\!\!\mid x \mid\!\!\} \cup \{\!\!\mid y \mid\!\!\}$
    unfolding
      join-prime-embedding-def
      join-primes-def
      join-prime-def
    by *blast*
next
  show $\{\!\!\mid x \mid\!\!\} \cup \{\!\!\mid y \mid\!\!\} \subseteq \{\!\!\mid x \sqcup y \mid\!\!\}$
    unfolding
      join-prime-embedding-def
    using
      le-supI1
      sup.absorb-iff1
      sup.assoc
    by *force*
qed

lemma (**in** *finite-boolean-algebra*) *join-primes-sup-homomorphism*:
  $\{\!\!\mid \bigsqcup X \mid\!\!\} = (\bigcup x \in X . \{\!\!\mid x \mid\!\!\})$
proof −
  have *finite X*
    by *simp*
  thus *?thesis*
  proof (*induct X rule*: *finite-induct*)

case *empty*
then show *?case* **by** (*simp add*: *join-primes-bot*)
next
case (*insert x X*)
then show *?case* **by** (*simp add*: *join-primes-join-homomorphism*)
qed
qed

lemma (**in** *finite-boolean-algebra*) *join-primes-meet-homomorphism*:
$\{\!| \ x \sqcap y \ |\!\} = \{\!| \ x \ |\!\} \cap \{\!| \ y \ |\!\}$
unfolding
join-prime-embedding-def
by *auto*

Arbitrary meets are also preserved, but relative to a top element $\mathcal{J}$. Perhaps a means of subtyping the algebra of sets of join prime elements would allow us to avoid this epicycle, but that is tricky to execute. We give a less elegant formulation here.

lemma (**in** *finite-boolean-algebra*) *join-primes-inf-homomorphism*:
$\{\!| \ \bigsqcap \ X \ |\!\} = \mathcal{J} \cap (\bigcap \ x \in X. \ \{\!| \ x \ |\!\})$
proof −
have *finite X*
by *simp*
thus *?thesis*
proof (*induct X rule*: *finite-induct*)
case *empty*
then show *?case* **by** (*simp add*: *join-primes-top*)
next
case (*insert x X*)
then show *?case* **by** (*simp add*: *join-primes-meet-homomorphism*, *blast*)
qed
qed

lemma (**in** *finite-boolean-algebra*) *join-primes-complement-homomorphism*:
$\{\!| \ - \ x \ |\!\} = \mathcal{J} - \{\!| \ x \ |\!\}$
proof
show $\{\!| \ - \ x \ |\!\} \subseteq \mathcal{J} - \{\!| \ x \ |\!\}$
proof
fix *j*
assume $j \in \{\!| \ - \ x \ |\!\}$
hence $j \notin \{\!| \ x \ |\!\}$
unfolding
join-prime-embedding-def
join-primes-def
join-prime-def
by (*metis*
(*mono-tags*, *lifting*)
CollectD

bot-unique
inf.boundedI
inf-compl-bot)
thus $j \in \mathcal{J} - \{\!\!\{\ x\ \}\!\!\}$
using ‹$j \in \{\!\!\{\ -x\ \}\!\!\}$›
unfolding
join-prime-embedding-def
by *blast*
qed
next
show $\mathcal{J} - \{\!\!\{\ x\ \}\!\!\} \subseteq \{\!\!\{\ -x\ \}\!\!\}$
proof
fix j
assume $j \in \mathcal{J} - \{\!\!\{\ x\ \}\!\!\}$
hence $j \in \mathcal{J}$ **and** $\neg\ j \leq x$
unfolding *join-prime-embedding-def*
by *blast+*
moreover have $j \leq x \sqcup -x$
by *auto*
ultimately have $j \leq -x$
unfolding
join-primes-def
join-prime-def
by *blast*
thus $j \in \{\!\!\{\ -x\ \}\!\!\}$
unfolding *join-prime-embedding-def*
using ‹$j \in \mathcal{J}$›
by *auto*
qed
qed

lemma (**in** *finite-boolean-algebra*) *join-primes-order-isomorphism*:
$x \leq y = (\{\!\!\{\ x\ \}\!\!\} \subseteq \{\!\!\{\ y\ \}\!\!\})$
by (
rule iffI,
simp add: *Collect-mono dual-order.trans join-prime-embedding-def*,
metis
(*full-types*)
Sup-subset-mono
sup-join-prime-embedding-ident)

2.3.5 Cardinality

Another consequence of Birkoff's theorem from §2.3.3 is that we can show every finite Boolean algebra has a cardinality which is a power of two. This gives a bound on the number of join prime elements, which must be logarithmic in the size of the finite Boolean algebra they belong to.

lemma (**in** *finite-boolean-algebra*) *UNIV-card*:

```
  card (UNIV::'a set) = card (Pow 𝒥)
  using bij-betw-same-card [where f=λx. ⦃ x ⦄]
        birkoffs-theorem
  by blast

lemma finite-Pow-card:
  assumes finite X
  shows card (Pow X) = 2 powr (card X)
  using assms
proof (induct X rule: finite-induct)
  case empty
  then show ?case by fastforce
next
  case (insert x X)
  have 0 ≤ (2 :: real) by auto
  hence two-powr-one: (2 :: real) = 2 powr 1 by fastforce
  have bij-betw (λ x. fst x ∪ snd x) ({{},{x}} × Pow X) (Pow (insert x X))
    unfolding bij-betw-def
  proof
    {
      fix y z
      assume y ∈ {{}, {x}} × Pow X
             z ∈ {{}, {x}} × Pow X
             fst y ∪ snd y = fst z ∪ snd z
             (is ?Uy = ?Uz)
      hence x ∉ snd y
            x ∉ snd z
            fst y = {x} ∨ fst y = {}
            fst z = {x} ∨ fst z = {}
        using insert.hyps(2) by auto
      hence x ∈ ?Uy ⟷ fst y = {x}
            x ∈ ?Uz ⟷ fst z = {x}
            x ∉ ?Uy ⟷ fst y = {}
            x ∉ ?Uz ⟷ fst z = {}
            snd y = ?Uy − {x}
            snd z = ?Uz − {x}
        by auto
      hence x ∈ ?Uy ⟷ y = ({x}, ?Uy − {x})
            x ∈ ?Uz ⟷ z = ({x}, ?Uz − {x})
            x ∉ ?Uy ⟷ y = ({}, ?Uy − {x})
            x ∉ ?Uz ⟷ z = ({}, ?Uz − {x})
        by (metis fst-conv prod.collapse)+
      hence y = z
        using ‹?Uy = ?Uz›
        by metis
    }
    thus inj-on (λx. fst x ∪ snd x) ({{}, {x}} × Pow X)
      unfolding inj-on-def
      by auto
```

next
show $(\lambda x.\ \mathit{fst}\ x \cup \mathit{snd}\ x)$ ` $(\{\{\},\ \{x\}\} \times \mathit{Pow}\ X) = \mathit{Pow}\ (\mathit{insert}\ x\ X)$
proof (*intro equalityI subsetI*)
fix y
assume $y \in (\lambda x.\ \mathit{fst}\ x \cup \mathit{snd}\ x)$ ` $(\{\{\},\ \{x\}\} \times \mathit{Pow}\ X)$
from *this* **obtain** z **where**
$z \in (\{\{\},\ \{x\}\} \times \mathit{Pow}\ X)$
$y = \mathit{fst}\ z \cup \mathit{snd}\ z$
by *auto*
hence $\mathit{snd}\ z \subseteq X$
$\mathit{fst}\ z \subseteq \mathit{insert}\ x\ X$
using *SigmaE* **by** *auto*
thus $y \in \mathit{Pow}\ (\mathit{insert}\ x\ X)$
using $\langle y = \mathit{fst}\ z \cup \mathit{snd}\ z\rangle$ **by** *blast*
next
fix y
assume $y \in \mathit{Pow}\ (\mathit{insert}\ x\ X)$
let $?z = (\mathit{if}\ x \in y\ \mathit{then}\ \{x\}\ \mathit{else}\ \{\},\ y - \{x\})$
have $?z \in (\{\{\},\ \{x\}\} \times \mathit{Pow}\ X)$
using $\langle y \in \mathit{Pow}\ (\mathit{insert}\ x\ X)\rangle$ **by** *auto*
moreover have $(\lambda x.\ \mathit{fst}\ x \cup \mathit{snd}\ x)\ ?z = y$
by *auto*
ultimately show $y \in (\lambda x.\ \mathit{fst}\ x \cup \mathit{snd}\ x)$ ` $(\{\{\},\ \{x\}\} \times \mathit{Pow}\ X)$
by *blast*
qed
qed
hence $\mathit{card}\ (\mathit{Pow}\ (\mathit{insert}\ x\ X)) = \mathit{card}\ (\{\{\},\{x\}\} \times \mathit{Pow}\ X)$
using *bij-betw-same-card* **by** *fastforce*
also have $\ldots = 2 * \mathit{card}\ (\mathit{Pow}\ X)$
by (*simp add*: *insert.hyps*(*1*))
also have $\ldots = 2 * (2\ \mathit{powr}\ (\mathit{card}\ X))$
by (*simp add*: *insert.hyps*(*3*))
also have $\ldots = (2\ \mathit{powr}\ 1) * 2\ \mathit{powr}\ (\mathit{card}\ X)$
using *two-powr-one*
by *fastforce*
also have $\ldots = 2\ \mathit{powr}\ (1 + \mathit{card}\ X)$
by (*simp add*: *powr-add*)
also have $\ldots = 2\ \mathit{powr}\ (\mathit{card}\ (\mathit{insert}\ x\ X))$
by (*simp add*: *insert.hyps*(*1*) *insert.hyps*(*2*))
finally show *?case* **.**
qed

lemma (**in** *finite-boolean-algebra*) *UNIV-card-powr-2*:
$\mathit{card}\ (\mathit{UNIV}::'a\ \mathit{set}) = 2\ \mathit{powr}\ (\mathit{card}\ \mathcal{J})$
using *finite* [*of* $\mathcal{J}$]
finite-Pow-card [*of* $\mathcal{J}$]
UNIV-card
by *linarith*

```
lemma (in finite-boolean-algebra) join-primes-card-log-2:
  card 𝒥 = log 2 (card (UNIV :: 'a set))
proof (cases card (UNIV :: 'a set) = 1)
  case True
  hence ∃ x :: 'a. UNIV = {x}
    using card-1-singletonE by blast
  hence ∀ x y :: 'a. x ∈ UNIV ⟶ y ∈ UNIV ⟶ x = y
    by (metis (mono-tags) singletonD)
  hence ∀ x y :: 'a. x = y
    by blast
  hence ∀ x. x = ⊥
    by blast
  hence 𝒥 = {}
    unfolding join-primes-def
              join-prime-def
    by blast
  hence card 𝒥 = (0 :: real)
    by simp
  moreover
  have log 2 (card (UNIV :: 'a set)) = 0
    by (simp add: True)
  ultimately show ?thesis by auto
next
  case False
  hence 0 < 2 powr (card 𝒥) 2 powr (card 𝒥) ≠ 1
    using finite-UNIV-card-ge-0 finite UNIV-card-powr-2
    by (simp, linarith)
  hence log 2 (2 powr (card 𝒥)) = card 𝒥
    by simp
  then show ?thesis
    using UNIV-card-powr-2
    by simp
qed

end
```

2.4 Finite Boolean Algebra Probability

```
theory Finite-Probability
  imports
    ../Logic/Probability-Logic
    Finite-Boolean-Algebra
begin

sledgehammer-params [smt-proofs = false]

no-notation
  verum (⊤) and
  falsum (⊥) and
```

disjunction (**infixr** $\sqcup$ *67*) **and**
conjunction (**infixr** $\sqcap$ *67*) **and**
arbitrary-conjunction ($\bigsqcap$) **and**
arbitrary-disjunction ($\bigsqcup$)

class $\mathcal{P}$ =
fixes $\mathcal{P}$ :: $'a \Rightarrow$ *real*

2.4.1 Definition of Finitely Additive Probability

TODO: cite [5], [7], "Elementary Theory of Probability" [18]

class *finitely-additive-probability* = $\mathcal{P}$ + *boolean-algebra* +
assumes *probability-non-negative*: $\mathcal{P}\ \varphi \geq 0$
assumes *probability-unity*: $\mathcal{P}\ \top = 1$
assumes *finite-additivity*: $\varphi \sqcap \psi = \bot \Longrightarrow \mathcal{P}\ (\varphi \sqcup \psi) = \mathcal{P}\ \varphi + \mathcal{P}\ \psi$

context *boolean-algebra* **begin**

2.4.2 Equivalence With Probability Logic

The Boolean algebra formulation of finitely additive probability is in fact a special case of probability logic as presented in §2.1.

definition *residual* (**infixr** $\Rightarrow$ *70*) **where**
$\varphi \Rightarrow \psi \equiv -\ \varphi \sqcup \psi$

lemma *residual-galois-connection*:
$A \sqcap B \leq C \longleftrightarrow B \leq A \Rightarrow C$
proof
assume $A \sqcap B \leq C$
have $B \sqcup (A \Rightarrow C) = A \Rightarrow C \sqcup B \sqcap \top$
unfolding *residual-def*
using *inf-top.right-neutral*
sup-commute
by *presburger*
moreover have $\top = A \Rightarrow C \sqcup A$
unfolding *residual-def*
using *sup-commute sup-compl-top-left2*
by *fastforce*
ultimately have $B \sqcup (A \Rightarrow C) = A \Rightarrow C \sqcup B \sqcap A$
unfolding *residual-def*
by (*simp add*: *sup-inf-distrib1*)
moreover have $A \sqcap B \sqcup C = C$
using ‹$A \sqcap B \leq C$› *sup.absorb-iff2* **by** *blast*
ultimately show $B \leq A \Rightarrow C$
unfolding *residual-def*
by (*metis*
inf-commute
sup.absorb-iff2

```
        sup.semigroup-axioms
        sup-commute
        semigroup.assoc)
next
  assume B ≤ A ⇒ C
  hence B ⊓ (A ⇒ C) = B
    using inf-absorb1
    unfolding residual-def
    by fastforce
  moreover have A ⇒ C = C ⊔ − A
    unfolding residual-def
    by (simp add: abel-semigroup.commute sup.abel-semigroup-axioms)
  moreover have A ⊓ B ⊓ C = A ⊓ (B ⊓ C)
      by (simp add: inf.semigroup-axioms semigroup.assoc)
  ultimately show A ⊓ B ≤ C
    unfolding residual-def
    by (metis
        (no-types)
        inf.orderI
        inf-compl-bot-right
        inf-sup-distrib1
        sup-bot.right-neutral)
qed

interpretation classical-logic (=) ⊤ (⇒) ⊥
proof standard
  fix φ ψ
  show ⊤ = φ ⇒ ψ ⇒ φ
    unfolding residual-def
    by (simp add: sup.commute)
next
  fix φ ψ χ
  show ⊤ = (φ ⇒ ψ ⇒ χ) ⇒ (φ ⇒ ψ) ⇒ φ ⇒ χ
  proof −
    have ⊤ = (φ ⇒ χ) ⇒ φ ⇒ χ
      unfolding residual-def
      by (metis compl-sup-top)
    moreover have − φ ⇒ φ ⇒ χ = − φ ⇒ (φ ⇒ ψ ⇒ χ) ⇒ φ ⇒ χ
      unfolding residual-def
      by (metis sup-compl-top-left2 sup-left-commute)
    moreover have ψ ⇒ (φ ⇒ ψ ⇒ χ) ⇒ φ ⇒ χ = χ ⇒ φ ⇒ χ
      unfolding residual-def
      by (metis compl-sup-top sup-compl-top-left2 sup-left-commute)
    ultimately have ⊤ = (φ ⇒ ψ ⇒ χ) ⇒ (φ ⇒ χ) ⊔ − (φ ⇒ ψ)
      unfolding residual-def
      using
        abel-semigroup.commute
        sup.abel-semigroup-axioms
        sup-inf-distrib1
```

```
      by fastforce
    hence ⊤ = (φ ⇒ ψ) ⇒ (φ ⇒ ψ ⇒ χ) ⇒ φ ⇒ χ
      unfolding residual-def
      by (simp add: abel-semigroup.commute sup.abel-semigroup-axioms)
    thus ?thesis
      unfolding residual-def
      by (simp add: sup-left-commute)
  qed
next
  fix φ ψ
  show ⊤ = φ ⇒ ψ ⟹ ⊤ = φ ⟹ ⊤ = ψ
    unfolding residual-def
    using compl-top-eq
    by auto
next
  fix φ
  show ⊤ = ((φ ⇒ ⊥) ⇒ ⊥) ⇒ φ
    unfolding residual-def
    by simp
qed

lemmas axiom-k = axiom-k
lemmas axiom-s = axiom-s
lemmas double-negation = double-negation
lemmas modus-ponens = modus-ponens
lemmas probabilities-def = probabilities-def

lemma probabilities-def':
  probabilities =
     { 𝒫. class.finitely-additive-probability
              𝒫 (−) uminus (⊓) (≤) (<) (⊔) ⊥ ⊤ }
  (is - = ?ba-probabilities)
proof
  show ?ba-probabilities ⊆ probabilities
  proof
    fix 𝒫
    assume 𝒫 ∈ ?ba-probabilities
    from this interpret
      finitely-additive-probability 𝒫
      unfolding probabilities-def
      by auto
    have class.probability-logic ((=) ⊤) (⇒) ⊥ 𝒫
    proof standard
      fix φ
      show 0 ≤ 𝒫 φ
        by (simp add: probability-non-negative)
    next
      fix φ
      show ⊤ = φ ⟹ 𝒫 φ = 1
```

```
        using probability-unity by blast
      next
        fix φ ψ
        assume ⊤ = (φ ⇒ ψ ⇒ ⊥)
        hence φ ⊓ ψ = ⊥
          unfolding residual-def
          using compl-top-eq by auto
        thus 𝒫 ((φ ⇒ ⊥) ⇒ ψ) = 𝒫 φ + 𝒫 ψ
          unfolding residual-def
          by (simp add: finite-additivity)
      qed
      thus 𝒫 ∈ probabilities
        unfolding probabilities-def
        by auto
    qed
  next
    show probabilities ⊆ ?ba-probabilities
    proof
      fix 𝒫
      assume 𝒫 ∈ probabilities
      from this interpret probability-logic (=) ⊤ (⇒) ⊥ 𝒫
        unfolding probabilities-def
        by auto
      have
        class.finitely-additive-probability
          𝒫 (−) uminus (⊓) (≤) (<) (⊔) ⊥ ⊤
      proof standard
        fix φ
        show 0 ≤ 𝒫 φ
          by (simp add: probability-non-negative)
      next
        show 𝒫 ⊤ = 1
          using probability-unity by blast
      next
        fix φ ψ
        assume φ ⊓ ψ = ⊥
        thus 𝒫 (φ ⊔ ψ) = 𝒫 φ + 𝒫 ψ
          using
            probability-implicational-additivity
            compl-bot-eq
            sup-bot.right-neutral
            residual-def
          by force
      qed
      thus 𝒫 ∈ ?ba-probabilities
        by auto
    qed
  qed
```

lemma *join-prime-to-dirac-measure*:
  assumes $\alpha \in \mathcal{J}$
  shows $(\lambda\ \varphi.\ \mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0) \in \mathit{dirac\text{-}measures}$
  (**is** $?\delta \in \mathit{dirac\text{-}measures}$)
proof −
  have $\mathit{class.probability\text{-}logic}\ ((=)\ \top)\ (\Rightarrow)\ \bot\ ?\delta$
  proof *standard*
    fix φ
    show $0 \leq\ ?\delta\ \varphi$
      by *fastforce*
  next
    fix φ
    show $\top = \varphi \Longrightarrow (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0) = 1$
      using *top-greatest* **by** *auto*
  next
    fix $\varphi\ \psi$
    assume $\top = \varphi \Rightarrow \psi \Rightarrow \bot$
    hence $\varphi \sqcap \psi = \bot$
      using *compl-top-eq residual-def* **by** *auto*
    hence $\neg\ \alpha \leq \varphi \vee \neg\ \alpha \leq \psi$
      using $\langle\alpha \in \mathcal{J}\rangle$
      unfolding *join-primes-def join-prime-def*
      using *bot-unique inf.boundedI* **by** *blast*
    moreover have $\alpha \leq \varphi \sqcup \psi \longleftrightarrow \alpha \leq \varphi \vee \alpha \leq \psi$
      using $\langle\alpha \in \mathcal{J}\rangle$
      unfolding *join-primes-def join-prime-def*
      using *le-supI1 le-supI2* **by** *blast*
    ultimately show $?\delta\ ((\varphi \Rightarrow \bot) \Rightarrow \psi) =\ ?\delta\ \varphi +\ ?\delta\ \psi$
      unfolding *residual-def*
      by *auto*
  qed
  thus *?thesis*
    unfolding *dirac-measures-def*
    by *simp*
qed

lemma *conditional-probability-measure*:
  fixes $\mathcal{P} :: 'a \Rightarrow \mathit{real}$
  assumes $\mathcal{P} \in \mathit{probabilities}$ **and** $\mathcal{P}\ \psi \neq 0$
  shows $(\lambda\ \varphi.\ \mathcal{P}\ (\varphi \sqcap \psi)\ /\ \mathcal{P}\ \psi) \in \mathit{probabilities}$
proof −
  from *assms* **interpret**
    finitely-additive-probability $\mathcal{P}$
    unfolding *probabilities-def'*
    by *auto*
  have $\mathcal{P}\ \psi > 0$
    using
      $\langle\mathcal{P}\ \psi \neq 0\rangle$
      probability-non-negative

order-class.dual-order.order-iff-strict
by *blast*
let $?\mathcal{P}' = \lambda\ \varphi.\ \mathcal{P}\ (\varphi \sqcap \psi)\ /\ \mathcal{P}\ \psi$
have *class.finitely-additive-probability*
$?\mathcal{P}'\ (-)\ uminus\ (\sqcap)\ (\leq)\ (<)\ (\sqcup)\ \bot\ \top$
proof *standard*
fix φ
show $0 \leq \mathcal{P}\ (\varphi \sqcap \psi)\ /\ \mathcal{P}\ \psi$
by (*simp add*: *probability-non-negative*)
next
show $\mathcal{P}\ (\top \sqcap \psi)\ /\ \mathcal{P}\ \psi = 1$
using ‹$0 < \mathcal{P}\ \psi$› *inf-top-left* **by** *auto*
next
fix $\varphi\ \chi$
assume $\varphi \sqcap \chi = \bot$
hence $\mathcal{P}\ ((\varphi \sqcup \chi) \sqcap \psi) = \mathcal{P}\ (\varphi \sqcap \psi) + \mathcal{P}\ (\chi \sqcap \psi)$
by (*metis*
finite-additivity
inf.assoc
inf.commute
inf-bot-right
inf-sup-distrib2)
thus $\mathcal{P}\ ((\varphi \sqcup \chi) \sqcap \psi)\ /\ \mathcal{P}\ \psi = \mathcal{P}\ (\varphi \sqcap \psi)\ /\ \mathcal{P}\ \psi + \mathcal{P}\ (\chi \sqcap \psi)\ /\ \mathcal{P}\ \psi$
by (*simp add*: *add-divide-distrib*)
qed
thus *?thesis*
unfolding *probabilities-def'*
by *blast*
qed

lemma *probabilities-convex*:
fixes $\mathcal{P}\ \mathcal{Q} :: 'a \Rightarrow real$ **and** $\alpha :: real$
assumes $\{\mathcal{P},\mathcal{Q}\} \subseteq probabilities$ **and** $0 \leq \alpha$ **and** $\alpha \leq 1$
shows $(\lambda\ \varphi.\ \alpha * \mathcal{P}\ \varphi + (1 - \alpha) * \mathcal{Q}\ \varphi) \in probabilities$
proof −
let $?\mathcal{M} = \lambda\ \varphi.\ \alpha * \mathcal{P}\ \varphi + (1 - \alpha) * \mathcal{Q}\ \varphi$
from *assms* **interpret** *finitely-additive-probability* $\mathcal{P}$
unfolding *probabilities-def'*
by *auto*
note $\mathcal{P}$*-probability-non-negative* = *probability-non-negative*
note $\mathcal{P}$*-probability-unity* = *probability-unity*
note $\mathcal{P}$*-finite-additivity* = *finite-additivity*
from *assms* **interpret** *finitely-additive-probability* $\mathcal{Q}$
unfolding *probabilities-def'*
by *auto*
have *class.finitely-additive-probability*
$?\mathcal{M}\ (-)\ uminus\ (\sqcap)\ (\leq)\ (<)\ (\sqcup)\ \bot\ \top$
proof *standard*
fix φ

```
    show 0 ≤ α ∗ 𝒫 φ + (1 − α) ∗ 𝒬 φ
      by (simp add:
            𝒫-probability-non-negative
            probability-non-negative
            ‹0 ≤ α›
            ‹α ≤ 1›)
  next
    show α ∗ 𝒫 ⊤ + (1 − α) ∗ 𝒬 ⊤ = 1
      using 𝒫-probability-unity probability-unity by auto
  next
    fix φ ψ
    assume φ ⊓ ψ = ⊥
    thus   α ∗ 𝒫 (φ ⊔ ψ) + (1 − α) ∗ 𝒬 (φ ⊔ ψ)
         = α ∗ 𝒫 φ + (1 − α) ∗ 𝒬 φ + (α ∗ 𝒫 ψ + (1 − α) ∗ 𝒬 ψ)
      by (simp add: 𝒫-finite-additivity distrib-left finite-additivity)
  qed
  thus ?thesis
    unfolding probabilities-def'
    by auto
qed

end

context finitely-additive-probability begin

interpretation classical-logic (=) ⊤ (⇒) ⊥
  by (standard,
        simp add: axiom-k,
        simp add: axiom-s,
        metis modus-ponens,
        simp add: double-negation)

interpretation probability-logic (=) ⊤ (⇒) ⊥ 𝒫
proof −
  have class.finitely-additive-probability
         𝒫 (−) uminus (⊓) (≤) (<) (⊔) ⊥ ⊤
    by standard
  hence 𝒫 ∈ probabilities
    unfolding probabilities-def'
    by auto

  thus class.probability-logic ((=) ⊤) (⇒) ⊥ 𝒫
    unfolding probabilities-def
    by blast
qed

lemma sum-rule: 𝒫 a + 𝒫 b = 𝒫 (a ⊓ b) + 𝒫 (a ⊔ b)
  by (metis compl-inf
            conjunction-def
```

```
            disjunction-def
            double-compl
            residual-def
            sum-rule
            sup.commute
            sup-bot.left-neutral)

lemma conditional-probability-join-prime:
  assumes α ∈ 𝒥 and 𝒫 α ≠ 0
  shows  𝒫 (φ ⊓ α) / 𝒫 α = (if α ≤ φ then 1 else 0)
proof (cases α ≤ φ)
  case True
  hence 𝒫 (φ ⊓ α) = 𝒫 α
    by (simp add: inf-absorb2)
  hence 𝒫 (φ ⊓ α) / 𝒫 α = 1
    using ‹𝒫 α ≠ 0› right-inverse-eq by blast
  then show ?thesis
    using ‹α ≤ φ› by simp
next
  case False
  hence α ≤ − φ
    using ‹α ∈ 𝒥› top-greatest
    unfolding join-primes-def join-prime-def
    by force
  hence φ ⊓ α = ⊥
    by (metis inf-absorb1 inf-compl-bot-right)
  hence 𝒫 (φ ⊓ α) / 𝒫 α = 0
    using finite-additivity inf-bot-right sup-bot.right-neutral by fastforce
  then show ?thesis
    using ‹¬ α ≤ φ› by auto
qed

lemma join-prime-conditional-probability:
  assumes ∀ φ. 𝒫 (φ ⊓ α) / 𝒫 α = (if α ≤ φ then 1 else 0)
  shows  α ∈ 𝒥
proof −
  have 𝒫 (⊤ ⊓ α) / 𝒫 α = 1
    using assms top-greatest by auto
  hence 𝒫 α > 0
    using less-eq-real-def probability-non-negative by fastforce
  hence α ≠ ⊥
    using gaines-weatherson-antithesis by auto
  moreover
  have ⋆: ∀ φ. 𝒫 (φ ⊓ α) = (if α ≤ φ then 𝒫 α else 0)
    by (metis ‹𝒫 (⊤ ⊓ α) / 𝒫 α = 1›
              ‹∀ φ. 𝒫 (φ ⊓ α) / 𝒫 α = (if α ≤ φ then 1 else 0)›
              divide-eq-0-iff
              inf.absorb2 zero-neq-one)
  {
```

fix φ ψ
assume $\alpha \leq \varphi \sqcup \psi$
have $\alpha \leq \varphi \vee \alpha \leq \psi$
proof (*rule ccontr*)
assume $\neg$ $(\alpha \leq \varphi \vee \alpha \leq \psi)$
hence $\mathcal{P}$ $(\varphi \sqcap \alpha) = \mathit{0}$
$\mathcal{P}$ $(\psi \sqcap \alpha) = \mathit{0}$
using $\star$ **by** *auto*
hence $\mathit{0} = \mathcal{P}$ $((\varphi \sqcap \alpha) \sqcap (\psi \sqcap \alpha)) + \mathcal{P}$ $((\varphi \sqcap \alpha) \sqcup (\psi \sqcap \alpha))$
using *sum-rule* **by** *auto*
hence $\mathit{0} = \mathcal{P}$ $(\varphi \sqcap \psi \sqcap \alpha) + \mathcal{P}$ $((\varphi \sqcup \psi) \sqcap \alpha)$
by (*simp add*: *inf.commute inf.left-commute inf-sup-distrib1*)
hence $\mathit{0} = \mathcal{P}$ $(\varphi \sqcap \psi \sqcap \alpha) + \mathcal{P}$ α
by (*simp add*: ‹$\alpha \leq \varphi \sqcup \psi$› *inf.absorb2*)
hence $\mathit{0} > \mathcal{P}$ $(\varphi \sqcap \psi \sqcap \alpha)$
using ‹$\mathit{0} < \mathcal{P}$ α› **by** *linarith*
thus *False*
using *probability-non-negative not-le* **by** *blast*
qed
}
ultimately show *?thesis*
unfolding *join-primes-def join-prime-def*
by *blast*
qed

lemma *monotonicity*: $a \leq b \Longrightarrow \mathcal{P}$ $a \leq \mathcal{P}$ b
by (*metis*
monotonicity
residual-def
sup.commute
sup.left-commute
sup-absorb1
sup-cancel-left1)

lemmas *gaines-weatherson-antithesis* = *gaines-weatherson-antithesis*

lemma *complementation*: $\mathcal{P}$ $(-\ \varphi) = \mathit{1} - \mathcal{P}$ φ
by (*metis add-diff-cancel-left'*
finite-additivity
probability-unity
inf-compl-bot
sup-compl-top)

lemma *finite-certainty*:
assumes *finite A* **and** $\forall$ $a \in A$. $\mathcal{P}$ $a = \mathit{1}$
shows $\mathcal{P}$ (*Finite-Set.fold* $(\sqcap)$ $\top$ A) $= \mathit{1}$
using *assms*
proof (*induct A rule*: *finite-induct*)
case *empty*

show $\mathcal{P}$ (*Finite-Set.fold* ($\sqcap$) $\top$ {}) = *1*
  by (*simp add*: *probability-unity*)
next
  case (*insert a A*)
  have $\star$: $\mathcal{P}$ (*Finite-Set.fold* ($\sqcap$) $\top$ (*insert a A*))
      $= \mathcal{P}$ ($a \sqcap$ *Finite-Set.fold* ($\sqcap$) $\top$ A)
    (**is** $\mathcal{P}$ *?A'* $= \mathcal{P}$ ($a \sqcap$ *?A*))
    by (*simp add*:
      comp-fun-idem.fold-insert-idem
      insert.hyps(*1*)
      comp-fun-idem-inf)
  have $\mathcal{P}$ *?A* = *1*
    using *insert.hyps*(*3*) *insert.prems* **by** *blast*
  moreover have $\mathcal{P}$ a = *1*
    by (*simp add*: *insert.prems*)
  moreover
  have $a \leq a \sqcup$ *?A* **by** *simp*
  hence *1* $\leq \mathcal{P}$ ($a \sqcup$ *?A*)
    using *monotonicity* ‹$\mathcal{P}$ a = *1*›
    by *fastforce*
  hence $\mathcal{P}$ ($a \sqcup$ *?A*) = *1*
    using *unity-upper-bound* [*of* $a \sqcup$ *?A*]
    by *linarith*
  ultimately have $\mathcal{P}$ ($a \sqcap$ *?A*) = *1*
    using *sum-rule* [**where** a=a **and** b=*?A*]
    by *linarith*
  thus $\mathcal{P}$ *?A'* = *1*
    using $\star$ **by** *auto*
qed

lemma *full-finite-additivity*:
  assumes *finite* A **and** $\forall\ a \in A.\ \forall\ a' \in A.\ a \neq a' \longrightarrow a \sqcap a' = \bot$
  shows $\mathcal{P}$ (*Finite-Set.fold* ($\sqcup$) $\bot$ A) = ($\sum\ a \in A.\ \mathcal{P}\ a$)
  using *assms*
proof (*induct A rule*: *finite-induct*)
  case *empty*
  then show *?case*
    using *gaines-weatherson-antithesis* **by** *fastforce*
next
  case (*insert a A*)
  hence $\forall\, a' \in A.\ a \sqcap a' = \bot$
    by *auto*
  with ‹*finite A*› ‹$a \notin A$›
    have $a \sqcap$ (*Finite-Set.fold* ($\sqcup$) $\bot$ A) = $\bot$ (**is** $a \sqcap$ *?UA* = $\bot$)
  proof (*induct A rule*: *finite-induct*)
    case *empty*
    then show *?case* **by** *auto*
  next
    case (*insert a' A*)

hence $a \sqcap (\mathit{Finite\text{-}Set.fold}\ (\sqcup)\ \bot\ A) = \bot$ (**is** $a \sqcap\ ?UA = \bot$)
$a \sqcap a' = \bot$
by *auto*
moreover
have $\mathit{Finite\text{-}Set.fold}\ (\sqcup)\ \bot\ (\{a'\} \cup A) = a' \sqcup\ ?UA$
(**is** $?UA' = \text{-}$)
by (*simp add*:
comp-fun-idem.fold-insert-idem
⟨*finite A*⟩
comp-fun-idem-sup)
hence $a \sqcap\ ?UA' = (a \sqcap a') \sqcup (a \sqcap\ ?UA)$
using *inf-sup-distrib1* **by** *auto*
ultimately show *?case*
by *auto*
qed
moreover have $\mathit{Finite\text{-}Set.fold}\ (\sqcup)\ \bot\ (\{a\} \cup A) = a \sqcup\ ?UA$
by (*simp add*: *comp-fun-idem.fold-insert-idem* ⟨*finite A*⟩ *comp-fun-idem-sup*)
moreover have $\mathcal{P}\ ?UA = (\sum\ a \in A.\ \mathcal{P}\ a)$
using *insert* **by** *blast*
ultimately show *?case*
by (*simp add*: ⟨*finite A*⟩ ⟨$a \notin A$⟩ *finite-additivity*)
qed

end

2.4.3 Collapse Theorem For Finite Boolean Algebras

context *finite-boolean-algebra* **begin**

interpretation *classical-logic* $(=)\ \top\ (\Rightarrow)\ \bot$
by (*standard*,
simp add: *axiom-k*,
simp add: *axiom-s*,
metis modus-ponens,
simp add: *double-negation*)

lemma *join-prime-decomposition*:
fixes $\mathcal{P} :: {'a} \Rightarrow \mathit{real}$
assumes $\mathcal{P} \in \mathit{probabilities}$
shows $\mathcal{P}\ \varphi = (\sum\ \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \le \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0))$
proof −
interpret *finitely-additive-probability* $\mathcal{P}$
using ⟨$\mathcal{P} \in \mathit{probabilities}$⟩
unfolding *probabilities-def'*
by *blast*
have $\star$: $\varphi = \bigsqcup\ \{\ \alpha \in \mathcal{J}.\ \alpha \le \varphi\ \}$ (**is** $\varphi = \bigsqcup\ ?\mathcal{J}\varphi$)
using
join-prime-embedding-def
sup-join-prime-embedding-ident

by *auto*
have $\forall\ \alpha \in\ ?\mathcal{J}\varphi.\ \forall\ \alpha' \in\ ?\mathcal{J}\varphi.\ \alpha \neq \alpha' \longrightarrow \alpha \sqcap \alpha' = \bot$
unfolding *join-primes-def*
by (*metis inf.cobounded1 inf.commute join-prime-def' mem-Collect-eq*)
hence $\mathcal{P}\ (\bigsqcup\ ?\mathcal{J}\varphi) = (\sum\ \alpha \in\ ?\mathcal{J}\varphi.\ \mathcal{P}\ \alpha)$
by (*simp add*: *Sup-def full-finite-additivity*)
with $\star$ **have** $\dagger$: $\mathcal{P}\ \varphi = (\sum\ \alpha \in\ ?\mathcal{J}\varphi.\ \mathcal{P}\ \alpha)$ **by** *auto*
have *finite* $?\mathcal{J}\varphi$ **by** *auto*
hence $(\sum\ \alpha \in\ ?\mathcal{J}\varphi.\ \mathcal{P}\ \alpha) = (\sum\ \alpha \in\ ?\mathcal{J}\varphi.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0))$
by (*induct* $?\mathcal{J}\varphi$ *rule*: *finite-induct*, *auto*)
with $\dagger$ **have** $\mathcal{P}\ \varphi = (\sum\ \alpha \in\ ?\mathcal{J}\varphi.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0))$
(**is** - = $?\Sigma 1$)
by *presburger*
moreover
let $?n\mathcal{J}\varphi = \{\ \alpha \in \mathcal{J}.\ \neg\ \alpha \leq \varphi\ \}$
have *finite* $?n\mathcal{J}\varphi$ **by** *auto*
hence $0 = (\sum\ \alpha \in\ ?n\mathcal{J}\varphi.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0))$
(**is** - = $?\Sigma 2$)
by (*induct* $?n\mathcal{J}\varphi$ *rule*: *finite-induct*, *auto*)
with $\dagger$ **have** $\ddagger$: $\mathcal{P}\ \varphi =\ ?\Sigma 1\ +\ ?\Sigma 2$ **by** *auto*
have $\forall \alpha \in\ ?\mathcal{J}\varphi \cap\ ?n\mathcal{J}\varphi.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0) = 0$ **by** *auto*
with $\ddagger$ **have** $\mathcal{P}\ \varphi = (\sum\ \alpha \in\ ?\mathcal{J}\varphi \cup\ ?n\mathcal{J}\varphi.\ \mathcal{P}\ \alpha * (\mathit{if}\ \alpha \leq \varphi\ \mathit{then}\ 1\ \mathit{else}\ 0))$
by (*simp add*: *sum.union-inter-neutral* [**where** $A=?\mathcal{J}\varphi$ **and** $B=?n\mathcal{J}\varphi$])
moreover have $\mathcal{J} =\ ?\mathcal{J}\varphi \cup\ ?n\mathcal{J}\varphi$ **by** *auto*
ultimately show *?thesis*
by *auto*
qed

lemma *dirac-measure-to-join-prime*:
assumes $\delta \in$ *dirac-measures*
shows $\bigsqcap\ \{\ \varphi\ .\ \delta\ \varphi = 1\ \} \in \mathcal{J}$
(**is** $?\alpha \in \mathcal{J}$)
proof −
have $\delta \in$ *probabilities*
using
‹$\delta \in$ *dirac-measures*›
probabilities-def
unfolding *dirac-measures-def*
by *blast*
interpret *finitely-additive-probability* δ
using ‹$\delta \in$ *probabilities*›
unfolding *probabilities-def'*
by *auto*
have $\forall\ \varphi \in \{\ \varphi\ .\ \delta\ \varphi = 1\ \}.\ \delta\ \varphi = 1$
(**is** $\forall\ \varphi \in\ ?A.\ \delta\ \varphi = 1$)
by *auto*
hence $\delta\ ?\alpha = 1$
using *finite-certainty Inf-def finite*
by *presburger*

```
  hence ?α ≠ ⊥
    using gaines-weatherson-antithesis
    by auto
  moreover
  {
    fix y z
    assume ?α ≤ y ⊔ z
    hence 1 ≤ δ (y ⊔ z)
      using ‹δ ?α = 1› monotonicity
      by fastforce
    hence δ (y ⊔ z) = 1
      by (metis
            probability-unity
            monotonicity
            sup.cobounded2
            sup-top-left
            order-class.eq-iff)
    moreover have δ y = 0 ⟹ δ z = 0 ⟹ δ (y ⊔ z) = 0
      by (metis
            add.right-neutral
            add-diff-cancel-left'
            diff-ge-0-iff-ge
            probability-non-negative
            sum-rule
            order-class.eq-iff)
    ultimately have δ y ≠ 0 ∨ δ z ≠ 0
      by linarith
    hence δ y = 1 ∨ δ z = 1
      using ‹δ ∈ dirac-measures›
      unfolding dirac-measures-def
      by auto
    hence y ∈ ?A ∨ z ∈ ?A
      by auto
    hence ?α ≤ y ∨ ?α ≤ z
      using Inf-lower by auto
  }
  ultimately show ?thesis
    unfolding join-primes-def join-prime-def
    by auto
qed

lemma dirac-to-join-prime-ident:
  assumes δ ∈ dirac-measures
  shows (λ φ. if ⨅ { φ . δ φ = 1 } ≤ φ then 1 else 0) = δ
proof
  have δ ∈ probabilities
    using
      ‹δ ∈ dirac-measures›
      probabilities-def
```

unfolding *dirac-measures-def*
    by *blast*
  interpret *finitely-additive-probability* δ
    using ‹$\delta \in$ *probabilities*›
    unfolding *probabilities-def'*
    by *auto*
  fix φ
  show $(if \bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq \varphi\ then\ 1\ else\ 0) = \delta\ \varphi$
  proof (*cases* $\delta\ \varphi = 1$)
    case *True*
    hence $\bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq \varphi$
      by (*fastforce simp add*: *Inf-lower*)
    hence $(if \bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq \varphi\ then\ 1\ else\ 0) = 1$
      by *auto*
    then show *?thesis*
      using ‹$\delta\ \varphi = 1$›
      by *simp*
  next
    have *join-prime* $(\bigsqcap \{ \varphi . \delta\ \varphi = 1 \})$
      using
        ‹$\delta \in$ *dirac-measures*›
        dirac-measure-to-join-prime
      unfolding *join-primes-def*
      by *blast*
    case *False*
    hence $\delta\ \varphi = 0$
      using ‹$\delta \in$ *dirac-measures*›
      unfolding *dirac-measures-def*
      by *auto*
    hence $\delta\ (-\ \varphi) = 1$
      using *complementation*
      by *auto*
    hence $\bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq -\ \varphi$
      by (*fastforce simp add*: *Inf-lower*)
    hence $\neg\ (\bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq \varphi)$
      using ‹*join-prime* $(\bigsqcap \{ \varphi . \delta\ \varphi = 1 \})$›
      unfolding *join-prime-def*
      by (*metis inf.boundedI inf-compl-bot le-bot*)
    hence $(if \bigsqcap \{ \varphi . \delta\ \varphi = 1 \} \leq \varphi\ then\ 1\ else\ 0) = 0$
      by *auto*
    then show *?thesis*
      using ‹$\delta\ \varphi = 0$›
      by *auto*
  qed
qed

lemma *join-prime-to-dirac-ident*:
  assumes $\alpha \in \mathcal{J}$
  shows $\bigsqcap\{ \varphi.\ (\lambda\ \varphi.\ if\ \alpha \leq \varphi\ then\ 1\ else\ 0)\ \varphi = (1 :: real)\} = \alpha$

```
  (is ?α = α)
proof (rule antisym)
  have α ∈ { φ. (λ φ. if α ≤ φ then 1 else 0) φ = 1 }
    by simp
  thus ?α ≤ α
    by (simp add: Inf-lower)
next
  {
    fix φ
    assume φ ∈ { φ. (λ φ. if α ≤ φ then 1 else 0) φ = (1 :: real) }
    hence (if α ≤ φ then 1 else 0) = (1 :: real)
      by fastforce
    hence α ≤ φ
      by (meson zero-neq-one)
  }
  hence ∀ φ ∈ { φ. (λ φ. if α ≤ φ then 1 else 0) φ = (1 :: real) } . α ≤ φ
    by blast
  thus α ≤ ?α
    using Inf-greatest by blast
qed

lemma dirac-join-prime-bij-betw:
  bij-betw (λ α φ. if α ≤ φ then 1 else 0 :: real) 𝒥 dirac-measures
  unfolding bij-betw-def
proof
  obtain to-δ where to-δ-def:
    to-δ = (λ α φ . if α ≤ φ then 1 else 0 :: real) by auto
  {
    fix α₁ α₂
    assume
      α₁ ∈ 𝒥
      α₂ ∈ 𝒥
      to-δ α₁ = to-δ α₂
    moreover from this have
      ⨅{ φ. (λ φ. if α₁ ≤ φ then 1 else 0) φ = (1 :: real) }
        = ⨅{ φ. (λ φ. if α₂ ≤ φ then 1 else 0) φ = (1 :: real) }
      unfolding to-δ-def
      by metis
    ultimately have α₁ = α₂
      using
        join-prime-to-dirac-ident [of α₁]
        join-prime-to-dirac-ident [of α₂]
      by presburger
  }
  hence inj-on to-δ 𝒥
    unfolding inj-on-def
    by blast
  thus inj-on (λ α φ. if α ≤ φ then 1 else 0 :: real) 𝒥
    unfolding to-δ-def
```

by *blast*
next
show $(\lambda\alpha\ \varphi.\ \textit{if}\ \ \alpha \leq \varphi\ \textit{then 1 else 0})$ ‘ $\mathcal{J} = \textit{dirac-measures}$
proof
{
fix α
assume $\alpha \in \mathcal{J}$
hence $(\lambda\varphi.\ \textit{if}\ \ \alpha \leq \varphi\ \textit{then 1 else 0}) \in \textit{dirac-measures}$
using *join-prime-to-dirac-measure* **by** *blast*
}
thus $(\lambda\alpha\ \varphi.\ \textit{if}\ \ \alpha \leq \varphi\ \textit{then 1 else 0})$ ‘ $\mathcal{J} \subseteq \textit{dirac-measures}$ **by** *blast*
next
{
fix δ
assume $\delta \in \textit{dirac-measures}$
let $?\alpha = \bigsqcap\ \{\ \varphi\ .\ \delta\ \varphi = \mathit{1}\ \}$
have $?\alpha \in \mathcal{J}$
using ‹$\delta \in \textit{dirac-measures}$› *dirac-measure-to-join-prime* **by** *blast*
moreover have $(\lambda\varphi.\ \textit{if}\ ?\alpha \leq \varphi\ \textit{then 1 else 0}) = \delta$
using ‹$\delta \in \textit{dirac-measures}$› *dirac-to-join-prime-ident* **by** *blast*
ultimately have $\delta \in (\lambda\alpha\ \varphi.\ \textit{if}\ \ \alpha \leq \varphi\ \textit{then 1 else 0})$ ‘ $\mathcal{J}$
using *image-iff* **by** *fastforce*
}
thus $\textit{dirac-measures} \subseteq (\lambda\alpha\ \varphi.\ \textit{if}\ \ \alpha \leq \varphi\ \textit{then 1 else 0})$ ‘ $\mathcal{J}$
using *subsetI*
by *blast*
qed
qed

lemma *dirac-join-prime-bij-betw-alt*:
bij-betw $(\lambda\ \delta.\ \bigsqcap\ \{\ \varphi\ .\ \delta\ \varphi = \mathit{1}\ \})$ *dirac-measures* $\mathcal{J}$
(**is** *bij-betw* $?to\text{-}\mathcal{J}$ - -)
unfolding *bij-betw-def*
proof
{
fix $\delta_1\ \delta_2$
assume
$\delta_1 \in \textit{dirac-measures}$
$\delta_2 \in \textit{dirac-measures}$
$?to\text{-}\mathcal{J}\ \delta_1 = ?to\text{-}\mathcal{J}\ \delta_2$
moreover from *this* **have**
$(\lambda\ \varphi.\ \textit{if}\ ?to\text{-}\mathcal{J}\ \delta_1 \leq \varphi\ \textit{then 1 else 0}) = \delta_1$
$(\lambda\ \varphi.\ \textit{if}\ ?to\text{-}\mathcal{J}\ \delta_2 \leq \varphi\ \textit{then 1 else 0}) = \delta_2$
using *dirac-to-join-prime-ident* **by** *blast+*
ultimately have $\delta_1 = \delta_2$
by *presburger*
}
thus *inj-on* $?to\text{-}\mathcal{J}$ *dirac-measures*
unfolding *inj-on-def*

```
      by auto
  next
    show ?to-𝒥 ` dirac-measures = 𝒥
    proof
      show (λδ. ⨅ {φ. δ φ = 1}) ` dirac-measures ⊆ 𝒥
        using dirac-measure-to-join-prime by blast
    next
      {
        fix α :: 'a
        assume α ∈ 𝒥
        hence (λφ. if α ≤ φ then 1 else 0 :: real) ∈ dirac-measures
          using join-prime-to-dirac-measure by blast
        moreover have ?to-𝒥 (λφ. if α ≤ φ then 1 else 0 :: real) = α
          by (simp add: ‹α ∈ 𝒥› join-prime-to-dirac-ident)
        ultimately have α ∈ ?to-𝒥 ` dirac-measures
          using image-iff by fastforce
      }
      thus 𝒥 ⊆ (λδ. ⨅ {φ. δ φ = 1}) ` dirac-measures
        using subsetI
        by blast
    qed
  qed

lemma special-dirac-collapse:
  (∀ 𝒫 ∈ probabilities. (∑φ←Φ. 𝒫 φ) + c ≤ (∑γ←Γ. 𝒫 γ))
    = (∀ 𝒫 ∈ dirac-measures. (∑φ←Φ. 𝒫 φ) + ⌈c⌉ ≤ (∑γ←Γ. 𝒫 γ))
proof
  assume ⋆: ∀ 𝒫 ∈ probabilities. (∑φ←Φ. 𝒫 φ) + c ≤ (∑γ←Γ. 𝒫 γ)
  {
    fix δ
    assume δ ∈ dirac-measures
    hence ∀ φ. δ φ = 1 ∨ δ φ = 0
      using dirac-measures-def by blast
    have A: (∑φ←Φ. δ φ) = ⌈∑φ←Φ. δ φ⌉
    proof (induct Φ)
      case Nil
      then show ?case using ‹∀ φ. δ φ = 1 ∨ δ φ = 0› by simp
    next
      case (Cons φ Φ)
      then show ?case
      proof (cases δ φ = 0)
        case True
        then show ?thesis
          using Cons.hyps by fastforce
      next
        case False
        hence δ φ = 1
          using ‹∀φ. δ φ = 1 ∨ δ φ = 0› by blast
        then show ?thesis
```

```
        by (simp,
            metis
              Cons.hyps
              add.commute
              ceiling-add-one
              of-int-1
              of-int-add)
    qed
  qed
  have B: (∑γ←Γ. δ γ) = ⌈∑γ←Γ. δ γ⌉
  proof (induct Γ)
    case Nil
    then show ?case using ‹∀ φ. δ φ = 1 ∨ δ φ = 0› by simp
  next
    case (Cons γ Γ)
    then show ?case
    proof (cases δ γ = 0)
      case True
      then show ?thesis
        using Cons.hyps by fastforce
    next
      case False
      hence δ γ = 1
        using ‹∀φ. δ φ = 1 ∨ δ φ = 0› by blast
      then show ?thesis
        by (simp,
            metis
              Cons.hyps
              add.commute
              ceiling-add-one
              of-int-1
              of-int-add)
    qed
  qed
  have δ ∈ probabilities
    using ‹δ ∈ dirac-measures› dirac-measures-subset by auto
  hence C: (∑φ←Φ. δ φ) + c ≤ (∑γ←Γ. δ γ)
    using ⋆
    by blast
  from A B C have ⌈(∑φ←Φ. δ φ)⌉ + c ≤ ⌈(∑γ←Γ. δ γ)⌉
    by simp
  hence ⌈(∑φ←Φ. δ φ)⌉ + ⌈c⌉ ≤ ⌈(∑γ←Γ. δ γ)⌉
    by linarith
  hence (∑φ←Φ. δ φ) + ⌈c⌉ ≤ (∑γ←Γ. δ γ)
    using A B C by simp
 }
 thus ∀ δ ∈ dirac-measures. (∑φ←Φ. δ φ) + ⌈c⌉ ≤ (∑γ←Γ. δ γ)
  by auto
next
```

assume $\star$: $\forall\ \delta \in$ *dirac-measures*. $(\sum\varphi\leftarrow\Phi.\ \delta\ \varphi) + \lceil c \rceil \leq (\sum\gamma\leftarrow\Gamma.\ \delta\ \gamma)$
let $?to\text{-}\delta = \lambda\ \alpha\ \varphi :: 'a.\ if\ \alpha \leq \varphi\ then\ 1 :: real\ else\ 0$
{
  fix $\mathcal{P}$
  assume $\mathcal{P} \in$ *probabilities*
  from *this* **interpret**
    finitely-additive-probability $\mathcal{P}$
    unfolding *probabilities-def'*
    by *auto*
  have *finite* $\mathcal{J}$ **by** *simp*
  {
    fix Φ :: *'a list*
    {
      fix A :: *'a set*
      assume *finite A*
      have $(\sum\varphi \leftarrow \Phi.\ (\sum\alpha \in A.\ \mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi))$
            $= (\sum\alpha \in A.\ \mathcal{P}\ \alpha * (\sum\varphi \leftarrow \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi))$
      proof (*induct* Φ)
        case *Nil*
        then show *?case* **by** *simp*
      next
        case (*Cons* φ' Φ)
        with ‹*finite A*› **show** *?case*
        proof (*induct A rule*: *finite-induct*)
          case *empty*
          then show *?case* **by** *simp*
        next
          case (*insert a A*)
          have
            $(\sum\varphi\leftarrow\varphi' \# \Phi.\ \sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi)$
             $= (\sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi')$
               $+ (\sum\varphi\leftarrow\Phi.\ \sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi)$
            by *simp*
          also have
            $\ldots = (\sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi')$
                $+ (\sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * (\sum\varphi\leftarrow\Phi.\ ?to\text{-}\delta\ \alpha\ \varphi))$
            using *insert.prems* **by** *linarith*
          also have
            $\ldots = (\sum\alpha\in insert\ a\ A.\ (\mathcal{P}\ \alpha * ?to\text{-}\delta\ \alpha\ \varphi')$
                $+ \mathcal{P}\ \alpha * (\sum\varphi\leftarrow\Phi.\ ?to\text{-}\delta\ \alpha\ \varphi))$
            by (*simp add*: *sum.distrib*)
          also have
            $\ldots = (\sum\alpha\in insert\ a\ A.\ \mathcal{P}\ \alpha * (\sum\varphi\leftarrow\varphi' \# \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi))$
            by (*simp add*: *distrib-left*)
          finally show *?case* **by** *simp*
        qed
      qed
    }
    note $\dagger$ = *this*

have $(\sum\varphi{\leftarrow}\Phi.\ \mathcal{P}\ \varphi) = (\sum\varphi{\leftarrow}\Phi.\ (\sum\alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * \textit{?to-}\delta\ \alpha\ \varphi))$
by (*induct* Φ,
auto,
metis join-prime-decomposition [*OF* ‹$\mathcal{P} \in$ *probabilities*›])
hence $(\sum\varphi{\leftarrow}\Phi.\ \mathcal{P}\ \varphi) = (\sum\ \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum\varphi \leftarrow \Phi.\ \textit{?to-}\delta\ \alpha\ \varphi))$
unfolding † [*OF* ‹*finite* $\mathcal{J}$›] **by** *auto*
}
hence X: $(\sum\varphi{\leftarrow}\Phi.\ \mathcal{P}\ \varphi) = (\sum\alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum\varphi \leftarrow \Phi.\ \textit{?to-}\delta\ \alpha\ \varphi))$
and Y: $(\sum\gamma{\leftarrow}\Gamma.\ \mathcal{P}\ \gamma) = (\sum\alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum\gamma \leftarrow \Gamma.\ \textit{?to-}\delta\ \alpha\ \gamma))$
by *auto*
{
fix A :: *'a set*
assume $A \subseteq \mathcal{J}$
hence *finite* A
by *simp*
hence $(\sum\alpha \in A.\ \mathcal{P}\ \alpha * ((\sum\varphi \leftarrow \Phi.\ \textit{?to-}\delta\ \alpha\ \varphi) + \lceil c\rceil))$
$\leq (\sum\alpha \in A.\ \mathcal{P}\ \alpha * (\sum\gamma \leftarrow \Gamma.\ \textit{?to-}\delta\ \alpha\ \gamma))$
using ‹$A \subseteq \mathcal{J}$›
proof (*induct* A *rule*: *finite-induct*)
case *empty*
then show *?case* **by** *auto*
next
case (*insert* α' A)
hence $\alpha' \in \mathcal{J}$
by *blast*
hence $\textit{?to-}\delta\ \alpha' \in$ *dirac-measures*
using *dirac-join-prime-bij-betw*
unfolding *bij-betw-def*
by *blast*
hence $(\sum\varphi{\leftarrow}\Phi.\ \textit{?to-}\delta\ \alpha'\ \varphi) + \lceil c\rceil \leq (\sum\gamma{\leftarrow}\Gamma.\ \textit{?to-}\delta\ \alpha'\ \gamma)$
using $\star$ **by** *blast*
moreover have $0 \leq \mathcal{P}\ \alpha'$
by (*simp add*: *probability-non-negative*)
ultimately have
$\mathcal{P}\ \alpha' * ((\sum\varphi{\leftarrow}\Phi.\ \textit{?to-}\delta\ \alpha'\ \varphi) + \lceil c\rceil) \leq \mathcal{P}\ \alpha' * (\sum\gamma{\leftarrow}\Gamma.\ \textit{?to-}\delta\ \alpha'\ \gamma)$
using *mult-left-mono* **by** *blast*
moreover have
$(\sum\alpha \in A.\ \mathcal{P}\ \alpha * ((\sum\varphi \leftarrow \Phi.\ \textit{?to-}\delta\ \alpha\ \varphi) + \lceil c\rceil))$
$\leq (\sum\alpha \in A.\ \mathcal{P}\ \alpha * (\sum\gamma \leftarrow \Gamma.\ \textit{?to-}\delta\ \alpha\ \gamma))$
using *insert.hyps insert.prems* **by** *blast*
ultimately show *?case*
using *insert.hyps*(*2*) **by** *auto*
qed
}
hence A:
$(\sum\alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * ((\sum\varphi \leftarrow \Phi.\ \textit{?to-}\delta\ \alpha\ \varphi) + \lceil c\rceil))$
$\leq (\sum\alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum\gamma \leftarrow \Gamma.\ \textit{?to-}\delta\ \alpha\ \gamma))$
by *blast*
{

fix A :: $'a\ set$
assume *finite* A
hence
$(\sum \alpha \in A.\ \mathcal{P}\ \alpha * ((\sum \varphi \leftarrow \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi) + \lceil c \rceil))$
$= (\sum \alpha \in A.\ \mathcal{P}\ \alpha * (\sum \varphi \leftarrow \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi)) + \lceil c \rceil * (\sum \alpha \in A.\ \mathcal{P}\ \alpha)$
by (*induct* A *rule*: *finite-induct*, *simp*, *simp add*: *distrib-left*)
}
with A ‹*finite* $\mathcal{J}$› **have** B:
$(\sum \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum \varphi \leftarrow \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi)) + \lceil c \rceil * (\sum \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha)$
$\leq (\sum \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum \gamma \leftarrow \Gamma.\ ?to\text{-}\delta\ \alpha\ \gamma))$
by *auto*
have $(\sum \alpha{\in}\mathcal{J}.\ \mathcal{P}\ \alpha) = 1$
using
join-prime-decomposition [*OF* ‹$\mathcal{P} \in$ *probabilities*›, **where** $\varphi{=}\top$]
top-greatest
unfolding *probability-unity*
by *auto*
hence $(\sum \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum \varphi \leftarrow \Phi.\ ?to\text{-}\delta\ \alpha\ \varphi)) + \lceil c \rceil$
$\leq (\sum \alpha \in \mathcal{J}.\ \mathcal{P}\ \alpha * (\sum \gamma \leftarrow \Gamma.\ ?to\text{-}\delta\ \alpha\ \gamma))$
using B **by** *auto*
hence $(\sum \varphi \leftarrow \Phi.\ \mathcal{P}\ \varphi) + c \leq (\sum \gamma \leftarrow \Gamma.\ \mathcal{P}\ \gamma)$
using X Y
by *linarith*
}
thus $\forall\ \mathcal{P} \in$ *probabilities*. $(\sum \varphi{\leftarrow}\Phi.\ \mathcal{P}\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ \mathcal{P}\ \gamma)$ **by** *auto*
qed

end

end

2.5 Completeness For Probability Inequalities

theory *Probability-Logic-Inequality-Completeness*
imports
Probability-Logic
begin

sledgehammer-params [*smt-proofs* = *false*]

2.5.1 Segmented Deduction

definition *uncurry* :: $('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow 'a \times 'b \Rightarrow 'c$
where *uncurry-def* [*simp*]: *uncurry* $f = (\lambda\ (x,\ y).\ f\ x\ y)$

primrec (**in** *classical-logic*)
segmented-deduction :: $'a\ list \Rightarrow 'a\ list \Rightarrow bool$ (- \$⊢ - [*60*,*100*] *60*)
where
Γ \$⊢ [] = *True*

| Γ \$$\vdash$ (φ # Φ) =
    ($\exists$ Ψ. *mset* (*map snd* Ψ) $\subseteq$# *mset* Γ
        $\wedge$ *map* (*uncurry* ($\sqcup$)) Ψ :$\vdash$ φ
        $\wedge$ *map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ) \$$\vdash$ Φ)

definition (**in** *implication-logic*)
  stronger-theory-relation :: *'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *bool* (**infix** $\preceq$ *100*)
  where
    $\Sigma \preceq \Gamma$ =
      ($\exists$ Φ. *map snd* Φ = Σ
        $\wedge$ *mset* (*map fst* Φ) $\subseteq$# *mset* Γ
        $\wedge$ ($\forall$ (γ,σ) $\in$ *set* Φ. $\vdash$ $\gamma \rightarrow \sigma$))

abbreviation (**in** *implication-logic*)
  stronger-theory-relation-op :: *'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *bool* (**infix** $\succeq$ *100*)
  where
    $\Gamma \succeq \Sigma \equiv \Sigma \preceq \Gamma$

lemma (**in** *implication-logic*) *msub-stronger-theory-intro*:
  assumes *mset* Σ $\subseteq$# *mset* Γ
  shows $\Sigma \preceq \Gamma$
proof −
  let *?*$\Delta\Sigma$ = *map* (λ *x*. (*x*,*x*)) Σ
  have *map snd ?*$\Delta\Sigma$ = Σ
    by (*induct* Σ, *simp*, *simp*)
  moreover have *map fst ?*$\Delta\Sigma$ = Σ
    by (*induct* Σ, *simp*, *simp*)
  hence *mset* (*map fst ?*$\Delta\Sigma$) $\subseteq$# *mset* Γ
    using *assms* **by** *simp*
  moreover have $\forall$ (γ,σ) $\in$ *set ?*$\Delta\Sigma$. $\vdash$ $\gamma \rightarrow \sigma$
    by (*induct* Σ, *simp*, *simp*,
        metis list-implication.simps(*1*) *list-implication-axiom-k*)
  ultimately show *?thesis* **using** *stronger-theory-relation-def* **by** (*simp*, *blast*)
qed

lemma (**in** *implication-logic*) *stronger-theory-reflexive* [*simp*]: $\Gamma \preceq \Gamma$
  using *msub-stronger-theory-intro* **by** *auto*

lemma (**in** *implication-logic*) *weakest-theory* [*simp*]: [] $\preceq$ Γ
  using *msub-stronger-theory-intro* **by** *auto*

lemma (**in** *implication-logic*) *stronger-theory-empty-list-intro* [*simp*]:
  assumes $\Gamma \preceq$ []
  shows Γ = []
  using *assms stronger-theory-relation-def* **by** *simp*

lemma (**in** *implication-logic*) *stronger-theory-right-permutation*:
  assumes $\Gamma \rightleftharpoons \Delta$
    and $\Sigma \preceq \Gamma$

shows $\Sigma \preceq \Delta$
proof $-$
from *assms(1)* **have** *mset* Γ = *mset* Δ
by (*simp add*: *mset-eq-perm*)
thus *?thesis*
using *assms(2)* *stronger-theory-relation-def*
by *fastforce*
qed

lemma (**in** *implication-logic*) *stronger-theory-left-permutation*:
assumes $\Sigma \rightleftharpoons \Delta$
and $\Sigma \preceq \Gamma$
shows $\Delta \preceq \Gamma$
proof $-$
have $\forall\ \Sigma\ \Gamma.\ \Sigma \rightleftharpoons \Delta \longrightarrow \Sigma \preceq \Gamma \longrightarrow \Delta \preceq \Gamma$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix $\Sigma\ \Gamma$
assume $\Sigma \rightleftharpoons (\delta \# \Delta)$ $\Sigma \preceq \Gamma$
from *this* **obtain** Φ **where** Φ:
map snd $\Phi = \Sigma$
mset (*map fst* Φ) $\subseteq\#$ *mset* Γ
$\forall\ (\gamma,\delta) \in$ *set* $\Phi.\ \vdash \gamma \rightarrow \delta$
using *stronger-theory-relation-def* **by** *fastforce*
with $\langle\Sigma \rightleftharpoons (\delta \# \Delta)\rangle$ **have** $\delta \in\#$ *mset* (*map snd* Φ)
by (*simp add*: *perm-set-eq*)
from *this* **obtain** γ **where** γ: $(\gamma, \delta) \in\#$ *mset* Φ
by (*induct* Φ, *fastforce+*)
let $?\Phi_0$ = *remove1* (γ, δ) Φ
let $?\Sigma_0$ = *map snd* $?\Phi_0$
from γ $\Phi(2)$ **have** *mset* (*map fst* $?\Phi_0$) $\subseteq\#$ *mset* (*remove1* γ Γ)
by (*metis ex-mset*
list-subtract-monotonic
list-subtract-mset-homomorphism
mset-remove1
remove1-pairs-list-projections-fst)
moreover have *mset* $?\Phi_0 \subseteq\#$ *mset* Φ **by** *simp*
with $\Phi(3)$ **have** $\forall\ (\gamma,\delta) \in$ *set* $?\Phi_0.\ \vdash \gamma \rightarrow \delta$ **by** *fastforce*
ultimately have $?\Sigma_0 \preceq$ *remove1* γ Γ
unfolding *stronger-theory-relation-def* **by** *blast*
moreover have $\Delta \rightleftharpoons$ (*remove1* δ Σ) **using** $\langle\Sigma \rightleftharpoons (\delta \# \Delta)\rangle$
by (*metis perm-remove-perm perm-sym remove-hd*)
moreover from γ $\Phi(1)$ **have** *mset* $?\Sigma_0$ = *mset* (*remove1* δ Σ)
using *remove1-pairs-list-projections-snd*
by *fastforce*

```
      hence ?Σ₀ ⇌ remove1 δ Σ
        using mset-eq-perm by blast
      ultimately have Δ ≼ remove1 γ Γ using Cons
        by (meson perm.trans perm-sym)
      from this obtain Ψ₀ where Ψ₀:
        map snd Ψ₀ = Δ
        mset (map fst Ψ₀) ⊆# mset (remove1 γ Γ)
        ∀ (γ,δ) ∈ set Ψ₀. ⊢ γ → δ
        using stronger-theory-relation-def by fastforce
      let ?Ψ = (γ, δ) # Ψ₀
      have map snd ?Ψ = (δ # Δ)
        by (simp add: Ψ₀(1))
      moreover have mset (map fst ?Ψ) ⊆# mset (γ # (remove1 γ Γ))
        using Ψ₀(2) by auto
      moreover from γ Φ(3) Ψ₀(3) have ∀ (γ,σ) ∈ set ?Ψ. ⊢ γ → σ by auto
      ultimately have (δ # Δ) ≼ (γ # (remove1 γ Γ))
        unfolding stronger-theory-relation-def by metis
      moreover from γ Φ(2) have γ ∈# mset Γ
        using mset-subset-eqD by fastforce
      hence (γ # (remove1 γ Γ)) ⇌ Γ
        by (simp add: perm-remove perm-sym)
      ultimately have (δ # Δ) ≼ Γ
        using stronger-theory-right-permutation by blast
    }
    then show ?case by blast
  qed
  with assms show ?thesis by blast
qed

lemma (in implication-logic) stronger-theory-transitive:
  assumes Σ ≼ Δ and Δ ≼ Γ
    shows Σ ≼ Γ
proof −
  have ∀ Δ Γ. Σ ≼ Δ ⟶ Δ ≼ Γ ⟶ Σ ≼ Γ
  proof (induct Σ)
    case Nil
    then show ?case using stronger-theory-relation-def by simp
  next
    case (Cons σ Σ)
    {
      fix Δ Γ
      assume (σ # Σ) ≼ Δ Δ ≼ Γ
      from this obtain Φ where Φ:
        map snd Φ = σ # Σ
        mset (map fst Φ) ⊆# mset Δ
        ∀ (δ,σ) ∈ set Φ. ⊢ δ → σ
        using stronger-theory-relation-def by (simp, metis)
      let ?δ = fst (hd Φ)
      from Φ(1) have Φ ≠ [] by (induct Φ, simp+)
```

hence *?δ* $\in\#$ *mset* (*map fst* Φ) **by** (*induct* Φ, *simp*+)
with $\Phi(2)$ **have** *?δ* $\in\#$ *mset* Δ **by** (*meson mset-subset-eqD*)
hence *mset* (*map fst* (*remove1* (*hd* Φ) Φ)) $\subseteq\#$ *mset* (*remove1 ?δ* Δ)
  using ‹$\Phi \neq []$› $\Phi(2)$
  by (*simp*,
    metis
      diff-single-eq-union
      hd-in-set
      image-mset-add-mset
      insert-subset-eq-iff
      set-mset-mset)
moreover have *remove1* (*hd* Φ) Φ = *tl* Φ
  using ‹$\Phi \neq []$›
  by (*induct* Φ, *simp*+)
moreover from $\Phi(1)$ **have** *map snd* (*tl* Φ) = Σ
  by (*simp add*: *map-tl*)
moreover from $\Phi(3)$ **have** $\forall$ (δ,σ) $\in$ *set* (*tl* Φ). $\vdash \delta \rightarrow \sigma$
  by (*simp add*: ‹$\Phi \neq []$› *list.set-sel*(*2*))
ultimately have $\Sigma \preceq$ *remove1 ?δ* Δ
  using *stronger-theory-relation-def* **by** *auto*
from ‹*?δ* $\in\#$ *mset* Δ› **have** *?δ* # (*remove1 ?δ* Δ) $\rightleftharpoons \Delta$
  by (*simp add*: *perm-remove perm-sym*)
with ‹$\Delta \preceq \Gamma$› **have** (*?δ* # (*remove1 ?δ* Δ)) $\preceq \Gamma$
  using *stronger-theory-left-permutation perm-sym* **by** *blast*
from *this* **obtain** Ψ **where** Ψ:
  map snd Ψ = (*?δ* # (*remove1 ?δ* Δ))
  mset (*map fst* Ψ) $\subseteq\#$ *mset* Γ
  $\forall$ (γ,δ) $\in$ *set* Ψ. $\vdash \gamma \rightarrow \delta$
  using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let *?γ* = *fst* (*hd* Ψ)
from $\Psi(1)$ **have** $\Psi \neq []$ **by** (*induct* Ψ, *simp*+)
hence *?γ* $\in\#$ *mset* (*map fst* Ψ) **by** (*induct* Ψ, *simp*+)
with $\Psi(2)$ **have** *?γ* $\in\#$ *mset* Γ **by** (*meson mset-subset-eqD*)
hence *mset* (*map fst* (*remove1* (*hd* Ψ) Ψ)) $\subseteq\#$ *mset* (*remove1 ?γ* Γ)
  using ‹$\Psi \neq []$› $\Psi(2)$
  by (*simp*,
    metis
      diff-single-eq-union
      hd-in-set
      image-mset-add-mset
      insert-subset-eq-iff
      set-mset-mset)
moreover from ‹$\Psi \neq []$› **have** *remove1* (*hd* Ψ) Ψ = *tl* Ψ
  by (*induct* Ψ, *simp*+)
moreover from $\Psi(1)$ **have** *map snd* (*tl* Ψ) = (*remove1 ?δ* Δ)
  by (*simp add*: *map-tl*)
moreover from $\Psi(3)$ **have** $\forall$ (γ,δ) $\in$ *set* (*tl* Ψ). $\vdash \gamma \rightarrow \delta$
  by (*simp add*: ‹$\Psi \neq []$› *list.set-sel*(*2*))
ultimately have *remove1 ?δ* $\Delta \preceq$ *remove1 ?γ* Γ

using *stronger-theory-relation-def* **by** *auto*
with ‹$\Sigma \preceq$ *remove1 ?δ* Δ› *Cons.hyps* **have** $\Sigma \preceq$ *remove1 ?γ* Γ
by *blast*
from *this* **obtain** Ω_0 **where** Ω_0:
map snd $\Omega_0 = \Sigma$
mset (*map fst* Ω_0) $\subseteq\#$ *mset* (*remove1 ?γ* Γ)
$\forall$ $(\gamma,\sigma) \in$ *set* Ω_0. $\vdash \gamma \to \sigma$
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let ?Ω = (?γ, σ) # Ω_0
from $\Omega_0(1)$ **have** *map snd ?Ω* = σ # Σ **by** *simp*
moreover from $\Omega_0(2)$ **have** *mset* (*map fst ?Ω*) $\subseteq\#$ *mset* (?γ # (*remove1* ?γ Γ))
by *simp*
moreover from Φ(*1*) Ψ(*1*) **have** σ = *snd* (*hd* Φ) ?δ = *snd* (*hd* Ψ) **by** *fastforce+*
with Φ(*3*) Ψ(*3*) ‹Φ ≠ []› ‹Ψ ≠ []› *hd-in-set* **have** ⊢ ?δ → σ ⊢ ?γ → ?δ
by *fastforce+*
hence ⊢ ?γ → σ **using** *modus-ponens hypothetical-syllogism* **by** *blast*
with $\Omega_0(3)$ **have** ∀ (γ,σ) ∈ *set* ?Ω. ⊢ γ → σ
by *auto*
ultimately have (σ # Σ) ⪯ (?γ # (*remove1* ?γ Γ))
unfolding *stronger-theory-relation-def*
by *metis*
moreover from ‹?γ ∈# *mset* Γ› **have** (?γ # (*remove1* ?γ Γ)) ⇌ Γ
by (*simp add*: *perm-remove perm-sym*)
ultimately have (σ # Σ) ⪯ Γ
using *stronger-theory-right-permutation*
by *blast*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *implication-logic*) *stronger-theory-witness*:
assumes σ ∈ *set* Σ
shows Σ ⪯ Γ = (∃ γ ∈ *set* Γ. ⊢ γ → σ ∧ (*remove1* σ Σ) ⪯ (*remove1* γ Γ))
proof (*rule iffI*)
assume Σ ⪯ Γ
from *this* **obtain** Φ **where** Φ:
map snd Φ = Σ
mset (*map fst* Φ) ⊆# *mset* Γ
∀ (γ,σ) ∈ *set* Φ. ⊢ γ → σ
unfolding *stronger-theory-relation-def* **by** *blast*
from *assms* Φ(*1*) **obtain** γ **where** γ: (γ, σ) ∈# *mset* Φ
by (*induct* Φ, *fastforce+*)
hence γ ∈# *mset* (*map fst* Φ) **by** *force*
hence γ ∈# *mset* Γ **using** Φ(*2*)
by (*meson mset-subset-eqD*)

```
  moreover
  let ?Φ₀ = remove1 (γ, σ) Φ
  let ?Σ₀ = map snd ?Φ₀
  from γ Φ(2) have mset (map fst ?Φ₀) ⊆# mset (remove1 γ Γ)
    by (metis ex-mset
              list-subtract-monotonic
              list-subtract-mset-homomorphism
              remove1-pairs-list-projections-fst
              mset-remove1)
  moreover have mset ?Φ₀ ⊆# mset Φ by simp
  with Φ(3) have ∀ (γ,σ) ∈ set ?Φ₀. ⊢ γ → σ by fastforce
  ultimately have ?Σ₀ ≼ remove1 γ Γ
    unfolding stronger-theory-relation-def by blast
  moreover from γ Φ(1) have mset ?Σ₀ = mset (remove1 σ Σ)
    using remove1-pairs-list-projections-snd
    by fastforce
  hence ?Σ₀ ⇌ remove1 σ Σ
    using mset-eq-perm by blast
  ultimately have remove1 σ Σ ≼ remove1 γ Γ
    using stronger-theory-left-permutation by auto
  moreover from γ Φ(3) have ⊢ γ → σ by (simp, fast)
  moreover from γ Φ(2) have γ ∈# mset Γ
    using mset-subset-eqD by fastforce
  ultimately show ∃ γ ∈ set Γ. ⊢ γ → σ ∧ (remove1 σ Σ) ≼ (remove1 γ Γ) by
auto
next
  assume ∃ γ ∈ set Γ. ⊢ γ → σ ∧ (remove1 σ Σ) ≼ (remove1 γ Γ)
  from this obtain Φ γ where γ: γ ∈ set Γ ⊢ γ → σ
                     and Φ: map snd Φ = (remove1 σ Σ)
                            mset (map fst Φ) ⊆# mset (remove1 γ Γ)
                            ∀ (γ,σ) ∈ set Φ. ⊢ γ → σ
    unfolding stronger-theory-relation-def by blast
  let ?Φ = (γ, σ) # Φ
  from Φ(1) have map snd ?Φ = σ # (remove1 σ Σ) by simp
  moreover from Φ(2) γ(1) have mset (map fst ?Φ) ⊆# mset Γ
    by (simp add: insert-subset-eq-iff)
  moreover from Φ(3) γ(2) have ∀ (γ,σ) ∈ set ?Φ. ⊢ γ → σ
    by auto
  ultimately have (σ # (remove1 σ Σ)) ≼ Γ
    unfolding stronger-theory-relation-def by metis
  moreover from assms have σ # (remove1 σ Σ) ⇌ Σ
    by (simp add: perm-remove perm-sym)
  ultimately show Σ ≼ Γ
    using stronger-theory-left-permutation by blast
qed

lemma (in implication-logic) stronger-theory-cons-witness:
  (σ # Σ) ≼ Γ = (∃ γ ∈ set Γ. ⊢ γ → σ ∧ Σ ≼ (remove1 γ Γ))
proof −
```

have $\sigma \in\#$ *mset* $(\sigma \# \Sigma)$ **by** *simp*
hence $(\sigma \# \Sigma) \preceq \Gamma = (\exists\ \gamma \in$ *set* $\Gamma. \vdash \gamma \to \sigma \wedge ($*remove1* σ $(\sigma \# \Sigma)) \preceq ($*remove1* γ $\Gamma))$
by (*meson list.set-intros*(*1*) *stronger-theory-witness*)
thus *?thesis* **by** *simp*
qed

lemma (**in** *implication-logic*) *stronger-theory-left-cons*:
assumes $(\sigma \# \Sigma) \preceq \Gamma$
shows $\Sigma \preceq \Gamma$
proof −
from *assms* **obtain** Φ **where** Φ:
map snd $\Phi = \sigma \# \Sigma$
mset (*map fst* Φ) $\subseteq\#$ *mset* Γ
$\forall\ (\delta,\sigma) \in$ *set* $\Phi. \vdash \delta \to \sigma$
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let $?\Phi' =$ *remove1* (*hd* Φ) Φ
from Φ(*1*) **have** *map snd* $?\Phi' = \Sigma$ **by** (*induct* Φ, *simp+*)
moreover from Φ(*2*) **have** *mset* (*map fst* $?\Phi'$) $\subseteq\#$ *mset* Γ
by (*metis diff-subset-eq-self*
list-subtract.simps(*1*)
list-subtract.simps(*2*)
list-subtract-mset-homomorphism
map-monotonic
subset-mset.dual-order.trans)
moreover from Φ(*3*) **have** $\forall\ (\delta,\sigma) \in$ *set* $?\Phi'. \vdash \delta \to \sigma$ **by** *fastforce*
ultimately show *?thesis* **unfolding** *stronger-theory-relation-def* **by** *blast*
qed

lemma (**in** *implication-logic*) *stronger-theory-right-cons*:
assumes $\Sigma \preceq \Gamma$
shows $\Sigma \preceq (\gamma \# \Gamma)$
proof −
from *assms* **obtain** Φ **where** Φ:
map snd $\Phi = \Sigma$
mset (*map fst* Φ) $\subseteq\#$ *mset* Γ
$\forall (\gamma, \sigma) \in$ *set* $\Phi. \vdash \gamma \to \sigma$
unfolding *stronger-theory-relation-def*
by *auto*
hence *mset* (*map fst* Φ) $\subseteq\#$ *mset* $(\gamma \# \Gamma)$
by (*metis Diff-eq-empty-iff-mset*
list-subtract.simps(*2*)
list-subtract-mset-homomorphism
mset-zero-iff remove1.simps(*1*))
with Φ(*1*) Φ(*3*) **show** *?thesis*
unfolding *stronger-theory-relation-def*
by *auto*
qed

lemma (**in** *implication-logic*) *stronger-theory-left-right-cons*:
  assumes $\vdash \gamma \to \sigma$
    and $\Sigma \preceq \Gamma$
  shows $(\sigma \# \Sigma) \preceq (\gamma \# \Gamma)$
proof −
  from *assms*(*2*) **obtain** Φ **where** Φ:
    map snd $\Phi = \Sigma$
    mset (*map fst* Φ) $\subseteq\#$ *mset* Γ
    $\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma$
    unfolding *stronger-theory-relation-def*
    by *auto*
  let $?\Phi = (\gamma, \sigma) \# \Phi$
  from *assms*(*1*) Φ **have**
    map snd $?\Phi = \sigma \# \Sigma$
    mset (*map fst* $?\Phi$) $\subseteq\#$ *mset* $(\gamma \# \Gamma)$
    $\forall (\gamma, \sigma) \in set\ ?\Phi.\ \vdash \gamma \to \sigma$
    by *fastforce+*
  thus *?thesis*
    unfolding *stronger-theory-relation-def*
    by *metis*
qed

lemma (**in** *implication-logic*) *stronger-theory-relation-alt-def*:
  $\Sigma \preceq \Gamma = (\exists \Phi.\ mset\ (map\ snd\ \Phi) = mset\ \Sigma\ \wedge$
      $mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ \Gamma\ \wedge$
      $(\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma))$
proof −
  have $\forall\ \Sigma\ .\ \Sigma \preceq \Gamma = (\exists \Phi.\ mset\ (map\ snd\ \Phi) = mset\ \Sigma\ \wedge$
      $mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ \Gamma\ \wedge$
      $(\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma))$
  proof (*induct* Γ)
    case *Nil*
    then show *?case*
      using *stronger-theory-empty-list-intro*
        stronger-theory-reflexive
      by (*simp*, *blast*)
  next
    case (*Cons* γ Γ)
    {
      fix Σ
      have $\Sigma \preceq (\gamma \# \Gamma) = (\exists \Phi.\ mset\ (map\ snd\ \Phi) = mset\ \Sigma\ \wedge$
          $mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ (\gamma \# \Gamma)\ \wedge$
          $(\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma))$
      proof (*rule iffI*)
        assume $\Sigma \preceq (\gamma \# \Gamma)$
        thus $\exists \Phi.\ mset\ (map\ snd\ \Phi) = mset\ \Sigma\ \wedge$
          $mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ (\gamma \# \Gamma)\ \wedge$
          $(\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma)$
          unfolding *stronger-theory-relation-def*

by *metis*
next
assume $\exists\Phi.\ \mathit{mset}\ (\mathit{map}\ \mathit{snd}\ \Phi) = \mathit{mset}\ \Sigma \land$
$\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ \Phi) \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma) \land$
$(\forall(\gamma, \sigma){\in}\mathit{set}\ \Phi.\ \vdash \gamma \to \sigma)$
from *this* **obtain** Φ **where** Φ:
$\mathit{mset}\ (\mathit{map}\ \mathit{snd}\ \Phi) = \mathit{mset}\ \Sigma$
$\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ \Phi) \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma)$
$\forall(\gamma, \sigma){\in}\mathit{set}\ \Phi.\ \vdash \gamma \to \sigma$
by *metis*
show $\Sigma \preceq (\gamma\ \#\ \Gamma)$
proof $(\mathit{cases}\ \exists\ \sigma.\ (\gamma, \sigma) \in \mathit{set}\ \Phi)$
assume $\exists\ \sigma.\ (\gamma, \sigma) \in \mathit{set}\ \Phi$
from *this* **obtain** σ **where** σ: $(\gamma, \sigma) \in \mathit{set}\ \Phi$ **by** *auto*
let $?\Phi = \mathit{remove1}\ (\gamma, \sigma)\ \Phi$
from σ **have** $\mathit{mset}\ (\mathit{map}\ \mathit{snd}\ ?\Phi) = \mathit{mset}\ (\mathit{remove1}\ \sigma\ \Sigma)$
using $\Phi(1)$ *remove1-pairs-list-projections-snd* **by** *force+*
moreover
from σ **have** $\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ ?\Phi) = \mathit{mset}\ (\mathit{remove1}\ \gamma\ (\mathit{map}\ \mathit{fst}\ \Phi))$
using $\Phi(1)$ *remove1-pairs-list-projections-fst* **by** *force+*
with $\Phi(2)$ **have** $\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ ?\Phi) \subseteq\#\ \mathit{mset}\ \Gamma$
by (*simp add*: *subset-eq-diff-conv*)
moreover from $\Phi(3)$ **have** $\forall(\gamma, \sigma){\in}\mathit{set}\ ?\Phi.\ \vdash \gamma \to \sigma$
by *fastforce*
ultimately have $\mathit{remove1}\ \sigma\ \Sigma \preceq \Gamma$ **using** *Cons* **by** *blast*
from *this* **obtain** Ψ **where** Ψ:
$\mathit{map}\ \mathit{snd}\ \Psi = \mathit{remove1}\ \sigma\ \Sigma$
$\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ \Psi) \subseteq\#\ \mathit{mset}\ \Gamma$
$\forall(\gamma, \sigma){\in}\mathit{set}\ \Psi.\ \vdash \gamma \to \sigma$
unfolding *stronger-theory-relation-def*
by *blast*
let $?\Psi = (\gamma, \sigma)\ \#\ \Psi$
from Ψ **have** $\mathit{map}\ \mathit{snd}\ ?\Psi = \sigma\ \#\ (\mathit{remove1}\ \sigma\ \Sigma)$
$\mathit{mset}\ (\mathit{map}\ \mathit{fst}\ ?\Psi) \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma)$
by *simp+*
moreover from $\Phi(3)$ σ **have** $\vdash \gamma \to \sigma$ **by** *auto*
with $\Psi(3)$ **have** $\forall(\gamma, \sigma){\in}\mathit{set}\ ?\Psi.\ \vdash \gamma \to \sigma$ **by** *auto*
ultimately have $(\sigma\ \#\ (\mathit{remove1}\ \sigma\ \Sigma)) \preceq (\gamma\ \#\ \Gamma)$
unfolding *stronger-theory-relation-def*
by *metis*
moreover
have $\sigma \in \mathit{set}\ \Sigma$
by (*metis* $\Phi(1)$ σ *set-mset-mset set-zip-rightD zip-map-fst-snd*)
hence $\Sigma \rightleftharpoons \sigma\ \#\ (\mathit{remove1}\ \sigma\ \Sigma)$
by (*simp add*: *perm-remove*)
hence $\Sigma \preceq (\sigma\ \#\ (\mathit{remove1}\ \sigma\ \Sigma))$
using *stronger-theory-reflexive*
stronger-theory-right-permutation
by *blast*

ultimately show *?thesis*
  using *stronger-theory-transitive*
  by *blast*
next
  assume $\nexists \sigma.\ (\gamma, \sigma) \in$ *set* Φ
  hence $\gamma \notin$ *set* (*map fst* Φ) **by** *fastforce*
  with $\Phi(2)$ **have** *mset* (*map fst* Φ) $\subseteq\#$ *mset* Γ
    by (*metis diff-single-trivial*
      in-multiset-in-set
      insert-DiffM2
      mset-remove1
      remove-hd
      subset-eq-diff-conv)
  hence $\Sigma \preceq \Gamma$
    using *Cons* $\Phi(1)$ $\Phi(3)$
    by *blast*
  thus *?thesis*
    using *stronger-theory-right-cons*
    by *auto*
qed
qed
}
then show *?case* **by** *auto*
qed
thus *?thesis* **by** *auto*
qed

lemma (**in** *implication-logic*) *stronger-theory-deduction-monotonic*:
  assumes $\Sigma \preceq \Gamma$
    and $\Sigma :\vdash \varphi$
  shows $\Gamma :\vdash \varphi$
using *assms*
proof −
  have $\forall\ \varphi.\ \Sigma \preceq \Gamma \longrightarrow \Sigma :\vdash \varphi \longrightarrow \Gamma :\vdash \varphi$
  proof (*induct* Σ)
    case *Nil*
    then show *?case*
      by (*simp add*: *list-deduction-weaken*)
  next
    case (*Cons* σ Σ)
    {
      fix φ
      assume $(\sigma\ \#\ \Sigma) \preceq \Gamma$ $(\sigma\ \#\ \Sigma) :\vdash \varphi$
      hence $\Sigma :\vdash \sigma \rightarrow \varphi$ $\Sigma \preceq \Gamma$
        using *list-deduction-theorem*
          stronger-theory-left-cons
        by (*blast*, *metis*)
      with *Cons* **have** $\Gamma :\vdash \sigma \rightarrow \varphi$ **by** *blast*
      moreover

```
      have σ ∈ set (σ # Σ) by auto
      with ‹(σ # Σ) ≼ Γ› obtain γ where γ: γ ∈ set Γ ⊢ γ → σ
        using stronger-theory-witness by blast
      hence Γ :⊢ σ
        using list-deduction-modus-ponens
              list-deduction-reflection
              list-deduction-weaken
        by blast
      ultimately have Γ :⊢ φ
        using list-deduction-modus-ponens by blast
    }
    then show ?case by blast
  qed
  with assms show ?thesis by blast
qed

lemma (in classical-logic) segmented-msub-left-monotonic:
  assumes mset Σ ⊆# mset Γ
      and Σ $⊢ Φ
    shows Γ $⊢ Φ
proof −
  have ∀ Σ Γ. mset Σ ⊆# mset Γ ⟶ Σ $⊢ Φ ⟶ Γ $⊢ Φ
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons φ Φ)
    {
      fix Σ Γ
      assume mset Σ ⊆# mset Γ Σ $⊢ (φ # Φ)
      from this obtain Ψ where Ψ:
        mset (map snd Ψ) ⊆# mset Σ
        map (uncurry (⊔)) Ψ :⊢ φ
        map (uncurry (→)) Ψ @ Σ ⊖ (map snd Ψ) $⊢ Φ
        using segmented-deduction.simps(2) by blast
      let ?Ψ = map snd Ψ
      let ?Ψ′ = map (uncurry (→)) Ψ
      let ?Σ′ = ?Ψ′ @ (Σ ⊖ ?Ψ)
      let ?Γ′ = ?Ψ′ @ (Γ ⊖ ?Ψ)
      from Ψ have mset ?Ψ ⊆# mset Γ
        using ‹mset Σ ⊆# mset Γ› subset-mset.order.trans by blast
      moreover have mset (Σ ⊖ ?Ψ) ⊆# mset (Γ ⊖ ?Ψ)
        by (metis ‹mset Σ ⊆# mset Γ› list-subtract-monotonic)
      hence mset ?Σ′ ⊆# mset ?Γ′
        by simp
      with Cons.hyps Ψ(3) have ?Γ′ $⊢ Φ by blast
      ultimately have Γ $⊢ (φ # Φ)
        using Ψ(2) by fastforce
    }
```

then show *?case*
by *simp*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *segmented-stronger-theory-intro*:
assumes $\Gamma \succeq \Sigma$
shows Γ \$⊢ Σ
proof −
have $\forall\ \Gamma.\ \Sigma \preceq \Gamma \longrightarrow \Gamma$ \$⊢ Σ
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *fastforce*
next
case (*Cons* σ Σ)
{
fix Γ
assume $(\sigma\ \#\ \Sigma) \preceq \Gamma$
from *this* **obtain** γ **where** γ: $\gamma \in$ *set* $\Gamma \vdash \gamma \rightarrow \sigma\ \Sigma \preceq$ (*remove1* γ Γ)
using *stronger-theory-cons-witness* **by** *blast*
let *?*Φ = $[(\gamma,\gamma)]$
from γ *Cons* **have** (*remove1* γ Γ) \$⊢ Σ **by** *blast*
moreover have *mset* (*remove1* γ Γ) ⊆# *mset* (*map* (*uncurry* ($\rightarrow$)) *?*Φ @ Γ ⊖ (*map snd ?*Φ))
by *simp*
ultimately have *map* (*uncurry* ($\rightarrow$)) *?*Φ @ Γ ⊖ (*map snd ?*Φ) \$⊢ Σ
using *segmented-msub-left-monotonic* **by** *blast*
moreover have *map* (*uncurry* (⊔)) *?*Φ :⊢ σ
by (*simp*, *metis* $\gamma(2)$
Peirces-law
disjunction-def
list-deduction-def
list-deduction-modus-ponens
list-deduction-weaken
list-implication.simps(*1*)
list-implication.simps(*2*)))
moreover from $\gamma(1)$ **have** *mset* (*map snd ?*Φ) ⊆# *mset* Γ **by** *simp*
ultimately have Γ \$⊢ $(\sigma\ \#\ \Sigma)$
using *segmented-deduction.simps*(*2*) **by** *blast*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *witness-weaker-theory*:
assumes *mset* (*map snd* Σ) ⊆# *mset* Γ
shows *map* (*uncurry* (⊔)) $\Sigma \preceq \Gamma$

proof −
have $\forall$ Γ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\longrightarrow$ *map* (*uncurry* ($\sqcup$)) Σ $\preceq$ Γ
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
{
fix Γ
assume *mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ
hence *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*remove1* (*snd* σ) Γ)
by (*simp add*: *insert-subset-eq-iff*)
with *Cons* **have** *map* (*uncurry* ($\sqcup$)) Σ $\preceq$ *remove1* (*snd* σ) Γ **by** *blast*
moreover have *uncurry* ($\sqcup$) = (λ σ. *fst* σ $\sqcup$ *snd* σ) **by** *fastforce*
hence *uncurry* ($\sqcup$) σ = *fst* σ $\sqcup$ *snd* σ **by** *simp*
moreover have $\vdash$ *snd* σ $\to$ (*fst* σ $\sqcup$ *snd* σ)
unfolding *disjunction-def*
by (*simp add*: *axiom-k*)
ultimately have *map* (*uncurry* ($\sqcup$)) (σ # Σ) $\preceq$ (*snd* σ # (*remove1* (*snd* σ)
Γ))
by (*simp add*: *stronger-theory-left-right-cons*)
moreover have *mset* (*snd* σ # (*remove1* (*snd* σ) Γ)) = *mset* Γ
using ‹*mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ›
by (*simp*, *meson insert-DiffM mset-subset-eq-insertD*)
ultimately have *map* (*uncurry* ($\sqcup$)) (σ # Σ) $\preceq$ Γ
unfolding *stronger-theory-relation-alt-def*
by *simp*
}
then show *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *segmented-deduction-one-collapse*:
Γ \$$\vdash$ [φ] = Γ :$\vdash$ φ
proof (*rule iffI*)
assume Γ \$$\vdash$ [φ]
from *this* **obtain** Σ **where**
Σ: *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ
map (*uncurry* ($\sqcup$)) Σ :$\vdash$ φ
by *auto*
hence *map* (*uncurry* ($\sqcup$)) Σ $\preceq$ Γ
using *witness-weaker-theory* **by** *blast*
thus Γ :$\vdash$ φ **using** Σ(*2*)
using *stronger-theory-deduction-monotonic* **by** *blast*
next
assume Γ :$\vdash$ φ
let *?*Σ = *map* (λ γ. (⊥, γ)) Γ
have Γ $\preceq$ *map* (*uncurry* ($\sqcup$)) *?*Σ

```
  proof (induct Γ)
    case Nil
    then show ?case by simp
  next
    case (Cons γ Γ)
    have ⊢ (⊥ ⊔ γ) → γ
      unfolding disjunction-def
      using ex-falso-quodlibet modus-ponens excluded-middle-elimination
      by blast
    then show ?case using Cons
      by (simp add: stronger-theory-left-right-cons)
  qed
  hence map (uncurry (⊔)) ?Σ :⊢ φ
    using ‹Γ :⊢ φ› stronger-theory-deduction-monotonic by blast
  moreover have mset (map snd ?Σ) ⊆# mset Γ by (induct Γ, simp+)
  ultimately show Γ $⊢ [φ]
    using segmented-deduction.simps(1)
          segmented-deduction.simps(2)
    by blast
qed

lemma (in implication-logic) stronger-theory-combine:
  assumes Φ ≼ Δ
      and Ψ ≼ Γ
    shows (Φ @ Ψ) ≼ (Δ @ Γ)
proof −
  have ∀ Φ. Φ ≼ Δ ⟶ (Φ @ Ψ) ≼ (Δ @ Γ)
  proof (induct Δ)
    case Nil
    then show ?case
      using assms(2) stronger-theory-empty-list-intro by fastforce
  next
    case (Cons δ Δ)
    {
      fix Φ
      assume Φ ≼ (δ # Δ)
      from this obtain Σ where Σ:
        map snd Σ = Φ
        mset (map fst Σ) ⊆# mset (δ # Δ)
        ∀ (δ,φ) ∈ set Σ. ⊢ δ → φ
        unfolding stronger-theory-relation-def
        by blast
      have (Φ @ Ψ) ≼ ((δ # Δ) @ Γ)
      proof (cases ∃ φ . (δ, φ) ∈ set Σ)
        assume ∃ φ . (δ, φ) ∈ set Σ
        from this obtain φ where φ: (δ, φ) ∈ set Σ by auto
        let ?Σ = remove1 (δ, φ) Σ
        from φ Σ(1) have mset (map snd ?Σ) = mset (remove1 φ Φ)
          using remove1-pairs-list-projections-snd by fastforce
```

moreover from φ **have** *mset* (*map fst* ?Σ) = *mset* (*remove1* δ (*map fst* Σ))
  using *remove1-pairs-list-projections-fst* **by** *fastforce*
hence *mset* (*map fst* ?Σ) $\subseteq$# *mset* Δ
  using Σ(*2*) *mset.simps*(*1*) *subset-eq-diff-conv* **by** *force*
moreover from Σ(*3*) **have** $\forall$ (δ,φ) $\in$ *set* ?Σ. $\vdash \delta \to \varphi$ **by** *auto*
ultimately have *remove1* φ $\Phi \preceq \Delta$
  unfolding *stronger-theory-relation-alt-def* **by** *blast*
hence (*remove1* φ Φ @ Ψ) $\preceq$ (Δ @ Γ) **using** *Cons* **by** *auto*
from *this* **obtain** Ω **where** Ω:
  map snd Ω = (*remove1* φ Φ) @ Ψ
  mset (*map fst* Ω) $\subseteq$# *mset* (Δ @ Γ)
  $\forall$ (α,β) $\in$ *set* Ω. $\vdash \alpha \to \beta$
  unfolding *stronger-theory-relation-def*
  by *blast*
let ?Ω = (δ, φ) # Ω
have *map snd* ?Ω = φ # *remove1* φ Φ @ Ψ
  using Ω(*1*) **by** *simp*
moreover have *mset* (*map fst* ?Ω) $\subseteq$# *mset* ((δ # Δ) @ Γ)
  using Ω(*2*) **by** *simp*
moreover have $\vdash \delta \to \varphi$
  using Σ(*3*) φ **by** *blast*
hence $\forall$ (α,β) $\in$ *set* ?Ω. $\vdash \alpha \to \beta$ **using** Ω(*3*) **by** *auto*
ultimately have (φ # *remove1* φ Φ @ Ψ) $\preceq$ ((δ # Δ) @ Γ)
  by (*metis stronger-theory-relation-def*)
moreover have $\varphi \in$ *set* Φ
  using Σ(*1*) φ **by** *force*
hence (φ # *remove1* φ Φ) $\rightleftharpoons$ Φ
  by (*simp add*: *perm-remove perm-sym*)
hence (φ # *remove1* φ Φ @ Ψ) $\rightleftharpoons$ Φ @ Ψ
  by (*metis append-Cons perm-append2*)
ultimately show *?thesis*
  using *stronger-theory-left-permutation* **by** *blast*
next
assume $\nexists\varphi$. (δ, φ) $\in$ *set* Σ
hence $\delta \notin$ *set* (*map fst* Σ)
    mset Δ + *add-mset* δ (*mset* []) = *mset* (δ # Δ)
  by *auto*
hence *mset* (*map fst* Σ) $\subseteq$# *mset* Δ
  by (*metis* (*no-types*) ‹*mset* (*map fst* Σ) $\subseteq$# *mset* (δ # Δ)›
    diff-single-trivial
    mset.simps(*1*)
    set-mset-mset
    subset-eq-diff-conv)
with Σ(*1*) Σ(*3*) **have** $\Phi \preceq \Delta$
  unfolding *stronger-theory-relation-def*
  by *blast*
hence (Φ @ Ψ) $\preceq$ (Δ @ Γ) **using** *Cons* **by** *auto*
then show *?thesis*

```
      by (simp add: stronger-theory-right-cons)
    qed
  }
  then show ?case by blast
 qed
 thus ?thesis using assms by blast
qed

lemma (in classical-logic) segmented-empty-deduction:
 [] $⊢ Φ = (∀ φ ∈ set Φ. ⊢ φ)
 by (induct Φ, simp, rule iffI, fastforce+)

lemma (in classical-logic) segmented-stronger-theory-left-monotonic:
 assumes Σ ≼ Γ
     and Σ $⊢ Φ
   shows Γ $⊢ Φ
proof −
 have ∀ Σ Γ. Σ ≼ Γ ⟶ Σ $⊢ Φ ⟶ Γ $⊢ Φ
 proof (induct Φ)
  case Nil
  then show ?case by simp
 next
  case (Cons φ Φ)
  {
    fix Σ Γ
    assume Σ $⊢ (φ # Φ) Σ ≼ Γ
    from this obtain Ψ Δ where
      Ψ: mset (map snd Ψ) ⊆# mset Σ
         map (uncurry (⊔)) Ψ :⊢ φ
         map (uncurry (→)) Ψ @ Σ ⊖ (map snd Ψ) $⊢ Φ
      and
      Δ: map snd Δ = Σ
         mset (map fst Δ) ⊆# mset Γ
         ∀ (γ,σ) ∈ set Δ. ⊢ γ → σ
      unfolding stronger-theory-relation-def
      by fastforce
    from ‹mset (map snd Ψ) ⊆# mset Σ›
         ‹map snd Δ = Σ›
    obtain Ω where Ω:
      map (λ (ψ, σ, -). (ψ, σ)) Ω = Ψ
      mset (map (λ (-, σ, γ). (γ, σ)) Ω) ⊆# mset Δ
      using triple-list-exists by blast
    let ?Θ = map (λ (ψ, -, γ). (ψ, γ)) Ω
    have map snd ?Θ = map fst (map (λ (-, σ, γ). (γ, σ)) Ω)
      by auto
    hence mset (map snd ?Θ) ⊆# mset Γ
      using Ω(2) Δ(2) map-monotonic subset-mset.order.trans
      by metis
    moreover have map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) ?Θ
```

proof −
  let $?\Phi = map\ (\lambda\ (\psi, \sigma, \gamma).\ (\psi \sqcup \gamma, \psi \sqcup \sigma))\ \Omega$
  have $map\ snd\ ?\Phi = map\ (uncurry\ (\sqcup))\ \Psi$
    using $\Omega(1)$ **by** *fastforce*
  moreover have $map\ fst\ ?\Phi = map\ (uncurry\ (\sqcup))\ ?\Theta$
    by *fastforce*
  hence $mset\ (map\ fst\ ?\Phi) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ ?\Theta)$
    by (*metis subset-mset.dual-order.refl*)
  moreover
  have $mset\ (map\ (\lambda(\psi, \sigma, \text{-}).\ (\psi, \sigma))\ \Omega) \subseteq\#\ mset\ \Psi$
    using $\Omega(1)$ **by** *simp*
  hence $\forall\ (\varphi,\chi) \in set\ ?\Phi.\ \vdash \varphi \to \chi$ **using** $\Omega(2)$
  proof (*induct* Ω)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* $\omega\ \Omega$)
    let $?\Phi = map\ (\lambda\ (\psi, \sigma, \gamma).\ (\psi \sqcup \gamma, \psi \sqcup \sigma))\ (\omega\ \#\ \Omega)$
    let $?\Phi' = map\ (\lambda\ (\psi, \sigma, \gamma).\ (\psi \sqcup \gamma, \psi \sqcup \sigma))\ \Omega$
    have $mset\ (map\ (\lambda(\psi, \sigma, \text{-}).\ (\psi, \sigma))\ \Omega) \subseteq\#\ mset\ \Psi$
        $mset\ (map\ (\lambda(\text{-}, \sigma, \gamma).\ (\gamma, \sigma))\ \Omega) \subseteq\#\ mset\ \Delta$
      using *Cons.prems(1) Cons.prems(2) subset-mset.dual-order.trans* **by** *fastforce+*
    with *Cons* **have** $\forall\ (\varphi,\chi) \in set\ ?\Phi'.\ \vdash \varphi \to \chi$ **by** *fastforce*
    moreover
    let $?\psi = (\lambda\ (\psi, \text{-}, \text{-}).\ \psi)\ \omega$
    let $?\sigma = (\lambda\ (\text{-}, \sigma, \text{-}).\ \sigma)\ \omega$
    let $?\gamma = (\lambda\ (\text{-}, \text{-}, \gamma).\ \gamma)\ \omega$
    have $(\lambda(\text{-}, \sigma, \gamma).\ (\gamma, \sigma)) = (\lambda\ \omega.\ ((\lambda\ (\text{-}, \text{-}, \gamma).\ \gamma)\ \omega,(\lambda\ (\text{-}, \sigma, \text{-}).\ \sigma)\ \omega))$ **by** *auto*
    hence $(\lambda(\text{-}, \sigma, \gamma).\ (\gamma, \sigma))\ \omega = (?\gamma,\ ?\sigma)$ **by** *metis*
    hence $\vdash\ ?\gamma \to\ ?\sigma$
      using *Cons.prems(2) mset-subset-eqD* $\Delta(3)$
      by *fastforce*
    hence $\vdash (?\psi \sqcup\ ?\gamma) \to (?\psi \sqcup\ ?\sigma)$
      unfolding *disjunction-def*
      using *modus-ponens hypothetical-syllogism*
      by *blast*
    moreover have
      $(\lambda(\psi, \sigma, \gamma).\ (\psi \sqcup \gamma, \psi \sqcup \sigma)) =$
       $(\lambda\ \omega.\ (((\lambda\ (\psi, \text{-}, \text{-}).\ \psi)\ \omega) \sqcup ((\lambda\ (\text{-}, \text{-}, \gamma).\ \gamma)\ \omega),$
           $((\lambda\ (\psi, \text{-}, \text{-}).\ \psi)\ \omega) \sqcup ((\lambda\ (\text{-}, \sigma, \text{-}).\ \sigma)\ \omega)))$
      by *auto*
    hence $(\lambda(\psi, \sigma, \gamma).\ (\psi \sqcup \gamma, \psi \sqcup \sigma))\ \omega = ((?\psi \sqcup\ ?\gamma), (?\psi \sqcup\ ?\sigma))$ **by** *metis*
    ultimately show *?case* **by** *simp*
  qed
  ultimately show *?thesis*
    unfolding *stronger-theory-relation-def*
    by *blast*

```
    qed
    hence map (uncurry (⊔)) ?Θ :⊢ φ
      using Ψ(2)
            stronger-theory-deduction-monotonic
              [where Σ=map (uncurry (⊔)) Ψ
                 and Γ=map (uncurry (⊔)) ?Θ
                 and φ=φ]
      by metis
    moreover have
      (map (uncurry (→)) Ψ @ Σ ⊖ (map snd Ψ)) ≼
       (map (uncurry (→)) ?Θ @ Γ ⊖ (map snd ?Θ))
    proof −
      have map (uncurry (→)) Ψ ≼ map (uncurry (→)) ?Θ
      proof −
        let ?Φ = map (λ (ψ, σ, γ). (ψ → γ, ψ → σ)) Ω
        have map snd ?Φ = map (uncurry (→)) Ψ
          using Ω(1) by fastforce
        moreover have map fst ?Φ = map (uncurry (→)) ?Θ
          by fastforce
        hence mset (map fst ?Φ) ⊆# mset (map (uncurry (→)) ?Θ)
          by (metis subset-mset.dual-order.refl)
        moreover
        have mset (map (λ(ψ, σ, -). (ψ, σ)) Ω) ⊆# mset Ψ
          using Ω(1) by simp
        hence ∀ (φ,χ) ∈ set ?Φ. ⊢ φ → χ using Ω(2)
        proof (induct Ω)
          case Nil
          then show ?case by simp
        next
          case (Cons ω Ω)
          let ?Φ = map (λ (ψ, σ, γ). (ψ → γ, ψ → σ)) (ω # Ω)
          let ?Φ' = map (λ (ψ, σ, γ). (ψ → γ, ψ → σ)) Ω
          have mset (map (λ(ψ, σ, -). (ψ, σ)) Ω) ⊆# mset Ψ
               mset (map (λ(-, σ, γ). (γ, σ)) Ω) ⊆# mset Δ
            using Cons.prems(1) Cons.prems(2) subset-mset.dual-order.trans by
fastforce+
          with Cons have ∀ (φ,χ) ∈ set ?Φ'. ⊢ φ → χ by fastforce
          moreover
          let ?ψ = (λ (ψ, -, -). ψ) ω
          let ?σ = (λ (-, σ, -). σ) ω
          let ?γ = (λ (-, -, γ). γ) ω
           have (λ(-, σ, γ). (γ, σ)) = (λ ω. ((λ (-, -, γ). γ) ω,(λ (-, σ, -). σ) ω))
by auto
          hence (λ(-, σ, γ). (γ, σ)) ω = (?γ, ?σ) by metis
          hence ⊢ ?γ → ?σ
            using Cons.prems(2) mset-subset-eqD Δ(3)
            by fastforce
          hence ⊢ (?ψ → ?γ) → (?ψ → ?σ)
            using modus-ponens hypothetical-syllogism
```

```
        by blast
      moreover have
        (λ(ψ, σ, γ). (ψ → γ, ψ → σ)) =
         (λ ω. (((λ (ψ, -, -). ψ) ω) → ((λ (-, -, γ). γ) ω),
               ((λ (ψ, -, -). ψ) ω) → ((λ (-, σ, -). σ) ω)))
        by auto
      hence (λ(ψ, σ, γ). (ψ → γ, ψ → σ)) ω = ((?ψ → ?γ), (?ψ → ?σ)) by
metis
      ultimately show ?case by simp
    qed
    ultimately show ?thesis
      unfolding stronger-theory-relation-def
      by blast
  qed
  moreover
  have (Σ ⊖ (map snd Ψ)) ≼ (Γ ⊖ (map snd ?Θ))
  proof −
    let ?Δ = Δ ⊖ (map (λ (-, σ, γ). (γ, σ)) Ω)
    have mset (map fst ?Δ) ⊆# mset (Γ ⊖ (map snd ?Θ))
      using Δ(2)
      by (metis Ω(2)
               ‹map snd (map (λ(ψ, -, γ). (ψ, γ)) Ω) =
               map fst (map (λ(-, σ, γ). (γ, σ)) Ω)›
               list-subtract-monotonic
               map-list-subtract-mset-equivalence)
    moreover
    from Ω(2) have mset ?Δ ⊆# mset Δ by simp
    hence ∀ (γ,σ) ∈ set ?Δ. ⊢ γ → σ
      using Δ(3)
      by (metis mset-subset-eqD set-mset-mset)
    moreover
    have map snd (map (λ(-, σ, γ). (γ, σ)) Ω) = map snd Ψ
      using Ω(1)
      by (induct Ω, simp, fastforce)
    hence mset (map snd ?Δ) = mset (Σ ⊖ (map snd Ψ))
      by (metis Δ(1) Ω(2) map-list-subtract-mset-equivalence)
    ultimately show ?thesis
      by (metis stronger-theory-relation-alt-def)
  qed
  ultimately show ?thesis using stronger-theory-combine by blast
 qed
 hence map (uncurry (→)) ?Θ @ Γ ⊖ (map snd ?Θ) $⊢ Φ
  using Ψ(3) Cons by blast
 ultimately have Γ $⊢ (φ # Φ)
  by (metis segmented-deduction.simps(2))
}
then show ?case by blast
qed
with assms show ?thesis by blast
```

qed

lemma (**in** *classical-logic*) *negated-segmented-deduction*:
  ∼ Γ $⊢ (φ # Φ) = (∃ Ψ. *mset* (*map fst* Ψ) ⊆# *mset* Γ ∧
                    ∼ (*map* (*uncurry* (\)) Ψ) :⊢ φ ∧
                    ∼ (*map* (*uncurry* (⊓)) Ψ @ Γ ⊖ (*map fst* Ψ)) $⊢ Φ)
proof (*rule iffI*)
  assume ∼ Γ $⊢ (φ # Φ)
  from *this* **obtain** Ψ **where** Ψ:
    mset (*map snd* Ψ) ⊆# *mset* (∼ Γ)
    map (*uncurry* (⊔)) Ψ :⊢ φ
    map (*uncurry* (→)) Ψ @ ∼ Γ ⊖ *map snd* Ψ $⊢ Φ
    using *segmented-deduction.simps*(*2*)
    by *metis*
  from *this* **obtain** Δ **where** Δ:
    mset Δ ⊆# *mset* Γ
    map snd Ψ = ∼ Δ
    unfolding *map-negation-def*
    using *mset-sub-map-list-exists* [**where** *f*=∼ **and** Γ=Γ]
    by *metis*
  let *?*Ψ = *zip* Δ (*map fst* Ψ)
  from Δ(*2*) **have** *map fst ?*Ψ = Δ
    unfolding *map-negation-def*
    by (*metis length-map map-fst-zip*)
  with Δ(*1*) **have** *mset* (*map fst ?*Ψ) ⊆# *mset* Γ
    by *simp*
  moreover have ∀ Δ. *map snd* Ψ = ∼ Δ ⟶
                  map (*uncurry* (⊔)) Ψ ⪯ ∼ (*map* (*uncurry* (\)) (*zip* Δ (*map fst*
Ψ)))
  proof (*induct* Ψ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ψ Ψ)
    let *?*ψ = *fst* ψ
    {
      fix Δ
      assume *map snd* (ψ # Ψ) = ∼ Δ
      from *this* **obtain** γ **where** γ: ∼ γ = *snd* ψ γ = *hd* Δ **by** *auto*
      from ‹*map snd* (ψ # Ψ) = ∼ Δ› **have** *map snd* Ψ = ∼ (*tl* Δ) **by** *auto*
      with *Cons.hyps* **have**
        map (*uncurry* (⊔)) Ψ ⪯ ∼ (*map* (*uncurry* (\)) (*zip* (*tl* Δ) (*map fst* Ψ)))
        by *auto*
      moreover
      {
        fix ψ γ
        have ⊢ ∼(γ \ ψ) → (ψ ⊔ ∼ γ)
          unfolding *disjunction-def*
                    subtraction-def

conjunction-def
negation-def
by (*meson modus-ponens*
flip-implication
hypothetical-syllogism)
} **note** *tautology* = *this*
have *uncurry* ($\sqcup$) = (λ ψ. (*fst* ψ) $\sqcup$ (*snd* ψ))
by *fastforce*
with γ **have** *uncurry* ($\sqcup$) ψ = *?*ψ $\sqcup$ $\sim$ γ
by *simp*
with *tautology* **have** $\vdash$ $\sim$(γ \ *?*ψ) $\rightarrow$ *uncurry* ($\sqcup$) ψ
by *simp*
ultimately have *map* (*uncurry* ($\sqcup$)) (ψ # Ψ) $\preceq$
$\sim$ (*map* (*uncurry* (\)) ((*zip* ((*hd* Δ) # (*tl* Δ)) (*map fst* (ψ #
Ψ)))))
using *stronger-theory-left-right-cons* γ(*2*)
by *simp*
hence *map* (*uncurry* ($\sqcup$)) (ψ # Ψ) $\preceq$
$\sim$ (*map* (*uncurry* (\)) (*zip* Δ (*map fst* (ψ # Ψ))))
using ‹*map snd* (ψ # Ψ) = $\sim$ Δ› **by** *force*
}
thus *?case* **by** *blast*
qed
with Ψ(*2*) Δ(*2*) **have** $\sim$ (*map* (*uncurry* (\)) *?*Ψ) :$\vdash$ φ
using *stronger-theory-deduction-monotonic* **by** *blast*
moreover
have (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\sim$ Γ $\ominus$ *map snd* Ψ) $\preceq$
$\sim$ (*map* (*uncurry* ($\sqcap$)) *?*Ψ @ Γ $\ominus$ (*map fst ?*Ψ))
proof −
from Δ(*1*) **have** *mset* ($\sim$ Γ $\ominus$ $\sim$ Δ) = *mset* ($\sim$ (Γ $\ominus$ Δ))
by (*simp add*: *image-mset-Diff*)
hence *mset* ($\sim$ Γ $\ominus$ *map snd* Ψ) = *mset* ($\sim$ (Γ $\ominus$ *map fst ?*Ψ))
using Ψ(*1*) Δ(*2*) ‹*map fst ?*Ψ = Δ› **by** *simp*
hence ($\sim$ Γ $\ominus$ *map snd* Ψ) $\preceq$ $\sim$ (Γ $\ominus$ *map fst ?*Ψ)
by (*simp add*: *msub-stronger-theory-intro*)
moreover have $\forall$ Δ. *map snd* Ψ = $\sim$ Δ $\longrightarrow$
map (*uncurry* ($\rightarrow$)) Ψ $\preceq$ $\sim$ (*map* (*uncurry* ($\sqcap$)) (*zip* Δ (*map*
fst Ψ)))
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
let *?*ψ = *fst* ψ
{
fix Δ
assume *map snd* (ψ # Ψ) = $\sim$ Δ
from *this* **obtain** γ **where** γ: $\sim$ γ = *snd* ψ γ = *hd* Δ **by** *auto*
from ‹*map snd* (ψ # Ψ) = $\sim$ Δ› **have** *map snd* Ψ = $\sim$ (*tl* Δ) **by** *auto*

```
        with Cons.hyps have
          map (uncurry (→)) Ψ ≼ ∼ (map (uncurry (⊓)) (zip (tl Δ) (map fst Ψ)))
          by simp
        moreover
        {
          fix ψ γ
          have ⊢ ∼(γ ⊓ ψ) → (ψ → ∼ γ)
            unfolding disjunction-def
                      conjunction-def
                      negation-def
            by (meson modus-ponens
                      flip-implication
                      hypothetical-syllogism)
        } note tautology = this
        have (uncurry (→)) = (λ ψ. (fst ψ) → (snd ψ))
          by fastforce
        with γ have uncurry (→) ψ = ?ψ → ∼ γ
          by simp
        with tautology have ⊢ ∼(γ ⊓ ?ψ) → (uncurry (→)) ψ
          by simp
        ultimately have map (uncurry (→)) (ψ # Ψ) ≼
                        ∼ (map (uncurry (⊓)) ((zip ((hd Δ) # (tl Δ)) (map fst (ψ #
Ψ)))))
          using stronger-theory-left-right-cons γ(2)
          by simp
        hence map (uncurry (→)) (ψ # Ψ) ≼
              ∼ (map (uncurry (⊓)) (zip Δ (map fst (ψ # Ψ))))
          using ‹map snd (ψ # Ψ) = ∼ Δ› by force
      }
      then show ?case by blast
    qed
    ultimately have (map (uncurry (→)) Ψ @ ∼ Γ ⊖ map snd Ψ) ≼
                    (∼ (map (uncurry (⊓)) ?Ψ) @ ∼ (Γ ⊖ (map fst ?Ψ)))
      using stronger-theory-combine Δ(2)
      by metis
    thus ?thesis by simp
  qed
  hence ∼ (map (uncurry (⊓)) ?Ψ @ Γ ⊖ (map fst ?Ψ)) $⊢ Φ
    using Ψ(3) segmented-stronger-theory-left-monotonic
    by blast
  ultimately show ∃ Ψ. mset (map fst Ψ) ⊆# mset Γ ∧
                        ∼ (map (uncurry (\)) Ψ) :⊢ φ ∧
                        ∼ (map (uncurry (⊓)) Ψ @ Γ ⊖ (map fst Ψ)) $⊢ Φ
    by metis
next
  assume ∃ Ψ. mset (map fst Ψ) ⊆# mset Γ ∧
              ∼ (map (uncurry (\)) Ψ) :⊢ φ ∧
              ∼ (map (uncurry (⊓)) Ψ @ Γ ⊖ map fst Ψ) $⊢ Φ
  from this obtain Ψ where Ψ:
```

```
    mset (map fst Ψ) ⊆# mset Γ
    ∼ (map (uncurry (\)) Ψ) :⊢ φ
    ∼ (map (uncurry (⊓)) Ψ @ Γ ⊖ map fst Ψ) $⊢ Φ
    by auto
  let ?Ψ = zip (map snd Ψ) (∼ (map fst Ψ))
  from Ψ(1) have mset (map snd ?Ψ) ⊆# mset (∼ Γ)
    by (simp, metis image-mset-subseteq-mono multiset.map-comp)
  moreover have ∼ (map (uncurry (\)) Ψ) ≼ map (uncurry (⊔)) ?Ψ
  proof (induct Ψ)
    case Nil
    then show ?case by simp
  next
    case (Cons ψ Ψ)
    let ?γ = fst ψ
    let ?ψ = snd ψ
    {
      fix ψ γ
      have ⊢ (ψ ⊔ ∼ γ) → ∼(γ \ ψ)
        unfolding disjunction-def
                  subtraction-def
                  conjunction-def
                  negation-def
        by (meson modus-ponens
                  flip-implication
                  hypothetical-syllogism)
    } note tautology = this
    have ∼ ∘ uncurry (\) = (λ ψ. ∼ ((fst ψ) \ (snd ψ)))
         uncurry (⊔) = (λ (ψ,γ). ψ ⊔ γ)
      by fastforce+
    with tautology have ⊢ uncurry (⊔) (?ψ, ∼ ?γ) → (∼ ∘ uncurry (\)) ψ
      by fastforce
    with Cons.hyps have
      ((∼ ∘ uncurry (\)) ψ # ∼ (map (uncurry (\)) Ψ)) ≼
       (uncurry (⊔) (?ψ, ∼ ?γ) # map (uncurry (⊔)) (zip (map snd Ψ) (∼ (map
fst Ψ))))
      using stronger-theory-left-right-cons by blast
    thus ?case by simp
  qed
  with Ψ(2) have map (uncurry (⊔)) ?Ψ :⊢ φ
    using stronger-theory-deduction-monotonic by blast
  moreover have ∼ (map (uncurry (⊓)) Ψ @ Γ ⊖ map fst Ψ) ≼
                (map (uncurry (→)) ?Ψ @ ∼ Γ ⊖ map snd ?Ψ)
  proof −
    have ∼ (map (uncurry (⊓)) Ψ) ≼ map (uncurry (→)) ?Ψ
    proof (induct Ψ)
      case Nil
      then show ?case by simp
    next
      case (Cons ψ Ψ)
```

```
      let ?γ = fst ψ
      let ?ψ = snd ψ
      {
        fix ψ γ
        have ⊢ (ψ → ∼ γ) → ∼(γ ⊓ ψ)
          unfolding disjunction-def
                    conjunction-def
                    negation-def
          by (meson modus-ponens
                    flip-implication
                    hypothetical-syllogism)
      } note tautology = this
      have ∼ ∘ uncurry (⊓) = (λ ψ. ∼ ((fst ψ) ⊓ (snd ψ)))
           uncurry (→) = (λ (ψ,γ). ψ → γ)
        by fastforce+
      with tautology have ⊢ uncurry (→) (?ψ, ∼ ?γ) → (∼ ∘ uncurry (⊓)) ψ
        by fastforce
      with Cons.hyps have
        ((∼ ∘ uncurry (⊓)) ψ # ∼ (map (uncurry (⊓)) Ψ)) ≼
          (uncurry (→) (?ψ, ∼ ?γ) # map (uncurry (→)) (zip (map snd Ψ) (∼
(map fst Ψ))))
        using stronger-theory-left-right-cons by blast
      then show ?case by simp
    qed
    moreover have mset (∼ (Γ ⊖ map fst Ψ)) = mset (∼ Γ ⊖ map snd ?Ψ)
      using Ψ(1)
      by (simp add: image-mset-Diff multiset.map-comp)
    hence ∼ (Γ ⊖ map fst Ψ) ≼ (∼ Γ ⊖ map snd ?Ψ)
      using stronger-theory-reflexive
            stronger-theory-right-permutation
            mset-eq-perm
      by blast
    ultimately show ?thesis
      using stronger-theory-combine
      by simp
  qed
  hence map (uncurry (→)) ?Ψ @ ∼ Γ ⊖ map snd ?Ψ $⊢ Φ
    using Ψ(3) segmented-stronger-theory-left-monotonic by blast
  ultimately show ∼ Γ $⊢ (φ # Φ)
    using segmented-deduction.simps(2) by blast
qed

lemma (in probability-logic) segmented-deduction-summation-introduction:
  assumes ∼ Γ $⊢ ∼ Φ
  shows (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
proof −
  have ∀ Γ. ∼ Γ $⊢ ∼ Φ ⟶ (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
  proof (induct Φ)
    case Nil
```

```
  then show ?case
   by (simp, metis (full-types) ex-map-conv probability-non-negative sum-list-nonneg)
next
  case (Cons φ Φ)
  {
    fix Γ
    assume ∼ Γ $⊢ ∼ (φ # Φ)
    hence ∼ Γ $⊢ (∼ φ # ∼ Φ) by simp
    from this obtain Ψ where Ψ:
      mset (map fst Ψ) ⊆# mset Γ
      ∼ (map (uncurry (\)) Ψ) :⊢ ∼ φ
      ∼ (map (uncurry (⊓)) Ψ @ Γ ⊖ (map fst Ψ)) $⊢ ∼ Φ
      using negated-segmented-deduction by blast
    let ?Γ = Γ ⊖ (map fst Ψ)
    let ?Ψ₁ = map (uncurry (\)) Ψ
    let ?Ψ₂ = map (uncurry (⊓)) Ψ
    have (∑φ'←Φ. Pr φ') ≤ (∑φ←(?Ψ₂ @ ?Γ). Pr φ)
      using Cons Ψ(3) by blast
    moreover
    have Pr φ ≤ (∑φ←?Ψ₁. Pr φ)
      using Ψ(2)
            biconditional-weaken
            list-deduction-def
            map-negation-list-implication
            set-deduction-base-theory
            implication-list-summation-inequality
      by blast
   ultimately have (∑φ'←(φ # Φ). Pr φ') ≤ (∑γ ←(?Ψ₁ @ ?Ψ₂ @ ?Γ). Pr
γ)
      by simp
    moreover have (∑φ'←(?Ψ₁ @ ?Ψ₂). Pr φ') = (∑γ←(map fst Ψ). Pr γ)
    proof (induct Ψ)
      case Nil
      then show ?case by simp
    next
      case (Cons ψ Ψ)
      let ?Ψ₁ = map (uncurry (\)) Ψ
      let ?Ψ₂ = map (uncurry (⊓)) Ψ
      let ?ψ₁ = uncurry (\) ψ
      let ?ψ₂ = uncurry (⊓) ψ
      assume (∑φ'←(?Ψ₁ @ ?Ψ₂). Pr φ') = (∑γ←(map fst Ψ). Pr γ)
      moreover
      {
        let ?γ = fst ψ
        let ?ψ = snd ψ
        have uncurry (\) = (λ ψ. (fst ψ) \ (snd ψ))
             uncurry (⊓) = (λ ψ. (fst ψ) ⊓ (snd ψ))
          by fastforce+
        moreover have Pr ?γ = Pr (?γ \ ?ψ) + Pr (?γ ⊓ ?ψ)
```

by (*simp add*: *subtraction-identity*)
ultimately have $Pr\ ?\gamma = Pr\ ?\psi_1 + Pr\ ?\psi_2$
by *simp*
}
moreover have $mset\ (?\psi_1\ \#\ ?\psi_2\ \#\ (?\Psi_1\ @\ ?\Psi_2)) =$
$mset\ (map\ (uncurry\ (\backslash))\ (\psi\ \#\ \Psi)\ @\ map\ (uncurry\ (\sqcap))\ (\psi\ \#\ \Psi))$
(**is** *mset* - = *mset ?rhs*)
by *simp*
hence $(\sum \varphi' \leftarrow (?\psi_1\ \#\ ?\psi_2\ \#\ (?\Psi_1\ @\ ?\Psi_2)).\ Pr\ \varphi') = (\sum \gamma \leftarrow ?rhs.\ Pr\ \gamma)$
by *auto*
ultimately show *?case* **by** *simp*
qed
moreover have $mset\ ((map\ fst\ \Psi)\ @\ ?\Gamma) = mset\ \Gamma$
using $\Psi(1)$
by *simp*
hence $(\sum \varphi' \leftarrow ((map\ fst\ \Psi)\ @\ ?\Gamma).\ Pr\ \varphi') = (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
by (*metis mset-map sum-mset-sum-list*)
ultimately have $(\sum \varphi' \leftarrow (\varphi\ \#\ \Phi).\ Pr\ \varphi') \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
by *simp*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

primrec (**in** *implication-logic*)
firstComponent :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{A})$
where
$\mathfrak{A}\ \Psi\ [] = []$
$|\ \mathfrak{A}\ \Psi\ (\delta\ \#\ \Delta) =$
$(case\ find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi\ of$
$None \Rightarrow \mathfrak{A}\ \Psi\ \Delta$
$|\ Some\ \psi \Rightarrow \psi\ \#\ (\mathfrak{A}\ (remove1\ \psi\ \Psi)\ \Delta))$

primrec (**in** *implication-logic*)
secondComponent :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{B})$
where
$\mathfrak{B}\ \Psi\ [] = []$
$|\ \mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta) =$
$(case\ find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi\ of$
$None \Rightarrow \mathfrak{B}\ \Psi\ \Delta$
$|\ Some\ \psi \Rightarrow \delta\ \#\ (\mathfrak{B}\ (remove1\ \psi\ \Psi)\ \Delta))$

lemma (**in** *implication-logic*) *firstComponent-secondComponent-mset-connection*:
$mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{A}\ \Psi\ \Delta)) = mset\ (map\ snd\ (\mathfrak{B}\ \Psi\ \Delta))$
proof −
have $\forall\ \Psi.\ mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{A}\ \Psi\ \Delta)) = mset\ (map\ snd\ (\mathfrak{B}\ \Psi\ \Delta))$

proof (*induct* Δ)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons* δ Δ)
  {
    fix Ψ
    have *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{A}$ Ψ (δ # Δ))) =
        mset (*map* *snd* ($\mathfrak{B}$ Ψ (δ # Δ)))
    proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
      case *True*
      then show *?thesis* **using** *Cons* **by** *simp*
    next
      case *False*
      from *this* **obtain** ψ **where**
        find ($\lambda\psi$. *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
        uncurry ($\rightarrow$) ψ = *snd* δ
        using *find-Some-predicate*
        by *fastforce*
      then show *?thesis* **using** *Cons* **by** *simp*
    qed
  }
  then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *secondComponent-right-empty* [*simp*]:
  $\mathfrak{B}$ [] Δ = []
  by (*induct* Δ, *simp*+)

lemma (**in** *implication-logic*) *firstComponent-msub*:
  mset ($\mathfrak{A}$ Ψ Δ) $\subseteq$# *mset* Ψ
proof –
  have $\forall$ Ψ. *mset* ($\mathfrak{A}$ Ψ Δ) $\subseteq$# *mset* Ψ
  proof(*induct* Δ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* δ Δ)
    {
      fix Ψ
      have *mset* ($\mathfrak{A}$ Ψ (δ # Δ)) $\subseteq$# *mset* Ψ
      proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
        case *True*
        then show *?thesis* **using** *Cons* **by** *simp*
      next
        case *False*
        from *this* **obtain** ψ **where**

```
      ψ: find (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
        ψ ∈ set Ψ
      using find-Some-set-membership
      by fastforce
    have mset (𝔄 (remove1 ψ Ψ) Δ) ⊆# mset (remove1 ψ Ψ)
      using Cons by metis
    thus ?thesis using ψ by (simp add: insert-subset-eq-iff)
  qed
 }
 then show ?case by blast
qed
thus ?thesis by blast
qed

lemma (in implication-logic) secondComponent-msub:
  mset (𝔅 Ψ Δ) ⊆# mset Δ
proof −
  have ∀Ψ. mset (𝔅 Ψ Δ) ⊆# mset Δ
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have mset (𝔅 Ψ (δ # Δ)) ⊆# mset (δ # Δ)
      using Cons
      by (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None,
            simp,
            metis add-mset-remove-trivial
                  diff-subset-eq-self
                  subset-mset.order-trans,
            auto)
    }
    thus ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in implication-logic) secondComponent-snd-projection-msub:
  mset (map snd (𝔅 Ψ Δ)) ⊆# mset (map (uncurry (→)) Ψ)
proof −
  have ∀Ψ. mset (map snd (𝔅 Ψ Δ)) ⊆# mset (map (uncurry (→)) Ψ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
```

fix Ψ
have *mset* (*map snd* ($\mathfrak{B}$ Ψ (δ # Δ))) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
case *True*
then show *?thesis*
using *Cons* **by** *simp*
next
case *False*
from *this* **obtain** ψ **where** ψ:
find (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *Some* ψ
by *auto*
hence $\mathfrak{B}$ Ψ (δ # Δ) = δ # ($\mathfrak{B}$ (*remove1* ψ Ψ) Δ)
using ψ **by** *fastforce*
with *Cons* **have** *mset* (*map snd* ($\mathfrak{B}$ Ψ (δ # Δ))) $\subseteq$#
mset ((*snd* δ) # *map* (*uncurry* ($\rightarrow$)) (*remove1* ψ Ψ))
by (*simp*, *metis mset-map mset-remove1*)
moreover from ψ **have** *snd* δ = (*uncurry* ($\rightarrow$)) ψ
using *find-Some-predicate* **by** *fastforce*
ultimately have *mset* (*map snd* ($\mathfrak{B}$ Ψ (δ # Δ))) $\subseteq$#
mset (*map* (*uncurry* ($\rightarrow$)) (ψ # (*remove1* ψ Ψ)))
by *simp*
thus *?thesis*
by (*metis* ψ *find-Some-set-membership mset-eq-perm mset-map perm-remove*)
qed
}
thus *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *secondComponent-diff-msub*:
assumes *mset* (*map snd* Δ) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ))
shows *mset* (*map snd* (Δ $\ominus$ ($\mathfrak{B}$ Ψ Δ))) $\subseteq$# *mset* (Γ $\ominus$ (*map snd* Ψ))
proof –
have $\forall$ Ψ Γ. *mset* (*map snd* Δ) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ)) $\longrightarrow$
mset (*map snd* (Δ $\ominus$ ($\mathfrak{B}$ Ψ Δ))) $\subseteq$# *mset* (Γ $\ominus$ (*map snd* Ψ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ Γ
assume $\diamond$: *mset* (*map snd* (δ # Δ)) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ *map snd* Ψ)
have *mset* (*map snd* ((δ # Δ) $\ominus$ $\mathfrak{B}$ Ψ (δ # Δ))) $\subseteq$# *mset* (Γ $\ominus$ *map snd* Ψ)
proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)

```
case True
hence A: snd δ ∉ set (map (uncurry (→)) Ψ)
proof (induct Ψ)
  case Nil
  then show ?case by simp
next
  case (Cons ψ Ψ)
  then show ?case
    by (cases uncurry (→) ψ = snd δ, simp+)
qed
moreover have mset (map snd Δ)
         ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ) − {#snd δ#}
  using ◇ insert-subset-eq-iff by fastforce
ultimately have mset (map snd Δ)
             ⊆# mset (map (uncurry (→)) Ψ @ (remove1 (snd δ) Γ) ⊖ map
snd Ψ)
  by (metis (no-types) mset-remove1
                       mset-eq-perm union-code
                       list-subtract.simps(2)
                       list-subtract-remove1-cons-perm
                       remove1-append)
hence B: mset (map snd (Δ ⊖ (𝔅 Ψ Δ))) ⊆# mset (remove1 (snd δ) Γ
⊖ (map snd Ψ))
  using Cons by blast
have C: snd δ ∈# mset (snd δ # map snd Δ @
                      (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ) ⊖ (snd δ #
map snd Δ))
  by (meson in-multiset-in-set list.set-intros(1))
have mset (map snd (δ # Δ))
   + (mset (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ)
     − mset (map snd (δ # Δ)))
  = mset (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ)
  using ◇ subset-mset.add-diff-inverse by blast
then have snd δ ∈# mset (map (uncurry (→)) Ψ) + (mset Γ − mset (map
snd Ψ))
  using C by simp
with A have snd δ ∈ set Γ
  by (metis (no-types) diff-subset-eq-self
                       in-multiset-in-set
                       subset-mset.add-diff-inverse
                       union-iff)
have D: 𝔅 Ψ Δ = 𝔅 Ψ (δ # Δ)
  using ‹find (λψ. uncurry (→) ψ = snd δ) Ψ = None›
  by simp
obtain diff :: 'a list ⇒ 'a list ⇒ 'a list where
  ∀x0 x1. (∃v2. x1 @ v2 ⇌ x0) = (x1 @ diff x0 x1 ⇌ x0)
  by moura
then have E: mset (map snd (𝔅 Ψ (δ # Δ))
           @ diff (map (uncurry (→)) Ψ) (map snd (𝔅 Ψ (δ # Δ))))
```

$= mset\ (map\ (uncurry\ (\rightarrow))\ \Psi)$
by (*meson secondComponent-snd-projection-msub mset-eq-perm mset-le-perm-append*)
have F: $\forall\, a\ m\ ma.\ (add\text{-}mset\ (a::'a)\ m \subseteq\#\ ma) = (a \in\#\ ma \wedge m \subseteq\#\ ma - \{\#a\#\})$
using *insert-subset-eq-iff* **by** *blast*
then have $snd\ \delta \in\#\ mset\ (map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))$
$@\ diff\ (map\ (uncurry\ (\rightarrow))\ \Psi)\ (map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))))$
$+\ mset\ (\Gamma \ominus map\ snd\ \Psi)$
using E $\diamondsuit$ **by** *force*
then have $snd\ \delta \in\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$
using A E **by** (*metis* (*no-types*) *in-multiset-in-set union-iff*)
then have G: $add\text{-}mset\ (snd\ \delta)\ (mset\ (map\ snd\ (\Delta \ominus \mathfrak{B}\ \Psi\ \Delta))) \subseteq\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$
using B F **by** *force*
have H: $\forall\, ps\ psa\ f.\ \neg\ mset\ (ps::('a \times 'a)\ list) \subseteq\#\ mset\ psa\ \vee$
$mset\ ((map\ f\ psa::'a\ list) \ominus map\ f\ ps) = mset\ (map\ f\ (psa \ominus ps))$
using *map-list-subtract-mset-equivalence* **by** *blast*
have $snd\ \delta \notin\#\ mset\ (map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)))$
$+\ mset\ (diff\ (map\ (uncurry\ (\rightarrow))\ \Psi)\ (map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))))$
using A E **by** *auto*
then have $add\text{-}mset\ (snd\ \delta)\ (mset\ (map\ snd\ (\Delta \ominus \mathfrak{B}\ \Psi\ \Delta)))$
$= mset\ (map\ snd\ (\delta\ \#\ \Delta) \ominus map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)))$
using D H *secondComponent-msub* **by** *auto*
then show *?thesis*
using G H **by** (*metis* (*no-types*) *secondComponent-msub*)
next
case *False*
from *this* **obtain** ψ **where** ψ: $find\ (\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
by *auto*
let $?\Psi' = remove1\ \psi\ \Psi$
let $?\Gamma' = remove1\ (snd\ \psi)\ \Gamma$
have $snd\ \delta = uncurry\ (\rightarrow)\ \psi$
$\psi \in set\ \Psi$
$mset\ ((\delta\ \#\ \Delta) \ominus \mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)) =$
$mset\ (\Delta \ominus \mathfrak{B}\ ?\Psi'\ \Delta)$
using ψ *find-Some-predicate find-Some-set-membership*
by *fastforce+*
moreover
have $mset\ (\Gamma \ominus map\ snd\ \Psi) = mset\ (?\Gamma' \ominus map\ snd\ ?\Psi')$
by (*simp*, *metis* ‹$\psi \in set\ \Psi$› *image-mset-add-mset in-multiset-in-set insert-DiffM*)
moreover
obtain $search :: ('a \times 'a)\ list \Rightarrow ('a \times 'a \Rightarrow bool) \Rightarrow 'a \times 'a$ **where**
$\forall\, xs\ P.\ (\exists\, x.\ x \in set\ xs \wedge P\ x) = (search\ xs\ P \in set\ xs \wedge P\ (search\ xs\ P))$
by *moura*
then have $\forall\, p\ ps.\ (find\ p\ ps \neq None \vee (\forall\, pa.\ pa \notin set\ ps \vee \neg\ p\ pa))$

$\wedge$ (*find p ps* = *None* $\vee$ *search ps p* $\in$ *set ps* $\wedge$ *p* (*search ps p*))
by (*metis* (*full-types*) *find-None-iff*)
then have (*find* ($\lambda p.$ *uncurry* ($\rightarrow$) *p* = *snd* δ) $\Psi \neq$ *None*
$\vee$ ($\forall p.$ $p \notin$ *set* Ψ $\vee$ *uncurry* ($\rightarrow$) $p \neq$ *snd* δ))
$\wedge$ (*find* ($\lambda p.$ *uncurry* ($\rightarrow$) *p* = *snd* δ) Ψ = *None*
$\vee$ *search* Ψ ($\lambda p.$ *uncurry* ($\rightarrow$) *p* = *snd* δ) $\in$ *set* Ψ
$\wedge$ *uncurry* ($\rightarrow$) (*search* Ψ ($\lambda p.$ *uncurry* ($\rightarrow$) *p* = *snd* δ)) = *snd* δ)
by *blast*
hence *snd* $\delta \in$ *set* (*map* (*uncurry* ($\rightarrow$)) Ψ)
by (*metis* (*no-types*) *False image-eqI image-set*)
moreover
have *A*: *add-mset* (*uncurry* ($\rightarrow$) ψ) (*image-mset snd* (*mset* Δ))
= *image-mset snd* (*add-mset* δ (*mset* Δ))
by (*simp add*: ‹*snd* δ = *uncurry* ($\rightarrow$) ψ›)
have *B*: {#*snd* δ#} $\subseteq$# *image-mset* (*uncurry* ($\rightarrow$)) (*mset* Ψ)
using ‹*snd* $\delta \in$ *set* (*map* (*uncurry* ($\rightarrow$)) Ψ)› **by** *force*
have *image-mset* (*uncurry* ($\rightarrow$)) (*mset* Ψ) $-$ {#*snd* δ#}
= *image-mset* (*uncurry* ($\rightarrow$)) (*mset* (*remove1* ψ Ψ))
by (*simp add*: ‹$\psi \in$ *set* Ψ› ‹*snd* δ = *uncurry* ($\rightarrow$) ψ› *image-mset-Diff*)
then have *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ (*remove1* ψ Ψ) Δ))
$\subseteq$# *mset* (*remove1* (*snd* ψ) $\Gamma \ominus$ *map snd* (*remove1* ψ Ψ))
by (*metis* (*no-types*)
A B $\diamondsuit$ *Cons.hyps*
calculation(*1*)
calculation(*4*)
insert-subset-eq-iff
mset.simps(*2*)
mset-map
subset-mset.diff-add-assoc2
union-code)
ultimately show *?thesis* **by** *fastforce*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *auto*
qed

primrec (**in** *classical-logic*)
mergeWitness :: ('*a* $\times$ '*a*) *list* $\Rightarrow$ ('*a* $\times$ '*a*) *list* $\Rightarrow$ ('*a* $\times$ '*a*) *list* ($\mathfrak{J}$)
where
$\mathfrak{J}$ Ψ [] = Ψ
| $\mathfrak{J}$ Ψ (δ # Δ) =
(*case find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
None $\Rightarrow$ δ # $\mathfrak{J}$ Ψ Δ
| *Some* ψ $\Rightarrow$ (*fst* $\delta \sqcap$ *fst* ψ, *snd* ψ) # ($\mathfrak{J}$ (*remove1* ψ Ψ) Δ))

lemma (**in** *classical-logic*) *mergeWitness-right-empty* [*simp*]:
$\mathfrak{J}$ [] Δ = Δ

by (*induct* Δ, *simp*+)

lemma (**in** *classical-logic*) *secondComponent-mergeWitness-snd-projection*:
mset (*map snd* Ψ @ *map snd* (Δ ⊖ (𝔅 Ψ Δ))) = *mset* (*map snd* (ℑ Ψ Δ))
proof −
have ∀ Ψ. *mset* (*map snd* Ψ @ *map snd* (Δ ⊖ (𝔅 Ψ Δ))) = *mset* (*map snd* (ℑ Ψ Δ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* Ψ @ *map snd* ((δ # Δ) ⊖ 𝔅 Ψ (δ # Δ))) =
mset (*map snd* (ℑ Ψ (δ # Δ)))
proof (*cases find* (λ ψ. (*uncurry* (→)) ψ = *snd* δ) Ψ = *None*)
case *True*
then show *?thesis*
using *Cons*
by (*simp*,
metis (*no-types*, *lifting*)
ab-semigroup-add-class.add-ac(*1*)
add-mset-add-single
image-mset-single
image-mset-union
secondComponent-msub
subset-mset.add-diff-assoc2)
next
case *False*
from *this* **obtain** ψ **where** ψ: *find* (λψ. *uncurry* (→) ψ = *snd* δ) Ψ = *Some* ψ
by *auto*
moreover have ψ ∈ *set* Ψ
by (*meson* ψ *find-Some-set-membership*)
moreover
let *?*Ψ′ = *remove1* ψ Ψ
from *Cons* **have**
mset (*map snd ?*Ψ′ @ *map snd* (Δ ⊖ 𝔅 *?*Ψ′ Δ)) =
mset (*map snd* (ℑ *?*Ψ′ Δ))
by *blast*
ultimately show *?thesis*
by (*simp*,
metis (*no-types*, *lifting*)
add-mset-remove-trivial-eq
image-mset-add-mset
in-multiset-in-set
union-mset-add-mset-left)
qed

```
    }
    then show ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in classical-logic) secondComponent-mergeWitness-stronger-theory:
  (map (uncurry (→)) Δ @ map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ Δ)) ≼
    map (uncurry (→)) (ℑ Ψ Δ)
proof −
  have ∀ Ψ. (map (uncurry (→)) Δ @
            map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ Δ)) ≼
            map (uncurry (→)) (ℑ Ψ Δ)
  proof (induct Δ)
    case Nil
    then show ?case
      by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have ⊢ (uncurry (→)) δ → (uncurry (→)) δ
        using axiom-k modus-ponens implication-absorption by blast
      have
        (map (uncurry (→)) (δ # Δ) @
          map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ))) ≼
          map (uncurry (→)) (ℑ Ψ (δ # Δ))
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        thus ?thesis
          using Cons
                ‹⊢ (uncurry (→)) δ → (uncurry (→)) δ›
          by (simp, metis stronger-theory-left-right-cons)
      next
        case False
         from this obtain ψ where ψ: find (λψ. uncurry (→) ψ = snd δ) Ψ =
Some ψ
          by auto
        from ψ have snd δ = uncurry (→) ψ
          using find-Some-predicate by fastforce
        from ψ ‹snd δ = uncurry (→) ψ› have
          mset (map (uncurry (→)) (δ # Δ) @
                  map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ))) =
           mset (map (uncurry (→)) (δ # Δ) @
                  map (uncurry (→)) (remove1 ψ Ψ) ⊖
                  map snd (𝔅 (remove1 ψ Ψ) Δ))
          by (simp add: find-Some-set-membership image-mset-Diff)
        hence
          (map (uncurry (→)) (δ # Δ) @
```

$map\ (uncurry\ (\rightarrow))\ \Psi \ominus map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))) \preceq$
$(map\ (uncurry\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$map\ (uncurry\ (\rightarrow))\ (remove1\ \psi\ \Psi) \ominus map\ snd\ (\mathfrak{B}\ (remove1\ \psi\ \Psi)\ \Delta))$
by (*simp add*: *msub-stronger-theory-intro*)
with *Cons* ‹$\vdash (uncurry\ (\rightarrow))\ \delta \rightarrow (uncurry\ (\rightarrow))\ \delta$› **have**
$(map\ (uncurry\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$map\ (uncurry\ (\rightarrow))\ \Psi \ominus map\ snd\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)))$
$\preceq ((uncurry\ (\rightarrow))\ \delta\ \#\ map\ (uncurry\ (\rightarrow))\ (\mathfrak{J}\ (remove1\ \psi\ \Psi)\ \Delta))$
using *stronger-theory-left-right-cons*
stronger-theory-transitive
by *fastforce*
moreover
let $?\alpha = fst\ \delta$
let $?\beta = fst\ \psi$
let $?\gamma = snd\ \psi$
have $uncurry\ (\rightarrow) = (\lambda\ \delta.\ fst\ \delta \rightarrow snd\ \delta)$ **by** *fastforce*
with ψ **have** $(uncurry\ (\rightarrow))\ \delta = ?\alpha \rightarrow ?\beta \rightarrow ?\gamma$
using *find-Some-predicate* **by** *fastforce*
hence $\vdash ((?\alpha \sqcap ?\beta) \rightarrow ?\gamma) \rightarrow (uncurry\ (\rightarrow))\ \delta$
using *biconditional-def curry-uncurry* **by** *auto*
with ψ **have**
$((uncurry\ (\rightarrow))\ \delta\ \#\ map\ (uncurry\ (\rightarrow))\ (\mathfrak{J}\ (remove1\ \psi\ \Psi)\ \Delta)) \preceq$
$map\ (uncurry\ (\rightarrow))\ (\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta))$
using *stronger-theory-left-right-cons* **by** *auto*
ultimately show *?thesis*
using *stronger-theory-transitive*
by *blast*
qed
}
then show *?case* **by** *simp*
qed
thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *mergeWitness-msub-intro*:
assumes $mset\ (map\ snd\ \Psi) \subseteq\#\ mset\ \Gamma$
and $mset\ (map\ snd\ \Delta) \subseteq\#\ mset\ (map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi))$
shows $mset\ (map\ snd\ (\mathfrak{J}\ \Psi\ \Delta)) \subseteq\#\ mset\ \Gamma$
proof −
have $\forall\,\Psi\ \Gamma.\ mset\ (map\ snd\ \Psi) \subseteq\#\ mset\ \Gamma \longrightarrow$
$mset\ (map\ snd\ \Delta) \subseteq\#\ mset\ (map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi)) \longrightarrow$
$mset\ (map\ snd\ (\mathfrak{J}\ \Psi\ \Delta)) \subseteq\#\ mset\ \Gamma$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)

{
  fix Ψ :: $('a \times 'a)$ *list*
  fix Γ :: *'a list*
  assume $\diamondsuit$: *mset* (*map snd* Ψ) $\subseteq\#$ *mset* Γ
      mset (*map snd* (δ # Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
  have *mset* (*map snd* ($\mathfrak{J}$ Ψ (δ # Δ))) $\subseteq\#$ *mset* Γ
  proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
    case *True*
    hence *snd* $\delta \notin$ *set* (*map* (*uncurry* ($\rightarrow$)) Ψ)
    proof (*induct* Ψ)
      case *Nil*
      then show *?case* **by** *simp*
    next
      case (*Cons* ψ Ψ)
      hence *uncurry* ($\rightarrow$) $\psi \neq$ *snd* δ **by** *fastforce*
      with *Cons* **show** *?case* **by** *fastforce*
    qed
    with $\diamondsuit$(*2*) **have** *snd* $\delta \in\#$ *mset* ($\Gamma \ominus$ *map snd* Ψ)
      using *mset-subset-eq-insertD* **by** *fastforce*
    with $\diamondsuit$(*1*) **have** *mset* (*map snd* Ψ) $\subseteq\#$ *mset* (*remove1* (*snd* δ) Γ)
      by (*metis list-subtract-mset-homomorphism*
          mset-remove1
          single-subset-iff
          subset-mset.add-diff-assoc
          subset-mset.add-diff-inverse
          subset-mset.le-iff-add)
    moreover
    have *add-mset* (*snd* δ) (*mset* ($\Gamma \ominus$ *map snd* Ψ) $-$ {#*snd* δ#}) = *mset* ($\Gamma \ominus$ *map snd* Ψ)
      by (*meson* ‹*snd* $\delta \in\#$ *mset* ($\Gamma \ominus$ *map snd* Ψ)› *insert-DiffM*)
      then have *image-mset snd* (*mset* Δ) $-$ (*mset* Γ $-$ *add-mset* (*snd* δ) (*image-mset snd* (*mset* Ψ)))
        $\subseteq\#$ {#$x \rightarrow y$. (x, y) $\in\#$ *mset* Ψ#}
      using $\diamondsuit$(*2*) **by** (*simp*, *metis add-mset-diff-bothsides*
              list-subtract-mset-homomorphism
              mset-map subset-eq-diff-conv)
    hence *mset* (*map snd* Δ)
      $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ (*remove1* (*snd* δ) Γ) $\ominus$ (*map snd* Ψ))
      using *subset-eq-diff-conv* **by** (*simp*, *blast*)
    ultimately have *mset* (*map snd* ($\mathfrak{J}$ Ψ Δ)) $\subseteq\#$ *mset* (*remove1* (*snd* δ) Γ)
      using *Cons* **by** *blast*
    hence *mset* (*map snd* (δ # ($\mathfrak{J}$ Ψ Δ))) $\subseteq\#$ *mset* Γ
      by (*simp*, *metis* ‹*snd* $\delta \in\#$ *mset* ($\Gamma \ominus$ *map snd* Ψ)›
            cancel-ab-semigroup-add-class.diff-right-commute
            diff-single-trivial
            insert-subset-eq-iff
            list-subtract-mset-homomorphism
            multi-drop-mem-not-eq)

```
    with ‹find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None›
    show ?thesis
      by simp
  next
    case False
    from this obtain ψ where ψ:
      find (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
      by fastforce
    let ?χ = fst ψ
    let ?γ = snd ψ
    have uncurry (→) = (λ ψ. fst ψ → snd ψ)
      by fastforce
    moreover
    from this have uncurry (→) ψ = ?χ → ?γ by fastforce
    with ψ have A: (?χ, ?γ) ∈ set Ψ
            and B: snd δ = ?χ → ?γ
      using find-Some-predicate
      by (simp add: find-Some-set-membership, fastforce)
    let ?Ψ′ = remove1 (?χ, ?γ) Ψ
    from B ♢(2) have
       mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ)
− {# ?χ → ?γ #}
      by (simp add: insert-subset-eq-iff)
    moreover
    have mset (map (uncurry (→)) Ψ)
        = add-mset (case (fst ψ, snd ψ) of (x, xa) ⇒ x → xa)
                (image-mset (uncurry (→)) (mset (remove1 (fst ψ, snd ψ) Ψ)))
      by (metis (no-types) A
                image-mset-add-mset
                in-multiset-in-set
                insert-DiffM
                mset-map
                mset-remove1
                uncurry-def)
    ultimately have
     mset (map snd Δ) ⊆# mset (map (uncurry (→)) ?Ψ′ @ Γ ⊖ map snd Ψ)
      using add-diff-cancel-left′
            add-diff-cancel-right
            diff-diff-add-mset
            diff-subset-eq-self
            mset-append
            subset-eq-diff-conv
            subset-mset.diff-add
      by auto
    moreover from A B ♢
   have mset (Γ ⊖ map snd Ψ) = mset((remove1 ?γ Γ) ⊖ (remove1 ?γ (map
snd Ψ)))
      by (metis image-eqI
                list-subtract-remove1-perm
```

mset-eq-perm
prod.sel(2)
set-map)
with *A* **have** *mset* $(\Gamma \ominus$ *map snd* $\Psi) =$ *mset*((*remove1* $?\gamma\ \Gamma) \ominus$ (*map snd* $?\Psi'$))
by (*metis remove1-pairs-list-projections-snd*
in-multiset-in-set
list-subtract-mset-homomorphism
mset-remove1)
ultimately have *mset* (*map snd* Δ) $\subseteq\#$
mset (*map* (*uncurry* $(\rightarrow)$) $?\Psi'$ @ (*remove1* $?\gamma\ \Gamma$) $\ominus$ *map snd* $?\Psi'$)
by *simp*
hence *mset* (*map snd* ($\mathfrak{J}\ ?\Psi'\ \Delta$)) $\subseteq\#$ *mset* (*remove1* $?\gamma\ \Gamma$)
using *Cons* $\Diamond$(*1*) *A*
by (*metis* (*no-types*, *lifting*)
image-mset-add-mset
in-multiset-in-set
insert-DiffM
insert-subset-eq-iff
mset-map mset-remove1
prod.collapse)
with $\Diamond$(*1*) *A* **have** *mset* (*map snd* ($\mathfrak{J}\ ?\Psi'\ \Delta$)) + $\{\#\ ?\gamma\ \#\} \subseteq\#$ *mset* Γ
by (*metis add-mset-add-single*
image-eqI
insert-subset-eq-iff
mset-remove1
mset-subset-eqD
set-map
set-mset-mset
snd-conv)
hence *mset* (*map snd* ((*fst* $\delta \sqcap ?\chi$, $?\gamma$) # ($\mathfrak{J}\ ?\Psi'\ \Delta$))) $\subseteq\#$ *mset* Γ
by *simp*
moreover from ψ **have**
$\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta) =$ (*fst* $\delta \sqcap ?\chi$, $?\gamma$) # ($\mathfrak{J}\ ?\Psi'\ \Delta$)
by *simp*
ultimately show *?thesis* **by** *simp*
qed
}
thus *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *right-mergeWitness-stronger-theory*:
map (*uncurry* $(\sqcup)$) $\Delta \preceq$ *map* (*uncurry* $(\sqcup)$) ($\mathfrak{J}\ \Psi\ \Delta$)
proof –
have $\forall\ \Psi.$ *map* (*uncurry* $(\sqcup)$) $\Delta \preceq$ *map* (*uncurry* $(\sqcup)$) ($\mathfrak{J}\ \Psi\ \Delta$)
proof (*induct* Δ)

```
  case Nil
  then show ?case by simp
next
  case (Cons δ Δ)
  {
    fix Ψ
    have map (uncurry (⊔)) (δ # Δ) ⪯ map (uncurry (⊔)) (𝔍 Ψ (δ # Δ))
    proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
      case True
      hence 𝔍 Ψ (δ # Δ) = δ # 𝔍 Ψ Δ
        by simp
      moreover have ⊢ (uncurry (⊔)) δ → (uncurry (⊔)) δ
        by (metis axiom-k axiom-s modus-ponens)
      ultimately show ?thesis using Cons
        by (simp add: stronger-theory-left-right-cons)
    next
      case False
      from this obtain ψ where ψ:
        find (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
        by fastforce
      let ?χ = fst ψ
      let ?γ = snd ψ
      let ?μ = fst δ
      have uncurry (→) = (λ ψ. fst ψ → snd ψ)
           uncurry (⊔) = (λ δ. fst δ ⊔ snd δ)
        by fastforce+
      hence uncurry (⊔) δ = ?μ ⊔ (?χ → ?γ)
        using ψ find-Some-predicate
        by fastforce
      moreover
      {
        fix μ χ γ
        have ⊢ ((μ ⊓ χ) ⊔ γ) → (μ ⊔ (χ → γ))
        proof −
          have ∀𝔐. 𝔐 ⊨prop ((⟨μ⟩ ⊓ ⟨χ⟩) ⊔ ⟨γ⟩) → (⟨μ⟩ ⊔ (⟨χ⟩ → ⟨γ⟩))
            by fastforce
          hence ⊢ ⦇ ((⟨μ⟩ ⊓ ⟨χ⟩) ⊔ ⟨γ⟩) → (⟨μ⟩ ⊔ (⟨χ⟩ → ⟨γ⟩)) ⦈
            using propositional-semantics by blast
          thus ?thesis
            by simp
       qed
      }
      ultimately show ?thesis
        using Cons ψ stronger-theory-left-right-cons
        by simp
    qed
  }
  thus ?case by blast
qed
```

  thus *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *left-mergeWitness-stronger-theory*:
  map (*uncurry* $(\sqcup)$) $\Psi \preceq$ *map* (*uncurry* $(\sqcup)$) $(\mathfrak{J}\ \Psi\ \Delta)$
proof −
  have $\forall\ \Psi$. *map* (*uncurry* $(\sqcup)$) $\Psi \preceq$ *map* (*uncurry* $(\sqcup)$) $(\mathfrak{J}\ \Psi\ \Delta)$
  proof (*induct* Δ)
    case *Nil*
    then show *?case*
      by *simp*
  next
    case (*Cons* δ Δ)
    {
      fix Ψ
      have *map* (*uncurry* $(\sqcup)$) $\Psi \preceq$ *map* (*uncurry* $(\sqcup)$) $(\mathfrak{J}\ \Psi\ (\delta \# \Delta))$
      proof (*cases find* $(\lambda\ \psi.$ (*uncurry* $(\rightarrow)$) $\psi =$ *snd* $\delta)$ $\Psi =$ *None*)
        case *True*
        then show *?thesis*
          using *Cons stronger-theory-right-cons*
          by *auto*
      next
        case *False*
        from *this* **obtain** ψ **where** ψ:
          find $(\lambda\psi.$ *uncurry* $(\rightarrow)$ $\psi =$ *snd* $\delta)$ $\Psi =$ *Some* ψ
          by *fastforce*
        let $?\chi =$ *fst* ψ
        let $?\gamma =$ *snd* ψ
        let $?\mu =$ *fst* δ
        have *uncurry* $(\rightarrow) = (\lambda\ \psi.$ *fst* $\psi \rightarrow$ *snd* $\psi)$
             uncurry $(\sqcup) = (\lambda\ \delta.$ *fst* $\delta \sqcup$ *snd* $\delta)$
          by *fastforce+*
        hence
          uncurry $(\sqcup)\ \delta = ?\mu \sqcup (?\chi \rightarrow ?\gamma)$
          uncurry $(\sqcup)\ \psi = ?\chi \sqcup ?\gamma$
          using ψ *find-Some-predicate*
          by *fastforce+*
        moreover
        {
          fix $\mu\ \chi\ \gamma$
          have $\vdash ((\mu \sqcap \chi) \sqcup \gamma) \rightarrow (\chi \sqcup \gamma)$
          proof −
            have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\mu\rangle \sqcap \langle\chi\rangle) \sqcup \langle\gamma\rangle) \rightarrow (\langle\chi\rangle \sqcup \langle\gamma\rangle))$
              by *fastforce*
            hence $\vdash (\!|\ (((\langle\mu\rangle \sqcap \langle\chi\rangle) \sqcup \langle\gamma\rangle) \rightarrow (\langle\chi\rangle \sqcup \langle\gamma\rangle))\ |\!)$
              using *propositional-semantics* **by** *blast*
            thus *?thesis*
              by *simp*
          qed

```
      }
      ultimately have
        map (uncurry (⊔)) (ψ # (remove1 ψ Ψ)) ≼
         map (uncurry (⊔)) (𝔍 Ψ (δ # Δ))
        using Cons ψ stronger-theory-left-right-cons
        by simp
      moreover from ψ have ψ ∈ set Ψ
        by (simp add: find-Some-set-membership)
      hence mset (map (uncurry (⊔)) (ψ # (remove1 ψ Ψ))) =
             mset (map (uncurry (⊔)) Ψ)
        by (metis insert-DiffM
                 mset.simps(2)
                 mset-map
                 mset-remove1
                 set-mset-mset)
      hence map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) (ψ # (remove1 ψ Ψ))
        by (simp add: msub-stronger-theory-intro)
      ultimately show ?thesis
        using stronger-theory-transitive by blast
    qed
  }
  then show ?case by blast
 qed
 thus ?thesis by blast
qed

lemma (in classical-logic) mergeWitness-segmented-deduction-intro:
 assumes mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
Ψ))
     and map (uncurry (→)) Δ @ (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ) ⊖
map snd Δ $⊢ Φ
         (is ?Γ₀ $⊢ Φ)
   shows map (uncurry (→)) (𝔍 Ψ Δ) @ Γ ⊖ map snd (𝔍 Ψ Δ) $⊢ Φ
         (is ?Γ $⊢ Φ)
proof -
 let ?Σ = 𝔅 Ψ Δ
 let ?A = map (uncurry (→)) Δ
 let ?B = map (uncurry (→)) Ψ
 let ?C = map snd ?Σ
 let ?D = Γ ⊖ (map snd Ψ)
 let ?E = map snd (Δ ⊖ ?Σ)
 have Σ: mset ?Σ ⊆# mset Δ
         mset ?C ⊆# mset ?B
         mset ?E ⊆# mset ?D
   using assms(1)
         secondComponent-msub
         secondComponent-snd-projection-msub
         secondComponent-diff-msub
   by simp+
```

moreover
from *calculation* **have** *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$\subseteq\#$ *mset* Γ $-$ *image-mset snd* (*mset* Ψ)
by *simp*
hence *mset* Γ $-$ *image-mset snd* (*mset* Ψ)
$-$ *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$+$ *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$=$ *mset* Γ $-$ *image-mset snd* (*mset* Ψ)
using *subset-mset.diff-add* **by** *blast*
then have *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$+$ ({#$x \rightarrow y$. (x, y) $\in\#$ *mset* Ψ#}
$+$ (*mset* Γ $-$ (*image-mset snd* (*mset* Ψ)
$+$ *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ)))))
$=$ {#$x \rightarrow y$. (x, y) $\in\#$ *mset* Ψ#} $+$ (*mset* Γ $-$ *image-mset snd* (*mset*
Ψ))
by (*simp add*: *union-commute*)
with *calculation* **have** *mset* $?\Gamma_0$ $=$ *mset* ($?A$ @ ($?B \ominus ?C$) @ ($?D \ominus ?E$))
by (*simp*, *metis* (*no-types*) *add-diff-cancel-left image-mset-union subset-mset.diff-add*)
moreover have ($?A$ @ ($?B \ominus ?C$)) $\preceq$ *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{J}$ Ψ Δ)
using *secondComponent-mergeWitness-stronger-theory* **by** *simp*
moreover have *mset* ($?D \ominus ?E$) $=$ *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{J}$ Ψ Δ))
using *secondComponent-mergeWitness-snd-projection*
by *simp*
with *calculation* **have** ($?A$ @ ($?B \ominus ?C$) @ ($?D \ominus ?E$)) $\preceq$ $?\Gamma$
by (*metis* (*no-types*, *lifting*)
stronger-theory-combine
append.assoc
list-subtract-mset-homomorphism
msub-stronger-theory-intro
map-list-subtract-mset-containment
map-list-subtract-mset-equivalence
mset-subset-eq-add-right
subset-mset.add-diff-inverse
subset-mset.diff-add-assoc2)
ultimately have $?\Gamma_0 \preceq ?\Gamma$
unfolding *stronger-theory-relation-alt-def*
by *simp*
thus *?thesis*
using *assms*(*2*) *segmented-stronger-theory-left-monotonic*
by *blast*
qed

lemma (**in** *classical-logic*) *segmented-formula-right-split*:
Γ $\$\vdash$ (φ # Φ) $=$ Γ $\$\vdash$ ($\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Φ)
proof (*rule iffI*)
assume Γ $\$\vdash$ (φ # Φ)
from *this* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq\#$ *mset* Γ
map (*uncurry* ($\sqcup$)) Ψ :$\vdash$ φ

```
  (map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)) $⊢ Φ
  by auto
 let ?Ψ₁ = zip (map (λ (χ,γ). ψ ⊔ χ) Ψ) (map snd Ψ)
 let ?Γ₁ = map (uncurry (→)) ?Ψ₁ @ Γ ⊖ (map snd ?Ψ₁)
 let ?Ψ₂ = zip (map (λ (χ,γ). ψ → χ) Ψ) (map (uncurry (→)) ?Ψ₁)
 let ?Γ₂ = map (uncurry (→)) ?Ψ₂ @ ?Γ₁ ⊖ (map snd ?Ψ₂)
 have map (uncurry (→)) Ψ ≼ map (uncurry (→)) ?Ψ₂
 proof (induct Ψ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Ψ)
  let ?χ = fst δ
  let ?γ = snd δ
  let ?Ψ₁ = zip (map (λ (χ,γ). ψ ⊔ χ) Ψ) (map snd Ψ)
  let ?Ψ₂ = zip (map (λ (χ,γ). ψ → χ) Ψ) (map (uncurry (→)) ?Ψ₁)
  let ?T₁ = λ Ψ. map (uncurry (→)) (zip (map (λ (χ,γ). ψ ⊔ χ) Ψ) (map snd
Ψ))
  let ?T₂ = λ Ψ. map (uncurry (→)) (zip (map (λ (χ,γ). ψ → χ) Ψ) (?T₁ Ψ))
  {
   fix δ :: 'a × 'a
   have (λ (χ,γ). ψ ⊔ χ) = (λ δ. ψ ⊔ (fst δ))
        (λ (χ,γ). ψ → χ) = (λ δ. ψ → (fst δ))
    by fastforce+
   note functional-identities = this
   have (λ (χ,γ). ψ ⊔ χ) δ = ψ ⊔ (fst δ)
        (λ (χ,γ). ψ → χ) δ = ψ → (fst δ)
    by (simp add: functional-identities)+
  }
  hence ?T₂ (δ # Ψ) = ((ψ → ?χ) → (ψ ⊔ ?χ) → ?γ) # (map (uncurry (→))
?Ψ₂)
   by simp
  moreover have map (uncurry (→)) (δ # Ψ) = (?χ → ?γ) # map (uncurry
(→)) Ψ
   by (simp add: case-prod-beta)
  moreover
  {
   fix χ ψ γ
   have ⊢ ((ψ → χ) → (ψ ⊔ χ) → γ) ↔ (χ → γ)
   proof -
    have ∀ 𝔐. 𝔐 ⊨_prop ((⟨ψ⟩ → ⟨χ⟩) → (⟨ψ⟩ ⊔ ⟨χ⟩) → ⟨γ⟩) ↔ (⟨χ⟩ → ⟨γ⟩)
     by fastforce
    hence ⊢ ⦇ ((⟨ψ⟩ → ⟨χ⟩) → (⟨ψ⟩ ⊔ ⟨χ⟩) → ⟨γ⟩) ↔ (⟨χ⟩ → ⟨γ⟩) ⦈
     using propositional-semantics by blast
    thus ?thesis by simp
   qed
  }
  hence identity: ⊢ ((ψ → ?χ) → (ψ ⊔ ?χ) → ?γ) → (?χ → ?γ)
   using biconditional-def by auto
```

assume $map\ (uncurry\ (\rightarrow))\ \Psi \preceq map\ (uncurry\ (\rightarrow))\ ?\Psi_2$
with *identity* **have** $((?\chi \rightarrow ?\gamma)\ \#\ map\ (uncurry\ (\rightarrow))\ \Psi) \preceq$
$(((\psi \rightarrow ?\chi) \rightarrow (\psi \sqcup ?\chi) \rightarrow ?\gamma)\ \#\ (map\ (uncurry\ (\rightarrow))\ ?\Psi_2))$
using *stronger-theory-left-right-cons* **by** *blast*
ultimately show *?case* **by** *simp*
qed
hence $(map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi)) \preceq$
$((map\ (uncurry\ (\rightarrow))\ ?\Psi_2)\ @\ \Gamma \ominus (map\ snd\ \Psi))$
using *stronger-theory-combine stronger-theory-reflexive* **by** *blast*
moreover have $mset\ ?\Gamma_2 = mset\ ((map\ (uncurry\ (\rightarrow))\ ?\Psi_2)\ @\ \Gamma \ominus (map\ snd$
$?\Psi_1))$
by *simp*
ultimately have $(map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi)) \preceq ?\Gamma_2$
by (*simp add*: *stronger-theory-relation-def*)
hence $?\Gamma_2\ \$\vdash\ \Phi$
using $\Psi(3)$ *segmented-stronger-theory-left-monotonic* **by** *blast*
moreover
have $(map\ (uncurry\ (\sqcup))\ ?\Psi_2) :\vdash \psi \rightarrow \varphi$
proof –
let $?\Gamma = map\ (\lambda\ (\chi, \gamma).\ (\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma)\ \Psi$
let $?\Sigma = map\ (\lambda\ (\chi, \gamma).\ (\psi \rightarrow (\chi \sqcup \gamma)))\ \Psi$
have $map\ (uncurry\ (\sqcup))\ ?\Psi_2 = ?\Gamma$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\chi\ \Psi$)
have $(\lambda\ \varphi.\ (case\ \varphi\ of\ (\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup (case\ \varphi\ of\ (\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow$
$snd\ \varphi) =$
$(\lambda\ \varphi.\ (case\ \varphi\ of\ (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma))$
by *fastforce*
hence $(case\ \chi\ of\ (\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup (case\ \chi\ of\ (\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow snd\ \chi$
$=$
$(case\ \chi\ of\ (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma)$
by *metis*
with *Cons* **show** *?case* **by** *simp*
qed
moreover have $?\Sigma \preceq ?\Gamma$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\delta\ \Psi$)
let $?\alpha = (\lambda\ (\chi, \gamma).\ (\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma)\ \delta$
let $?\beta = (\lambda\ (\chi, \gamma).\ (\psi \rightarrow (\chi \sqcup \gamma)))\ \delta$
let $?\chi = fst\ \delta$
let $?\gamma = snd\ \delta$
have $(\lambda\ \delta.\ (case\ \delta\ of\ (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma)) =$
$(\lambda\ \delta.\ \psi \rightarrow fst\ \delta \sqcup (\psi \sqcup fst\ \delta) \rightarrow snd\ \delta)$

$(\lambda\ \delta.\ (case\ \delta\ of\ (\chi,\ \gamma) \Rightarrow \psi \to (\chi \sqcup \gamma))) = (\lambda\ \delta.\ \psi \to (fst\ \delta \sqcup snd\ \delta))$
by *fastforce+*
hence $?\alpha = (\psi \to ?\chi) \sqcup (\psi \sqcup ?\chi) \to ?\gamma$
$?\beta = \psi \to (?\chi \sqcup ?\gamma)$
by *metis+*
moreover
{
fix $\psi\ \chi\ \gamma$
have $\vdash ((\psi \to \chi) \sqcup (\psi \sqcup \chi) \to \gamma) \to (\psi \to (\chi \sqcup \gamma))$
proof $-$
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\psi\rangle \to \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \to (\langle\psi\rangle \to (\langle\chi\rangle$
$\sqcup \langle\gamma\rangle)))$
by *fastforce*
hence $\vdash (\!|\ (((\langle\psi\rangle \to \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \to (\langle\psi\rangle \to (\langle\chi\rangle \sqcup \langle\gamma\rangle)))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
ultimately have $\vdash ?\alpha \to ?\beta$ **by** *simp*
thus *?case*
using *Cons*
stronger-theory-left-right-cons
by *simp*
qed
moreover have $\forall\ \varphi.\ (map\ (uncurry\ (\sqcup))\ \Psi) :\vdash \varphi \longrightarrow ?\Sigma :\vdash \psi \to \varphi$
proof (*induct* Ψ)
case *Nil*
then show *?case*
using *axiom-k modus-ponens*
by *fastforce*
next
case (*Cons* $\delta\ \Psi$)
let $?\delta' = (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta$
let $?\Sigma = map\ (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \Psi$
let $?\Sigma' = map\ (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ (\delta\ \#\ \Psi)$
{
fix φ
assume $map\ (uncurry\ (\sqcup))\ (\delta\ \#\ \Psi) :\vdash \varphi$
hence $map\ (uncurry\ (\sqcup))\ \Psi :\vdash (uncurry\ (\sqcup))\ \delta \to \varphi$
using *list-deduction-theorem*
by *simp*
hence $?\Sigma :\vdash \psi \to (uncurry\ (\sqcup))\ \delta \to \varphi$
using *Cons*
by *blast*
moreover
{
fix $\alpha\ \beta\ \gamma$
have $\vdash (\alpha \to \beta \to \gamma) \to ((\alpha \to \beta) \to \alpha \to \gamma)$
using *axiom-s* **by** *auto*

}
ultimately have $?\Sigma :\vdash (\psi \to (\mathit{uncurry}\ (\sqcup))\ \delta) \to \psi \to \varphi$
using *list-deduction-weaken* [**where** $?\Gamma{=}?\Sigma$]
list-deduction-modus-ponens [**where** $?\Gamma{=}?\Sigma$]
by *metis*
moreover
have $(\lambda\ \delta.\ \psi \to (\mathit{uncurry}\ (\sqcup))\ \delta) = (\lambda\ \delta.\ (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta)$
by *fastforce*
ultimately have $?\Sigma :\vdash (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta \to \psi \to \varphi$
by *metis*
hence $?\Sigma' :\vdash \psi \to \varphi$
using *list-deduction-theorem*
by *simp*
}
then show *?case* **by** *simp*
qed
with $\Psi(2)$ **have** $?\Sigma :\vdash \psi \to \varphi$
by *blast*
ultimately show *?thesis*
using *stronger-theory-deduction-monotonic* **by** *auto*
qed
moreover have $\mathit{mset}\ (\mathit{map}\ \mathit{snd}\ ?\Psi_2) \subseteq\#\ \mathit{mset}\ ?\Gamma_1$ **by** *simp*
ultimately have $?\Gamma_1\ \$\vdash (\psi \to \varphi\ \#\ \Phi)$ **using** *segmented-deduction.simps(2)* **by** *blast*
moreover have $\vdash (\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Psi :\to \varphi) \to (\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ ?\Psi_1) :\to (\psi \sqcup \varphi)$
proof (*induct* Ψ)
case *Nil*
then show *?case*
unfolding *disjunction-def*
using *axiom-k modus-ponens*
by *fastforce*
next
case (*Cons* $\nu\ \Psi$)
let $?\Delta = \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Psi$
let $?\Delta' = \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\nu\ \#\ \Psi)$
let $?\Sigma = \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathit{zip}\ (\mathit{map}\ (\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi)\ (\mathit{map}\ \mathit{snd}\ \Psi))$
let $?\Sigma' = \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathit{zip}\ (\mathit{map}\ (\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ (\nu\ \#\ \Psi))\ (\mathit{map}\ \mathit{snd}\ (\nu\ \#\ \Psi)))$
have $\vdash (?\Delta' :\to \varphi) \to (\mathit{uncurry}\ (\sqcup))\ \nu \to ?\Delta :\to \varphi$
by (*simp*, *metis axiom-k axiom-s modus-ponens*)
with *Cons* **have** $\vdash (?\Delta' :\to \varphi) \to (\mathit{uncurry}\ (\sqcup))\ \nu \to ?\Sigma :\to (\psi \sqcup \varphi)$
using *hypothetical-syllogism modus-ponens*
by *blast*
hence $(?\Delta' :\to \varphi)\ \#\ ((\mathit{uncurry}\ (\sqcup))\ \nu)\ \#\ ?\Sigma :\vdash \psi \sqcup \varphi$
by (*simp add*: *list-deduction-def*)
moreover have $\mathit{set}\ ((?\Delta' :\to \varphi)\ \#\ ((\mathit{uncurry}\ (\sqcup))\ \nu)\ \#\ ?\Sigma) =$
$\mathit{set}\ (((\mathit{uncurry}\ (\sqcup))\ \nu)\ \#\ (?\Delta' :\to \varphi)\ \#\ ?\Sigma)$
by *fastforce*

ultimately have $((uncurry\ (\sqcup))\ \nu)\ \#\ (?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash \psi \sqcup \varphi$
  using *list-deduction-monotonic* **by** *blast*
hence $(?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash ((uncurry\ (\sqcup))\ \nu) \rightarrow (\psi \sqcup \varphi)$
  using *list-deduction-theorem*
  by *simp*
moreover
let $?\chi = fst\ \nu$
let $?\gamma = snd\ \nu$
have $(\lambda\ \nu\ .\ (uncurry\ (\sqcup))\ \nu) = (\lambda\ \nu.\ fst\ \nu \sqcup snd\ \nu)$
  by *fastforce*
hence $(uncurry\ (\sqcup))\ \nu = ?\chi \sqcup ?\gamma$ **by** *simp*
ultimately have $(?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash (?\chi \sqcup ?\gamma) \rightarrow (\psi \sqcup \varphi)$ **by** *simp*
moreover
{
  fix $\alpha\ \beta\ \delta\ \gamma$
  have $\vdash ((\beta \sqcup \alpha) \rightarrow (\gamma \sqcup \delta)) \rightarrow ((\gamma \sqcup \beta) \sqcup \alpha) \rightarrow (\gamma \sqcup \delta)$
  proof −
    have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\beta\rangle \sqcup \langle\alpha\rangle) \rightarrow (\langle\gamma\rangle \sqcup \langle\delta\rangle)) \rightarrow (((\langle\gamma\rangle \sqcup \langle\beta\rangle) \sqcup \langle\alpha\rangle) \rightarrow (\langle\gamma\rangle \sqcup \langle\delta\rangle)$
      by *fastforce*
    hence $\vdash (\!|\ (((\langle\beta\rangle \sqcup \langle\alpha\rangle) \rightarrow (\langle\gamma\rangle \sqcup \langle\delta\rangle)) \rightarrow (((\langle\gamma\rangle \sqcup \langle\beta\rangle) \sqcup \langle\alpha\rangle) \rightarrow (\langle\gamma\rangle \sqcup \langle\delta\rangle)\ |\!)$
      using *propositional-semantics* **by** *blast*
    thus *?thesis* **by** *simp*
  qed
}
hence $(?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash ((?\chi \sqcup ?\gamma) \rightarrow (\psi \sqcup \varphi)) \rightarrow ((\psi \sqcup ?\chi) \sqcup ?\gamma) \rightarrow (\psi \sqcup \varphi)$
  using *list-deduction-weaken* **by** *blast*
ultimately have $(?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash ((\psi \sqcup ?\chi) \sqcup ?\gamma) \rightarrow (\psi \sqcup \varphi)$
  using *list-deduction-modus-ponens* **by** *blast*
hence $((\psi \sqcup ?\chi) \sqcup ?\gamma)\ \#\ (?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma :\vdash \psi \sqcup \varphi$
  using *list-deduction-theorem*
  by *simp*
moreover have $set\ (((\psi \sqcup ?\chi) \sqcup ?\gamma)\ \#\ (?\Delta' :\rightarrow\ \varphi)\ \#\ ?\Sigma) =$
    $set\ ((?\Delta' :\rightarrow\ \varphi)\ \#\ ((\psi \sqcup ?\chi) \sqcup ?\gamma)\ \#\ ?\Sigma)$
  by *fastforce*
moreover have
  $map\ (uncurry\ (\sqcup))\ (\nu\ \#\ \Psi) :\rightarrow \varphi$
  $\#\ (\psi \sqcup fst\ \nu) \sqcup snd\ \nu$
  $\#\ map\ (uncurry\ (\sqcup))\ (zip\ (map\ (\lambda(\text{-},\ a).\ \psi \sqcup a)\ \Psi)\ (map\ snd\ \Psi)) :\vdash (\psi \sqcup fst\ \nu) \sqcup snd\ \nu$
  by (*meson list.set-intros*(*1*)
    list-deduction-monotonic
    list-deduction-reflection
    set-subset-Cons)
ultimately have $(?\Delta' :\rightarrow\ \varphi)\ \#\ ((\psi \sqcup ?\chi) \sqcup ?\gamma)\ \#\ ?\Sigma :\vdash \psi \sqcup \varphi$
  using *list-deduction-modus-ponens list-deduction-monotonic* **by** *blast*
moreover

```
    have (λ ν. ψ ⊔ fst ν) = (λ (χ, γ). ψ ⊔ χ)
      by fastforce
    hence ψ ⊔ fst ν = (λ (χ, γ). ψ ⊔ χ) ν
      by metis
    hence ((ψ ⊔ ?χ) ⊔ ?γ) # ?Σ = ?Σ'
      by simp
    ultimately have (?Δ' :→ φ) # ?Σ' :⊢ ψ ⊔ φ by simp
    then show ?case by (simp add: list-deduction-def)
  qed
  with Ψ(2) have map (uncurry (⊔)) ?Ψ₁ :⊢ (ψ ⊔ φ)
    unfolding list-deduction-def
    using modus-ponens
    by blast
  moreover have mset (map snd ?Ψ₁) ⊆# mset Γ using Ψ(1) by simp
  ultimately show Γ $⊢ (ψ ⊔ φ # ψ → φ # Φ)
    using segmented-deduction.simps(2) by blast
next
  assume Γ $⊢ (ψ ⊔ φ # ψ → φ # Φ)
  from this obtain Ψ where Ψ:
    mset (map snd Ψ) ⊆# mset Γ
    map (uncurry (⊔)) Ψ :⊢ ψ ⊔ φ
    map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ) $⊢ (ψ → φ # Φ)
    using segmented-deduction.simps(2) by blast
  let ?Γ' = map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)
  from Ψ obtain Δ where Δ:
    mset (map snd Δ) ⊆# mset ?Γ'
    map (uncurry (⊔)) Δ :⊢ ψ → φ
    (map (uncurry (→)) Δ @ ?Γ' ⊖ (map snd Δ)) $⊢ Φ
    using segmented-deduction.simps(2) by blast
  let ?Ω = 𝔍 Ψ Δ
  have mset (map snd ?Ω) ⊆# mset Γ
    using Δ(1) Ψ(1) mergeWitness-msub-intro
    by blast
  moreover have map (uncurry (⊔)) ?Ω :⊢ φ
  proof −
    have map (uncurry (⊔)) ?Ω :⊢ ψ ⊔ φ
         map (uncurry (⊔)) ?Ω :⊢ ψ → φ
      using Ψ(2) Δ(2)
            stronger-theory-deduction-monotonic
            right-mergeWitness-stronger-theory
            left-mergeWitness-stronger-theory
      by blast+
    moreover
    have ⊢ (ψ ⊔ φ) → (ψ → φ) → φ
      unfolding disjunction-def
      using modus-ponens excluded-middle-elimination flip-implication
      by blast
    ultimately show ?thesis
      using list-deduction-weaken list-deduction-modus-ponens
```

    by *blast*
  qed
  moreover have *map* (*uncurry* ($\rightarrow$)) *?*Ω @ $\Gamma \ominus$ (*map snd ?*Ω) \$$\vdash$ Φ
    using Δ(*1*) Δ(*3*) Ψ(*1*) *mergeWitness-segmented-deduction-intro* **by** *blast*
  ultimately show Γ \$$\vdash$ (φ # Φ)
    using *segmented-deduction.simps*(*2*) **by** *blast*
qed

primrec (**in** *implication-logic*)
  XWitness :: ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* ($\mathfrak{X}$)
  where
    $\mathfrak{X}$ Ψ [] = []
  | $\mathfrak{X}$ Ψ (δ # Δ) =
      (*case find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
          None $\Rightarrow$ δ # $\mathfrak{X}$ Ψ Δ
        | *Some* ψ $\Rightarrow$ (*fst* ψ $\rightarrow$ *fst* δ, *snd* ψ) # ($\mathfrak{X}$ (*remove1* ψ Ψ) Δ))

primrec (**in** *implication-logic*)
  XComponent :: ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* ($\mathfrak{X}_\bullet$)
  where
    $\mathfrak{X}_\bullet$ Ψ [] = []
  | $\mathfrak{X}_\bullet$ Ψ (δ # Δ) =
      (*case find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
          None $\Rightarrow$ $\mathfrak{X}_\bullet$ Ψ Δ
        | *Some* ψ $\Rightarrow$ (*fst* ψ $\rightarrow$ *fst* δ, *snd* ψ) # ($\mathfrak{X}_\bullet$ (*remove1* ψ Ψ) Δ))

primrec (**in** *implication-logic*)
  YWitness :: ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* ($\mathfrak{Y}$)
  where
    $\mathfrak{Y}$ Ψ [] = Ψ
  | $\mathfrak{Y}$ Ψ (δ # Δ) =
      (*case find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
          None $\Rightarrow$ $\mathfrak{Y}$ Ψ Δ
        | *Some* ψ $\Rightarrow$ (*fst* ψ, (*fst* ψ $\rightarrow$ *fst* δ) $\rightarrow$ *snd* ψ) #
                ($\mathfrak{Y}$ (*remove1* ψ Ψ) Δ))

primrec (**in** *implication-logic*)
  YComponent :: ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* ($\mathfrak{Y}_\bullet$)
  where
    $\mathfrak{Y}_\bullet$ Ψ [] = []
  | $\mathfrak{Y}_\bullet$ Ψ (δ # Δ) =
      (*case find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
          None $\Rightarrow$ $\mathfrak{Y}_\bullet$ Ψ Δ
        | *Some* ψ $\Rightarrow$ (*fst* ψ, (*fst* ψ $\rightarrow$ *fst* δ) $\rightarrow$ *snd* ψ) #
                ($\mathfrak{Y}_\bullet$ (*remove1* ψ Ψ) Δ))

lemma (**in** *implication-logic*) *XWitness-right-empty* [*simp*]:
  $\mathfrak{X}$ [] Δ = Δ
  by (*induct* Δ, *simp*+)

lemma (**in** *implication-logic*) *YWitness-right-empty* [*simp*]:
$\mathfrak{Y}$ [] Δ = []
by (*induct* Δ, *simp*+)

lemma (**in** *implication-logic*) *XWitness-map-snd-decomposition*:
mset (*map snd* ($\mathfrak{X}$ Ψ Δ)) = *mset* (*map snd* (($\mathfrak{A}$ Ψ Δ) @ ($\Delta \ominus$ ($\mathfrak{B}$ Ψ Δ))))
proof −
have $\forall\,\Psi$. *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ)) = *mset* (*map snd* (($\mathfrak{A}$ Ψ Δ) @ ($\Delta \ominus$ ($\mathfrak{B}$ Ψ Δ))))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* ($\mathfrak{X}$ Ψ (δ # Δ)))
= *mset* (*map snd* ($\mathfrak{A}$ Ψ (δ # Δ) @ (δ # Δ) $\ominus$ $\mathfrak{B}$ Ψ (δ # Δ)))
using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*,
simp,
metis (*no-types*, *lifting*)
add-mset-add-single
image-mset-single
image-mset-union
mset-subset-eq-multiset-union-diff-commute
secondComponent-msub,
fastforce)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *YWitness-map-snd-decomposition*:
mset (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* (($\Psi \ominus$ ($\mathfrak{A}$ Ψ Δ)) @ ($\mathfrak{Y}_\bullet$ Ψ Δ)))
proof −
have $\forall$ Ψ. *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* (($\Psi \ominus$ ($\mathfrak{A}$ Ψ Δ)) @ ($\mathfrak{Y}_\bullet$ Ψ Δ)))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* ($\mathfrak{Y}$ Ψ (δ # Δ))) = *mset* (*map snd* ($\Psi \ominus \mathfrak{A}$ Ψ (δ # Δ) @ $\mathfrak{Y}_\bullet$ Ψ (δ # Δ)))

using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\to$)) ψ = *snd* δ) Ψ = *None*, *fastforce*+)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *XWitness-msub*:
assumes *mset* (*map snd* Ψ) $\subseteq\#$ *mset* Γ
and *mset* (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
shows *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
proof −
have *mset* (*map snd* ($\Delta \ominus$ ($\mathfrak{B}$ Ψ Δ))) $\subseteq\#$ *mset* ($\Gamma \ominus$ (*map snd* Ψ))
using *assms secondComponent-diff-msub* **by** *blast*
moreover have *mset* (*map snd* ($\mathfrak{A}$ Ψ Δ)) $\subseteq\#$ *mset* (*map snd* Ψ)
using *firstComponent-msub*
by (*simp add*: *image-mset-subseteq-mono*)
moreover have *mset* ((*map snd* Ψ) @ ($\Gamma \ominus$ *map snd* Ψ)) = *mset* Γ
using *assms*(*1*)
by *simp*
moreover have *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* ($\Delta \ominus \mathfrak{B}$ Ψ Δ))
= *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ))
using *XWitness-map-snd-decomposition* **by** *force*
ultimately
show *?thesis*
by (*metis* (*no-types*) *mset-append mset-map subset-mset.add-mono*)
qed

lemma (**in** *implication-logic*) *YComponent-msub*:
mset (*map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{X}$ Ψ Δ))
proof −
have $\forall$ Ψ. *mset* (*map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{X}$ Ψ Δ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* ($\mathfrak{Y}_\bullet$ Ψ (δ # Δ))) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{X}$ Ψ (δ # Δ)))
using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\to$)) ψ = *snd* δ) Ψ = *None*,
simp, *metis add-mset-add-single*
mset-subset-eq-add-left
subset-mset.order-trans,

fastforce)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *YWitness-msub*:
assumes *mset* (*map snd* Ψ) $\subseteq\#$ *mset* Γ
and *mset* (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
shows *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ)) $\subseteq\#$
mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}$ Ψ Δ) @ $\Gamma \ominus$ *map snd* ($\mathfrak{X}$ Ψ Δ))
proof −
have *A*: *image-mset snd* (*mset* Ψ) $\subseteq\#$ *mset* Γ **using** *assms* **by** *simp*
have *B*: *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
using *A XWitness-map-snd-decomposition assms*(*2*) *XWitness-msub* **by** *auto*
have *mset* Γ − *image-mset snd* (*mset* Ψ) = *mset* ($\Gamma \ominus$ *map snd* Ψ)
by *simp*
then have *C*: *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ Ψ Δ)) + *image-mset snd* (*mset* Ψ) $\subseteq\#$ *mset* Γ
using *A* **by** (*metis* (*full-types*) *assms*(*2*) *secondComponent-diff-msub subset-mset.le-diff-conv2*)
have *image-mset snd* (*mset* ($\Psi \ominus \mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) = *image-mset snd* (*mset* Ψ)
by (*metis* (*no-types*) *image-mset-union*
list-subtract-mset-homomorphism
firstComponent-msub
subset-mset.diff-add)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}$ Ψ Δ))
+ (*image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)))
= *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ Ψ Δ)) + *image-mset snd* (*mset* Ψ)
by (*simp add*: *union-commute*)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}$ Ψ Δ))
$\subseteq\#$ *mset* Γ − (*image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)))
by (*metis* (*no-types*) *B C subset-mset.le-diff-conv2*)
hence *mset* (*map snd* ($\Psi \ominus \mathfrak{A}$ Ψ Δ)) $\subseteq\#$ *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{X}$ Ψ Δ))
using *assms XWitness-map-snd-decomposition*
by *simp*
thus *?thesis*
using *YComponent-msub*
YWitness-map-snd-decomposition
by (*simp add*: *mset-subset-eq-mono-add union-commute*)
qed

lemma (**in** *classical-logic*) *XWitness-right-stronger-theory*:
map (*uncurry* ($\sqcup$)) $\Delta \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{X}$ Ψ Δ)

proof −
  have $\forall\ \Psi.$ *map* (*uncurry* ($\sqcup$)) $\Delta \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{X}\ \Psi\ \Delta$)
  proof (*induct* Δ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* δ Δ)
    {
      fix Ψ
      have *map* (*uncurry* ($\sqcup$)) (δ # Δ) $\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{X}\ \Psi$ (δ # Δ))
      proof (*cases find* ($\lambda\ \psi.$ (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
        case *True*
        then show *?thesis*
          using *Cons*
          by (*simp add*: *stronger-theory-left-right-cons*
                    trivial-implication)
      next
        case *False*
        from *this* **obtain** ψ **where**
          ψ: *find* ($\lambda\psi.$ *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
            $\psi \in$ *set* Ψ
            (*fst* $\psi \rightarrow$ *snd* ψ) = *snd* δ
          using *find-Some-set-membership*
              find-Some-predicate
          by *fastforce*
        let *?*Ψ' = *remove1* ψ Ψ
        let *?*α = *fst* ψ
        let *?*β = *snd* ψ
        let *?*γ = *fst* δ
        have *map* (*uncurry* ($\sqcup$)) $\Delta \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{X}$ *?*Ψ' Δ)
          using *Cons* **by** *simp*
        moreover
        have (*uncurry* ($\sqcup$)) = ($\lambda\ \delta.$ *fst* $\delta \sqcup$ *snd* δ) **by** *fastforce*
        hence (*uncurry* ($\sqcup$)) δ = *?*$\gamma \sqcup$ (*?*$\alpha \rightarrow$ *?*β) **using** ψ(*3*) **by** *fastforce*
        moreover
        {
          fix $\alpha\ \beta\ \gamma$
          have $\vdash (\alpha \rightarrow \gamma \sqcup \beta) \rightarrow (\gamma \sqcup (\alpha \rightarrow \beta))$
          proof −
            let *?*$\varphi = (\langle\alpha\rangle \rightarrow \langle\gamma\rangle \sqcup \langle\beta\rangle) \rightarrow (\langle\gamma\rangle \sqcup (\langle\alpha\rangle \rightarrow \langle\beta\rangle))$
            have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
            hence $\vdash (\!|$ *?*φ $|\!)$ **using** *propositional-semantics* **by** *blast*
            thus *?thesis* **by** *simp*
          qed
        }
        hence $\vdash$ (*?*$\alpha \rightarrow$ *?*$\gamma \sqcup$ *?*β) $\rightarrow$ (*?*$\gamma \sqcup$ (*?*$\alpha \rightarrow$ *?*β)) **by** *simp*
        ultimately
        show *?thesis* **using** ψ
          by (*simp add*: *stronger-theory-left-right-cons*)

```
      qed
    }
    then show ?case by simp
  qed
  thus ?thesis by simp
qed

lemma (in classical-logic) YWitness-left-stronger-theory:
  map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) (𝔜 Ψ Δ)
proof −
  have ∀ Ψ. map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) (𝔜 Ψ Δ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) (𝔜 Ψ (δ # Δ))
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        then show ?thesis using Cons by simp
      next
        case False
        from this obtain ψ where
          ψ: find (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
             ψ ∈ set Ψ
             (uncurry (⊔)) ψ = fst ψ ⊔ snd ψ
          using find-Some-set-membership
          by fastforce
        let ?φ = fst ψ ⊔ (fst ψ → fst δ) → snd ψ
        let ?Ψ' = remove1 ψ Ψ
        have map (uncurry (⊔)) ?Ψ' ≼ map (uncurry (⊔)) (𝔜 ?Ψ' Δ)
          using Cons by simp
        moreover
        {
          fix α β γ
          have ⊢ (α ⊔ (α → γ) → β) → (α ⊔ β)
          proof −
            let ?φ = (⟨α⟩ ⊔ (⟨α⟩ → ⟨γ⟩) → ⟨β⟩) → (⟨α⟩ ⊔ ⟨β⟩)
            have ∀𝔐. 𝔐 ⊨prop ?φ by fastforce
            hence ⊢ ⦇ ?φ ⦈ using propositional-semantics by blast
            thus ?thesis by simp
          qed
        }
        hence ⊢ ?φ → (uncurry (⊔)) ψ using ψ(3) by auto
        ultimately
         have map (uncurry (⊔)) (ψ # ?Ψ') ≼ (?φ # map (uncurry (⊔)) (𝔜 ?Ψ'
Δ))
```

```
        by (simp add: stronger-theory-left-right-cons)
      moreover
      from ψ have mset (map (uncurry (⊔)) (ψ # ?Ψ')) = mset (map (uncurry
(⊔)) Ψ)
        by (metis mset-eq-perm mset-map perm-remove)
      ultimately show ?thesis
        using stronger-theory-relation-alt-def ψ(1) by auto
    qed
  }
  then show ?case by blast
 qed
 thus ?thesis by blast
qed

lemma (in implication-logic) XWitness-secondComponent-diff-decomposition:
 mset (𝔛 Ψ Δ) = mset (𝔛• Ψ Δ @ Δ ⊖ 𝔅 Ψ Δ)
proof −
 have ∀ Ψ. mset (𝔛 Ψ Δ) = mset (𝔛• Ψ Δ @ Δ ⊖ 𝔅 Ψ Δ)
 proof (induct Δ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Δ)
  {
   fix Ψ
   have mset (𝔛 Ψ (δ # Δ)) =
        mset (𝔛• Ψ (δ # Δ) @ (δ # Δ) ⊖ 𝔅 Ψ (δ # Δ))
     using Cons
     by (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None,
      simp, metis add-mset-add-single secondComponent-msub subset-mset.diff-add-assoc2,
        fastforce)
  }
  then show ?case by blast
 qed
 thus ?thesis by blast
qed

lemma (in implication-logic) YWitness-firstComponent-diff-decomposition:
 mset (𝔜 Ψ Δ) = mset (Ψ ⊖ 𝔄 Ψ Δ @ 𝔜• Ψ Δ)
proof −
 have ∀ Ψ. mset (𝔜 Ψ Δ) = mset (Ψ ⊖ 𝔄 Ψ Δ @ 𝔜• Ψ Δ)
 proof (induct Δ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Δ)
  {
   fix Ψ
   have mset (𝔜 Ψ (δ # Δ)) =
```

$mset\ (\Psi \ominus \mathfrak{A}\ \Psi\ (\delta\ \#\ \Delta)\ @\ \mathfrak{Y}_{\bullet}\ \Psi\ (\delta\ \#\ \Delta))$
using *Cons*
by (*cases find* ($\lambda\ \psi$. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*, *simp*, *fastforce*)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *YWitness-right-stronger-theory*:
map (*uncurry* ($\rightarrow$)) $\Delta \preceq$ *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{Y}\ \Psi\ \Delta \ominus (\Psi \ominus \mathfrak{A}\ \Psi\ \Delta)\ @\ (\Delta \ominus \mathfrak{B}\ \Psi\ \Delta)$)
proof −
let $?\mathfrak{f} = \lambda\Psi\ \Delta.\ (\Psi \ominus \mathfrak{A}\ \Psi\ \Delta)$
let $?\mathfrak{g} = \lambda\ \Psi\ \Delta.\ (\Delta \ominus \mathfrak{B}\ \Psi\ \Delta)$
have $\forall\ \Psi$. *map* (*uncurry* ($\rightarrow$)) $\Delta \preceq$ *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{Y}\ \Psi\ \Delta \ominus ?\mathfrak{f}\ \Psi\ \Delta\ @\ ?\mathfrak{g}\ \Psi\ \Delta$)
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
let $?\delta$ = (*uncurry* ($\rightarrow$)) δ
{
fix Ψ
have *map* (*uncurry* ($\rightarrow$)) ($\delta\ \#\ \Delta$)
$\preceq$ *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{Y}\ \Psi\ (\delta\ \#\ \Delta) \ominus ?\mathfrak{f}\ \Psi\ (\delta\ \#\ \Delta)\ @\ ?\mathfrak{g}\ \Psi\ (\delta\ \#\ \Delta)$)
proof (*cases find* ($\lambda\ \psi$. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
case *True*
moreover
from *Cons* **have**
map (*uncurry* ($\rightarrow$)) ($\delta\ \#\ \Delta$) $\preceq$ *map* (*uncurry* ($\rightarrow$)) ($\delta\ \#\ \mathfrak{Y}\ \Psi\ \Delta \ominus ?\mathfrak{f}\ \Psi\ \Delta\ @\ ?\mathfrak{g}\ \Psi\ \Delta$)
by (*simp add*: *stronger-theory-left-right-cons trivial-implication*)
moreover
have *mset* (*map* (*uncurry* ($\rightarrow$)) ($\delta\ \#\ \mathfrak{Y}\ \Psi\ \Delta \ominus ?\mathfrak{f}\ \Psi\ \Delta\ @\ ?\mathfrak{g}\ \Psi\ \Delta$))
= *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{Y}\ \Psi\ \Delta \ominus ?\mathfrak{f}\ \Psi\ \Delta\ @\ ((\delta\ \#\ \Delta) \ominus \mathfrak{B}\ \Psi\ \Delta)$))
by (*simp*,
metis (*no-types*, *lifting*)
add-mset-add-single
image-mset-single
image-mset-union
secondComponent-msub
mset-subset-eq-multiset-union-diff-commute)
moreover have
$\forall\Psi\ \Phi.\ \Psi \preceq \Phi$
= ($\exists\Sigma$. *map snd* $\Sigma = \Psi$
$\wedge$ *mset* (*map fst* Σ) $\subseteq\#$ *mset* Φ
$\wedge$ ($\forall\xi.\ \xi \notin$ *set* $\Sigma \vee \vdash$ (*uncurry* ($\rightarrow$) ξ)))

by (*simp add*: *Ball-def-raw stronger-theory-relation-def*)
moreover have
$((\mathit{uncurry}\ (\rightarrow)\ \delta)\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Delta)$
$\preceq ((\mathit{uncurry}\ (\rightarrow)\ \delta)\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathfrak{Y}\ \Psi\ \Delta \ominus (?\mathfrak{f}\ \Psi\ \Delta))$
$@\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (?\mathfrak{g}\ \Psi\ \Delta))$
using *calculation* **by** *auto*
ultimately show *?thesis*
by (*simp*, *metis union-mset-add-mset-right*)
next
case *False*
from *this* **obtain** ψ **where**
ψ: *find* $(\lambda\psi.\ \mathit{uncurry}\ (\rightarrow)\ \psi = \mathit{snd}\ \delta)\ \Psi = \mathit{Some}\ \psi$
$\mathit{uncurry}\ (\rightarrow)\ \psi = \mathit{snd}\ \delta$
using *find-Some-predicate*
by *fastforce*
let $?\alpha = \mathit{fst}\ \psi$
let $?\beta = \mathit{fst}\ \delta$
let $?\gamma = \mathit{snd}\ \psi$
have $(\lambda\ \delta.\ \mathit{fst}\ \delta \rightarrow \mathit{snd}\ \delta) = \mathit{uncurry}\ (\rightarrow)$ **by** *fastforce*
hence $?\beta \rightarrow ?\alpha \rightarrow ?\gamma = \mathit{uncurry}\ (\rightarrow)\ \delta$ **using** $\psi(2)$ **by** *metis*
moreover
let $?A = \mathfrak{Y}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta$
let $?B = \mathfrak{A}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta$
let $?C = \mathfrak{B}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta$
let $?D = ?A \ominus ((\mathit{remove1}\ \psi\ \Psi) \ominus ?B)$
have $\mathit{mset}\ ((\mathit{remove1}\ \psi\ \Psi) \ominus ?B) \subseteq\#\ \mathit{mset}\ ?A$
using *YWitness-firstComponent-diff-decomposition* **by** *simp*
hence $\mathit{mset}\ (\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))$
$(((?\alpha,\ (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma)\ \#\ ?A) \ominus \mathit{remove1}\ \psi\ (\Psi \ominus ?B)$
$@\ (\mathit{remove1}\ \delta\ ((\delta\ \#\ \Delta) \ominus ?C))))$
$= \mathit{mset}\ ((?\alpha \rightarrow (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma)\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (?D\ @\ (\Delta \ominus$
$?C)))$
by (*simp*, *metis* (*no-types*, *hide-lams*)
add-mset-add-single
image-mset-add-mset
prod.simps(*2*)
subset-mset.diff-add-assoc2)
moreover
have $\vdash (?\alpha \rightarrow (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma) \rightarrow ?\beta \rightarrow ?\alpha \rightarrow ?\gamma$
proof −
let $?\Gamma = [(?\alpha \rightarrow (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma),\ ?\beta,\ ?\alpha]$
have $?\Gamma :\vdash ?\alpha \rightarrow (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma$
$?\Gamma :\vdash ?\alpha$
by (*simp add*: *list-deduction-reflection*)+
hence $?\Gamma :\vdash (?\alpha \rightarrow ?\beta) \rightarrow ?\gamma$
using *list-deduction-modus-ponens* **by** *blast*
moreover have $?\Gamma :\vdash ?\beta$
by (*simp add*: *list-deduction-reflection*)
hence $?\Gamma :\vdash ?\alpha \rightarrow ?\beta$

using *axiom-k list-deduction-modus-ponens list-deduction-weaken* **by** *blast*
ultimately have *?*Γ :$\vdash$ *?*γ
using *list-deduction-modus-ponens* **by** *blast*
thus *?thesis*
unfolding *list-deduction-def* **by** *simp*
qed
hence (*?*β $\rightarrow$ *?*α $\rightarrow$ *?*γ # *map* (*uncurry* ($\rightarrow$)) Δ) $\preceq$
(*?*α $\rightarrow$ (*?*α $\rightarrow$ *?*β) $\rightarrow$ *?*γ # *map* (*uncurry* ($\rightarrow$)) (*?D* @ (Δ $\ominus$ *?C*)))
using *Cons stronger-theory-left-right-cons* **by** *blast*
ultimately show *?thesis*
using ψ **by** (*simp add*: *stronger-theory-relation-alt-def*)
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *implication-logic*) *xcomponent-ycomponent-connection*:
map (*uncurry* ($\rightarrow$)) ($\mathfrak{X}_\bullet$ Ψ Δ) = *map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)
proof −
have $\forall$ Ψ. *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}_\bullet$ Ψ Δ) = *map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}_\bullet$ Ψ (δ # Δ)) = *map snd* ($\mathfrak{Y}_\bullet$ Ψ (δ # Δ))
using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*, *simp*, *fastforce*)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *xwitness-ywitness-segmented-deduction-intro*:
assumes *mset* (*map snd* Ψ) $\subseteq$# *mset* Γ
and *mset* (*map snd* Δ) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ))
and *map* (*uncurry* ($\rightarrow$)) Δ @ (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ *map snd* Ψ) $\ominus$ *map snd* Δ \$$\vdash$ Φ
(**is** *?*Γ_0 \$$\vdash$ Φ)
shows *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{Y}$ Ψ Δ) @
(*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}$ Ψ Δ) @ Γ $\ominus$ *map snd* ($\mathfrak{X}$ Ψ Δ)) $\ominus$
map snd ($\mathfrak{Y}$ Ψ Δ) \$$\vdash$ Φ
(**is** *?*Γ \$$\vdash$ Φ)

proof −
let $?A = map\ (uncurry\ (\rightarrow))\ (\mathfrak{Y}\ \Psi\ \Delta)$
let $?B = map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}\ \Psi\ \Delta)$
let $?C = \Psi \ominus \mathfrak{A}\ \Psi\ \Delta$
let $?D = map\ (uncurry\ (\rightarrow))\ ?C$
let $?E = \Delta \ominus \mathfrak{B}\ \Psi\ \Delta$
let $?F = map\ (uncurry\ (\rightarrow))\ ?E$
let $?G = map\ snd\ (\mathfrak{B}\ \Psi\ \Delta)$
let $?H = map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}_\bullet\ \Psi\ \Delta)$
let $?I = \mathfrak{A}\ \Psi\ \Delta$
let $?J = map\ snd\ (\mathfrak{X}\ \Psi\ \Delta)$
let $?K = map\ snd\ (\mathfrak{Y}\ \Psi\ \Delta)$
have $mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{Y}\ \Psi\ \Delta \ominus\ ?C\ @\ ?E)) = mset\ (?A \ominus\ ?D\ @\ ?F)$
by (*simp add*: *YWitness-firstComponent-diff-decomposition*)
hence $(map\ (uncurry\ (\rightarrow))\ \Delta) \preceq (?A \ominus\ ?D\ @\ ?F)$
using *YWitness-right-stronger-theory*
stronger-theory-relation-alt-def
by (*simp*, *metis* (*no-types*, *lifting*))
hence $?\Gamma_0 \preceq ((?A \ominus\ ?D\ @\ ?F)\ @\ (map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus map\ snd\ \Psi)$
$\ominus\ map\ snd\ \Delta)$
using *stronger-theory-combine stronger-theory-reflexive* **by** *blast*
moreover
have ♠: $mset\ ?G \subseteq\#\ mset\ (map\ (uncurry\ (\rightarrow))\ \Psi)$
$mset\ (\mathfrak{B}\ \Psi\ \Delta) \subseteq\#\ mset\ \Delta$
$mset\ (map\ snd\ ?E) \subseteq\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$
$mset\ (map\ (uncurry\ (\rightarrow))\ \Psi \ominus\ ?G) = mset\ ?D$
$mset\ ?D \subseteq\#\ mset\ ?A$
$mset\ (map\ snd\ ?I) \subseteq\#\ mset\ (map\ snd\ \Psi)$
$mset\ (map\ snd\ ?I) \subseteq\#\ mset\ \Gamma$
$mset\ (map\ snd\ (?I\ @\ ?E)) = mset\ ?J$
using *secondComponent-msub*
secondComponent-diff-msub
secondComponent-snd-projection-msub
firstComponent-secondComponent-mset-connection
XWitness-map-snd-decomposition
by (*simp*,
simp,
metis assms(*2*),
simp add: *image-mset-Diff firstComponent-msub*,
simp add: *YWitness-firstComponent-diff-decomposition*,
simp add: *image-mset-subseteq-mono firstComponent-msub*,
metis assms(*1*) *firstComponent-msub map-monotonic subset-mset.dual-order.trans*,
simp)
hence $mset\ \Delta - mset\ (\mathfrak{B}\ \Psi\ \Delta) + mset\ (\mathfrak{B}\ \Psi\ \Delta) = mset\ \Delta$
by *simp*
hence ♡: $\{\#x \rightarrow y.\ (x,\ y) \in\#\ mset\ \Psi\#\} + (mset\ \Gamma - image\text{-}mset\ snd\ (mset$
$\Psi))$
$-\ image\text{-}mset\ snd\ (mset\ \Delta)$
$= \{\#x \rightarrow y.\ (x,\ y) \in\#\ mset\ \Psi\#\} + (mset\ \Gamma - image\text{-}mset\ snd\ (mset$

Ψ))
$-$ *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$-$ *image-mset snd* (*mset* ($\mathfrak{B}$ Ψ Δ))
image-mset snd (*mset* Ψ $-$ *mset* ($\mathfrak{A}$ Ψ Δ)) $+$ *image-mset snd* (*mset* ($\mathfrak{A}$
Ψ Δ))
$=$ *image-mset snd* (*mset* Ψ)
using ♠
by (*metis* (*no-types*) *diff-diff-add-mset image-mset-union*,
metis (*no-types*) *image-mset-union firstComponent-msub subset-mset.diff-add*)
then have *mset* Γ $-$ *image-mset snd* (*mset* Ψ)
$-$ *image-mset snd* (*mset* Δ $-$ *mset* ($\mathfrak{B}$ Ψ Δ))
$=$ *mset* Γ $-$ (*image-mset snd* (*mset* Ψ $-$ *mset* ($\mathfrak{A}$ Ψ Δ))
$+$ *image-mset snd* (*mset* ($\mathfrak{X}$ Ψ Δ)))
using ♠ **by** (*simp*, *metis* (*full-types*) *diff-diff-add-mset*)
hence *mset* ((*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ *map snd* Ψ) $\ominus$ *map snd* Δ)
$=$ *mset* (*?D* @ (Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
using ♡ ♠ **by** (*simp*, *metis* (*no-types*) *add.commute subset-mset.add-diff-assoc*)
ultimately have $?\Gamma_0 \preceq$ ((*?A* $\ominus$ *?D* @ *?F*) @ *?D* @ (Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
unfolding *stronger-theory-relation-alt-def*
by *simp*
moreover
have *mset ?F* $=$ *mset* (*?B* $\ominus$ *?H*)
mset ?D $\subseteq$# *mset ?A*
mset (*map snd* (Ψ $\ominus$ *?I*)) $\subseteq$# *mset* (Γ $\ominus$ *?J*)
by (*simp add*: *XWitness-secondComponent-diff-decomposition*,
simp add: *YWitness-firstComponent-diff-decomposition*,
simp, *metis* (*no-types*, *lifting*)
♡(*2*) ♠(*8*) *add.assoc assms*(*1*) *assms*(*2*) *image-mset-union*
XWitness-msub mergeWitness-msub-intro
secondComponent-mergeWitness-snd-projection
mset-map
subset-mset.le-diff-conv2
union-code)
hence *mset* ((*?A* $\ominus$ *?D* @ *?F*) @ *?D* @ (Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
$=$ *mset* (*?A* @ (*?B* $\ominus$ *?H* @ Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
mset ?H $\subseteq$# *mset ?B*
{#$x \rightarrow y$. (x, y) $\in$# *mset* ($\mathfrak{X}_\bullet$ Ψ Δ)#} $=$ *mset* (*map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ))
by (*simp add*: *subset-mset.diff-add-assoc*,
simp add: *XWitness-secondComponent-diff-decomposition*,
metis xcomponent-ycomponent-connection mset-map uncurry-def)
hence *mset* ((*?A* $\ominus$ *?D* @ *?F*) @ *?D* @ (Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
$=$ *mset* (*?A* @ (*?B* @ Γ $\ominus$ *?J*) $\ominus$ (*?H* @ *map snd ?C*))
{#$x \rightarrow y$. (x, y) $\in$# *mset* ($\mathfrak{X}_\bullet$ Ψ Δ)#} $+$ *image-mset snd* (*mset* Ψ $-$ *mset*
($\mathfrak{A}$ Ψ Δ))
$=$ *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ))
using *YWitness-map-snd-decomposition*
by (*simp add*: *subset-mset.diff-add-assoc*, *force*)
hence *mset* ((*?A* $\ominus$ *?D* @ *?F*) @ *?D* @ (Γ $\ominus$ *?J*) $\ominus$ *map snd ?C*)
$=$ *mset* (*?A* @ (*?B* @ Γ $\ominus$ *?J*) $\ominus$ *?K*)

by (*simp*)
ultimately have $?\Gamma_0 \preceq (?A\ @\ (?B\ @\ \Gamma \ominus ?J) \ominus ?K)$
unfolding *stronger-theory-relation-alt-def*
by *metis*
thus *?thesis*
using *assms*(*3*) *segmented-stronger-theory-left-monotonic*
by *blast*
qed

lemma (**in** *classical-logic*) *segmented-cons-cons-right-permute*:
assumes $\Gamma \mathrel{\$\vdash} (\varphi \# \psi \# \Phi)$
shows $\Gamma \mathrel{\$\vdash} (\psi \# \varphi \# \Phi)$
proof −
from *assms* **obtain** Ψ **where** Ψ:
$mset\ (map\ snd\ \Psi) \subseteq\# \ mset\ \Gamma$
$map\ (uncurry\ (\sqcup))\ \Psi :\vdash \varphi$
$map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi) \mathrel{\$\vdash} (\psi \# \Phi)$
by *fastforce*
let $?\Gamma_0 = map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus (map\ snd\ \Psi)$
from Ψ(*3*) **obtain** Δ **where** Δ:
$mset\ (map\ snd\ \Delta) \subseteq\# \ mset\ ?\Gamma_0$
$map\ (uncurry\ (\sqcup))\ \Delta :\vdash \psi$
$(map\ (uncurry\ (\rightarrow))\ \Delta\ @\ ?\Gamma_0 \ominus (map\ snd\ \Delta)) \mathrel{\$\vdash} \Phi$
using *segmented-deduction.simps*(*2*) **by** *blast*
let $?\Psi' = \mathfrak{X}\ \Psi\ \Delta$
let $?\Gamma_1 = map\ (uncurry\ (\rightarrow))\ ?\Psi'\ @\ \Gamma \ominus (map\ snd\ ?\Psi')$
let $?\Delta' = \mathfrak{Y}\ \Psi\ \Delta$
have $(map\ (uncurry\ (\rightarrow))\ ?\Delta'\ @\ ?\Gamma_1 \ominus (map\ snd\ ?\Delta')) \mathrel{\$\vdash} \Phi$
$map\ (uncurry\ (\sqcup))\ \Psi \preceq map\ (uncurry\ (\sqcup))\ ?\Delta'$
using Ψ(*1*) Δ(*1*) Δ(*3*)
xwitness-ywitness-segmented-deduction-intro
YWitness-left-stronger-theory
by *auto*
hence $?\Gamma_1 \mathrel{\$\vdash} (\varphi \# \Phi)$
using Ψ(*1*) Ψ(*2*) Δ(*1*)
YWitness-msub segmented-deduction.simps(*2*)
stronger-theory-deduction-monotonic
by *blast*
thus *?thesis*
using Ψ(*1*) Δ(*1*) Δ(*2*)
XWitness-msub
XWitness-right-stronger-theory
segmented-deduction.simps(*2*)
stronger-theory-deduction-monotonic
by *blast*
qed

lemma (**in** *classical-logic*) *segmented-cons-remove1*:
assumes $\varphi \in set\ \Phi$

shows $\Gamma \ \$\vdash \ \Phi = \Gamma \ \$\vdash \ (\varphi \ \# \ (\textit{remove1} \ \varphi \ \Phi))$
proof −
from ‹$\varphi \in \textit{set} \ \Phi$›
have $\forall \ \Gamma. \ \Gamma \ \$\vdash \ \Phi = \Gamma \ \$\vdash \ (\varphi \ \# \ (\textit{remove1} \ \varphi \ \Phi))$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* χ Φ)
{
fix Γ
have $\Gamma \ \$\vdash \ (\chi \ \# \ \Phi) = \Gamma \ \$\vdash \ (\varphi \ \# \ (\textit{remove1} \ \varphi \ (\chi \ \# \ \Phi)))$
proof (*cases* $\chi = \varphi$)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
hence $\varphi \in \textit{set} \ \Phi$
using *Cons.prems* **by** *simp*
with *Cons.hyps* **have** $\Gamma \ \$\vdash \ (\chi \ \# \ \Phi) = \Gamma \ \$\vdash \ (\chi \ \# \ \varphi \ \# \ (\textit{remove1} \ \varphi \ \Phi))$
by *fastforce*
hence $\Gamma \ \$\vdash \ (\chi \ \# \ \Phi) = \Gamma \ \$\vdash \ (\varphi \ \# \ \chi \ \# \ (\textit{remove1} \ \varphi \ \Phi))$
using *segmented-cons-cons-right-permute* **by** *blast*
then show *?thesis* **using** ‹$\chi \neq \varphi$› **by** *simp*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *witness-stronger-theory*:
assumes $\textit{mset} \ (\textit{map} \ \textit{snd} \ \Psi) \subseteq\# \ \textit{mset} \ \Gamma$
shows $(\textit{map} \ (\textit{uncurry} \ (\rightarrow)) \ \Psi \ @ \ \Gamma \ominus (\textit{map} \ \textit{snd} \ \Psi)) \preceq \Gamma$
proof −
have $\forall \ \Gamma. \ \textit{mset} \ (\textit{map} \ \textit{snd} \ \Psi) \subseteq\# \ \textit{mset} \ \Gamma \longrightarrow (\textit{map} \ (\textit{uncurry} \ (\rightarrow)) \ \Psi \ @ \ \Gamma \ominus (\textit{map} \ \textit{snd} \ \Psi)) \preceq \Gamma$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
let $?\gamma = \textit{snd} \ \psi$
{
fix Γ
assume $\textit{mset} \ (\textit{map} \ \textit{snd} \ (\psi \ \# \ \Psi)) \subseteq\# \ \textit{mset} \ \Gamma$
hence $\textit{mset} \ (\textit{map} \ \textit{snd} \ \Psi) \subseteq\# \ \textit{mset} \ (\textit{remove1} \ (\textit{snd} \ \psi) \ \Gamma)$
by (*simp add*: *insert-subset-eq-iff*)
with *Cons* **have**

$(map\ (uncurry\ (\to))\ \Psi\ @\ (remove1\ (snd\ \psi)\ \Gamma) \ominus (map\ snd\ \Psi)) \preceq (remove1\ ?\gamma\ \Gamma)$
by *blast*
hence $(map\ (uncurry\ (\to))\ \Psi\ @\ \Gamma \ominus (map\ snd\ (\psi\ \#\ \Psi))) \preceq (remove1\ ?\gamma\ \Gamma)$
by (*simp add*: *stronger-theory-relation-alt-def*)
moreover
have $(uncurry\ (\to)) = (\lambda\ \psi.\ fst\ \psi \to snd\ \psi)$
by *fastforce*
hence $\vdash\ ?\gamma \to uncurry\ (\to)\ \psi$
using *axiom-k* **by** *simp*
ultimately have
$(map\ (uncurry\ (\to))\ (\psi\ \#\ \Psi)\ @\ \Gamma \ominus (map\ snd\ (\psi\ \#\ \Psi))) \preceq (?\gamma\ \#\ (remove1\ ?\gamma\ \Gamma))$
using *stronger-theory-left-right-cons* **by** *auto*
hence $(map\ (uncurry\ (\to))\ (\psi\ \#\ \Psi)\ @\ \Gamma \ominus (map\ snd\ (\psi\ \#\ \Psi))) \preceq \Gamma$
using *stronger-theory-relation-alt-def*
$\langle mset\ (map\ snd\ (\psi\ \#\ \Psi)) \subseteq\#\ mset\ \Gamma\rangle$
mset-subset-eqD
by *fastforce*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *segmented-msub-weaken*:
assumes $mset\ \Psi \subseteq\#\ mset\ \Phi$
and $\Gamma\ \$\vdash\ \Phi$
shows $\Gamma\ \$\vdash\ \Psi$
proof −
have $\forall\,\Psi\ \Gamma.\ mset\ \Psi \subseteq\#\ mset\ \Phi \longrightarrow \Gamma\ \$\vdash\ \Phi \longrightarrow \Gamma\ \$\vdash\ \Psi$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\varphi\ \Phi$)
{
fix $\Psi\ \Gamma$
assume $mset\ \Psi \subseteq\#\ mset\ (\varphi\ \#\ \Phi)$
$\Gamma\ \$\vdash\ (\varphi\ \#\ \Phi)$
hence $\Gamma\ \$\vdash\ \Phi$
using *segmented-deduction.simps(2)*
segmented-stronger-theory-left-monotonic
witness-stronger-theory
by *blast*
have $\Gamma\ \$\vdash\ \Psi$
proof (*cases* $\varphi \in set\ \Psi$)
case *True*
hence $mset\ (remove1\ \varphi\ \Psi) \subseteq\#\ mset\ \Phi$

```
        using ‹mset Ψ ⊆# mset (φ # Φ)›
              subset-eq-diff-conv
        by force
      hence ∀Γ. Γ $⊢ Φ ⟶ Γ $⊢ (remove1 φ Ψ)
        using Cons by blast
      hence Γ $⊢ (φ # (remove1 φ Ψ))
        using ‹Γ $⊢ (φ # Φ)› by fastforce
      then show ?thesis
        using ‹φ ∈ set Ψ›
              segmented-cons-remove1
        by blast
    next
      case False
      have mset Ψ ⊆# mset Φ + add-mset φ (mset [])
        using ‹mset Ψ ⊆# mset (φ # Φ)› by auto
      hence mset Ψ ⊆# mset Φ
        by (metis (no-types) False
                  diff-single-trivial
                  in-multiset-in-set mset.simps(1)
                  subset-eq-diff-conv)
      then show ?thesis
        using ‹Γ $⊢ Φ› Cons
        by blast
    qed
  }
  then show ?case by blast
 qed
 with assms show ?thesis by blast
qed

lemma (in classical-logic) segmented-stronger-theory-right-antitonic:
 assumes Ψ ≼ Φ
    and Γ $⊢ Φ
  shows Γ $⊢ Ψ
proof −
 have ∀Ψ Γ. Ψ ≼ Φ ⟶ Γ $⊢ Φ ⟶ Γ $⊢ Ψ
 proof (induct Φ)
  case Nil
  then show ?case
    using segmented-deduction.simps(1)
          stronger-theory-empty-list-intro
    by blast
 next
  case (Cons φ Φ)
  {
    fix Ψ Γ
    assume Γ $⊢ (φ # Φ)
           Ψ ≼ (φ # Φ)
    from this obtain Σ where
```

```
    Σ: map snd Σ = Ψ
       mset (map fst Σ) ⊆# mset (φ # Φ)
       ∀(φ,ψ)∈set Σ. ⊢ φ → ψ
    unfolding stronger-theory-relation-def
    by auto
  hence Γ $⊢ Ψ
  proof (cases φ ∈ set (map fst Σ))
    case True
    from this obtain ψ where (φ,ψ) ∈ set Σ
      by (induct Σ, simp, fastforce)
    hence A: mset (map snd (remove1 (φ, ψ) Σ)) = mset (remove1 ψ Ψ)
      and B: mset (map fst (remove1 (φ, ψ) Σ)) ⊆# mset Φ
      using Σ remove1-pairs-list-projections-snd
              remove1-pairs-list-projections-fst
              subset-eq-diff-conv
      by fastforce+
    have ∀(φ,ψ)∈set (remove1 (φ, ψ) Σ). ⊢ φ → ψ
      using Σ(3) by fastforce+
    hence (remove1 ψ Ψ) ≼ Φ
      unfolding stronger-theory-relation-alt-def using A B by blast
    moreover
    from ‹Γ $⊢ (φ # Φ)› obtain Δ where
      Δ: mset (map snd Δ) ⊆# mset Γ
         map (uncurry (⊔)) Δ :⊢ φ
         (map (uncurry (→)) Δ @ Γ ⊖ (map snd Δ)) $⊢ Φ
      by auto
   ultimately have (map (uncurry (→)) Δ @ Γ ⊖ (map snd Δ)) $⊢ remove1
ψ Ψ
      using Cons by blast
    moreover have map (uncurry (⊔)) Δ :⊢ ψ
      using Δ(2) Σ(3) ‹(φ,ψ) ∈ set Σ›
            list-deduction-weaken
            list-deduction-modus-ponens
      by blast
    ultimately have ‹Γ $⊢ (ψ # (remove1 ψ Ψ))›
      using Δ(1) by auto
    moreover from ‹(φ,ψ) ∈ set Σ› Σ(1) have ψ ∈ set Ψ
      by force
    hence mset Ψ ⊆# mset (ψ # (remove1 ψ Ψ))
      by auto
    ultimately show ?thesis using segmented-msub-weaken by blast
  next
    case False
    hence mset (map fst Σ) ⊆# mset Φ
      using Σ(2)
      by (simp,
          metis add-mset-add-single
                diff-single-trivial
                mset-map set-mset-mset
```

subset-eq-diff-conv)
hence $\Psi \preceq \Phi$
using $\Sigma(1)$ $\Sigma(3)$
unfolding *stronger-theory-relation-def*
by *auto*
moreover from ‹Γ $\Vdash (\varphi \# \Phi)$› **have** Γ $\Vdash \Phi$
using *segmented-deduction.simps*(*2*)
segmented-stronger-theory-left-monotonic
witness-stronger-theory
by *blast*
ultimately show *?thesis* **using** *Cons* **by** *blast*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *segmented-witness-right-split*:
assumes *mset* (*map snd* Ψ) $\subseteq\#$ *mset* Φ
shows Γ \$$\vdash$ (*map* (*uncurry* ($\sqcup$)) Ψ @ *map* (*uncurry* ($\rightarrow$)) Ψ @ $\Phi \ominus$ (*map snd* Ψ)) = Γ \$$\vdash$ Φ
proof –
have $\forall$ Γ Φ. *mset* (*map snd* Ψ) $\subseteq\#$ *mset* $\Phi \longrightarrow$
Γ \$$\vdash$ Φ = Γ \$$\vdash$ (*map* (*uncurry* ($\sqcup$)) Ψ @ *map* (*uncurry* ($\rightarrow$)) Ψ @ $\Phi \ominus$ (*map snd* Ψ))
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
{
fix Γ Φ
let *?*χ = *fst* ψ
let *?*φ = *snd* ψ
let *?*Φ' = *map* (*uncurry* ($\sqcup$)) ($\psi \# \Psi$) @
map (*uncurry* ($\rightarrow$)) ($\psi \# \Psi$) @
$\Phi \ominus$ *map snd* ($\psi \# \Psi$)
let *?*Φ_0 = *map* (*uncurry* ($\sqcup$)) Ψ @
map (*uncurry* ($\rightarrow$)) Ψ @
(*remove1 ?*φ Φ) $\ominus$ *map snd* Ψ
assume *mset* (*map snd* ($\psi \# \Psi$)) $\subseteq\#$ *mset* Φ
hence *mset* (*map snd* Ψ) $\subseteq\#$ *mset* (*remove1 ?*φ Φ)
mset (*?*φ # *remove1 ?*φ Φ) = *mset* Φ
by (*simp add*: *insert-subset-eq-iff*)+
hence Γ \$$\vdash$ Φ = Γ \$$\vdash$ (*?*φ # *remove1 ?*φ Φ)
$\forall$ Γ. Γ \$$\vdash$ (*remove1 ?*φ Φ) = Γ \$$\vdash$ *?*Φ_0
by (*metis list.set-intros*(*1*) *segmented-cons-remove1 set-mset-mset*,
metis Cons.hyps)

```
      moreover
      have (uncurry (⊔)) = (λ ψ. fst ψ ⊔ snd ψ)
           (uncurry (→)) = (λ ψ. fst ψ → snd ψ)
        by fastforce+
      hence mset ?Φ' ⊆# mset (?χ ⊔ ?φ # ?χ → ?φ # ?Φ₀)
            mset (?χ ⊔ ?φ # ?χ → ?φ # ?Φ₀) ⊆# mset ?Φ'
            (is mset ?X ⊆# mset ?Y)
        by fastforce+
      hence Γ $⊢ ?Φ' = Γ $⊢ (?φ # ?Φ₀)
        using segmented-formula-right-split
              segmented-msub-weaken
        by blast
      ultimately have Γ $⊢ Φ = Γ $⊢ ?Φ'
        by fastforce
    }
    then show ?case by blast
  qed
  with assms show ?thesis by blast
qed

primrec (in classical-logic)
  submergeWitness :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (𝔈)
  where
    𝔈 Σ [] = map (λ σ. (⊥, (uncurry (⊔)) σ)) Σ
  | 𝔈 Σ (δ # Δ) =
       (case find (λ σ. (uncurry (→)) σ = snd δ) Σ of
             None ⇒ 𝔈 Σ Δ
           | Some σ ⇒ (fst σ, (fst δ ⊓ fst σ) ⊔ snd σ) # (𝔈 (remove1 σ Σ) Δ))

lemma (in classical-logic) submergeWitness-stronger-theory-left:
   map (uncurry (⊔)) Σ ≼ map (uncurry (⊔)) (𝔈 Σ Δ)
proof −
  have ∀ Σ. map (uncurry (⊔)) Σ ≼ map (uncurry (⊔)) (𝔈 Σ Δ)
  proof (induct Δ)
    case Nil
    {
      fix Σ
      {
        fix φ
        have ⊢ (⊥ ⊔ φ) → φ
          unfolding disjunction-def
         using ex-falso-quodlibet modus-ponens excluded-middle-elimination by blast
      }
      note tautology = this
      have map (uncurry (⊔)) Σ ≼ map (uncurry (⊔)) (𝔈 Σ [])
        by (induct Σ,
            simp,
            simp add: stronger-theory-left-right-cons tautology)
    }
```

then show *?case* **by** *auto*
next
case (*Cons* δ Δ)
{
fix Σ
have *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{C}$ Σ (δ # Δ))
proof (*cases find* (λ σ. (*uncurry* ($\rightarrow$)) σ = *snd* δ) Σ = *None*)
case *True*
then show *?thesis* **using** *Cons* **by** *simp*
next
case *False*
from *this* **obtain** σ **where**
σ: *find* ($\lambda\sigma$. *uncurry* ($\rightarrow$) σ = *snd* δ) Σ = *Some* σ
uncurry ($\rightarrow$) σ = *snd* δ
$\sigma \in$ *set* Σ
using *find-Some-predicate find-Some-set-membership*
by *fastforce*
{
fix α β γ
have $\vdash$ ($\alpha \sqcup (\gamma \sqcap \alpha) \sqcup \beta) \rightarrow (\alpha \sqcup \beta)$
proof −
let *?*φ = ($\langle\alpha\rangle \sqcup (\langle\gamma\rangle \sqcap \langle\alpha\rangle) \sqcup \langle\beta\rangle) \rightarrow (\langle\alpha\rangle \sqcup \langle\beta\rangle)$
have $\forall\mathfrak{M}$. $\mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ (| *?*φ |) **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
note *tautology* = *this*
let *?*α = *fst* σ
let *?*β = *snd* σ
let *?*γ = *fst* δ
have (*uncurry* ($\sqcup$)) = (λ σ. *fst* σ $\sqcup$ *snd* σ) **by** *fastforce*
hence (*uncurry* ($\sqcup$)) σ = *?*α $\sqcup$ *?*β **by** *simp*
hence *A*: $\vdash$ (*?*α $\sqcup$ (*?*γ $\sqcap$ *?*α) $\sqcup$ *?*β) $\rightarrow$ (*uncurry* ($\sqcup$)) σ **using** *tautology*
by *simp*
moreover
have *map* (*uncurry* ($\sqcup$)) (*remove1* σ Σ)
$\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{C}$ (*remove1* σ Σ) Δ)
using *Cons* **by** *simp*
ultimately have *A*:
map (*uncurry* ($\sqcup$)) (σ # (*remove1* σ Σ))
$\preceq$ (*?*α $\sqcup$ (*?*γ $\sqcap$ *?*α) $\sqcup$ *?*β # *map* (*uncurry* ($\sqcup$)) ($\mathfrak{C}$ (*remove1* σ Σ) Δ))
using *stronger-theory-left-right-cons* **by** *fastforce*
from σ(*3*) **have** *mset* Σ = *mset* (σ # (*remove1* σ Σ))
by *simp*
hence *mset* (*map* (*uncurry* ($\sqcup$)) Σ) = *mset* (*map* (*uncurry* ($\sqcup$)) (σ #
(*remove1* σ Σ)))
by (*metis mset-map*)
hence *B*: *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq$ *map* (*uncurry* ($\sqcup$)) (σ # (*remove1* σ Σ))

```
        by (simp add: msub-stronger-theory-intro)
      have (  fst σ
             ⊔ (fst δ ⊓ fst σ)
             ⊔ snd σ # map (λ(x, y). x ⊔ y) (𝔈 (remove1 σ Σ) Δ)) ≽ map (λ(x,
y). x ⊔ y) Σ
        by (metis (no-types, hide-lams) A B stronger-theory-transitive uncurry-def)
      thus ?thesis using A B σ by simp
    qed
  }
  then show ?case by auto
 qed
 thus ?thesis by blast
qed

lemma (in classical-logic) submergeWitness-msub:
  mset (map snd (𝔈 Σ Δ)) ⊆# mset (map (uncurry (⊔)) (𝔍 Σ Δ))
proof −
  have ∀ Σ. mset (map snd (𝔈 Σ Δ)) ⊆# mset (map (uncurry (⊔)) (𝔍 Σ Δ))
  proof (induct Δ)
    case Nil
    {
      fix Σ
      have mset (map snd (𝔈 Σ [])) ⊆#
             mset (map (uncurry (⊔)) (𝔍 Σ []))
        by (induct Σ, simp+)
    }
    then show ?case by blast
  next
    case (Cons δ Δ)
    {
      fix Σ
      have mset (map snd (𝔈 Σ (δ # Δ))) ⊆#
             mset (map (uncurry (⊔)) (𝔍 Σ (δ # Δ)))
        using Cons
        by (cases find (λ σ. (uncurry (→)) σ = snd δ) Σ = None,
             simp,
             meson diff-subset-eq-self
                   insert-subset-eq-iff
                   mset-subset-eq-add-mset-cancel
                   subset-mset.dual-order.trans,
             fastforce)
    }
    then show ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in classical-logic) submergeWitness-stronger-theory-right:
   map (uncurry (⊔)) Δ
```

$\preceq$ (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ Σ Δ) @ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{E}$ Σ Δ))
proof –
have $\forall$ Σ. *map* (*uncurry* ($\sqcup$)) Δ
$\preceq$ (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ Σ Δ) @ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{E}$ Σ Δ))
proof(*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Σ
have *map* (*uncurry* ($\sqcup$)) (δ # Δ) $\preceq$
(*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ Σ (δ # Δ))
@ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ (δ # Δ))
$\ominus$ *map snd* ($\mathfrak{E}$ Σ (δ # Δ)))
proof (*cases find* (λ σ. (*uncurry* ($\rightarrow$)) σ = *snd* δ) Σ = *None*)
case *True*
from *Cons* **obtain** Φ **where** Φ:
map snd Φ = *map* (*uncurry* ($\sqcup$)) Δ
mset (*map fst* Φ) $\subseteq$#
mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ Σ Δ)
@ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{E}$ Σ Δ))
$\forall$(γ, σ)$\in$*set* Φ. $\vdash$ γ $\rightarrow$ σ
unfolding *stronger-theory-relation-def*
by *fastforce*
let *?*Φ' = (*uncurry* ($\sqcup$) δ, (*uncurry* ($\sqcup$)) δ) # Φ
have *map snd ?*Φ' = *map* (*uncurry* ($\sqcup$)) (δ # Δ) **using** Φ(*1*) **by** *simp*
moreover
from Φ(*2*) **have** *A*:
image-mset fst (*mset* Φ)
$\subseteq$# {#*x* $\rightarrow$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{E}$ Σ Δ)#}
+ ({#*x* $\sqcup$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#} – *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ)))
by *simp*
have *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ)) $\subseteq$# {#*x* $\sqcup$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#}
using *submergeWitness-msub* **by** *force*
then have *B*: {#*case* δ *of* (*x*, *xa*) $\Rightarrow$ *x* $\sqcup$ *xa*#}
$\subseteq$# *add-mset* (*case* δ *of* (*x*, *xa*) $\Rightarrow$ *x* $\sqcup$ *xa*)
{#*x* $\sqcup$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#} – *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ))
by (*metis add-mset-add-single subset-mset.le-add-diff*)
have *add-mset* (*case* δ *of* (*x*, *xa*) $\Rightarrow$ *x* $\sqcup$ *xa*) {#*x* $\sqcup$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#}
– *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ)) – {#*case* δ *of* (*x*, *xa*) $\Rightarrow$ *x* $\sqcup$ *xa*#}
= {#*x* $\sqcup$ *y*. (*x*, *y*) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#} – *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ))

by *force*

then have *add-mset* (*case* δ *of* (x, *xa*) $\Rightarrow$ $x \sqcup$ *xa*) (*image-mset fst* (*mset* Φ))

− (*add-mset* (*case* δ *of* (x, *xa*) $\Rightarrow$ $x \sqcup$ *xa*) {#$x \sqcup y$. (x, y) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#}

− *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ)))

$\subseteq$# {#$x \rightarrow y$. (x, y) $\in$# *mset* ($\mathfrak{E}$ Σ Δ)#}

using A B **by** (*metis* (*no-types*) *add-mset-add-single*
subset-eq-diff-conv
subset-mset.diff-diff-right)

hence *add-mset* (*case* δ *of* (x, *xa*) $\Rightarrow$ $x \sqcup$ *xa*) (*image-mset fst* (*mset* Φ))

$\subseteq$# {#$x \rightarrow y$. (x, y) $\in$# *mset* ($\mathfrak{E}$ Σ Δ)#}

+ (*add-mset* (*case* δ *of* (x, *xa*) $\Rightarrow$ $x \sqcup$ *xa*) {#$x \sqcup y$. (x, y) $\in$# *mset* ($\mathfrak{J}$ Σ Δ)#}

− *image-mset snd* (*mset* ($\mathfrak{E}$ Σ Δ)))

using *subset-eq-diff-conv* **by** *blast*

hence

mset (*map fst* ?Φ') $\subseteq$#

mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ Σ (δ # Δ))

@ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ (δ # Δ))

$\ominus$ *map snd* ($\mathfrak{E}$ Σ (δ # Δ)))

using *True* Φ(*2*)

by *simp*

moreover have $\forall$(γ, σ)$\in$*set* ?Φ'. $\vdash \gamma \rightarrow \sigma$

using Φ(*3*) *trivial-implication* **by** *auto*

ultimately show *?thesis*

unfolding *stronger-theory-relation-def*

by *blast*

next

case *False*

from *this* **obtain** σ **where**

σ: *find* ($\lambda\sigma$. *uncurry* ($\rightarrow$) σ = *snd* δ) Σ = *Some* σ

uncurry ($\rightarrow$) σ = *snd* δ

using *find-Some-predicate*

by *fastforce*

moreover from *Cons* **have**

map (*uncurry* ($\sqcup$)) Δ $\preceq$

(*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{E}$ (*remove1* σ Σ) Δ) @

remove1 ((*fst* δ $\sqcap$ *fst* σ) $\sqcup$ *snd* σ)

(((*fst* δ $\sqcap$ *fst* σ) $\sqcup$ *snd* σ # *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ (*remove1* σ Σ) Δ))

$\ominus$ *map snd* ($\mathfrak{E}$ (*remove1* σ Σ) Δ)))

unfolding *stronger-theory-relation-alt-def*

by *simp*

moreover

{

fix α β γ

have $\vdash$ ($\alpha \rightarrow$ (($\gamma \sqcap \alpha$) $\sqcup$ β)) $\rightarrow$ ($\gamma \sqcup$ ($\alpha \rightarrow \beta$))

proof −

let ?φ = ($\langle\alpha\rangle$ $\rightarrow$ (($\langle\gamma\rangle$ $\sqcap$ $\langle\alpha\rangle$) $\sqcup$ $\langle\beta\rangle$)) $\rightarrow$ ($\langle\gamma\rangle$ $\sqcup$ ($\langle\alpha\rangle$ $\rightarrow$ $\langle\beta\rangle$))

have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
note *tautology* = *this*
let $?\alpha = fst\ \sigma$
let $?\beta = snd\ \sigma$
let $?\gamma = fst\ \delta$
have $(\lambda\ \delta.\ uncurry\ (\sqcup)\ \delta) = (\lambda\ \delta.\ fst\ \delta \sqcup snd\ \delta)$
$(\lambda\ \sigma.\ uncurry\ (\rightarrow)\ \sigma) = (\lambda\ \sigma.\ fst\ \sigma \rightarrow snd\ \sigma)$ **by** *fastforce+*
hence $(uncurry\ (\sqcup)\ \delta) = (?\gamma \sqcup (?\alpha \rightarrow ?\beta))$ **using** $\sigma(2)$ **by** *simp*
hence $\vdash (?\alpha \rightarrow ((?\gamma \sqcap ?\alpha) \sqcup ?\beta)) \rightarrow (uncurry\ (\sqcup)\ \delta)$ **using** *tautology* **by** *auto*
ultimately show *?thesis*
using *stronger-theory-left-right-cons*
by *fastforce*
qed
}
then show *?case* **by** *auto*
qed
thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *mergeWitness-cons-segmented-deduction*:
assumes $map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi$
and $mset\ (map\ snd\ \Delta) \subseteq\#\ mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma)$
and $map\ (uncurry\ (\sqcup))\ \Delta\ \$\vdash \Phi$
shows $map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta)\ \$\vdash (\varphi\ \#\ \Phi)$
proof −
let $?\Sigma' = \mathfrak{E}\ \Sigma\ \Delta$
let $?\Gamma = map\ (uncurry\ (\rightarrow))\ ?\Sigma'\ @\ map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta) \ominus map\ snd\ ?\Sigma'$
have $?\Gamma\ \$\vdash \Phi$
using *assms(3)*
submergeWitness-stronger-theory-right
segmented-stronger-theory-left-monotonic
by *blast*
moreover have $map\ (uncurry\ (\sqcup))\ ?\Sigma' :\vdash \varphi$
using *assms(1)*
stronger-theory-deduction-monotonic
submergeWitness-stronger-theory-left
by *blast*
ultimately show *?thesis*
using *submergeWitness-msub*
by *fastforce*
qed

primrec (**in** *classical-logic*)
recoverWitnessA :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{P})$

where
  $\mathfrak{P}\ \Sigma\ []\ = \Sigma$
| $\mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta) =$
    $(\mathit{case}\ \mathit{find}\ (\lambda\ \sigma.\ \mathit{snd}\ \sigma = (\mathit{uncurry}\ (\sqcup))\ \delta)\ \Sigma\ \mathit{of}$
        $\mathit{None} \Rightarrow \mathfrak{P}\ \Sigma\ \Delta$
      $|\ \mathit{Some}\ \sigma \Rightarrow (\mathit{fst}\ \sigma \sqcup \mathit{fst}\ \delta,\ \mathit{snd}\ \delta)\ \#\ (\mathfrak{P}\ (\mathit{remove1}\ \sigma\ \Sigma)\ \Delta))$

primrec (**in** *classical-logic*)
  recoverComplementA :: $('a \times 'a)\ \mathit{list} \Rightarrow ('a \times 'a)\ \mathit{list} \Rightarrow ('a \times 'a)\ \mathit{list}\ (\mathfrak{P}^C)$
  where
  $\mathfrak{P}^C\ \Sigma\ []\ = []$
| $\mathfrak{P}^C\ \Sigma\ (\delta\ \#\ \Delta) =$
    $(\mathit{case}\ \mathit{find}\ (\lambda\ \sigma.\ \mathit{snd}\ \sigma = (\mathit{uncurry}\ (\sqcup))\ \delta)\ \Sigma\ \mathit{of}$
        $\mathit{None} \Rightarrow \delta\ \#\ \mathfrak{P}^C\ \Sigma\ \Delta$
      $|\ \mathit{Some}\ \sigma \Rightarrow (\mathfrak{P}^C\ (\mathit{remove1}\ \sigma\ \Sigma)\ \Delta))$

primrec (**in** *classical-logic*)
  recoverWitnessB :: $('a \times 'a)\ \mathit{list} \Rightarrow ('a \times 'a)\ \mathit{list} \Rightarrow ('a \times 'a)\ \mathit{list}\ (\mathfrak{Q})$
  where
  $\mathfrak{Q}\ \Sigma\ []\ = []$
| $\mathfrak{Q}\ \Sigma\ (\delta\ \#\ \Delta) =$
    $(\mathit{case}\ \mathit{find}\ (\lambda\ \sigma.\ (\mathit{snd}\ \sigma) = (\mathit{uncurry}\ (\sqcup))\ \delta)\ \Sigma\ \mathit{of}$
        $\mathit{None} \Rightarrow \delta\ \#\ \mathfrak{Q}\ \Sigma\ \Delta$
      $|\ \mathit{Some}\ \sigma \Rightarrow (\mathit{fst}\ \delta,\ (\mathit{fst}\ \sigma \sqcup \mathit{fst}\ \delta) \to \mathit{snd}\ \delta)\ \#\ (\mathfrak{Q}\ (\mathit{remove1}\ \sigma\ \Sigma)\ \Delta))$

lemma (**in** *classical-logic*) *recoverWitnessA-left-stronger-theory*:
  $\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Sigma \preceq \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ \Delta)$
proof −
  have $\forall\ \Sigma.\ \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Sigma \preceq \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ \Delta)$
  proof (*induct* Δ)
    case *Nil*
    {
      fix Σ
      have $\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Sigma \preceq \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ [])$
        by(*induct* Σ, *simp*+)
    }
    then show *?case* **by** *auto*
  next
    case (*Cons* δ Δ)
    {
      fix Σ
      have $\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Sigma \preceq \mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta))$
      proof (*cases* $\mathit{find}\ (\lambda\ \sigma.\ \mathit{snd}\ \sigma = \mathit{uncurry}\ (\sqcup)\ \delta)\ \Sigma = \mathit{None}$)
        case *True*
        then show *?thesis* **using** *Cons* **by** *simp*
      next
        case *False*
        from *this* **obtain** σ **where**
          σ: $\mathit{find}\ (\lambda\sigma.\ \mathit{snd}\ \sigma = \mathit{uncurry}\ (\sqcup)\ \delta)\ \Sigma = \mathit{Some}\ \sigma$

```
          snd σ = uncurry (⊔) δ
          σ ∈ set Σ
        using find-Some-predicate
              find-Some-set-membership
        by fastforce
      let ?α = fst σ
      let ?β = fst δ
      let ?γ = snd δ
      have uncurry (⊔) = (λδ. fst δ ⊔ snd δ) by fastforce
      hence ⊢ ((?α ⊔ ?β) ⊔ ?γ) → uncurry (⊔) σ
        using σ(2) biconditional-def disjunction-associativity
        by auto
      moreover
      have map (uncurry (⊔)) (remove1 σ Σ)
          ≼ map (uncurry (⊔)) (𝔓 (remove1 σ Σ) Δ)
        using Cons by simp
      ultimately have map (uncurry (⊔)) (σ # (remove1 σ Σ))
                      ≼ map (uncurry (⊔)) (𝔓 Σ (δ # Δ))
        using σ(1)
        by (simp, metis stronger-theory-left-right-cons)
      moreover
      from σ(3) have mset Σ = mset (σ # (remove1 σ Σ))
        by simp
        hence mset (map (uncurry (⊔)) Σ) = mset (map (uncurry (⊔)) (σ #
(remove1 σ Σ)))
        by (metis mset-map)
      hence map (uncurry (⊔)) Σ ≼ map (uncurry (⊔)) (σ # (remove1 σ Σ))
        by (simp add: msub-stronger-theory-intro)
      ultimately show ?thesis
        using stronger-theory-transitive by blast
    qed
  }
  then show ?case by blast
 qed
 thus ?thesis by auto
qed

lemma (in classical-logic) recoverWitnessA-mset-equiv:
 assumes mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
 shows mset (map snd (𝔓 Σ Δ @ 𝔓^C Σ Δ)) = mset (map snd Δ)
proof −
 have ∀ Σ. mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
       ⟶ mset (map snd (𝔓 Σ Δ @ 𝔓^C Σ Δ)) = mset (map snd Δ)
 proof (induct Δ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Δ)
  {
```

fix Σ :: $('a \times 'a)$ *list*
assume $\star$: *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) (δ # Δ))
have *mset* (*map snd* ($\mathfrak{P}$ Σ (δ # Δ) @ $\mathfrak{P}^C$ Σ (δ # Δ))) = *mset* (*map snd* (δ # Δ))
proof (*cases find* (λ σ. *snd* σ = *uncurry* $(\sqcup)$ δ) Σ = *None*)
  case *True*
  hence *uncurry* $(\sqcup)$ $\delta \notin$ *set* (*map snd* Σ)
  proof (*induct* Σ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* σ Σ)
    then show *?case*
      by (*cases* (*uncurry* $(\sqcup)$) δ = *snd* σ, *fastforce*+)
  qed
  moreover have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ) + {#*uncurry* $(\sqcup)$ δ#}
    using $\star$ **by** *fastforce*
  ultimately have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ)
    by (*metis diff-single-trivial*
          in-multiset-in-set
          subset-eq-diff-conv)
  then show *?thesis* **using** *Cons True* **by** *simp*
next
  case *False*
  from *this* **obtain** σ **where**
    σ: *find* ($\lambda\sigma$. *snd* σ = *uncurry* $(\sqcup)$ δ) Σ = *Some* σ
      snd σ = *uncurry* $(\sqcup)$ δ
      $\sigma \in$ *set* Σ
    using *find-Some-predicate*
        find-Some-set-membership
    by *fastforce*
  have *A*: *mset* (*map snd* Σ)
      $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ) + *add-mset* (*uncurry* $(\sqcup)$ δ) (*mset* [])
    using $\star$ **by** *auto*
  have (*fst* σ, *uncurry* $(\sqcup)$ δ) $\in\#$ *mset* Σ
    by (*metis* (*no-types*) $\sigma(2)$ $\sigma(3)$ *prod.collapse set-mset-mset*)
  then have *B*: *mset* (*map snd* (*remove1* (*fst* σ, *uncurry* $(\sqcup)$ δ) Σ))
          = *mset* (*map snd* Σ) − {#*uncurry* $(\sqcup)$ δ#}
    by (*meson remove1-pairs-list-projections-snd*)
  have (*fst* σ, *uncurry* $(\sqcup)$ δ) = σ
    by (*metis* $\sigma(2)$ *prod.collapse*)
  then have *mset* (*map snd* Σ) − *add-mset* (*uncurry* $(\sqcup)$ δ) (*mset* [])
        = *mset* (*map snd* (*remove1* σ Σ))
    using *B* **by** *simp*
  hence *mset* (*map snd* (*remove1* σ Σ)) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ)
    using *A* **by** (*metis* (*no-types*) *subset-eq-diff-conv*)
  with $\sigma(1)$ *Cons* **show** *?thesis* **by** *simp*
qed

    }
    then show *?case* **by** *simp*
  qed
  with *assms* **show** *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *recoverWitnessB-stronger-theory*:
  assumes *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
  shows (*map* (*uncurry* ($\rightarrow$)) Σ @ *map* (*uncurry* ($\sqcup$)) Δ $\ominus$ *map snd* Σ)
        $\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ Σ Δ)
proof −
  have $\forall$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
       $\longrightarrow$ (*map* (*uncurry* ($\rightarrow$)) Σ @ *map* (*uncurry* ($\sqcup$)) Δ $\ominus$ *map snd* Σ)
          $\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ Σ Δ)
  proof(*induct* Δ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* δ Δ)
    {
      fix Σ :: ($'a \times 'a$) *list*
      assume $\star$: *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) (δ # Δ))
      have (*map* (*uncurry* ($\rightarrow$)) Σ @ *map* (*uncurry* ($\sqcup$)) (δ # Δ) $\ominus$ *map snd* Σ)
          $\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ Σ (δ # Δ))
      proof (*cases find* (λ σ. *snd* σ = *uncurry* ($\sqcup$) δ) Σ = *None*)
        case *True*
        hence *uncurry* ($\sqcup$) $\delta \notin$ *set* (*map snd* Σ)
        proof (*induct* Σ)
          case *Nil*
          then show *?case* **by** *simp*
        next
          case (*Cons* σ Σ)
          then show *?case*
            by (*cases uncurry* ($\sqcup$) δ = *snd* σ, *fastforce*+)
        qed
        hence *mset* (*map* (*uncurry* ($\rightarrow$)) Σ @ (*map* (*uncurry* ($\sqcup$)) (δ # Δ)) $\ominus$ *map*
snd Σ)
              = *mset* (*uncurry* ($\sqcup$) δ # *map* (*uncurry* ($\rightarrow$)) Σ
                    @ *map* (*uncurry* ($\sqcup$)) Δ $\ominus$ *map snd* Σ)
              mset (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
          using $\star$
          by (*simp*, *simp*,
              metis add-mset-add-single
                    diff-single-trivial
                    image-set
                    mset-map
                    set-mset-mset
                    subset-eq-diff-conv)
        moreover from *this* **have**

$(map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ map\ (uncurry\ (\sqcup))\ \Delta \ominus map\ snd\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ \Delta)$
using *Cons*
by *auto*
hence $(uncurry\ (\sqcup)\ \delta\ \#\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ map\ (uncurry\ (\sqcup))\ \Delta \ominus map\ snd\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ (\delta\ \#\ \Delta))$
using *True*
by (*simp add*: *stronger-theory-left-right-cons trivial-implication*)
ultimately show *?thesis*
unfolding *stronger-theory-relation-alt-def*
by *simp*
next
case *False*
let $?\Gamma = map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ (map\ (uncurry\ (\sqcup))\ (\delta\ \#\ \Delta)) \ominus map\ snd\ \Sigma$
from *False* **obtain** σ **where**
σ: *find* $(\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
$snd\ \sigma = uncurry\ (\sqcup)\ \delta$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
let $?\Gamma_0 = map\ (uncurry\ (\rightarrow))\ (remove1\ \sigma\ \Sigma)$
$@\ (map\ (uncurry\ (\sqcup))\ \Delta) \ominus map\ snd\ (remove1\ \sigma\ \Sigma)$
let $?\alpha = fst\ \sigma$
let $?\beta = fst\ \delta$
let $?\gamma = snd\ \delta$
have $uncurry\ (\sqcup) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$
$uncurry\ (\rightarrow) = (\lambda\ \sigma.\ fst\ \sigma \rightarrow snd\ \sigma)$
by *fastforce+*
hence $uncurry\ (\rightarrow)\ \sigma = ?\alpha \rightarrow (?\beta \sqcup ?\gamma)$
using $\sigma(2)$
by *simp*
from $\sigma(3)$ **have** $mset\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)) = mset\ \Sigma$ **by** *simp*
hence ♠: $mset\ (map\ snd\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))) = mset\ (map\ snd\ \Sigma)$
$mset\ (map\ (uncurry\ (\rightarrow))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))) = mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma)$
by (*metis mset-map*)+
hence $mset\ ?\Gamma = mset\ (map\ (uncurry\ (\rightarrow))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
$@\ (uncurry\ (\sqcup)\ \delta\ \#\ map\ (uncurry\ (\sqcup))\ \Delta)$
$\ominus map\ snd\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)))$
by *simp*
hence $?\Gamma \preceq (?\alpha \rightarrow (?\beta \sqcup ?\gamma)\ \#\ ?\Gamma_0)$
using $\sigma(2)$ $\langle uncurry\ (\rightarrow)\ \sigma = ?\alpha \rightarrow (?\beta \sqcup ?\gamma)\rangle$
by (*simp add*: *msub-stronger-theory-intro*)
moreover have $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
using ♠(1)

by (*simp*,
metis (*no-types*, *lifting*)
$\star$ $\sigma(2)$
list.simps(*9*)
mset.simps(*2*)
mset-map
uncurry-def
mset-subset-eq-add-mset-cancel)
with *Cons* **have** $\heartsuit$: $?\Gamma_0 \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ (*remove1* σ Σ) Δ) **by** *simp*
{
fix α β γ
have $\vdash (\beta \sqcup (\alpha \sqcup \beta) \to \gamma) \to (\alpha \to (\beta \sqcup \gamma))$
proof $-$
let $?\varphi = (\langle\beta\rangle \sqcup (\langle\alpha\rangle \sqcup \langle\beta\rangle) \to \langle\gamma\rangle) \to (\langle\alpha\rangle \to (\langle\beta\rangle \sqcup \langle\gamma\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
hence $\vdash (?\beta \sqcup (?\alpha \sqcup ?\beta) \to ?\gamma) \to (?\alpha \to (?\beta \sqcup ?\gamma))$
by *simp*
hence $(?\alpha \to (?\beta \sqcup ?\gamma)\ \#\ ?\Gamma_0) \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ Σ (δ # Δ))
using $\sigma(1)$ $\heartsuit$
by (*simp*, *metis stronger-theory-left-right-cons*)
ultimately show *?thesis*
using *stronger-theory-transitive* **by** *blast*
qed
}
then show *?case* **by** *simp*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *classical-logic*) *recoverWitnessB-mset-equiv*:
assumes *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
shows *mset* (*map snd* ($\mathfrak{Q}$ Σ Δ))
= *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{P}$ Σ Δ) @ *map snd* Δ $\ominus$ *map snd* ($\mathfrak{P}$ Σ Δ))
proof $-$
have $\forall$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
$\longrightarrow$ *mset* (*map snd* ($\mathfrak{Q}$ Σ Δ)) = *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{P}$ Σ Δ) @ *map snd* ($\mathfrak{P}^C$ Σ Δ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Σ :: ($'a \times 'a$) *list*

assume $\star$: *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) (δ # Δ))
have *mset* (*map snd* ($\mathfrak{Q}$ Σ (δ # Δ)))
= *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ Σ (δ # Δ)) @ *map snd* ($\mathfrak{P}^C$ Σ (δ # Δ)))
proof (*cases find* (λ σ. *snd* σ = *uncurry* ($\sqcup$) δ) Σ = *None*)
case *True*
hence *uncurry* ($\sqcup$) $\delta \notin$ *set* (*map snd* Σ)
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
then show *?case*
by (*cases* (*uncurry* ($\sqcup$)) δ = *snd* σ, *fastforce*+)
qed
moreover have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ) + {#*uncurry* ($\sqcup$) δ#}
using $\star$ **by** *force*
ultimately have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
by (*metis diff-single-trivial in-multiset-in-set subset-eq-diff-conv*)
then show *?thesis* **using** *True Cons* **by** *simp*
next
case *False*
from *this* **obtain** σ **where**
σ: *find* ($\lambda\sigma$. *snd* σ = *uncurry* ($\sqcup$) δ) Σ = *Some* σ
snd σ = *uncurry* ($\sqcup$) δ
$\sigma \in$ *set* Σ
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
hence (*fst* σ, *uncurry* ($\sqcup$) δ) $\in\#$ *mset* Σ
by (*metis* (*full-types*) *prod.collapse set-mset-mset*)
then have *mset* (*map snd* (*remove1* (*fst* σ, *uncurry* ($\sqcup$) δ) Σ))
= *mset* (*map snd* Σ) − {#*uncurry* ($\sqcup$) δ#}
by (*meson remove1-pairs-list-projections-snd*)
moreover have
mset (*map snd* Σ)
$\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ) + *add-mset* (*uncurry* ($\sqcup$) δ) (*mset* [])
using $\star$ **by** *force*
ultimately have *mset* (*map snd* (*remove1* σ Σ))
$\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
by (*metis* (*no-types*) σ(*2*) *mset.simps*(*1*) *prod.collapse subset-eq-diff-conv*)
with σ(*1*) *Cons* **show** *?thesis* **by** *simp*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis*
using *assms recoverWitnessA-mset-equiv*
by (*simp*, *metis add-diff-cancel-left*′)

```
qed

lemma (in classical-logic) recoverWitnessB-right-stronger-theory:
  map (uncurry (→)) Δ ≼ map (uncurry (→)) (𝔔 Σ Δ)
proof −
  have ∀ Σ. map (uncurry (→)) Δ ≼ map (uncurry (→)) (𝔔 Σ Δ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Σ
      have map (uncurry (→)) (δ # Δ) ≼ map (uncurry (→)) (𝔔 Σ (δ # Δ))
      proof (cases find (λ σ. snd σ = uncurry (⊔) δ) Σ = None)
        case True
        then show ?thesis
          using Cons
          by (simp add: stronger-theory-left-right-cons trivial-implication)
      next
        case False
        from this obtain σ where σ:
          find (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
          by fastforce
        let ?α = fst δ
        let ?β = snd δ
        let ?γ = fst σ
        have uncurry (→) = (λδ. fst δ → snd δ) by fastforce
        hence uncurry (→) δ = ?α → ?β by auto
        moreover have ⊢ (?α → (?γ ⊔ ?α) → ?β) → ?α → ?β
          unfolding disjunction-def
          using axiom-k axiom-s modus-ponens flip-implication
          by blast
        ultimately show ?thesis
          using Cons σ
          by (simp add: stronger-theory-left-right-cons)
      qed
    }
    then show ?case by simp
  qed
  thus ?thesis by simp
qed

lemma (in classical-logic) recoverWitnesses-mset-equiv:
  assumes mset (map snd Δ) ⊆# mset Γ
      and mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
    shows mset (Γ ⊖ map snd Δ)
        = mset ((map (uncurry (→)) (𝔓 Σ Δ) @ Γ ⊖ map snd (𝔓 Σ Δ)) ⊖ map
snd (𝔔 Σ Δ))
```

proof −
  have *mset* (Γ ⊖ *map snd* Δ) = *mset* (Γ ⊖ *map snd* ($\mathfrak{P}^C$ Σ Δ) ⊖ *map snd* ($\mathfrak{P}$ Σ Δ))
    using *assms*(*2*) *recoverWitnessA-mset-equiv*
    by (*simp add*: *union-commute*)
  moreover have ∀ Σ. *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
                ⟶ *mset* (Γ ⊖ *map snd* ($\mathfrak{P}^C$ Σ Δ))
                  = (*mset* ((*map* (*uncurry* (→)) ($\mathfrak{P}$ Σ Δ) @ Γ) ⊖ *map snd* ($\mathfrak{Q}$ Σ Δ)))
    using *assms*(*1*)
  proof (*induct* Δ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* δ Δ)
    from *Cons.prems* **have** *snd* δ ∈ *set* Γ
      using *mset-subset-eqD* **by** *fastforce*
    from *Cons.prems* **have** ♡: *mset* (*map snd* Δ) ⊆# *mset* Γ
      using *subset-mset.dual-order.trans*
      by *fastforce*
    {
      fix Σ :: (′*a* × ′*a*) *list*
      assume ⋆: *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) (δ # Δ))
      have *mset* (Γ ⊖ *map snd* ($\mathfrak{P}^C$ Σ (δ # Δ)))
          = *mset* ((*map* (*uncurry* (→)) ($\mathfrak{P}$ Σ (δ # Δ)) @ Γ) ⊖ *map snd* ($\mathfrak{Q}$ Σ (δ # Δ)))
      proof (*cases find* (λ σ. *snd* σ = *uncurry* (⊔) δ) Σ = *None*)
        case *True*
        hence *uncurry* (⊔) δ ∉ *set* (*map snd* Σ)
        proof (*induct* Σ)
          case *Nil*
          then show *?case* **by** *simp*
        next
          case (*Cons* σ Σ)
          then show *?case*
            by (*cases* (*uncurry* (⊔)) δ = *snd* σ, *fastforce*+)
        qed
         moreover have *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ) + {#*uncurry* (⊔) δ#}
          using ⋆ **by** *auto*
        ultimately have *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
        by (*metis* (*full-types*) *diff-single-trivial in-multiset-in-set subset-eq-diff-conv*)
        with *Cons.hyps* ♡ **have** *mset* (Γ ⊖ *map snd* ($\mathfrak{P}^C$ Σ Δ))
                        = *mset* ((*map* (*uncurry* (→)) ($\mathfrak{P}$ Σ Δ) @ Γ) ⊖ *map snd* ($\mathfrak{Q}$ Σ Δ))
          by *simp*
        thus *?thesis* **using** *True* ‹*snd* δ ∈ *set* Γ› **by** *simp*
      next
        case *False*

from *this* **obtain** σ **where** σ:
find $(\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
$snd\ \sigma = uncurry\ (\sqcup)\ \delta$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
with $\star$ **have** $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
by (*simp*, *metis* (*no-types*, *lifting*)
add-mset-remove-trivial-eq
image-mset-add-mset
in-multiset-in-set
mset-subset-eq-add-mset-cancel)
with *Cons.hyps* **have** $mset\ (\Gamma \ominus map\ snd\ (\mathfrak{P}^C\ (remove1\ \sigma\ \Sigma)\ \Delta))$
$= mset\ ((map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ (remove1\ \sigma\ \Sigma)\ \Delta)\ @\ \Gamma)$
$\ominus map\ snd\ (\mathfrak{Q}\ (remove1\ \sigma\ \Sigma)\ \Delta))$
using $\heartsuit$ **by** *blast*
then show *?thesis* **using** σ **by** *simp*
qed
}
then show *?case* **by** *blast*
qed
moreover have $image\text{-}mset\ snd\ (mset\ (\mathfrak{P}^C\ \Sigma\ \Delta)) = mset\ (map\ snd\ \Delta \ominus map\ snd\ (\mathfrak{P}\ \Sigma\ \Delta))$
using *assms*(*2*) *recoverWitnessA-mset-equiv*
by (*simp*, *metis* (*no-types*) *diff-union-cancelL list-subtract-mset-homomorphism mset-map*)
then have $mset\ \Gamma - (image\text{-}mset\ snd\ (mset\ (\mathfrak{P}^C\ \Sigma\ \Delta)) + image\text{-}mset\ snd\ (mset\ (\mathfrak{P}\ \Sigma\ \Delta)))$
$= \{\#x \rightarrow y.\ (x,\ y) \in\#\ mset\ (\mathfrak{P}\ \Sigma\ \Delta)\#\}$
$+ (mset\ \Gamma - image\text{-}mset\ snd\ (mset\ (\mathfrak{P}\ \Sigma\ \Delta))) - image\text{-}mset\ snd\ (mset\ (\mathfrak{Q}\ \Sigma\ \Delta))$
using *calculation*
assms(*2*)
recoverWitnessA-mset-equiv
recoverWitnessB-mset-equiv
by *fastforce*
ultimately
show *?thesis*
using *assms recoverWitnessA-mset-equiv*
by *simp*
qed

theorem (**in** *classical-logic*) *segmented-deduction-generalized-witness*:
$\Gamma\ \$\vdash\ (\Phi\ @\ \Psi) = (\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
$map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ \Phi\ \wedge$
$(map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus (map\ snd\ \Sigma))\ \$\vdash\ \Psi)$
proof −

```
have ∀ Γ Ψ. Γ $⊢ (Φ @ Ψ) = (∃ Σ. mset (map snd Σ) ⊆# mset Γ ∧
                              map (uncurry (⊔)) Σ $⊢ Φ ∧
                             (map (uncurry (→)) Σ @ Γ ⊖ (map snd Σ)) $⊢ Ψ)
proof (induct Φ)
  case Nil
  {
    fix Γ Ψ
    have Γ $⊢ ([] @ Ψ) = (∃Σ. mset (map snd Σ) ⊆# mset Γ ∧
                             map (uncurry (⊔)) Σ $⊢ [] ∧
                             map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ)
    proof (rule iffI)
      assume Γ $⊢ ([] @ Ψ)
      moreover
      have Γ $⊢ ([] @ Ψ) = (mset (map snd []) ⊆# mset Γ ∧
                            map (uncurry (⊔)) [] $⊢ [] ∧
                            map (uncurry (→)) [] @ Γ ⊖ (map snd []) $⊢ Ψ)
        by simp
      ultimately show ∃Σ. mset (map snd Σ) ⊆# mset Γ ∧
                          map (uncurry (⊔)) Σ $⊢ [] ∧
                          map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ
        by metis
    next
      assume ∃Σ. mset (map snd Σ) ⊆# mset Γ ∧
                 map (uncurry (⊔)) Σ $⊢ [] ∧
                 map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ
      from this obtain Σ where
        Σ: mset (map snd Σ) ⊆# mset Γ
           map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ ([] @ Ψ)
        by fastforce
      hence (map (uncurry (→)) Σ @ Γ ⊖ map snd Σ) ≼ Γ
        using witness-stronger-theory by auto
      with Σ(2) show Γ $⊢ ([] @ Ψ)
        using segmented-stronger-theory-left-monotonic by blast
    qed
  }
  then show ?case by blast
next
  case (Cons φ Φ)
  {
    fix Γ Ψ
    have Γ $⊢ ((φ # Φ) @ Ψ) = (∃Σ. mset (map snd Σ) ⊆# mset Γ ∧
                                  map (uncurry (⊔)) Σ $⊢ (φ # Φ) ∧
                                  map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ)
    proof (rule iffI)
      assume Γ $⊢ ((φ # Φ) @ Ψ)
      from this obtain Σ where
        Σ: mset (map snd Σ) ⊆# mset Γ
           map (uncurry (⊔)) Σ :⊢ φ
           map (uncurry (→)) Σ @ Γ ⊖ (map snd Σ) $⊢ (Φ @ Ψ)
```

(**is** $?\Gamma_0$ $\$\vdash$ $(\Phi @ \Psi)$)
by *auto*
from *this*(*3*) **obtain** Δ **where**
Δ: *mset* (*map snd* Δ) $\subseteq\#$ *mset* $?\Gamma_0$
map (*uncurry* $(\sqcup)$) Δ $\$\vdash$ Φ
map (*uncurry* $(\rightarrow)$) Δ @ $?\Gamma_0 \ominus$ (*map snd* Δ) $\$\vdash$ Ψ
using *Cons*
by *auto*
let $?\Sigma' = \mathfrak{J}\ \Sigma\ \Delta$
have *map* (*uncurry* $(\sqcup)$) $?\Sigma'$ $\$\vdash$ $(\varphi \# \Phi)$
using $\Delta(1)$ $\Delta(2)$ $\Sigma(2)$ *mergeWitness-cons-segmented-deduction* **by** *blast*
moreover have *mset* (*map snd* $?\Sigma'$) $\subseteq\#$ *mset* Γ
using $\Delta(1)$ $\Sigma(1)$ *mergeWitness-msub-intro* **by** *blast*
moreover have *map* (*uncurry* $(\rightarrow)$) $?\Sigma'$ @ $\Gamma \ominus$ *map snd* $?\Sigma'$ $\$\vdash$ Ψ
using $\Delta(1)$ $\Delta(3)$ *mergeWitness-segmented-deduction-intro* **by** *blast*
ultimately show
$\exists\Sigma$. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
map (*uncurry* $(\sqcup)$) Σ $\$\vdash$ $(\varphi \# \Phi)$ $\wedge$
map (*uncurry* $(\rightarrow)$) Σ @ $\Gamma \ominus$ *map snd* Σ $\$\vdash$ Ψ
by *fast*
next
assume $\exists\Sigma$. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
map (*uncurry* $(\sqcup)$) Σ $\$\vdash$ $(\varphi \# \Phi)$ $\wedge$
map (*uncurry* $(\rightarrow)$) Σ @ $\Gamma \ominus$ *map snd* Σ $\$\vdash$ Ψ
from *this* **obtain** Δ **where** Δ:
mset (*map snd* Δ) $\subseteq\#$ *mset* Γ
map (*uncurry* $(\sqcup)$) Δ $\$\vdash$ $(\varphi \# \Phi)$
map (*uncurry* $(\rightarrow)$) Δ @ $\Gamma \ominus$ *map snd* Δ $\$\vdash$ Ψ
by *auto*
from *this* **obtain** Σ **where** Σ:
mset (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ)
map (*uncurry* $(\sqcup)$) Σ $:\vdash$ φ
map (*uncurry* $(\rightarrow)$) Σ @ (*map* (*uncurry* $(\sqcup)$) Δ) $\ominus$ *map snd* Σ $\$\vdash$ Φ
by *auto*
let $?\Omega = \mathfrak{P}\ \Sigma\ \Delta$
let $?\Xi = \mathfrak{Q}\ \Sigma\ \Delta$
let $?\Gamma_0$ = *map* (*uncurry* $(\rightarrow)$) $?\Omega$ @ $\Gamma \ominus$ *map snd* $?\Omega$
let $?\Gamma_1$ = *map* (*uncurry* $(\rightarrow)$) $?\Xi$ @ $?\Gamma_0 \ominus$ *map snd* $?\Xi$
have *mset* ($\Gamma \ominus$ *map snd* Δ) = *mset* ($?\Gamma_0 \ominus$ *map snd* $?\Xi$)
using $\Delta(1)$ $\Sigma(1)$ *recoverWitnesses-mset-equiv* **by** *blast*
hence ($\Gamma \ominus$ *map snd* Δ) $\preceq$ ($?\Gamma_0 \ominus$ *map snd* $?\Xi$)
by (*simp add*: *msub-stronger-theory-intro*)
hence $?\Gamma_1$ $\$\vdash$ Ψ
using $\Delta(3)$ *segmented-stronger-theory-left-monotonic*
stronger-theory-combine
recoverWitnessB-right-stronger-theory
by *blast*
moreover
have *mset* (*map snd* $?\Xi$) $\subseteq\#$ *mset* $?\Gamma_0$

using $\Sigma(1)$ $\Delta(1)$ *recoverWitnessB-mset-equiv*
by (*simp*,
metis list-subtract-monotonic
list-subtract-mset-homomorphism
mset-map)
moreover
have *map* (*uncurry* ($\sqcup$)) *?*Ξ \$$\vdash$ Φ
using $\Sigma(1)$ *recoverWitnessB-stronger-theory*
$\Sigma(3)$ *segmented-stronger-theory-left-monotonic* **by** *blast*
ultimately have *?*Γ_0 \$$\vdash$ (Φ @ Ψ)
using *Cons* **by** *fast*
moreover
have *mset* (*map snd ?*Ω) $\subseteq\#$ *mset* (*map snd* Δ)
using $\Sigma(1)$ *recoverWitnessA-mset-equiv*
by (*simp*, *metis mset-subset-eq-add-left*)
hence *mset* (*map snd ?*Ω) $\subseteq\#$ *mset* Γ **using** $\Delta(1)$ **by** *simp*
moreover
have *map* (*uncurry* ($\sqcup$)) *?*Ω :$\vdash$ φ
using $\Sigma(2)$
recoverWitnessA-left-stronger-theory
stronger-theory-deduction-monotonic
by *blast*
ultimately show Γ \$$\vdash$ ((φ # Φ) @ Ψ)
by (*simp*, *blast*)
qed
}
then show *?case* **by** *metis*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *segmented-list-deduction-antitonic*:
assumes Γ \$$\vdash$ Ψ
and Ψ :$\vdash$ φ
shows Γ :$\vdash$ φ
proof –
have $\forall$ Γ φ. Γ \$$\vdash$ Ψ $\longrightarrow$ Ψ :$\vdash$ φ $\longrightarrow$ Γ :$\vdash$ φ
proof (*induct* Ψ)
case *Nil*
then show *?case*
using *list-deduction-weaken*
by *simp*
next
case (*Cons* ψ Ψ)
{
fix Γ φ
assume Γ \$$\vdash$ (ψ # Ψ)
and ψ # Ψ :$\vdash$ φ
hence Ψ :$\vdash$ $\psi \to \varphi$

```
      using list-deduction-theorem by blast
    from ‹Γ $⊢ (ψ # Ψ)› obtain Σ where Σ:
      mset (map snd Σ) ⊆# mset Γ
      map (uncurry (⊔)) Σ :⊢ ψ
      map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ
      by auto
    hence Γ :⊢ ψ → φ
      using segmented-stronger-theory-left-monotonic
            witness-stronger-theory
            ‹Ψ :⊢ ψ → φ›
            Cons
      by blast
    moreover
    have Γ :⊢ ψ
      using Σ(1) Σ(2)
            stronger-theory-deduction-monotonic
            witness-weaker-theory
      by blast
    ultimately have Γ :⊢ φ using list-deduction-modus-ponens by auto
  }
  then show ?case by simp
 qed
 thus ?thesis using assms by auto
qed

theorem (in classical-logic) segmented-transitive:
 assumes Γ $⊢ Λ and Λ $⊢ Δ
  shows Γ $⊢ Δ
proof −
 have ∀ Γ Λ. Γ $⊢ Λ ⟶ Λ $⊢ Δ ⟶ Γ $⊢ Δ
 proof (induct Δ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Δ)
  {
    fix Γ Λ
    assume Λ $⊢ (δ # Δ)
    from this obtain Σ where Σ:
      mset (map snd Σ) ⊆# mset Λ
      map (uncurry (⊔)) Σ :⊢ δ
      map (uncurry (→)) Σ @ Λ ⊖ map snd Σ $⊢ Δ
      by auto
    assume Γ $⊢ Λ
    hence Γ $⊢ (map (uncurry (⊔)) Σ @ map (uncurry (→)) Σ @ Λ ⊖ (map snd
Σ))
      using Σ(1) segmented-witness-right-split
      by simp
    from this obtain Ψ where Ψ:
```

$mset\ (map\ snd\ \Psi) \subseteq\# \ mset\ \Gamma$
$map\ (uncurry\ (\sqcup))\ \Psi\ \$\vdash\ map\ (uncurry\ (\sqcup))\ \Sigma$
$map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus map\ snd\ \Psi\ \$\vdash\ (map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Lambda \ominus map\ snd\ \Sigma)$
using *segmented-deduction-generalized-witness*
by *fastforce*
have $map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus map\ snd\ \Psi\ \$\vdash\ \Delta$
using $\Sigma(3)$ $\Psi(3)$ *Cons*
by *auto*
moreover
have $map\ (uncurry\ (\sqcup))\ \Psi :\vdash \delta$
using $\Psi(2)$ $\Sigma(2)$ *segmented-list-deduction-antitonic*
by *blast*
ultimately have $\Gamma\ \$\vdash\ (\delta\ \#\ \Delta)$
using $\Psi(1)$
by *fastforce*
}
then show *?case* **by** *auto*
qed
with *assms* **show** *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *segmented-formula-left-split*:
$\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ \Phi = \varphi\ \#\ \Gamma\ \$\vdash\ \Phi$
proof (*rule iffI*)
assume $\varphi\ \#\ \Gamma\ \$\vdash\ \Phi$
have $\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ (\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma)$
using *segmented-stronger-theory-intro*
stronger-theory-reflexive
by *blast*
hence $\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ (\varphi\ \#\ \Gamma)$
using *segmented-formula-right-split* **by** *blast*
with ‹$\varphi\ \#\ \Gamma\ \$\vdash\ \Phi$› **show** $\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ \Phi$
using *segmented-transitive* **by** *blast*
next
assume $\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ \Phi$
have $\varphi\ \#\ \Gamma\ \$\vdash\ (\varphi\ \#\ \Gamma)$
using *segmented-stronger-theory-intro*
stronger-theory-reflexive
by *blast*
hence $\varphi\ \#\ \Gamma\ \$\vdash\ (\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma)$
using *segmented-formula-right-split* **by** *blast*
with ‹$\psi \sqcup \varphi\ \#\ \psi \rightarrow \varphi\ \#\ \Gamma\ \$\vdash\ \Phi$› **show** $\varphi\ \#\ \Gamma\ \$\vdash\ \Phi$
using *segmented-transitive* **by** *blast*
qed

lemma (**in** *classical-logic*) *segmented-witness-left-split* [*simp*]:
assumes $mset\ (map\ snd\ \Sigma) \subseteq\# \ mset\ \Gamma$
shows $(map\ (uncurry\ (\sqcup))\ \Sigma\ @\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus (map\ snd\ \Sigma))\ \$\vdash$

$\Phi = \Gamma$ \$⊢ Φ
proof $-$
have $\forall\ \Gamma$. *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma \longrightarrow$
(*map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ (*map snd* Σ)) \$⊢ $\Phi =$ Γ \$⊢ Φ
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
{
fix Γ
let *?*χ = *fst* σ
let *?*γ = *snd* σ
let *?*Γ_0 = *map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* (σ # Σ)
let *?*Γ' = *map* (*uncurry* ($\sqcup$)) (σ # Σ) @ *map* (*uncurry* ($\rightarrow$)) (σ # Σ) @ Γ $\ominus$ *map snd* (σ # Σ)
assume *mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ
hence *A*: *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ)) $\subseteq\#$ *mset* Γ **by** *simp*
hence *B*: *image-mset snd* (*mset* Σ) + (*mset* Γ $-$ *image-mset snd* (*mset* Σ))
= *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ))
+ (*mset* Γ $-$ *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ)))
by (*metis* (*no-types*) *mset-subset-eq-insertD subset-mset.add-diff-inverse subset-mset-def*)
have {#$x \rightarrow y$. (x, y) $\in\#$ *mset* Σ#} + *mset* Γ $-$ *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ))
= {#$x \rightarrow y$. (x, y) $\in\#$ *mset* Σ#} + (*mset* Γ $-$ *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ)))
using *A subset-mset.diff-add-assoc* **by** *blast*
hence {#$x \rightarrow y$. (x, y) $\in\#$ *mset* Σ#} + (*mset* Γ $-$ *image-mset snd* (*mset* Σ))
= *add-mset* (*snd* σ) ({#$x \rightarrow y$. (x, y) $\in\#$ *mset* Σ#}
+ *mset* Γ $-$ *add-mset* (*snd* σ) (*image-mset snd* (*mset* Σ)))
using *B* **by** *auto*
hence *C*:
mset (*map snd* Σ) $\subseteq\#$ *mset* Γ
mset (*map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ)
= *mset* (*?*γ # *?*Γ_0)
using ‹*mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ›
subset-mset.dual-order.trans
by (*fastforce*+)
hence Γ \$⊢ Φ = (*?*χ $\sqcup$ *?*γ # *?*χ $\rightarrow$ *?*γ # *?*Γ_0) \$⊢ Φ
proof $-$
have $\forall\Gamma\ \Delta$. $\neg$ *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ
$\vee$ $\neg$ Γ \$⊢ Φ
$\vee$ $\neg$ *mset* (*map* (*uncurry* ($\sqcup$)) Σ
@ *map* (*uncurry* ($\rightarrow$)) Σ
@ $\Gamma \ominus$ *map snd* Σ)

```
                  ⊆# mset Δ
                ∨ Δ $⊢ Φ
          using Cons.hyps segmented-msub-left-monotonic by blast
        moreover
        { assume ¬ Γ $⊢ Φ
          then have ∃ Δ. mset (snd σ # map (uncurry (⊔)) Σ
                                @ map (uncurry (→)) Σ
                                @ Γ ⊖ map snd (σ # Σ))
                             ⊆# mset Δ
                           ∧ ¬ Γ $⊢ Φ
                           ∧ ¬ Δ $⊢ Φ
            by (metis (no-types) Cons.hyps C subset-mset.dual-order.refl)
          then have ?thesis
              using segmented-formula-left-split segmented-msub-left-monotonic by
blast }
        ultimately show ?thesis
       by (metis (full-types) C segmented-formula-left-split subset-mset.dual-order.refl)
      qed
      moreover
      have (uncurry (⊔)) = (λ ψ. fst ψ ⊔ snd ψ)
           (uncurry (→)) = (λ ψ. fst ψ → snd ψ)
        by fastforce+
      hence mset ?Γ' = mset (?χ ⊔ ?γ # ?χ → ?γ # ?Γ₀)
        by fastforce
      hence (?χ ⊔ ?γ # ?χ → ?γ # ?Γ₀) $⊢ Φ = ?Γ' $⊢ Φ
        by (metis (mono-tags, lifting)
                segmented-msub-left-monotonic
                subset-mset.dual-order.refl)
      ultimately have Γ $⊢ Φ = ?Γ' $⊢ Φ
        by fastforce
    }
    then show ?case by blast
  qed
  with assms show ?thesis by blast
qed

lemma (in classical-logic) segmented-tautology-right-cancel:
  assumes ⊢ φ
  shows Γ $⊢ (φ # Φ) = Γ $⊢ Φ
proof (rule iffI)
  assume Γ $⊢ (φ # Φ)
  from this obtain Σ where Σ:
    mset (map snd Σ) ⊆# mset Γ
    map (uncurry (⊔)) Σ :⊢ φ
    map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Φ
    by auto
  thus Γ $⊢ Φ
    using segmented-stronger-theory-left-monotonic
          witness-stronger-theory
```

```
    by blast
next
  assume Γ $⊢ Φ
  hence map (uncurry (→)) [] @ Γ ⊖ map snd [] $⊢ Φ
        mset (map snd []) ⊆# mset Γ
        map (uncurry (⊔)) [] :⊢ φ
    using assms
    by simp+
  thus Γ $⊢ (φ # Φ)
    using segmented-deduction.simps(2)
    by blast
qed

lemma (in classical-logic) segmented-tautology-left-cancel [simp]:
  assumes ⊢ γ
  shows (γ # Γ) $⊢ Φ = Γ $⊢ Φ
proof (rule iffI)
  assume (γ # Γ) $⊢ Φ
  moreover have Γ $⊢ Γ
    by (simp add: segmented-stronger-theory-intro)
  hence Γ $⊢ (γ # Γ)
    using assms segmented-tautology-right-cancel
    by simp
  ultimately show Γ $⊢ Φ
    using segmented-transitive by blast
next
  assume Γ $⊢ Φ
  moreover have mset Γ ⊆# mset (γ # Γ)
    by simp
  hence (γ # Γ) $⊢ Γ
    using msub-stronger-theory-intro
          segmented-stronger-theory-intro
    by blast
  ultimately show (γ # Γ) $⊢ Φ
    using segmented-transitive by blast
qed

lemma (in classical-logic) segmented-cancel:
  (Δ @ Γ) $⊢ (Δ @ Φ) = Γ $⊢ Φ
proof −
  {
    fix Δ Γ Φ
    assume Γ $⊢ Φ
    hence (Δ @ Γ) $⊢ (Δ @ Φ)
    proof (induct Δ)
      case Nil
      then show ?case by simp
    next
      case (Cons δ Δ)
```

let *?Σ* = [(δ, δ)]
have *map* (*uncurry* (⊔)) *?Σ* :⊢ δ
unfolding *disjunction-def list-deduction-def*
by (*simp add*: *Peirces-law*)
moreover have *mset* (*map snd ?Σ*) ⊆# *mset* (δ # Δ) **by** *simp*
moreover have *map* (*uncurry* (→)) *?Σ* @ ((δ # Δ) @ Γ) ⊖ *map snd ?Σ* $⊢ (Δ @ Φ)
using *Cons*
by (*simp add*: *trivial-implication*)
moreover have *map snd* [(δ, δ)] = [δ] **by** *force*
ultimately show *?case*
by (*metis* (*no-types*) *segmented-deduction.simps*(*2*)
append-Cons
list.set-intros(*1*)
mset.simps(*1*)
mset.simps(*2*)
mset-subset-eq-single
set-mset-mset)
qed
} **note** *forward-direction* = *this*
{
assume (Δ @ Γ) $⊢ (Δ @ Φ)
hence Γ $⊢ Φ
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
have *mset* ((δ # Δ) @ Φ) = *mset* ((Δ @ Φ) @ [δ]) **by** *simp*
with *Cons.prems* **have** ((δ # Δ) @ Γ) $⊢ ((Δ @ Φ) @ [δ])
by (*metis segmented-msub-weaken*
subset-mset.dual-order.refl)
from *this* **obtain** Σ **where** Σ:
mset (*map snd* Σ) ⊆# *mset* ((δ # Δ) @ Γ)
map (*uncurry* (⊔)) Σ $⊢ (Δ @ Φ)
map (*uncurry* (→)) Σ @ ((δ # Δ) @ Γ) ⊖ *map snd* Σ $⊢ [δ]
by (*metis append-assoc segmented-deduction-generalized-witness*)
show *?case*
proof (*cases find* (λ σ. *snd* σ = δ) Σ = *None*)
case *True*
hence δ ∉ *set* (*map snd* Σ)
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
then show *?case* **by** (*cases snd* σ = δ, *simp*+)
qed
with Σ(*1*) **have** *mset* (*map snd* Σ) ⊆# *mset* (Δ @ Γ)

by (*simp*, *metis add-mset-add-single*
diff-single-trivial
mset-map
set-mset-mset
subset-eq-diff-conv)
thus *?thesis*
using *segmented-stronger-theory-left-monotonic*
witness-weaker-theory
Cons.hyps $\Sigma(2)$
by *blast*
next
case *False*
from *this* **obtain** σ χ **where**
σ: $\sigma = (\chi, \delta)$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
let $?\Sigma' = remove1\ \sigma\ \Sigma$
let $?\Sigma_A = map\ (uncurry\ (\sqcup))\ ?\Sigma'$
let $?\Sigma_B = map\ (uncurry\ (\rightarrow))\ ?\Sigma'$
have $mset\ \Sigma = mset\ (?\Sigma'\ @\ [(\chi, \delta)])$
$mset\ \Sigma = mset\ ((\chi, \delta)\ \#\ ?\Sigma')$
using σ **by** *simp+*
hence $mset\ (map\ (uncurry\ (\sqcup))\ \Sigma) = mset\ (map\ (uncurry\ (\sqcup))\ (?\Sigma'\ @\ [(\chi, \delta)]))$
$mset\ (map\ snd\ \Sigma) = mset\ (map\ snd\ ((\chi, \delta)\ \#\ ?\Sigma'))$
$mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma) = mset\ (map\ (uncurry\ (\rightarrow))\ ((\chi, \delta)\ \#\ ?\Sigma'))$
by (*metis mset-map*)+
hence $mset\ (map\ (uncurry\ (\sqcup))\ \Sigma) = mset\ (?\Sigma_A\ @\ [\chi \sqcup \delta])$
$mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ ((\delta\ \#\ \Delta)\ @\ \Gamma) \ominus map\ snd\ \Sigma)$
$= mset\ (\chi \rightarrow \delta\ \#\ ?\Sigma_B\ @\ (\Delta\ @\ \Gamma) \ominus map\ snd\ ?\Sigma')$
by *simp+*
hence
$?\Sigma_A\ @\ [\chi \sqcup \delta]\ \$\vdash (\Delta\ @\ \Phi)$
$\chi \rightarrow \delta\ \#\ (?\Sigma_B\ @\ (\Delta\ @\ \Gamma) \ominus map\ snd\ ?\Sigma')\ \$\vdash [\delta]$
using $\Sigma(2)$ $\Sigma(3)$
by (*metis segmented-msub-left-monotonic subset-mset.dual-order.refl*, *simp*)
moreover
have $\vdash ((\chi \rightarrow \delta) \rightarrow \delta) \rightarrow (\chi \sqcup \delta)$
unfolding *disjunction-def*
using *modus-ponens*
pseudo-scotus
flip-hypothetical-syllogism
by *blast*
ultimately have $(?\Sigma_A\ @\ ?\Sigma_B\ @\ (\Delta\ @\ \Gamma) \ominus map\ snd\ ?\Sigma')\ \$\vdash (\Delta\ @\ \Phi)$
using *segmented-deduction-one-collapse*
list-deduction-theorem

list-deduction-modus-ponens
list-deduction-weaken
forward-direction
segmented-transitive
by *meson*
moreover
have $\delta = snd\ \sigma$
$snd\ \sigma \in set\ (map\ snd\ \Sigma)$
by (*simp add*: $\sigma(1)$, *simp add*: $\sigma(2)$)
with $\Sigma(1)$ **have** $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) \subseteq\#\ mset\ (remove1\ \delta\ ((\delta$
$\#\ \Delta)\ @\ \Gamma))$
by (*metis insert-DiffM*
insert-subset-eq-iff
mset-remove1
$\sigma(1)\ \sigma(2)$
remove1-pairs-list-projections-snd
set-mset-mset)
hence $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) \subseteq\#\ mset\ (\Delta\ @\ \Gamma)$ **by** *simp*
ultimately show *?thesis*
using *segmented-witness-left-split Cons.hyps*
by *blast*
qed
qed
}
with *forward-direction* **show** *?thesis* **by** *auto*
qed

lemma (**in** *classical-logic*) *segmented-biconditional-cancel*:
assumes $\vdash \gamma \leftrightarrow \varphi$
shows $(\gamma\ \#\ \Gamma)\ \$\vdash\ (\varphi\ \#\ \Phi) = \Gamma\ \$\vdash\ \Phi$
proof −
from *assms* **have** $(\gamma\ \#\ \Phi) \preceq (\varphi\ \#\ \Phi)$ $(\varphi\ \#\ \Phi) \preceq (\gamma\ \#\ \Phi)$
unfolding *biconditional-def*
by (*simp add*: *stronger-theory-left-right-cons*)+
hence $(\gamma\ \#\ \Phi)\ \$\vdash\ (\varphi\ \#\ \Phi)$
$(\varphi\ \#\ \Phi)\ \$\vdash\ (\gamma\ \#\ \Phi)$
using *segmented-stronger-theory-intro* **by** *blast*+
moreover
have $\Gamma\ \$\vdash\ \Phi = (\gamma\ \#\ \Gamma)\ \$\vdash\ (\gamma\ \#\ \Phi)$
by (*metis append-Cons append-Nil segmented-cancel*)+
ultimately
have $\Gamma\ \$\vdash\ \Phi \Longrightarrow \gamma\ \#\ \Gamma\ \$\vdash\ (\varphi\ \#\ \Phi)$
$\gamma\ \#\ \Gamma\ \$\vdash\ (\varphi\ \#\ \Phi) \Longrightarrow \Gamma\ \$\vdash\ \Phi$
using *segmented-transitive* **by** *blast*+
thus *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *right-segmented-sub*:
assumes $\vdash \varphi \leftrightarrow \psi$

shows Γ $\Vdash (\varphi \# \Phi) = \Gamma$ $\Vdash (\psi \# \Phi)$
proof –
have Γ $\Vdash (\varphi \# \Phi) = (\psi \# \Gamma)$ $\Vdash (\psi \# \varphi \# \Phi)$
using *segmented-cancel* [**where** $\Delta=[\psi]$ **and** $\Gamma=\Gamma$ **and** $\Phi=\varphi \# \Phi$] **by** *simp*
also have ... $= (\psi \# \Gamma)$ $\Vdash (\varphi \# \psi \# \Phi)$
using *segmented-cons-cons-right-permute* **by** *blast*
also have ... $= \Gamma$ $\Vdash (\psi \# \Phi)$
using *assms biconditional-symmetry-rule segmented-biconditional-cancel* **by** *blast*
finally show *?thesis* **.**
qed

lemma (**in** *classical-logic*) *left-segmented-sub*:
assumes $\vdash \gamma \leftrightarrow \chi$
shows $(\gamma \# \Gamma)$ $\Vdash \Phi = (\chi \# \Gamma)$ $\Vdash \Phi$
proof –
have $(\gamma \# \Gamma)$ $\Vdash \Phi = (\chi \# \gamma \# \Gamma)$ $\Vdash (\chi \# \Phi)$
using *segmented-cancel* [**where** $\Delta=[\chi]$ **and** $\Gamma=(\gamma \# \Gamma)$ **and** $\Phi=\Phi$] **by** *simp*
also have ... $= (\gamma \# \chi \# \Gamma)$ $\Vdash (\chi \# \Phi)$
by (*metis segmented-msub-left-monotonic mset-eq-perm perm.swap subset-mset.dual-order.refl*)
also have ... $= (\chi \# \Gamma)$ $\Vdash \Phi$
using *assms biconditional-symmetry-rule segmented-biconditional-cancel* **by** *blast*
finally show *?thesis* **.**
qed

lemma (**in** *classical-logic*) *right-segmented-sum-rule*:
Γ $\Vdash (\alpha \# \beta \# \Phi) = \Gamma$ $\Vdash (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Phi)$
proof –
have A: *mset* $(\alpha \sqcup \beta \# \beta \to \alpha \# \beta \# \Phi) =$ *mset* $(\beta \to \alpha \# \beta \# \alpha \sqcup \beta \# \Phi)$ **by** *simp*
have B: $\vdash (\beta \to \alpha) \leftrightarrow (\beta \to (\alpha \sqcap \beta))$
proof –
let $?\varphi = (\langle\beta\rangle \to \langle\alpha\rangle) \leftrightarrow (\langle\beta\rangle \to (\langle\alpha\rangle \sqcap \langle\beta\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
have C: $\vdash \beta \leftrightarrow (\beta \sqcup (\alpha \sqcap \beta))$
proof –
let $?\varphi = \langle\beta\rangle \leftrightarrow (\langle\beta\rangle \sqcup (\langle\alpha\rangle \sqcap \langle\beta\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
have Γ $\Vdash (\alpha \# \beta \# \Phi) = \Gamma$ $\Vdash (\beta \sqcup \alpha \# \beta \to \alpha \# \beta \# \Phi)$
using *segmented-formula-right-split* **by** *blast*
also have ... $= \Gamma$ $\Vdash (\alpha \sqcup \beta \# \beta \to \alpha \# \beta \# \Phi)$
using *disjunction-commutativity right-segmented-sub* **by** *blast*

also have ... = Γ $\$\vdash$ ($\beta \to \alpha$ # β # $\alpha \sqcup \beta$ # Φ)
by (*metis A segmented-msub-weaken subset-mset.dual-order.refl*)
also have ... = Γ $\$\vdash$ ($\beta \to (\alpha \sqcap \beta)$ # β # $\alpha \sqcup \beta$ # Φ)
using *B right-segmented-sub* by *blast*
also have ... = Γ $\$\vdash$ (β # $\beta \to (\alpha \sqcap \beta)$ # $\alpha \sqcup \beta$ # Φ)
using *segmented-cons-cons-right-permute* by *blast*
also have ... = Γ $\$\vdash$ ($\beta \sqcup (\alpha \sqcap \beta)$ # $\beta \to (\alpha \sqcap \beta)$ # $\alpha \sqcup \beta$ # Φ)
using *C right-segmented-sub* by *blast*
also have ... = Γ $\$\vdash$ ($\alpha \sqcap \beta$ # $\alpha \sqcup \beta$ # Φ)
using *segmented-formula-right-split* by *blast*
finally show *?thesis*
using *segmented-cons-cons-right-permute* by *blast*
qed

lemma (in *classical-logic*) *left-segmented-sum-rule*:
(α # β # Γ) $\$\vdash$ Φ = ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) $\$\vdash$ Φ
proof −
have $\star$: *mset* ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # α # β # Γ) = *mset* (α # β # $\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) by *simp*
have (α # β # Γ) $\$\vdash$ Φ = ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # α # β # Γ) $\$\vdash$ ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Φ)
using *segmented-cancel* [where Δ=[$\alpha \sqcup \beta$, $\alpha \sqcap \beta$] and Γ=(α # β # Γ) and Φ=Φ] by *simp*
also have ... = ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # α # β # Γ) $\$\vdash$ (α # β # Φ)
using *right-segmented-sum-rule* by *blast*
also have ... = (α # β # $\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) $\$\vdash$ (α # β # Φ)
by (*metis* $\star$ *segmented-msub-left-monotonic subset-mset.dual-order.refl*)
also have ... = ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) $\$\vdash$ Φ
using *segmented-cancel* [where Δ=[α, β] and Γ=($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) and Φ=Φ] by *simp*
finally show *?thesis* .
qed

lemma (in *classical-logic*) *segmented-exchange*:
(γ # Γ) $\$\vdash$ (φ # Φ) = ($\varphi \to \gamma$ # Γ) $\$\vdash$ ($\gamma \to \varphi$ # Φ)
proof −
have (γ # Γ) $\$\vdash$ (φ # Φ)
= ($\varphi \sqcup \gamma$ # $\varphi \to \gamma$ # Γ) $\$\vdash$ ($\gamma \sqcup \varphi$ # $\gamma \to \varphi$ # Φ)
using *segmented-formula-left-split*
segmented-formula-right-split
by *blast+*
thus *?thesis*
using *segmented-biconditional-cancel*
disjunction-commutativity
by *blast*
qed

lemma (in *classical-logic*) *segmented-negation-swap*:
Γ $\$\vdash$ (φ # Φ) = ($\sim \varphi$ # Γ) $\$\vdash$ ($\bot$ # Φ)

proof −
  have Γ \$⊢ $(\varphi \# \Phi) = (\bot \# \Gamma)$ \$⊢ $(\bot \# \varphi \# \Phi)$
    by (*metis append-Cons append-Nil segmented-cancel*)
  also have ... = $(\bot \# \Gamma)$ \$⊢ $(\varphi \# \bot \# \Phi)$
    using *segmented-cons-cons-right-permute* **by** *blast*
  also have ... = $(\sim \varphi \# \Gamma)$ \$⊢ $(\bot \to \varphi \# \bot \# \Phi)$
    unfolding *negation-def*
    using *segmented-exchange*
    by *blast*
  also have ... = $(\sim \varphi \# \Gamma)$ \$⊢ $(\bot \# \Phi)$
    using *ex-falso-quodlibet*
        segmented-tautology-right-cancel
    by *blast*
  finally show *?thesis* **.**
qed

primrec (**in** *classical-logic*)
  stratified-deduction :: *'a list* ⇒ *nat* ⇒ *'a* ⇒ *bool* (- #⊢ - - [*60*,*100*,*59*] *60*)
  where
    Γ #⊢ *0* φ = *True*
  | Γ #⊢ (*Suc n*) φ = ($\exists$ Ψ. *mset* (*map snd* Ψ) ⊆# *mset* Γ $\wedge$
                map (*uncurry* (⊔)) Ψ :⊢ φ $\wedge$
                map (*uncurry* (→)) Ψ @ Γ ⊖ (*map snd* Ψ) #⊢ *n* φ)

lemma (**in** *classical-logic*) *stratified-segmented-deduction-replicate*:
  Γ #⊢ *n* φ = Γ \$⊢ (*replicate n* φ)
proof −
  have $\forall$ Γ. Γ #⊢ *n* φ = Γ \$⊢ (*replicate n* φ)
    by (*induct n*, *simp+*)
  thus *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *stratified-deduction-tautology-weaken*:
  assumes ⊢ φ
  shows Γ #⊢ *n* φ
proof (*induct n*)
  case *0*
  then show *?case* **by** *simp*
next
  case (*Suc n*)
  hence Γ \$⊢ (φ # *replicate n* φ)
    using *assms*
        stratified-segmented-deduction-replicate
        segmented-tautology-right-cancel
    by *blast*
  hence Γ \$⊢ *replicate* (*Suc n*) φ
    by *simp*
  then show *?case*
    using *stratified-segmented-deduction-replicate*

```
    by blast
qed

lemma (in classical-logic) stratified-deduction-weaken:
  assumes n ≤ m
      and Γ #⊢ m φ
    shows Γ #⊢ n φ
proof −
  have Γ $⊢ replicate m φ
    using assms(2) stratified-segmented-deduction-replicate
    by blast
  hence Γ $⊢ replicate n φ
    by (metis append-Nil2
              assms(1)
              le-iff-add
              segmented-deduction.simps(1)
              segmented-deduction-generalized-witness
              replicate-add)
  thus ?thesis
    using stratified-segmented-deduction-replicate
    by blast
qed

lemma (in classical-logic) stratified-deduction-implication:
  assumes ⊢ φ → ψ
     and Γ #⊢ n φ
   shows Γ #⊢ n ψ
proof −
  have replicate n ψ ≼ replicate n φ
    using stronger-theory-left-right-cons assms(1)
    by (induct n, auto)
  thus ?thesis
    using assms(2)
          segmented-stronger-theory-right-antitonic
          stratified-segmented-deduction-replicate
    by blast
qed

theorem (in classical-logic) segmented-stratified-falsum-equiv:
  Γ $⊢ Φ = (∼ Φ @ Γ) #⊢ (length Φ) ⊥
proof −
  have ∀ Γ Ψ. Γ $⊢ (Φ @ Ψ) = (∼ Φ @ Γ) $⊢ (replicate (length Φ) ⊥ @ Ψ)
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons φ Φ)
    {
      fix Γ Ψ
```

have Γ \$⊢ ((φ # Φ) @ Ψ) = ($\sim \varphi$ # Γ) \$⊢ ($\bot$ # Φ @ Ψ)
using *segmented-negation-swap* **by** *auto*
moreover have *mset* (Φ @ ($\bot$ # Ψ)) = *mset* ($\bot$ # Φ @ Ψ)
by *simp*
ultimately have Γ \$⊢ ((φ # Φ) @ Ψ) = ($\sim \varphi$ # Γ) \$⊢ (Φ @ ($\bot$ # Ψ))
by (*metis segmented-msub-weaken subset-mset.order-refl*)
hence Γ \$⊢ ((φ # Φ) @ Ψ) = (**∼** Φ @ ($\sim \varphi$ # Γ)) \$⊢ (*replicate* (*length* Φ) $\bot$ @ ($\bot$ # Ψ))
using *Cons*
by *blast*
moreover have *mset* (**∼** Φ @ ($\sim \varphi$ # Γ)) = *mset* (**∼** (φ # Φ) @ Γ)
mset (*replicate* (*length* Φ) $\bot$ @ ($\bot$ # Ψ))
= *mset* (*replicate* (*length* (φ # Φ)) $\bot$ @ Ψ)
by *simp+*
ultimately have
Γ \$⊢ ((φ # Φ) @ Ψ) = **∼** (φ # Φ) @ Γ \$⊢ (*replicate* (*length* (φ # Φ)) $\bot$ @ Ψ)
by (*metis append.assoc*
append-Cons
append-Nil
length-Cons
replicate-append-same
list-subtract.simps(*1*)
map-ident replicate-Suc
segmented-msub-left-monotonic
map-list-subtract-mset-containment)
}
then show *?case* **by** *blast*
qed
thus *?thesis*
by (*metis append-Nil2 stratified-segmented-deduction-replicate*)
qed

2.5.2 MaxSAT

definition (**in** *implication-logic*) *unproving-core* :: *'a list* ⇒ *'a* ⇒ *'a list set* ($\mathcal{C}$)
where
$\mathcal{C}$ Γ φ = {Φ. *mset* Φ ⊆# *mset* Γ
∧ ¬ Φ :⊢ φ
∧ (∀ Ψ. *mset* Ψ ⊆# *mset* Γ ⟶ ¬ Ψ :⊢ φ ⟶ *length* Ψ ≤ *length* Φ)}

lemma (**in** *implication-logic*) *unproving-core-finite*:
finite ($\mathcal{C}$ Γ φ)
proof –
{
fix Φ
assume $\Phi \in \mathcal{C}$ Γ φ
hence *set* Φ ⊆ *set* Γ

```
        length Φ ≤ length Γ
      unfolding unproving-core-def
      using mset-subset-eqD
            length-sub-mset
            mset-eq-length
      by fastforce+
  }
  hence 𝒞 Γ φ ⊆ {xs. set xs ⊆ set Γ ∧ length xs ≤ length Γ}
    by auto
  moreover
  have finite {xs. set xs ⊆ set Γ ∧ length xs ≤ length Γ}
    using finite-lists-length-le by blast
  ultimately show ?thesis using rev-finite-subset by auto
qed

lemma (in implication-logic) unproving-core-existence:
  (¬ ⊢ φ) = (∃ Σ. Σ ∈ 𝒞 Γ φ)
proof (rule iffI)
  assume ¬ ⊢ φ
  show ∃Σ. Σ ∈ 𝒞 Γ φ
  proof (rule ccontr)
    assume ∄Σ. Σ ∈ 𝒞 Γ φ
    hence ◇: ∀ Φ. mset Φ ⊆# mset Γ ⟶
                ¬ Φ :⊢ φ ⟶
                (∃Ψ. mset Ψ ⊆# mset Γ ∧ ¬ Ψ :⊢ φ ∧ length Ψ > length Φ)
      unfolding unproving-core-def
      by fastforce
    {
      fix n
      have ∃ Ψ. mset Ψ ⊆# mset Γ ∧ ¬ Ψ :⊢ φ ∧ length Ψ > n
        using ◇
        by (induct n,
            metis ‹¬ ⊢ φ›
                  list-deduction-base-theory
                  mset.simps(1)
                  neq0-conv
                  subset-mset.bot.extremum,
            fastforce)
    }
    hence ∃ Ψ. mset Ψ ⊆# mset Γ ∧ length Ψ > length Γ
      by auto
    thus False
      using size-mset-mono by fastforce
  qed
next
  assume ∃Σ. Σ ∈ 𝒞 Γ φ
  thus ¬ ⊢ φ
    unfolding unproving-core-def
    using list-deduction-weaken
```

    by *blast*
qed

lemma (**in** *implication-logic*) *unproving-core-complement-deduction*:
  assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
    and $\psi \in$ *set* $(\Gamma \ominus \Phi)$
  shows $\Phi :\vdash \psi \to \varphi$
proof (*rule ccontr*)
  assume $\neg\ \Phi :\vdash \psi \to \varphi$
  hence $\neg\ (\psi\ \#\ \Phi) :\vdash \varphi$
    by (*simp add*: *list-deduction-theorem*)
  moreover
  have *mset* $\Phi \subseteq\#$ *mset* Γ $\psi \in\#$ *mset* $(\Gamma \ominus \Phi)$
    using *assms*
    unfolding *unproving-core-def*
    by (*blast*, *meson in-multiset-in-set*)
  hence *mset* $(\psi\ \#\ \Phi) \subseteq\#$ *mset* Γ
    by (*simp*, *metis add-mset-add-single*
        mset-subset-eq-mono-add-left-cancel
        mset-subset-eq-single
        subset-mset.add-diff-inverse)
  ultimately have *length* $(\psi\ \#\ \Phi) \leq$ *length* (Φ)
    using *assms*
    unfolding *unproving-core-def*
    by *blast*
  thus *False*
    by *simp*
qed

lemma (**in** *implication-logic*) *unproving-core-set-complement* [*simp*]:
  assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
  shows *set* $(\Gamma \ominus \Phi) =$ *set* $\Gamma -$ *set* Φ
proof (*rule equalityI*)
  show *set* $(\Gamma \ominus \Phi) \subseteq$ *set* $\Gamma -$ *set* Φ
  proof (*rule subsetI*)
    fix ψ
    assume $\psi \in$ *set* $(\Gamma \ominus \Phi)$
    moreover from *this* **have** $\Phi :\vdash \psi \to \varphi$
      using *assms*
      using *unproving-core-complement-deduction*
      by *blast*
    hence $\psi \notin$ *set* Φ
      using *assms*
        list-deduction-modus-ponens
        list-deduction-reflection
        unproving-core-def
      by *blast*
    ultimately show $\psi \in$ *set* $\Gamma -$ *set* Φ
      using *list-subtract-set-trivial-upper-bound* [**where** Γ=Γ **and** Φ=Φ]

by *blast*
qed
next
show *set* Γ − *set* $\Phi \subseteq$ *set* $(\Gamma \ominus \Phi)$
by (*simp add*: *list-subtract-set-difference-lower-bound*)
qed

lemma (**in** *implication-logic*) *unproving-core-complement-equiv*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
and $\psi \in$ *set* Γ
shows $\Phi :\vdash \psi \to \varphi = (\psi \notin$ *set* $\Phi)$
proof (*rule iffI*)
assume $\Phi :\vdash \psi \to \varphi$
thus $\psi \notin$ *set* Φ
using *assms*(*1*)
list-deduction-modus-ponens
list-deduction-reflection
unproving-core-def
by *blast*
next
assume $\psi \notin$ *set* Φ
thus $\Phi :\vdash \psi \to \varphi$
using *assms unproving-core-complement-deduction*
by *auto*
qed

lemma (**in** *implication-logic*) *unproving-length-equiv*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
and $\Psi \in \mathcal{C}\ \Gamma\ \varphi$
shows *length* Φ = *length* Ψ
using *assms*
by (*simp add*: *dual-order.antisym unproving-core-def*)

lemma (**in** *implication-logic*) *unproving-list-subtract-length-equiv*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
and $\Psi \in \mathcal{C}\ \Gamma\ \varphi$
shows *length* $(\Gamma \ominus \Phi)$ = *length* $(\Gamma \ominus \Psi)$
proof −
have *length* Φ = *length* Ψ
using *assms unproving-length-equiv*
by *blast*
moreover
have *mset* $\Phi \subseteq\#$ *mset* Γ
mset $\Psi \subseteq\#$ *mset* Γ
using *assms unproving-core-def* **by** *blast*+
hence *length* $(\Gamma \ominus \Phi)$ = *length* Γ − *length* Φ
length $(\Gamma \ominus \Psi)$ = *length* Γ − *length* Ψ
by (*metis list-subtract-mset-homomorphism size-Diff-submset size-mset*)+
ultimately show *?thesis* **by** *metis*

qed

lemma (**in** *implication-logic*) *unproving-core-max-list-deduction*:
  $\Gamma :\vdash \varphi = (\forall\ \Phi \in \mathcal{C}\ \Gamma\ \varphi.\ \mathit{1} \leq \mathit{length}\ (\Gamma \ominus \Phi))$
proof *cases*
  assume $\vdash \varphi$
  hence $\Gamma :\vdash \varphi\ \mathcal{C}\ \Gamma\ \varphi = \{\}$
    unfolding *unproving-core-def*
    by (*simp add*: *list-deduction-weaken*)+
  then show *?thesis* **by** *blast*
next
  assume $\neg \vdash \varphi$
  from *this* **obtain** Ω **where** Ω: $\Omega \in \mathcal{C}\ \Gamma\ \varphi$
    using *unproving-core-existence* **by** *blast*
  from *this* **have** *mset* $\Omega \subseteq\#$ *mset* Γ
    unfolding *unproving-core-def* **by** *blast*
  hence $\diamondsuit$: *length* $(\Gamma \ominus \Omega) =$ *length* $\Gamma -$ *length* Ω
    by (*metis list-subtract-mset-homomorphism*
            size-Diff-submset
            size-mset)
  show *?thesis*
  proof (*cases* $\Gamma :\vdash \varphi$)
    assume $\Gamma :\vdash \varphi$
    from Ω **have** *mset* $\Omega \subset\#$ *mset* Γ
      by (*metis* (*no-types*, *lifting*)
              Diff-cancel
              Diff-eq-empty-iff
              ‹$\Gamma :\vdash \varphi$›
              list-deduction-monotonic
              unproving-core-def
              mem-Collect-eq
              mset-eq-setD
              subset-mset.dual-order.not-eq-order-implies-strict)
    hence *length* $\Omega <$ *length* Γ
      using *mset-subset-size* **by** *fastforce*
    hence $\mathit{1} \leq$ *length* $\Gamma -$ *length* Ω
      by (*simp add*: *Suc-leI*)
    with $\diamondsuit$ **have** $\mathit{1} \leq$ *length* $(\Gamma \ominus \Omega)$
      by *simp*
    with ‹$\Gamma :\vdash \varphi$› Ω **show** *?thesis*
      by (*metis unproving-list-subtract-length-equiv*)
  next
    assume $\neg\ \Gamma :\vdash \varphi$
    moreover have *mset* $\Gamma \subseteq\#$ *mset* Γ
      by *simp*
    moreover have *length* $\Omega \leq$ *length* Γ
      using ‹*mset* $\Omega \subseteq\#$ *mset* Γ› *length-sub-mset mset-eq-length*
      by *fastforce*
    ultimately have *length* $\Omega =$ *length* Γ

using Ω
unfolding *unproving-core-def*
by (*simp add*: *dual-order.antisym*)
hence $1 > length\ (\Gamma \ominus \Omega)$
using $\diamondsuit$
by *simp*
with $\langle\neg\ \Gamma :\vdash \varphi\rangle\ \Omega$ **show** *?thesis*
by *fastforce*
qed
qed

2.6 Abstract MaxSAT

definition (**in** *implication-logic*) *core-size* :: $'a\ list \Rightarrow 'a \Rightarrow nat$ (| - |- [45])
where
$(|\ \Gamma\ |_\varphi) = (if\ \mathcal{C}\ \Gamma\ \varphi = \{\}\ then\ 0\ else\ Max\ \{\ length\ \Phi\ |\ \Phi.\ \Phi \in \mathcal{C}\ \Gamma\ \varphi\ \})$

abbreviation (**in** *classical-logic*) *MaxSAT* :: $'a\ list \Rightarrow nat$
where
$MaxSAT\ \Gamma \equiv |\ \Gamma\ |_\bot$

definition (**in** *implication-logic*) *complement-core-size* :: $'a\ list \Rightarrow 'a \Rightarrow nat$ (‖ - ‖- [45])
where
$(\|\ \Gamma\ \|_\varphi) = length\ \Gamma - |\ \Gamma\ |_\varphi$

lemma (**in** *implication-logic*) *core-size-intro*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
shows $length\ \Phi = |\ \Gamma\ |_\varphi$
proof –
have $\forall\ n \in \{\ length\ \Psi\ |\ \Psi.\ \Psi \in \mathcal{C}\ \Gamma\ \varphi\ \}.\ n \leq length\ \Phi$
$length\ \Phi \in \{\ length\ \Psi\ |\ \Psi.\ \Psi \in \mathcal{C}\ \Gamma\ \varphi\ \}$
using *assms unproving-core-def*
by *auto*
moreover
have *finite* $\{\ length\ \Psi\ |\ \Psi.\ \Psi \in \mathcal{C}\ \Gamma\ \varphi\ \}$
using *finite-imageI unproving-core-finite*
by *simp*
ultimately have $Max\ \{\ length\ \Psi\ |\ \Psi.\ \Psi \in \mathcal{C}\ \Gamma\ \varphi\ \} = length\ \Phi$
using *Max-eqI*
by *blast*
thus *?thesis*
using *assms core-size-def*
by *auto*
qed

lemma (**in** *implication-logic*) *complement-core-size-intro*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
shows $length\ (\Gamma \ominus \Phi) = \|\ \Gamma\ \|_\varphi$

proof −
  have *mset* $\Phi \subseteq\#$ *mset* Γ
    using *assms*
    unfolding *unproving-core-def*
    by *auto*
  moreover from *this* **have** *length* $(\Gamma \ominus \Phi)$ = *length* Γ − *length* Φ
    by (*metis list-subtract-mset-homomorphism size-Diff-submset size-mset*)
  ultimately show *?thesis*
    unfolding *complement-core-size-def*
    by (*metis assms core-size-intro*)
qed

lemma (**in** *implication-logic*) *length-core-decomposition*:
  length $\Gamma = (\mid \Gamma \mid_{\varphi}) + \parallel \Gamma \parallel_{\varphi}$
proof (*cases* $\mathcal{C}\ \Gamma\ \varphi = \{\}$)
  case *True*
  then show *?thesis*
    unfolding *core-size-def*
          complement-core-size-def
    by *simp*
next
  case *False*
  from *this* **obtain** Φ **where** $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
    by *fast*
  moreover from *this* **have** *mset* $\Phi \subseteq\#$ *mset* Γ
    unfolding *unproving-core-def*
    by *auto*
  moreover from *this* **have** *length* $(\Gamma \ominus \Phi)$ = *length* Γ − *length* Φ
    by (*metis list-subtract-mset-homomorphism size-Diff-submset size-mset*)
  ultimately show *?thesis*
    unfolding *complement-core-size-def*
    using *list-subtract-msub-eq core-size-intro*
    by *fastforce*
qed

primrec *core-optimal-pre-witness* :: *'a list* ⇒ (*'a list* × *'a*) *list* ($\mathfrak{V}$)
  where
    $\mathfrak{V}$ [] = []
  | $\mathfrak{V}$ $(\psi \# \Psi) = (\Psi, \psi) \# \mathfrak{V}\ \Psi$

lemma *core-optimal-pre-witness-element-inclusion*:
  $\forall\ (\Delta,\delta) \in$ *set* $(\mathfrak{V}\ \Psi)$. *set* $(\mathfrak{V}\ \Delta) \subseteq$ *set* $(\mathfrak{V}\ \Psi)$
  by (*induct* Ψ, *fastforce+*)

lemma *core-optimal-pre-witness-nonelement*:
  assumes *length* $\Delta \geq$ *length* Ψ
  shows $(\Delta,\delta) \notin$ *set* $(\mathfrak{V}\ \Psi)$
  using *assms*
proof (*induct* Ψ)

case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
hence $\Psi \neq \Delta$ **by** *auto*
then show *?case* **using** *Cons* **by** *simp*
qed

lemma *core-optimal-pre-witness-distinct*: *distinct* ($\mathfrak{V}$ Ψ)
by (*induct* Ψ, *simp*, *simp add*: *core-optimal-pre-witness-nonelement*)

lemma *core-optimal-pre-witness-length-iff-eq*:
$\forall$ (Δ,δ) $\in$ *set* ($\mathfrak{V}$ Ψ). $\forall$ (Σ,σ) $\in$ *set* ($\mathfrak{V}$ Ψ). (*length* Δ = *length* Σ) = ((Δ, δ) = (Σ,σ))
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
{
fix Δ
fix δ
assume (Δ,δ) $\in$ *set* ($\mathfrak{V}$ (ψ # Ψ))
and *length* Δ = *length* Ψ
hence (Δ,δ) = (Ψ, ψ)
by (*simp add*: *core-optimal-pre-witness-nonelement*)
}
hence $\forall$ (Δ,δ) $\in$ *set* ($\mathfrak{V}$ (ψ # Ψ)). (*length* Δ = *length* Ψ) = ((Δ,δ) = (Ψ,ψ))
by *blast*
with *Cons* **show** *?case*
by *auto*
qed

lemma *mset-distinct-msub-down*:
assumes *mset* A $\subseteq$# *mset* B
and *distinct* B
shows *distinct* A
using *assms*
by (*meson distinct-append mset-le-perm-append perm-distinct-iff*)

lemma *mset-remdups-set-sub-iff*:
(*mset* (*remdups* A) $\subseteq$# *mset* (*remdups* B)) = (*set* A $\subseteq$ *set* B)
proof −
have $\forall B$. (*mset* (*remdups* A) $\subseteq$# *mset* (*remdups* B)) = (*set* A $\subseteq$ *set* B)
proof (*induct* A)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* *a* A)

```
    then show ?case
    proof (cases a ∈ set A)
      case True
      then show ?thesis using Cons by auto
    next
      case False
      {
        fix B
        have (mset (remdups (a # A)) ⊆# mset (remdups B)) = (set (a # A) ⊆
set B)
        proof (rule iffI)
          assume assm: mset (remdups (a # A)) ⊆# mset (remdups B)
          hence mset (remdups A) ⊆# mset (remdups B) − {#a#}
            using False
            by (simp add: insert-subset-eq-iff)
          hence mset (remdups A) ⊆# mset (remdups (removeAll a B))
            by (metis diff-subset-eq-self
                      distinct-remdups
                      distinct-remove1-removeAll
                      mset-distinct-msub-down
                      mset-remove1
                      set-eq-iff-mset-eq-distinct
                      set-remdups set-removeAll)
          hence set A ⊆ set (removeAll a B)
            using Cons.hyps by blast
          moreover from assm False have a ∈ set B
            using mset-subset-eq-insertD by fastforce
          ultimately show set (a # A) ⊆ set B
            by auto
        next
          assume assm: set (a # A) ⊆ set B
          hence set A ⊆ set (removeAll a B) using False
            by auto
          hence mset (remdups A) ⊆# mset (remdups B) − {#a#}
            by (metis Cons.hyps
                      distinct-remdups
                      mset-remdups-subset-eq
                      mset-remove1 remove-code(1)
                      set-remdups set-remove1-eq
                      set-removeAll
                      subset-mset.dual-order.trans)
          moreover from assm False have a ∈ set B by auto
          ultimately show mset (remdups (a # A)) ⊆# mset (remdups B)
            by (simp add: False insert-subset-eq-iff)
        qed
      }
      then show ?thesis by simp
    qed
  qed
```

thus *?thesis* **by** *blast*
qed

lemma *range-characterization*:
shows $(mset\ X = mset\ [0..<length\ X]) = (distinct\ X \wedge (\forall\ x \in set\ X.\ x < length\ X))$
proof (*rule iffI*)
assume $mset\ X = mset\ [0..<length\ X]$
thus $distinct\ X \wedge (\forall x \in set\ X.\ x < length\ X)$
by (*metis atLeastLessThan-iff count-mset-0-iff distinct-count-atmost-1 distinct-upt set-upt*)
next
assume $distinct\ X \wedge (\forall x \in set\ X.\ x < length\ X)$
moreover
{
fix *n*
have $\forall\ X.\ n = length\ X \longrightarrow$
$distinct\ X \wedge (\forall x \in set\ X.\ x < length\ X) \longrightarrow$
$mset\ X = mset\ [0..<length\ X]$
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix *X*
assume *A*: $n + 1 = length\ X$
and *B*: *distinct X*
and *C*: $\forall x \in set\ X.\ x < length\ X$
have $n \in set\ X$
proof (*rule ccontr*)
assume $n \notin set\ X$
from *A* **have** *A'*: $n = length\ (tl\ X)$
by *simp*
from *B* **have** *B'*: *distinct* (*tl X*)
by (*simp add*: *distinct-tl*)
have *C'*: $\forall x \in set\ (tl\ X).\ x < length\ (tl\ X)$
by (*metis A A' C* ‹$n \notin set\ X$›
Suc-eq-plus1
Suc-le-eq
Suc-le-mono
le-less
list.set-sel(*2*)
list.size(*3*)
nat.simps(*3*))
from *A' B' C' Suc* **have** $mset\ (tl\ X) = mset\ [0..<n]$
by *blast*
from *A* **have** $X = hd\ X\ \#\ tl\ X$
by (*metis Suc-eq-plus1 list.exhaust-sel list.size*(*3*) *nat.simps*(*3*))

```
          with B ‹mset (tl X) = mset [0..<n]› have hd X ∉ set [0..<n]
            by (metis distinct.simps(2) mset-eq-setD)
          hence hd X ≥ n by simp
          with C ‹n ∉ set X› ‹X = hd X # tl X› show False
           by (metis A Suc-eq-plus1 Suc-le-eq le-neq-trans list.set-intros(1) not-less)
        qed
        let ?X' = remove1 n X
        have A': n = length ?X'
          by (metis A ‹n ∈ set X› diff-add-inverse2 length-remove1)
        have B': distinct ?X'
          by (simp add: B)
        have C': ∀x∈set ?X'. x < length ?X'
          by (metis A A' B C
                   DiffE
                   Suc-eq-plus1
                   Suc-le-eq
                   Suc-le-mono
                   le-neq-trans
                   set-remove1-eq
                   singletonI)
        hence mset ?X' = mset [0..<n]
          using A' B' C' Suc
          by auto
        hence mset (n # ?X') = mset [0..<n+1]
          by simp
        hence mset X = mset [0..<length X]
          by (metis A B
                   ‹n ∈ set X›
                   distinct-upt
                   perm-remove
                   perm-set-eq
                   set-eq-iff-mset-eq-distinct
                   set-mset-mset)
      }
      then show ?case by fastforce
    qed
  }
  ultimately show mset X = mset [0..<length X]
    by blast
qed

lemma distinct-pigeon-hole:
  assumes distinct X
      and X ≠ []
    shows ∃ n ∈ set X. n + 1 ≥ length X
proof (rule ccontr)
  assume ⋆: ¬ (∃ n ∈ set X. length X ≤ n + 1)
  hence ∀ n ∈ set X. n < length X by fastforce
  hence mset X = mset [0..<length X]
```

using *assms*(*1*) *range-characterization*
by *fastforce*
with *assms*(*2*) **have** *length X* − *1* ∈ *set X*
by (*metis diff-zero last-in-set last-upt length-greater-0-conv length-upt mset-eq-setD*)
with ⋆ **show** *False*
by (*metis One-nat-def Suc-eq-plus1 Suc-pred le-refl length-pos-if-in-set*)
qed

lemma *core-optimal-pre-witness-pigeon-hole*:
assumes *mset* Σ ⊆# *mset* (𝔙 Ψ)
and Σ ≠ []
shows ∃ (Δ, δ) ∈ *set* Σ. *length* Δ + *1* ≥ *length* Σ
proof −
have *distinct* Σ
using *assms*
core-optimal-pre-witness-distinct
mset-distinct-msub-down
by *blast*
with *assms*(*1*) **have** *distinct* (*map* (*length* ∘ *fst*) Σ)
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
hence *mset* Σ ⊆# *mset* (𝔙 Ψ)
distinct Σ
by (*metis mset.simps*(*2*) *mset-subset-eq-insertD subset-mset-def*, *simp*)
with *Cons.hyps* **have** *distinct* (*map* (λ*a*. *length* (*fst a*)) Σ) **by** *simp*
moreover
obtain δ Δ **where** σ = (Δ, δ)
by *fastforce*
hence (Δ, δ) ∈ *set* (𝔙 Ψ)
using *Cons.prems mset-subset-eq-insertD*
by *fastforce*
hence ∀ (Σ,σ) ∈ *set* (𝔙 Ψ). (*length* Δ = *length* Σ) = ((Δ, δ) = (Σ, σ))
using *core-optimal-pre-witness-length-iff-eq* [**where** Ψ=Ψ]
by *fastforce*
hence ∀ (Σ,σ) ∈ *set* Σ. (*length* Δ = *length* Σ) = ((Δ, δ) = (Σ, σ))
using ‹*mset* Σ ⊆# *mset* (𝔙 Ψ)›
by (*metis* (*no-types*, *lifting*) *Un-iff mset-le-perm-append perm-set-eq set-append*)
hence *length* (*fst* σ) ∉ *set* (*map* (λ*a*. *length* (*fst a*)) Σ)
using *Cons.prems*(*2*) ‹σ = (Δ, δ)›
by *fastforce*
ultimately show *?case* **by** *simp*
qed
moreover have *length* (*map* (*length* ∘ *fst*) Σ) = *length* Σ **by** *simp*
moreover have *map* (*length* ∘ *fst*) Σ ≠ [] **using** *assms* **by** *simp*
ultimately show *?thesis*
using *distinct-pigeon-hole*

  by *fastforce*
qed

abbreviation (**in** *classical-logic*)
  core-optimal-witness :: $'a \Rightarrow 'a\ list \Rightarrow ('a \times 'a)\ list$ ($\mathfrak{W}$)
  where $\mathfrak{W}\ \varphi\ \Xi \equiv map\ (\lambda(\Psi,\psi).\ (\Psi :\to \varphi,\ \psi))\ (\mathfrak{V}\ \Xi)$

abbreviation (**in** *classical-logic*)
  disjunction-core-optimal-witness :: $'a \Rightarrow 'a\ list \Rightarrow 'a\ list$ ($\mathfrak{W}_{\sqcup}$)
  where $\mathfrak{W}_{\sqcup}\ \varphi\ \Psi \equiv map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ \Psi)$

abbreviation (**in** *classical-logic*)
  implication-core-optimal-witness :: $'a \Rightarrow 'a\ list \Rightarrow 'a\ list$ ($\mathfrak{W}_{\to}$)
  where $\mathfrak{W}_{\to}\ \varphi\ \Psi \equiv map\ (uncurry\ (\to))\ (\mathfrak{W}\ \varphi\ \Psi)$

lemma (**in** *classical-logic*) *core-optimal-witness-conjunction-identity*:
  $\vdash \bigsqcap\ (\mathfrak{W}_{\sqcup}\ \varphi\ \Psi) \leftrightarrow (\varphi \sqcup \bigsqcap\ \Psi)$
proof (*induct* Ψ)
  case *Nil*
  then show *?case*
    unfolding *biconditional-def*
              disjunction-def
    using *axiom-k*
          modus-ponens
          verum-tautology
    by (*simp*, *blast*)
next
  case (*Cons* ψ Ψ)
  have $\vdash (\Psi :\to \varphi) \leftrightarrow (\bigsqcap\ \Psi \to \varphi)$
    by (*simp add*: *list-curry-uncurry*)
  hence $\vdash \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Psi)))$
        $\leftrightarrow ((\bigsqcap\ \Psi \to \varphi \sqcup \psi) \sqcap \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ \Psi)))$
    unfolding *biconditional-def*
    using *conjunction-monotonic*
          disjunction-monotonic
    by *simp*
  moreover have $\vdash ((\bigsqcap\ \Psi \to \varphi \sqcup \psi) \sqcap \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ \Psi)))$
                $\leftrightarrow ((\bigsqcap\ \Psi \to \varphi \sqcup \psi) \sqcap (\varphi \sqcup \bigsqcap\ \Psi))$
    using *Cons.hyps biconditional-conjunction-weaken-rule*
    by *blast*
  moreover
  {
    fix φ ψ χ
    have $\vdash ((\chi \to \varphi \sqcup \psi) \sqcap (\varphi \sqcup \chi)) \leftrightarrow (\varphi \sqcup (\psi \sqcap \chi))$
    proof –
      let $?\varphi = ((\langle\chi\rangle \to \langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcup \langle\chi\rangle)) \leftrightarrow (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcap \langle\chi\rangle))$
      have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
      hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
      thus *?thesis* **by** *simp*

```
    qed
  }
  ultimately have ⊢ ⨅ (map (uncurry (⊔)) (𝔚 φ (ψ # Ψ))) ↔ (φ ⊔ (ψ ⊓ ⨅
Ψ))
    using biconditional-transitivity-rule
    by blast
  then show ?case by simp
qed

lemma (in classical-logic) core-optimal-witness-deduction:
  ⊢ 𝔚⊔ φ Ψ :→ φ ↔ Ψ :→ φ
proof −
  have ⊢ 𝔚⊔ φ Ψ :→ φ ↔ (⨅ (𝔚⊔ φ Ψ) → φ)
    by (simp add: list-curry-uncurry)
  moreover
  {
    fix α β γ
    have ⊢ (α ↔ β) → ((α → γ) ↔ (β → γ))
    proof −
      let ?φ = (⟨α⟩ ↔ ⟨β⟩) → ((⟨α⟩ → ⟨γ⟩) ↔ (⟨β⟩ → ⟨γ⟩))
      have ∀𝔐. 𝔐 ⊨prop ?φ by fastforce
      hence ⊢ ⦇ ?φ ⦈ using propositional-semantics by blast
      thus ?thesis by simp
    qed
  }
  ultimately have ⊢ 𝔚⊔ φ Ψ :→ φ ↔ ((φ ⊔ ⨅ Ψ) → φ)
    using modus-ponens
          biconditional-transitivity-rule
          core-optimal-witness-conjunction-identity
    by blast
  moreover
  {
    fix α β
    have ⊢ ((α ⊔ β) → α) ↔ (β → α)
    proof −
      let ?φ = ((⟨α⟩ ⊔ ⟨β⟩) → ⟨α⟩) ↔ (⟨β⟩ → ⟨α⟩)
      have ∀𝔐. 𝔐 ⊨prop ?φ by fastforce
      hence ⊢ ⦇ ?φ ⦈ using propositional-semantics by blast
      thus ?thesis by simp
    qed
  }
  ultimately have ⊢ 𝔚⊔ φ Ψ :→ φ ↔ (⨅ Ψ → φ)
    using biconditional-transitivity-rule by blast
  thus ?thesis
    using biconditional-symmetry-rule
          biconditional-transitivity-rule
          list-curry-uncurry
    by blast
qed
```

lemma (**in** *classical-logic*) *optimal-witness-split-identity*:
$\vdash (\mathfrak{W}_{\sqcup}\ \varphi\ (\psi\ \#\ \Xi)) :\to \varphi \to (\mathfrak{W}_{\to}\ \varphi\ (\psi\ \#\ \Xi)) :\to \varphi \to \Xi :\to \varphi$
proof (*induct* Ξ)
case *Nil*
have $\vdash ((\varphi \sqcup \psi) \to \varphi) \to ((\varphi \to \psi) \to \varphi) \to \varphi$
proof –
let $?\varphi = ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \to \langle\varphi\rangle) \to ((\langle\varphi\rangle \to \langle\psi\rangle) \to \langle\varphi\rangle) \to \langle\varphi\rangle$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
let $?A = \mathfrak{W}_{\sqcup}\ \varphi\ \Xi :\to \varphi$
let $?B = \mathfrak{W}_{\to}\ \varphi\ \Xi :\to \varphi$
let $?X = \Xi :\to \varphi$
from *Cons.hyps* **have** $\vdash ((?X \sqcup \psi) \to ?A) \to ((?X \to \psi) \to ?B) \to ?X$ **by** *simp*
moreover
have $\vdash (((?X \sqcup \psi) \to ?A) \to ((?X \to \psi) \to ?B) \to ?X)$
$\to ((\xi \to ?X \sqcup \psi) \to (?X \sqcup \xi) \to ?A) \to (((\xi \to ?X) \to \psi) \to (?X \to \xi) \to ?B) \to \xi \to ?X$
proof –
let $?\varphi =(((\langle ?X\rangle \sqcup \langle\psi\rangle) \to \langle ?A\rangle) \to ((\langle ?X\rangle \to \langle\psi\rangle) \to \langle ?B\rangle) \to \langle ?X\rangle) \to$
$((\langle\xi\rangle \to \langle ?X\rangle \sqcup \langle\psi\rangle) \to (\langle ?X\rangle \sqcup \langle\xi\rangle) \to \langle ?A\rangle) \to$
$(((\langle\xi\rangle \to \langle ?X\rangle) \to \langle\psi\rangle) \to (\langle ?X\rangle \to \langle\xi\rangle) \to \langle ?B\rangle) \to$
$\langle\xi\rangle \to$
$\langle ?X\rangle$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately
have $\vdash ((\xi \to ?X \sqcup \psi) \to (?X \sqcup \xi) \to ?A) \to (((\xi \to ?X) \to \psi) \to (?X \to \xi) \to ?B) \to \xi \to ?X$
using *modus-ponens*
by *blast*
thus *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *disj-conj-impl-duality*:
$\vdash (\varphi \to \chi \sqcap \psi \to \chi) \leftrightarrow ((\varphi \sqcup \psi) \to \chi)$
proof –
let $?\varphi = (\langle\varphi\rangle \to \langle\chi\rangle \sqcap \langle\psi\rangle \to \langle\chi\rangle) \leftrightarrow ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \to \langle\chi\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*

qed

lemma (**in** *classical-logic*) *weak-disj-of-conj-equiv*:
  $(\forall \sigma \in set\ \Sigma.\ \sigma :\vdash \varphi) = \vdash \bigsqcup\ (map\ \bigsqcap\ \Sigma) \to \varphi$
proof (*induct* Σ)
  case *Nil*
  then show *?case*
    by (*simp add*: *ex-falso-quodlibet*)
next
  case (*Cons* σ Σ)
  have $(\forall \sigma' \in set\ (\sigma \# \Sigma).\ \sigma' :\vdash \varphi) = (\sigma :\vdash \varphi \wedge (\forall \sigma' \in set\ \Sigma.\ \sigma' :\vdash \varphi))$ **by** *simp*
  also have ... $= (\vdash \sigma :\to \varphi \wedge \vdash \bigsqcup\ (map\ \bigsqcap\ \Sigma) \to \varphi)$ **using** *Cons.hyps list-deduction-def* **by** *simp*
  also have ... $= (\vdash \bigsqcap\ \sigma \to \varphi \wedge \vdash \bigsqcup\ (map\ \bigsqcap\ \Sigma) \to \varphi)$
    using *list-curry-uncurry weak-biconditional-weaken* **by** *blast*
  also have ... $= (\vdash \bigsqcap\ \sigma \to \varphi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Sigma) \to \varphi)$ **by** *simp*
  also have ... $= (\vdash (\bigsqcap\ \sigma \sqcup \bigsqcup\ (map\ \bigsqcap\ \Sigma)) \to \varphi)$
    using *disj-conj-impl-duality weak-biconditional-weaken* **by** *blast*
  finally show *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *arbitrary-disj-concat-equiv*:
  $\vdash \bigsqcup\ (\Phi\ @\ \Psi) \leftrightarrow (\bigsqcup\ \Phi \sqcup \bigsqcup\ \Psi)$
proof (*induct* Φ)
  case *Nil*
  then show *?case*
    by (*simp*,
        meson ex-falso-quodlibet
              modus-ponens
              biconditional-introduction
              disjunction-elimination
              disjunction-right-introduction
              trivial-implication)
next
  case (*Cons* φ Φ)
  have $\vdash \bigsqcup\ (\Phi\ @\ \Psi) \leftrightarrow (\bigsqcup\ \Phi \sqcup \bigsqcup\ \Psi) \to (\varphi \sqcup \bigsqcup\ (\Phi\ @\ \Psi)) \leftrightarrow ((\varphi \sqcup \bigsqcup\ \Phi) \sqcup \bigsqcup\ \Psi)$
  proof −
    let $?\varphi =$
      $(\langle \bigsqcup\ (\Phi\ @\ \Psi) \rangle \leftrightarrow (\langle \bigsqcup\ \Phi \rangle \sqcup \langle \bigsqcup\ \Psi \rangle)) \to (\langle \varphi \rangle \sqcup \langle \bigsqcup\ (\Phi\ @\ \Psi) \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcup \langle \bigsqcup\ \Phi \rangle) \sqcup \langle \bigsqcup\ \Psi \rangle)$
    have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
    hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
    thus *?thesis* **by** *simp*
  qed
  then show *?case* **using** *Cons modus-ponens* **by** *simp*
qed

lemma (**in** *classical-logic*) *arbitrary-conj-concat-equiv*:

$\vdash \bigsqcap (\Phi @ \Psi) \leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcap \Psi)$
proof (*induct* Φ)
  case *Nil*
  then show *?case*
    by (*simp*,
        meson modus-ponens
            biconditional-introduction
            conjunction-introduction
            conjunction-right-elimination
            verum-tautology)
next
  case (*Cons* φ Φ)
  have $\vdash \bigsqcap (\Phi @ \Psi) \leftrightarrow (\bigsqcap \Phi \sqcap \bigsqcap \Psi) \rightarrow (\varphi \sqcap \bigsqcap (\Phi @ \Psi)) \leftrightarrow ((\varphi \sqcap \bigsqcap \Phi) \sqcap \bigsqcap \Psi)$
  proof −
    let $?\varphi =$
      $(\langle \bigsqcap (\Phi @ \Psi) \rangle \leftrightarrow (\langle \bigsqcap \Phi \rangle \sqcap \langle \bigsqcap \Psi \rangle)) \rightarrow (\langle \varphi \rangle \sqcap \langle \bigsqcap (\Phi @ \Psi) \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcap \langle \bigsqcap \Phi \rangle) \sqcap \langle \bigsqcap \Psi \rangle)$
    have $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
    hence $\vdash (\!| ?\varphi |\!)$ **using** *propositional-semantics* **by** *blast*
    thus *?thesis* **by** *simp*
  qed
  then show *?case* **using** *Cons modus-ponens* **by** *simp*
qed

lemma (**in** *classical-logic*) *conj-absorption*:
  assumes $\chi \in set\ \Phi$
  shows $\vdash \bigsqcap \Phi \leftrightarrow (\chi \sqcap \bigsqcap \Phi)$
  using *assms*
proof (*induct* Φ)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons* φ Φ)
  then show *?case*
  proof (*cases* $\varphi = \chi$)
    case *True*
    then show *?thesis*
      by (*simp*,
          metis biconditional-def
              implication-distribution
              trivial-implication
              weak-biconditional-weaken
              weak-conjunction-deduction-equivalence)
  next
    case *False*
    then show *?thesis*
      by (*metis Cons.prems*
              arbitrary-conjunction.simps(*2*)

modus-ponens
arbitrary-conjunction-antitone
biconditional-introduction
remdups.simps(*2*)
set-remdups
set-subset-Cons)
qed
qed

lemma (**in** *classical-logic*) *conj-extract*: $\vdash \bigsqcup$ (*map* (($\sqcap$) φ) Ψ) $\leftrightarrow$ ($\varphi \sqcap \bigsqcup$ Ψ)
proof (*induct* Ψ)
case *Nil*
then show *?case*
by (*simp add*: *ex-falso-quodlibet biconditional-def conjunction-right-elimination*)
next
case (*Cons* ψ Ψ)
have $\vdash \bigsqcup$ (*map* (($\sqcap$) φ) Ψ) $\leftrightarrow$ ($\varphi \sqcap \bigsqcup$ Ψ)
$\rightarrow$ (($\varphi \sqcap \psi$) $\sqcup \bigsqcup$ (*map* (($\sqcap$) φ) Ψ)) $\leftrightarrow$ ($\varphi \sqcap$ ($\psi \sqcup \bigsqcup$ Ψ))
proof $-$
let *?*φ = $\langle\bigsqcup$ (*map* (($\sqcap$) φ) Ψ)$\rangle \leftrightarrow$ ($\langle\varphi\rangle \sqcap \langle\bigsqcup$ $\Psi\rangle$)
$\rightarrow$ ((($\langle\varphi\rangle \sqcap \langle\psi\rangle$) $\sqcup \langle\bigsqcup$ (*map* (($\sqcap$) φ) Ψ)$\rangle$) $\leftrightarrow$ ($\langle\varphi\rangle \sqcap$ ($\langle\psi\rangle \sqcup \langle\bigsqcup$ $\Psi\rangle$)))
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ ⦇ *?*φ ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
then show *?case* **using** *Cons modus-ponens* **by** *simp*
qed

lemma (**in** *classical-logic*) *conj-multi-extract*:
$\vdash \bigsqcup$ (*map* $\bigsqcap$ (*map* ((@) Δ) Σ)) $\leftrightarrow$ ($\bigsqcap$ $\Delta \sqcap \bigsqcup$ (*map* $\bigsqcap$ Σ))
proof (*induct* Σ)
case *Nil*
then show *?case*
by (*simp*, *metis list.simps*(*8*) *arbitrary-disjunction.simps*(*1*) *conj-extract*)
next
case (*Cons* σ Σ)
moreover have
$\vdash$ $\bigsqcup$ (*map* $\bigsqcap$ (*map* ((@) Δ) Σ)) $\leftrightarrow$ ($\bigsqcap$ $\Delta \sqcap \bigsqcup$ (*map* $\bigsqcap$ Σ))
$\rightarrow \bigsqcap$ (Δ @ σ) $\leftrightarrow$ ($\bigsqcap$ $\Delta \sqcap \bigsqcap$ σ)
$\rightarrow$ ($\bigsqcap$ (Δ @ σ) $\sqcup \bigsqcup$ (*map* ($\bigsqcap \circ$ (@) Δ) Σ)) $\leftrightarrow$ ($\bigsqcap$ $\Delta \sqcap$ ($\bigsqcap$ $\sigma \sqcup \bigsqcup$ (*map* $\bigsqcap$
Σ)))
proof $-$
let *?*φ =
$\langle\bigsqcup$ (*map* $\bigsqcap$ (*map* ((@) Δ) Σ))$\rangle \leftrightarrow$ ($\langle\bigsqcap$ $\Delta\rangle \sqcap \langle\bigsqcup$ (*map* $\bigsqcap$ Σ)$\rangle$)
$\rightarrow \langle\bigsqcap$ (Δ @ σ)$\rangle \leftrightarrow$ ($\langle\bigsqcap$ $\Delta\rangle \sqcap \langle\bigsqcap$ $\sigma\rangle$)
$\rightarrow$ ($\langle\bigsqcap$ (Δ @ σ)$\rangle \sqcup \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (@) Δ) Σ)$\rangle$) $\leftrightarrow$ ($\langle\bigsqcap$ $\Delta\rangle \sqcap$ ($\langle\bigsqcap$ $\sigma\rangle \sqcup \langle\bigsqcup$
(*map* $\bigsqcap$ Σ)$\rangle$)))
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ ⦇ *?*φ ⦈ **using** *propositional-semantics* **by** *blast*

thus *?thesis* **by** *simp*
qed
hence
$\vdash (\bigsqcap\ (\Delta\ @\ \sigma) \sqcup \bigsqcup\ (map\ (\bigsqcap \circ (@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap (\bigsqcap\ \sigma \sqcup \bigsqcup\ (map\ \bigsqcap\ \Sigma)))$
using *Cons.hyps arbitrary-conj-concat-equiv modus-ponens* **by** *blast*
then show *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *extract-inner-concat*:
$\vdash \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ (@)\ \Delta))\ \Psi) \leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ map\ snd)\ \Psi))$
proof (*induct* Δ)
case *Nil*
then show *?case*
by (*simp*,
meson modus-ponens
biconditional-introduction
conjunction-introduction
conjunction-right-elimination
verum-tautology)
next
case (*Cons* χ Δ)
let $?\Delta' = map\ snd\ \Delta$
let $?\chi' = snd\ \chi$
let $?\Pi = \lambda\varphi.\ \bigsqcap\ (map\ snd\ \varphi)$
let $?\Pi\Delta = \lambda\varphi.\ \bigsqcap\ (?\Delta'\ @\ map\ snd\ \varphi)$
from *Cons* **have**
$\vdash \bigsqcup\ (map\ ?\Pi\Delta\ \Psi) \leftrightarrow (\bigsqcap\ ?\Delta' \sqcap \bigsqcup\ (map\ ?\Pi\ \Psi))$
by *auto*
moreover have $\star$: $map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi) = map\ ((\sqcap)\ ?\chi') \circ map\ ?\Pi\Delta$
by *fastforce*
have $\bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) = \bigsqcup\ (map\ ((\sqcap)\ ?\chi')\ (map\ ?\Pi\Delta\ \Psi))$
by (*simp add*: $\star$)
hence
$\vdash \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) \leftrightarrow (?\chi' \sqcap \bigsqcup\ (map\ (\lambda\varphi.\ ?\Pi\Delta\ \varphi)\ \Psi))$
using *conj-extract* **by** *presburger*
moreover have
$\vdash \bigsqcup\ (map\ ?\Pi\Delta\ \Psi) \leftrightarrow (\bigsqcap\ ?\Delta' \sqcap \bigsqcup\ (map\ ?\Pi\ \Psi))$
$\rightarrow \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) \leftrightarrow (?\chi' \sqcap \bigsqcup\ (map\ ?\Pi\Delta\ \Psi))$
$\rightarrow \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) \leftrightarrow ((?\chi' \sqcap \bigsqcap\ ?\Delta') \sqcap \bigsqcup\ (map\ ?\Pi\ \Psi))$
proof –
let $?\varphi = \langle\bigsqcup\ (map\ ?\Pi\Delta\ \Psi)\rangle \leftrightarrow (\langle\bigsqcap\ ?\Delta'\rangle \sqcap \langle\bigsqcup\ (map\ ?\Pi\ \Psi)\rangle)$
$\rightarrow \langle\bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi)\rangle \leftrightarrow (\langle ?\chi'\rangle \sqcap \langle\bigsqcup\ (map\ ?\Pi\Delta\ \Psi)\rangle)$
$\rightarrow \langle\bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi)\rangle \leftrightarrow ((\langle ?\chi'\rangle \sqcap \langle\bigsqcap\ ?\Delta'\rangle) \sqcap \langle\bigsqcup\ (map\ ?\Pi\ \Psi)\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*

qed
ultimately have $\vdash \bigsqcup$ $(map\ (\lambda\varphi.\ ?\chi' \sqcap \prod\ (?\Delta'\ @\ map\ snd\ \varphi))\ \Psi)$
$\leftrightarrow ((?\chi' \sqcap \prod\ ?\Delta') \sqcap \bigsqcup\ (map\ (\lambda\varphi.\ \prod\ (map\ snd\ \varphi))\ \Psi))$
using *modus-ponens* **by** *blast*
thus *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *extract-inner-concat-remdups*:
$\vdash \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \leftrightarrow$
$(\prod\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))\ \Psi))$
proof –
have $\forall\ \Psi.\ \vdash \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \leftrightarrow$
$(\prod\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))\ \Psi))$
proof (*induct* Δ)
case *Nil*
then show *?case*
by (*simp*,
meson modus-ponens
biconditional-introduction
conjunction-introduction
conjunction-right-elimination
verum-tautology)
next
case (*Cons* δ Δ)
{
fix Ψ
have $\vdash \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)$
$\leftrightarrow (\prod\ (map\ snd\ (\delta\ \#\ \Delta)) \sqcap \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))\ \Psi))$
proof (*cases* $\delta \in set\ \Delta$)
assume $\delta \in set\ \Delta$
have
$\vdash \prod\ (map\ snd\ \Delta) \leftrightarrow (snd\ \delta \sqcap \prod\ (map\ snd\ \Delta))$
$\rightarrow \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
$\leftrightarrow (\prod\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))\ \Psi))$
$\rightarrow \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
$\leftrightarrow ((snd\ \delta \sqcap \prod\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))$
$\Psi))$
proof –
let $?\varphi = \langle \prod\ (map\ snd\ \Delta) \rangle \leftrightarrow (\langle snd\ \delta \rangle \sqcap \langle \prod\ (map\ snd\ \Delta) \rangle)$
$\rightarrow \langle \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \rangle$
$\leftrightarrow (\langle \prod\ (map\ snd\ \Delta) \rangle \sqcap \langle \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups))$
$\Psi) \rangle)$
$\rightarrow \langle \bigsqcup\ (map\ (\prod \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \rangle$
$\leftrightarrow ((\langle snd\ \delta \rangle \sqcap \langle \prod\ (map\ snd\ \Delta) \rangle) \sqcap \langle \bigsqcup\ (map\ (\prod \circ (map\ snd \circ$
$remdups))\ \Psi) \rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

moreover have $\vdash \bigsqcap$ (*map snd* Δ) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
  by (*simp add*: ‹$\delta \in$ *set* Δ› *conj-absorption*)
ultimately have
  $\vdash$ $\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) Ψ)
    $\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups*)) Ψ))
  using *Cons.hyps modus-ponens* **by** *blast*
moreover have *map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ) = *map snd* $\circ$ *remdups* $\circ$ (@) Δ
  using ‹$\delta \in$ *set* Δ› **by** *fastforce*
ultimately show *?thesis* **using** *Cons* **by** *simp*
next
assume $\delta \notin$ *set* Δ
hence †:
  $\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups*) = ($\lambda\psi$. $\bigsqcap$ (*map snd* (*remdups* ψ)))
  ($\lambda\psi$. $\bigsqcap$ (*map snd* (*if* $\delta \in$ *set* ψ *then remdups* (Δ @ ψ) *else* δ # *remdups* (Δ @ ψ))))
  = $\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))
  by *fastforce+*
show *?thesis*
proof (*induct* Ψ)
  case *Nil*
  then show *?case*
  by (*simp*, *metis list.simps(8) arbitrary-disjunction.simps(1) conj-extract*)
next
  case (*Cons* ψ Ψ)
  have $\vdash \bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) [ψ])
    $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups*)) [ψ]))
    using ‹$\forall\,\Psi$. $\vdash$ $\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) Ψ)
      $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups*)) Ψ))›
    by *blast*
  hence
    $\vdash$ ($\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))) $\sqcup \bot$)
    $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcap$ (*map snd* (*remdups* ψ)) $\sqcup \bot$)
  by *simp*
  hence ⋆:
    $\vdash \bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))) $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcap$ (*map snd* (*remdups* ψ)))
    by (*metis* (*no-types*, *hide-lams*)
        biconditional-conjunction-weaken-rule
        biconditional-symmetry-rule
        biconditional-transitivity-rule
        disjunction-def
        double-negation-biconditional
        negation-def)
  have $\vdash$ $\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
    $\leftrightarrow$ ($\bigsqcap$ (*map snd* (δ # Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups*)) Ψ))

using *Cons* **by** *blast*
hence $\Diamond$: $\vdash \quad \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ))
by *simp*
show *?case*
proof (*cases* $\delta \in$ *set* ψ)
assume $\delta \in$ *set* ψ
have *snd* $\delta \in$ *set* (*map snd* (*remdups* ψ))
using ‹$\delta \in$ *set* ψ› **by** *auto*
hence $\spadesuit$: $\vdash \bigsqcap$ (*map snd* (*remdups* ψ)) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* ψ)))
using *conj-absorption* **by** *blast*
have
$\vdash \quad$ ($\bigsqcap$ (*map snd* (*remdups* ψ)) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* ψ))))
$\rightarrow$ ($\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)))
$\rightarrow$ ($\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))) $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcap$ (*map snd* (*remdups* ψ))))
$\rightarrow \quad$ ($\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))
$\sqcup \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ))
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
$\sqcap$ ($\bigsqcap$ (*map snd* (*remdups* ψ)) $\sqcup \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)))
proof −
let *?φ* =
(($\langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle \leftrightarrow$ ($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle$)))
$\rightarrow \quad$ (($\langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)$\rangle$
$\leftrightarrow$ ((($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* Δ)$\rangle$) $\sqcap \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)$\rangle$)))
$\rightarrow \quad$ (($\langle\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))$\rangle$
$\leftrightarrow$ ($\langle\bigsqcap$ (*map snd* Δ)$\rangle \sqcap \langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle$)))
$\rightarrow \quad$ (($\langle\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))$\rangle$
$\sqcup \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)$\rangle$)
$\leftrightarrow$ ((($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* Δ)$\rangle$)
$\sqcap$ ($\langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle \sqcup \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)$\rangle$)))
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence
$\vdash \quad$ ($\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))
$\sqcup \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ))
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
$\sqcap$ ($\bigsqcap$ (*map snd* (*remdups* ψ)) $\sqcup \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$

remdups)) Ψ)))
using $\star$ $\diamondsuit$ $\spadesuit$ *modus-ponens* **by** *blast*
thus *?thesis* **using** ‹$\delta \notin$ *set* Δ› ‹$\delta \in$ *set* ψ›
by (*simp add*: †)
next
assume $\delta \notin$ *set* ψ
have
$\vdash$ $(\bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$
remdups)) Ψ)))
$\rightarrow$ $(\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))) $\leftrightarrow$ $(\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcap$ (*map*
snd (*remdups* ψ))))
$\rightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))))
$\sqcup \bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ))
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
$\sqcap$ $(\bigsqcap$ (*map snd* (*remdups* ψ)) $\sqcup \bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$
remdups)) Ψ)))
proof −
let *?φ* =
$(\langle\bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)$\rangle$
$\leftrightarrow$ (($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* Δ)$\rangle$) $\sqcap \langle\bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$
remdups)) Ψ)$\rangle$))
$\rightarrow$ $(\langle\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))$\rangle$
$\leftrightarrow$ $(\langle\bigsqcap$ (*map snd* Δ)$\rangle \sqcap \langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle$)))
$\rightarrow$ (($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ)))$\rangle$)
$\sqcup \langle\bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)$\rangle$)
$\leftrightarrow$ (($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* Δ)$\rangle$)
$\sqcap$ $(\langle\bigsqcap$ (*map snd* (*remdups* ψ))$\rangle \sqcup \langle\bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$
remdups)) Ψ)$\rangle$)))
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence
$\vdash$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))))
$\sqcup \bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ))
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
$\sqcap$ $(\bigsqcap$ (*map snd* (*remdups* ψ)) $\sqcup \bigsqcup$ (*map* $(\bigsqcap \circ$ (*map snd* $\circ$
remdups)) Ψ)))
using $\star$ $\diamondsuit$ *modus-ponens* **by** *blast*
then show *?thesis* **using** ‹$\delta \notin$ *set* ψ› ‹$\delta \notin$ *set* Δ› **by** (*simp add*: †)
qed
qed
qed
}
then show *?case* **by** *fastforce*
qed
thus *?thesis* **by** *blast*
qed

lemma *remove1-remdups-removeAll*: *remove1 x* (*remdups A*) = *remdups* (*removeAll x A*)
proof (*induct A*)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons a A*)
  then show *?case*
    by (*cases a* = *x*, (*simp add*: *Cons*)+)
qed

lemma *mset-remdups*:
  assumes *mset A* = *mset B*
  shows *mset* (*remdups A*) = *mset* (*remdups B*)
proof −
  have ∀ *B*. *mset A* = *mset B* ⟶ *mset* (*remdups A*) = *mset* (*remdups B*)
  proof (*induct A*)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons a A*)
    {
      fix *B*
      assume *mset* (*a* # *A*) = *mset B*
      hence *mset A* = *mset* (*remove1 a B*)
        by (*metis add-mset-add-mset-same-iff*
              list.set-intros(*1*)
              mset.simps(*2*)
              mset-eq-perm
              mset-eq-setD
              perm-remove)
      hence *mset* (*remdups A*) = *mset* (*remdups* (*remove1 a B*))
        using *Cons.hyps* **by** *blast*
      hence *mset* (*remdups* (*a* # (*remdups A*))) = *mset* (*remdups* (*a* # (*remdups* (*remove1 a B*))))
        by (*metis mset-eq-setD set-eq-iff-mset-remdups-eq list.simps*(*15*))
      hence *mset* (*remdups* (*a* # (*removeAll a* (*remdups A*))))
            = *mset* (*remdups* (*a* # (*removeAll a* (*remdups* (*remove1 a B*)))))
      by (*metis insert-Diff-single list.set*(*2*) *set-eq-iff-mset-remdups-eq set-removeAll*)
      hence *mset* (*remdups* (*a* # (*remdups* (*removeAll a A*))))
            = *mset* (*remdups* (*a* # (*remdups* (*removeAll a* (*remove1 a B*)))))
      by (*metis distinct-remdups distinct-remove1-removeAll remove1-remdups-removeAll*)
      hence *mset* (*remdups* (*remdups* (*a* # *A*))) = *mset* (*remdups* (*remdups* (*a* # (*remove1 a B*))))
        by (*metis* ‹*mset A* = *mset* (*remove1 a B*)›
              list.set(*2*)
              mset-eq-setD
              set-eq-iff-mset-remdups-eq)

hence *mset* (*remdups* (*a* # *A*)) = *mset* (*remdups* (*a* # (*remove1 a B*)))
  by (*metis remdups-remdups*)
hence *mset* (*remdups* (*a* # *A*)) = *mset* (*remdups B*)
  using ‹*mset* (*a* # *A*) = *mset B*› *mset-eq-setD set-eq-iff-mset-remdups-eq* **by** *blast*
  }
  then show *?case* **by** *simp*
 qed
 thus *?thesis* **using** *assms* **by** *blast*
qed

lemma *mset-mset-map-snd-remdups*:
 assumes *mset* (*map mset A*) = *mset* (*map mset B*)
 shows *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *A*) = *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *B*)
proof −
 {
  fix *B* :: ($'a \times 'b$) *list list*
  fix *b* :: ($'a \times 'b$) *list*
  assume $b \in$ *set B*
  hence *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*b* # (*remove1 b B*)))
    = *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *B*)
  proof (*induct B*)
   case *Nil*
   then show *?case* **by** *simp*
  next
   case (*Cons b′ B*)
   then show *?case*
   by (*cases b* = *b′*, *simp*+)
  qed
 }
 note ♢ = *this*
 have
  ∀ *B* :: ($'a \times 'b$) *list list*.
   mset (*map mset A*) = *mset* (*map mset B*)
    ⟶ *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *A*) = *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *B*)
 proof (*induct A*)
  case *Nil*
  then show *?case* **by** *simp*
 next
  case (*Cons a A*)
  {
   fix *B*
   assume ♠: *mset* (*map mset* (*a* # *A*)) = *mset* (*map mset B*)
   hence *mset a* ∈# *mset* (*map mset B*)
    by (*simp*,
      metis ♠
        image-set

list.set-intros(1)
list.simps(9)
mset-eq-setD)
from *this* **obtain** *b* **where** †:
b ∈ *set B*
mset a = *mset b*
by *auto*
with ♠ **have** *mset* (*map mset A*) = *mset* (*remove1* (*mset b*) (*map mset B*))
by (*simp add*: *union-single-eq-diff*)
moreover have *mset B* = *mset* (*b* # *remove1 b B*) **using** † **by** *simp*
hence *mset* (*map mset B*) = *mset* (*map mset* (*b* # (*remove1 b B*)))
by (*simp*,
metis image-mset-add-mset
mset.simps(2)
mset-remove1)
ultimately have *mset* (*map mset A*) = *mset* (*map mset* (*remove1 b B*))
by *simp*
hence *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *A*)
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*remove1 b B*))
using *Cons.hyps* **by** *blast*
moreover have (*mset* ∘ (*map snd*) ∘ *remdups*) *a* = (*mset* ∘ (*map snd*) ∘ *remdups*) *b*
using †(*2*) *mset-remdups* **by** *fastforce*
ultimately have
mset (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*a* # *A*))
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*b* # (*remove1 b B*)))
by *simp*
moreover have
mset (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*b* # (*remove1 b B*)))
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *B*)
using †(*1*) ♢ **by** *blast*
ultimately have
mset (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*a* # *A*))
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) *B*)
by *simp*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma *image-mset-cons-homomorphism*:
image-mset mset (*image-mset* ((#) φ) Φ) = *image-mset* ((+) {# φ #}) (*image-mset mset* Φ)
by (*induct* Φ, *simp*+)

lemma *image-mset-append-homomorphism*:
image-mset mset (*image-mset* ((@) Δ) Φ) = *image-mset* ((+) (*mset* Δ)) (*image-mset mset* Φ)

by (*induct* Φ, *simp*+)

lemma *image-mset-add-collapse*:
fixes A B :: $'a$ *multiset*
shows *image-mset* ((+) A) (*image-mset* ((+) B) X) = *image-mset* ((+) (A + B)) X
by (*induct* X, *simp*, *simp*)

lemma *mset-remdups-append-msub*:
mset (*remdups* A) $\subseteq\#$ *mset* (*remdups* (B @ A))
proof −
have $\forall$ B. *mset* (*remdups* A) $\subseteq\#$ *mset* (*remdups* (B @ A))
proof (*induct* A)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* a A)
{
fix B
have †: *mset* (*remdups* (B @ (a # A))) = *mset* (*remdups* (a # (B @ A)))
by (*induct* B, *simp*+)
have *mset* (*remdups* (a # A)) $\subseteq\#$ *mset* (*remdups* (B @ (a # A)))
proof (*cases* $a \in$ *set* B $\wedge$ $a \notin$ *set* A)
case *True*
hence †: *mset* (*remove1* a (*remdups* (B @ A))) = *mset* (*remdups* ((*removeAll* a B) @ A))
by (*simp add*: *remove1-remdups-removeAll*)
hence (*add-mset* a (*mset* (*remdups* A)) $\subseteq\#$ *mset* (*remdups* (B @ A)))
= (*mset* (*remdups* A) $\subseteq\#$ *mset* (*remdups* ((*removeAll* a B) @ A)))
using *True*
by (*simp add*: *insert-subset-eq-iff*)
then show *?thesis*
by (*metis* † *Cons True*
Un-insert-right
list.set(*2*)
mset.simps(*2*)
mset-subset-eq-insertD
remdups.simps(*2*)
set-append
set-eq-iff-mset-remdups-eq
set-mset-mset set-remdups)
next
case *False*
then show *?thesis* **using** † *Cons* **by** *simp*
qed
}
thus *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*

qed

lemma (**in** *classical-logic*) *optimal-witness-list-intersect-biconditional*:
  assumes *mset* $\Xi \subseteq\#$ *mset* Γ
    and *mset* $\Phi \subseteq\#$ *mset* $(\Gamma \ominus \Xi)$
    and *mset* $\Psi \subseteq\#$ *mset* $(\mathfrak{W}_{\rightarrow}\ \varphi\ \Xi)$
  shows $\exists\ \Sigma.\ \vdash ((\Phi\ @\ \Psi) :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ \Sigma) \rightarrow \varphi)$
    $\wedge\ (\forall\ \sigma \in set\ \Sigma.\ mset\ \sigma \subseteq\#\ mset\ \Gamma \wedge length\ \sigma + 1 \geq length\ (\Phi\ @$
$\Psi))$
proof −
  have $\exists\ \Sigma.\ \vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ \Sigma) \rightarrow \varphi)$
    $\wedge\ (\forall\ \sigma \in set\ \Sigma.\ mset\ \sigma \subseteq\#\ mset\ \Xi \wedge length\ \sigma + 1 \geq length\ \Psi)$
  proof −
    from *assms*(*3*) **obtain** Ψ_0 :: ('*a list* × '*a*) *list* **where** Ψ_0:
      mset $\Psi_0 \subseteq\#$ *mset* $(\mathfrak{V}\ \Xi)$
      map $(\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ \Psi_0 = \Psi$
      using *mset-sub-map-list-exists* **by** *fastforce*
    let $?\Pi_C = \lambda\ (\Delta,\delta)\ \Sigma.\ (map\ ((\#)\ (\Delta,\ \delta))\ \Sigma)\ @\ (map\ ((@)\ (\mathfrak{V}\ \Delta))\ \Sigma)$
    let $?T_\Sigma = \lambda\ \Psi.\ foldr\ ?\Pi_C\ \Psi\ [[]]$
    let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi_0)$
    have I: $\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
    proof −
      let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi_0)$
      let $?\Psi' = map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ \Psi_0$
      {
        fix Ψ :: ('*a list* × '*a*) *list*
        let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi)$
        let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi)$
        have $\vdash (\bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
        proof (*induct* Ψ)
          case *Nil*
          then show *?case* **by** (*simp add*: *biconditional-reflection*)
        next
          case (*Cons* $\Delta\delta\ \Psi$)
          let $?\Delta = fst\ \Delta\delta$
          let $?\delta = snd\ \Delta\delta$
          let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi)$
          let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi)$
          let $?\Sigma_\alpha' = map\ (map\ snd)\ (?T_\Sigma\ ((?\Delta,?\delta)\ \#\ \Psi))$
          let $?\Sigma' = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ ((?\Delta,?\delta)\ \#\ \Psi))$
          {
            fix Δ :: '*a list*
            fix δ :: '*a*
            let $?\Sigma_\alpha' = map\ (map\ snd)\ (?T_\Sigma\ ((\Delta,\delta)\ \#\ \Psi))$
            let $?\Sigma' = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ ((\Delta,\delta)\ \#\ \Psi))$
            let $?\Phi = map\ (map\ snd \circ (@)\ [(\Delta,\ \delta)])\ (?T_\Sigma\ \Psi)$
            let $?\Psi = map\ (map\ snd \circ (@)\ (\mathfrak{V}\ \Delta))\ (?T_\Sigma\ \Psi)$
            let $?\Delta = map\ (map\ snd \circ remdups \circ (@)\ [(\Delta,\ \delta)])\ (?T_\Sigma\ \Psi)$
            let $?\Omega = map\ (map\ snd \circ remdups \circ (@)\ (\mathfrak{V}\ \Delta))\ (?T_\Sigma\ \Psi)$

have $\vdash (\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Phi) \sqcup \bigsqcup\ (map \prod\ ?\Psi))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta) \sqcup \bigsqcup\ (map \prod\ ?\Omega))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Phi) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Psi) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Delta) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Omega) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma))) \rightarrow$
$((\bigsqcup\ (map \prod\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Sigma) \rightarrow \varphi)) \rightarrow$
$((\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \rightarrow \varphi))$
proof −
let $?\varphi =$
$((\langle\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi)\rangle \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Phi)\rangle \sqcup \langle\bigsqcup\ (map \prod\ ?\Psi)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega)\rangle \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Delta)\rangle \sqcup \langle\bigsqcup\ (map \prod\ ?\Omega)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Phi)\rangle \leftrightarrow (\langle\prod\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Psi)\rangle \leftrightarrow (\langle\prod\ \Delta\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Delta)\rangle \leftrightarrow (\langle\prod\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Omega)\rangle \leftrightarrow (\langle\prod\ \Delta\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle)) \rightarrow$
$((\langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle \rightarrow \langle\varphi\rangle) \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle \rightarrow \langle\varphi\rangle)) \rightarrow$
$((\langle\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi)\rangle \rightarrow \langle\varphi\rangle) \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega)\rangle \rightarrow \langle\varphi\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
moreover
have *map snd* $(\mathfrak{V}\ \Delta) = \Delta$ **by** (*induct* Δ, *auto*)
hence $\vdash \bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Phi) \sqcup \bigsqcup\ (map \prod\ ?\Psi))$
$\vdash \bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta) \sqcup \bigsqcup\ (map \prod\ ?\Omega))$
$\vdash \bigsqcup\ (map \prod\ ?\Phi) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))$
$\vdash \bigsqcup\ (map \prod\ ?\Psi) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))$
$\vdash \bigsqcup\ (map \prod\ ?\Delta) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma))$
$\vdash \bigsqcup\ (map \prod\ ?\Omega) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma))$
using *arbitrary-disj-concat-equiv*
extract-inner-concat [**where** $\Delta = [(\Delta, \delta)]$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat [**where** $\Delta = \mathfrak{V}\ \Delta$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat-remdups [**where** $\Delta = [(\Delta, \delta)]$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat-remdups [**where** $\Delta = \mathfrak{V}\ \Delta$ **and** $\Psi = ?T_\Sigma\ \Psi$]
by *auto*
ultimately have
$\vdash ((\bigsqcup\ (map \prod\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Sigma) \rightarrow \varphi)) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \rightarrow \varphi)$

using *modus-ponens* **by** *blast*
moreover have $(\#)\ (\Delta,\ \delta) = (@)\ [(\Delta,\ \delta)]$ **by** *fastforce*
ultimately have
$\vdash ((\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)) \rightarrow$
$((\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha{}') \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma') \rightarrow \varphi))$
by *auto*
}
hence
$\vdash ((\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha{}') \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma') \rightarrow \varphi))$
using *Cons modus-ponens* **by** *blast*
moreover have $\Delta\delta = (?\Delta, ?\delta)$ **by** *fastforce*
ultimately show *?case* **by** *metis*
qed
}
hence $\vdash (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)$ **by** *blast*
moreover have $\vdash (?\Psi' :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi)$
proof $(induct\ \Psi_0)$
case *Nil*
have $\vdash \varphi \leftrightarrow ((\top \sqcup \bot) \rightarrow \varphi)$
proof $-$
let $?\varphi = \langle\varphi\rangle \leftrightarrow ((\top \sqcup \bot) \rightarrow \langle\varphi\rangle)$
have $\forall\,\mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
thus *?case* **by** *simp*
next
case $(Cons\ \psi_0\ \Psi_0)$
let $?\Xi = fst\ \psi_0$
let $?\delta = snd\ \psi_0$
let $?\Psi' = map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ \Psi_0$
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi_0)$
{
fix $\Xi :: 'a\ list$
have $map\ snd\ (\mathfrak{V}\ \Xi) = \Xi$ **by** $(induct\ \Xi,\ auto)$
hence $map\ snd \circ (@)\ (\mathfrak{V}\ \Xi) = (@)\ \Xi \circ map\ snd$ **by** *fastforce*
}
moreover have $(map\ snd \circ (\#)\ (?\Xi,\ ?\delta)) = (@)\ [?\delta] \circ map\ snd$ **by** *fastforce*
ultimately have †:
$map\ (map\ snd)\ (?T_\Sigma\ (\psi_0\ \#\ \Psi_0)) = map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha\ @\ map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha$
$map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ (\psi_0\ \#\ \Psi_0) = ?\Xi :\rightarrow \varphi \rightarrow ?\delta\ \#\ ?\Psi'$
by (*simp add*: *case-prod-beta'*)+
have A: $\vdash (?\Psi' :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi)$ **using** *Cons.hyps* **by** *auto*
have B: $\vdash (?\Xi :\rightarrow \varphi) \leftrightarrow (\bigsqcap\ ?\Xi \rightarrow \varphi)$
by (*simp add*: *list-curry-uncurry*)
have C: $\vdash\ \ \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)$

$?\Sigma_\alpha))$

$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \sqcup \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)))$

using *arbitrary-disj-concat-equiv* **by** *blast*

have $map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha) = (map\ ((\sqcap)\ ?\delta)\ (map\ \bigsqcap\ ?\Sigma_\alpha))$ **by** *auto*

hence D: $\vdash \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \leftrightarrow (?\delta \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$

using *conj-extract* **by** *presburger*

have E: $\vdash \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \leftrightarrow (\bigsqcap\ ?\Xi \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$

using *conj-multi-extract* **by** *blast*

have

$\vdash\ (?\Psi' :\to \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \to \varphi)$

$\to\ (?\Xi :\to \varphi) \leftrightarrow (\bigsqcap\ ?\Xi \to \varphi)$

$\to\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))$

$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \sqcup \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)))$

$\to\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \leftrightarrow (?\delta \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$

$\to\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \leftrightarrow (\bigsqcap\ ?\Xi \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$

$\to\ ((?\Xi :\to \varphi \to ?\delta) \to ?\Psi' :\to \varphi)$

$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \to \varphi)$

proof $-$

let $?\varphi =$

$\langle ?\Psi' :\to \varphi\rangle \leftrightarrow (\langle \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle \to \langle\varphi\rangle)$

$\to\ \langle (?\Xi :\to \varphi)\rangle \leftrightarrow (\langle \bigsqcap\ ?\Xi\rangle \to \langle\varphi\rangle)$

$\to\ \langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))\rangle$

$\leftrightarrow (\langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha))\rangle \sqcup \langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))\rangle)$

$\to\ \langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha))\rangle \leftrightarrow (\langle ?\delta\rangle \sqcap \langle \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle)$

$\to\ \langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))\rangle \leftrightarrow (\langle \bigsqcap\ ?\Xi\rangle \sqcap \langle \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle)$

$\to\ ((\langle ?\Xi :\to \varphi\rangle \to \langle ?\delta\rangle) \to \langle ?\Psi' :\to \varphi\rangle)$

$\leftrightarrow (\langle \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))\rangle \to \langle\varphi\rangle)$

have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*

hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*

thus *?thesis* **by** *simp*

qed

hence

$\vdash\ ((?\Xi :\to \varphi \to ?\delta) \to ?\Psi' :\to \varphi)$

$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \to \varphi)$

using *A B C D E modus-ponens* **by** *blast*

thus *?case* **using** † **by** *simp*

qed

ultimately show *?thesis* **using** *biconditional-transitivity-rule* Ψ_0 **by** *blast*

qed

have II: $\forall\ \sigma \in set\ ?\Sigma.\ length\ \sigma + 1 \geq length\ \Psi$

proof −
let $?\mathcal{M}$ = *length* ∘ *fst*
let $?\mathcal{S}$ = *sort-key* (− $?\mathcal{M}$)
let $?\Sigma'$ = *map* (*map snd* ∘ *remdups*) ($?T_\Sigma$ ($?\mathcal{S}$ Ψ_0))
have *mset* Ψ_0 = *mset* ($?\mathcal{S}$ Ψ_0) **by** *simp*

have ∀ Φ. *mset* Ψ_0 = *mset* Φ ⟶ *mset* (*map mset* ($?T_\Sigma$ Ψ_0)) = *mset* (*map mset* ($?T_\Sigma$ Φ))
proof (*induct* Ψ_0)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ_0)
obtain Δ δ **where** ψ = (Δ,δ) **by** *fastforce*
{
fix Φ
assume *mset* (ψ # Ψ_0) = *mset* Φ
hence *mset* Ψ_0 = *mset* (*remove1* ψ Φ)
by (*simp add*: *union-single-eq-diff*)
have ψ ∈ *set* Φ **using** ‹*mset* (ψ # Ψ_0) = *mset* Φ›
using *mset-eq-setD* **by** *fastforce*
hence *mset* (*map mset* ($?T_\Sigma$ Φ)) = *mset* (*map mset* ($?T_\Sigma$ (ψ # (*remove1* ψ Φ))))
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
then show *?case* **proof** (*cases* φ = ψ)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
let $?\Sigma'$ = $?T_\Sigma$ (ψ # (*remove1* ψ Φ))
have †: *mset* (*map mset* $?\Sigma'$) = *mset* (*map mset* ($?T_\Sigma$ Φ))
using *Cons False* **by** *simp*
obtain Δ′ δ′
where φ = (Δ′,δ′)
by *fastforce*
let $?\Sigma$ = $?T_\Sigma$ (*remove1* ψ Φ)
let $?\mathfrak{m}$ = *image-mset mset*
have
mset (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
mset (*map mset* ($?\Pi_C$ ψ ($?\Pi_C$ φ $?\Sigma$)))
using *False* **by** *simp*
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
($?\mathfrak{m}$ ∘ (*image-mset* ((#) ψ) ∘ *image-mset* ((#) φ))) (*mset* $?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((#) ψ) ∘ *image-mset* ((@) ($\mathfrak{V}$ Δ′)))) (*mset* $?\Sigma$) +

$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta)) \circ image\text{-}mset\ ((\#)\ \varphi)))\ (mset$
$?\Sigma) +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta)) \circ image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta'))))$
$(mset\ ?\Sigma)$
using $\langle\psi = (\Delta,\delta)\rangle$ $\langle\varphi = (\Delta',\delta')\rangle$
by (*simp add*: *multiset.map-comp*)
hence $mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ remove1\ \psi\ (\varphi\ \#\ \Phi)))) =$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((\#)\ \varphi) \circ image\text{-}mset\ ((\#)\ \psi)))\ (mset\ ?\Sigma) +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta')) \circ image\text{-}mset\ ((\#)\ \psi)))\ (mset$
$?\Sigma) +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((\#)\ \varphi) \circ image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta))))\ (mset$
$?\Sigma) +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta')) \circ image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta))))$
$(mset\ ?\Sigma)$
by (*simp add*: *image-mset-cons-homomorphism*
image-mset-append-homomorphism
image-mset-add-collapse
add-mset-commute
add.commute)
hence $mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ remove1\ \psi\ (\varphi\ \#\ \Phi)))) =$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((\#)\ \varphi)))\ (mset\ ?\Sigma') +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta'))))\ (mset\ ?\Sigma')$
using $\langle\psi = (\Delta,\delta)\rangle$
by (*simp add*: *multiset.map-comp*)
hence $mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ remove1\ \psi\ (\varphi\ \#\ \Phi)))) =$
$image\text{-}mset\ ((+)\ \{\#\varphi\#\})\ (mset\ (map\ mset\ ?\Sigma')) +$
$image\text{-}mset\ ((+)\ (mset\ (\mathfrak{V}\ \Delta')))\ (mset\ (map\ mset\ ?\Sigma'))$
by (*simp add*: *image-mset-cons-homomorphism*
image-mset-append-homomorphism)
hence $mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ remove1\ \psi\ (\varphi\ \#\ \Phi)))) =$
$image\text{-}mset\ ((+)\ \{\#\varphi\#\})\ (mset\ (map\ mset\ (?T_\Sigma\ \Phi))) +$
$image\text{-}mset\ ((+)\ (mset\ (\mathfrak{V}\ \Delta')))\ (mset\ (map\ mset\ (?T_\Sigma\ \Phi)))$
using † **by** *auto*
hence $mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ remove1\ \psi\ (\varphi\ \#\ \Phi)))) =$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((\#)\ \varphi)))\ (mset\ (?T_\Sigma\ \Phi)) +$
$(?\mathfrak{m} \circ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta'))))\ (mset\ (?T_\Sigma\ \Phi))$
by (*simp add*: *image-mset-cons-homomorphism*
image-mset-append-homomorphism)
thus *?thesis* **using** $\langle\varphi = (\Delta',\delta')\rangle$ **by** (*simp add*: *multiset.map-comp*)
qed
qed
hence $image\text{-}mset\ mset\ (image\text{-}mset\ ((\#)\ \psi)\ (mset\ (?T_\Sigma\ (remove1\ \psi$
$\Phi)))) +$
$image\text{-}mset\ mset\ (image\text{-}mset\ ((@)\ (\mathfrak{V}\ \Delta))\ (mset\ (?T_\Sigma\ (remove1$
$\psi\ \Phi))))$
$= image\text{-}mset\ mset\ (mset\ (?T_\Sigma\ \Phi))$
by (*simp add*: $\langle\psi = (\Delta,\delta)\rangle$ *multiset.map-comp*)
hence
$image\text{-}mset\ ((+)\ \{\#\ \psi\ \#\})\ (image\text{-}mset\ mset\ (mset\ (?T_\Sigma\ (remove1\ \psi$

$\Phi)))) +$

$\quad$ *image-mset* $((+)$ (*mset* $(\mathfrak{V}\ \Delta)))$ (*image-mset mset* (*mset* (*?T*$_\Sigma$ (*remove1*
$\psi\ \Phi))))$

$\quad = $ *image-mset mset* (*mset* (*?T*$_\Sigma$ Φ))

by (*simp add*: *image-mset-cons-homomorphism image-mset-append-homomorphism*)

hence

image-mset $((+)\ \{\#\ \psi\ \#\})$ (*image-mset mset* (*mset* (*?T*$_\Sigma$ Ψ_0))) +
image-mset $((+)$ (*mset* $(\mathfrak{V}\ \Delta)))$ (*image-mset mset* (*mset* (*?T*$_\Sigma$ Ψ_0)))
$=$ *image-mset mset* (*mset* (*?T*$_\Sigma$ Φ))

using *Cons* ‹*mset* Ψ_0 = *mset* (*remove1* $\psi\ \Phi$)›

by *fastforce*

hence

image-mset mset (*image-mset* $((\#)\ \psi)$ (*mset* (*?T*$_\Sigma$ Ψ_0))) +
image-mset mset (*image-mset* $((@)\ (\mathfrak{V}\ \Delta))$ (*mset* (*?T*$_\Sigma$ Ψ_0)))
$=$ *image-mset mset* (*mset* (*?T*$_\Sigma$ Φ))

by (*simp add*: *image-mset-cons-homomorphism image-mset-append-homomorphism*)

hence *mset* (*map mset* (*?T*$_\Sigma$ $(\psi\ \#\ \Psi_0)$)) = *mset* (*map mset* (*?T*$_\Sigma$ Φ))

by (*simp add*: ‹$\psi = (\Delta,\delta)$› *multiset.map-comp*)

}

then show *?case* **by** *blast*

qed

hence *mset* (*map mset* (*?T*$_\Sigma$ Ψ_0)) = *mset* (*map mset* (*?T*$_\Sigma$ (*?S* Ψ_0)))

using ‹*mset* Ψ_0 = *mset* (*?S* Ψ_0)› **by** *blast*

hence *mset* (*map* (*mset* $\circ$ (*map snd*) $\circ$ *remdups*) (*?T*$_\Sigma$ Ψ_0))
$=$ *mset* (*map* (*mset* $\circ$ (*map snd*) $\circ$ *remdups*) (*?T*$_\Sigma$ (*?S* Ψ_0)))

using *mset-mset-map-snd-remdups* **by** *blast*

hence *mset* (*map mset* *?*Σ) = *mset* (*map mset* *?*Σ')

by (*simp add*: *fun.map-comp*)

hence *set* (*map mset* *?*Σ) = *set* (*map mset* *?*Σ')

using *mset-eq-setD* **by** *blast*

hence $\forall\ \sigma \in$ *set ?*Σ. $\exists\ \sigma' \in$ *set ?*Σ'. *mset* σ = *mset* σ'

by *fastforce*

hence $\forall\ \sigma \in$ *set ?*Σ. $\exists\ \sigma' \in$ *set ?*Σ'. *length* σ = *length* σ'

using *mset-eq-length* **by** *blast*

have *mset* (*?S* Ψ_0) $\subseteq\#$ *mset* $(\mathfrak{V}\ \Xi)$

by (*simp add*: $\Psi_0(1)$)

{

fix n

have $\forall\ \Psi$. *mset* $\Psi \subseteq\#$ *mset* $(\mathfrak{V}\ \Xi) \longrightarrow$
sorted (*map* ($-$ *?*$\mathcal{M}$) Ψ) $\longrightarrow$
length $\Psi = n \longrightarrow$
($\forall\ \sigma' \in$ *set* (*map* (*map snd* $\circ$ *remdups*) (*?T*$_\Sigma$ Ψ)). *length* σ' + *1*
$\geq n$)

proof (*induct* n)

case *0*

then show *?case* **by** *simp*

next

case (*Suc* n)

{

fix Ψ :: $('a\ list \times 'a)\ list$
assume A: *mset* $\Psi \subseteq\#$ *mset* $(\mathfrak{V}\ \Xi)$
and B: *sorted* (*map* ($-$ *?*$\mathcal{M}$) Ψ)
and C: *length* $\Psi = n + 1$
obtain $\Delta\ \delta$ **where** $(\Delta, \delta) = hd\ \Psi$
using *prod.collapse* **by** *blast*
let $?\Psi' = tl\ \Psi$
have *mset* $?\Psi' \subseteq\#$ *mset* $(\mathfrak{V}\ \Xi)$ **using** A
by (*induct* Ψ, *simp*, *simp*, *meson mset-subset-eq-insertD subset-mset-def*)
moreover
have *sorted* (*map* ($-$ *?*$\mathcal{M}$) (*tl* Ψ))
using B
by (*simp add*: *map-tl sorted-tl*)
moreover have *length* $?\Psi' = n$ **using** C
by *simp*
ultimately have $\star$: $\forall\ \sigma' \in$ *set* (*map* (*map snd* $\circ$ *remdups*) ($?T_\Sigma\ ?\Psi'$)).
length $\sigma' + 1 \geq n$
using *Suc*
by *blast*
from C **have** $\Psi = (\Delta, \delta)\ \#\ ?\Psi'$
by (*metis* ‹$(\Delta, \delta) = hd\ \Psi$›
One-nat-def
add-is-0
list.exhaust-sel
list.size(*3*)
nat.simps(*3*))
have *distinct* ($(\Delta, \delta)\ \#\ ?\Psi'$)
using A ‹$\Psi = (\Delta, \delta)\ \#\ ?\Psi'$›
core-optimal-pre-witness-distinct
mset-distinct-msub-down
by *fastforce*
hence *set* ($(\Delta, \delta)\ \#\ ?\Psi'$) $\subseteq$ *set* $(\mathfrak{V}\ \Xi)$
by (*metis* A ‹$\Psi = (\Delta, \delta)\ \#\ ?\Psi'$›
Un-iff
mset-le-perm-append
perm-set-eq set-append
subsetI)
hence $\forall\ (\Delta', \delta') \in$ *set* $?\Psi'$. $(\Delta, \delta) \neq (\Delta', \delta')$
$\forall\ (\Delta', \delta') \in$ *set* $(\mathfrak{V}\ \Xi)$. $((\Delta, \delta) \neq (\Delta', \delta')) \longrightarrow$ (*length* $\Delta \neq$ *length*
Δ')
set $?\Psi' \subseteq$ *set* $(\mathfrak{V}\ \Xi)$
using *core-optimal-pre-witness-length-iff-eq* [**where** Ψ=Ξ]
‹*distinct* ($(\Delta, \delta)\ \#\ ?\Psi'$)›
by *auto*
hence $\forall\ (\Delta', \delta') \in$ *set* $?\Psi'$. *length* $\Delta \neq$ *length* Δ'
sorted (*map* ($-$ *?*$\mathcal{M}$) ($(\Delta, \delta)\ \#\ ?\Psi'$))
using B ‹$\Psi = (\Delta, \delta)\ \#\ ?\Psi'$›
by (*fastforce*, *auto*)
hence $\forall\ (\Delta', \delta') \in$ *set* $?\Psi'$. *length* $\Delta >$ *length* Δ'

```
    by fastforce
  {
    fix σ' :: 'a list
    assume σ' ∈ set (map (map snd ∘ remdups) (?T_Σ Ψ))
    hence σ' ∈ set (map (map snd ∘ remdups) (?T_Σ ((Δ, δ) # ?Ψ')))
      using ‹Ψ = (Δ, δ) # ?Ψ'›
      by simp
    from this obtain ψ where ψ:
      ψ ∈ set (?T_Σ ?Ψ')
      σ' = (map snd ∘ remdups ∘ (#) (Δ, δ)) ψ ∨
       σ' = (map snd ∘ remdups ∘ (@) (𝔙 Δ)) ψ
      by fastforce
    hence length σ' ≥ n
    proof (cases σ' = (map snd ∘ remdups ∘ (#) (Δ, δ)) ψ)
      case True
      {
        fix Ψ :: ('a list × 'a) list
        fix n :: nat
        assume ∀ (Δ, δ) ∈ set Ψ. n > length Δ
        hence ∀ σ ∈ set (?T_Σ Ψ). ∀ (Δ, δ) ∈ set σ. n > length Δ
        proof (induct Ψ)
          case Nil
          then show ?case by simp
        next
          case (Cons ψ Ψ)
          obtain Δ δ where ψ = (Δ, δ)
            by fastforce
          hence n > length Δ using Cons.prems by fastforce
          have 0: ∀ σ ∈ set (?T_Σ Ψ). ∀ (Δ', δ') ∈ set σ. n > length Δ'
            using Cons by simp
          {
            fix σ :: ('a list × 'a) list
            fix ψ' :: 'a list × 'a
            assume 1: σ ∈ set (?T_Σ (ψ # Ψ))
               and 2: ψ' ∈ set σ
            obtain Δ' δ' where ψ' = (Δ', δ')
              by fastforce
             have 3: σ ∈ (#) (Δ, δ) ` set (?T_Σ Ψ) ∨ σ ∈ (@) (𝔙 Δ) ` set
(?T_Σ Ψ)
              using 1 ‹ψ = (Δ, δ)› by simp
            have n > length Δ'
            proof (cases σ ∈ (#) (Δ, δ) ` set (?T_Σ Ψ))
              case True
              from this obtain σ' where
                set σ = insert (Δ, δ) (set σ')
                σ' ∈ set (?T_Σ Ψ)
                by auto
              then show ?thesis
                using 0 ‹ψ' ∈ set σ› ‹ψ' = (Δ', δ')› ‹n > length Δ›
```

by *auto*
next
case *False*
from *this* **and** *3* **obtain** σ' **where** σ':
set σ = *set* ($\mathfrak{V}$ Δ) $\cup$ (*set* σ')
σ' $\in$ *set* (*?T*$_\Sigma$ Ψ)
by *auto*
have $\forall$ (Δ', δ') $\in$ *set* ($\mathfrak{V}$ Δ). *length* Δ > *length* Δ'
by (*metis* (*mono-tags*, *lifting*)
case-prodI2
core-optimal-pre-witness-nonelement
not-le)
hence $\forall$ (Δ', δ') $\in$ *set* ($\mathfrak{V}$ Δ). *n* > *length* Δ'
using ‹*n* > *length* Δ› **by** *auto*
then show *?thesis* **using** *0* σ' ‹ψ' $\in$ *set* σ› ‹ψ' = (Δ', δ')› **by**
fastforce
qed
hence *n* > *length* (*fst* ψ') **using** ‹ψ' = (Δ', δ')› **by** *fastforce*
}
then show *?case* **by** *fastforce*
qed
}
hence $\forall$ σ $\in$ *set* (*?T*$_\Sigma$ *?*Ψ'). $\forall$ (Δ', δ') $\in$ *set* σ. *length* Δ > *length*
Δ'
using ‹$\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* Δ > *length* Δ'›
by *blast*
then show *?thesis* **using** *True* $\star$ ψ(*1*) **by** *fastforce*
next
case *False*
have $\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* Δ $\geq$ *length* Δ'
using ‹$\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* Δ > *length* Δ'›
by *auto*
hence $\forall$ (Δ', δ') $\in$ *set* Ψ. *length* Δ $\geq$ *length* Δ'
using ‹Ψ = (Δ, δ) # *?*Ψ'›
by (*metis case-prodI2 eq-iff prod.sel*(*1*) *set-ConsD*)
hence *length* Δ + *1* $\geq$ *length* Ψ
using *A core-optimal-pre-witness-pigeon-hole*
by *fastforce*
hence *length* Δ $\geq$ *n*
using *C*
by *simp*
have *length* Δ = *length* ($\mathfrak{V}$ Δ)
by (*induct* Δ, *simp*+)
hence *length* (*remdups* ($\mathfrak{V}$ Δ)) = *length* ($\mathfrak{V}$ Δ)
by (*simp add*: *core-optimal-pre-witness-distinct*)
hence *length* (*remdups* ($\mathfrak{V}$ Δ)) $\geq$ *n*
using ‹*length* Δ = *length* ($\mathfrak{V}$ Δ)› ‹*n* $\leq$ *length* Δ›
by *linarith*
have *mset* (*remdups* ($\mathfrak{V}$ Δ @ ψ)) = *mset* (*remdups* (ψ @ $\mathfrak{V}$ Δ))

by (*simp add*: *mset-remdups*)
hence *length* (*remdups* ($\mathfrak{V}$ Δ @ ψ)) $\geq$ *length* (*remdups* ($\mathfrak{V}$ Δ))
by (*metis le-cases length-sub-mset mset-remdups-append-msub size-mset*)
hence *length* (*remdups* ($\mathfrak{V}$ Δ @ ψ)) $\geq$ *n*
using ‹*n* $\leq$ *length* (*remdups* ($\mathfrak{V}$ Δ))› *dual-order.trans* **by** *blast*
thus *?thesis* **using** *False* ψ(*2*)
by *simp*
qed
}
hence $\forall$ σ' $\in$ *set* (*map* (*map snd* $\circ$ *remdups*) (*?T*$_\Sigma$ Ψ)). *length* σ' $\geq$ *n*
by *blast*
}
then show *?case* **by** *fastforce*
qed
}
hence $\forall$ σ' $\in$ *set ?*Σ'. *length* σ' + *1* $\geq$ *length* (*?*$\mathcal{S}$ Ψ_0)
using ‹*mset* (*?*$\mathcal{S}$ Ψ_0) $\subseteq$# *mset* ($\mathfrak{V}$ Ξ)›
by *fastforce*
hence $\forall$ σ' $\in$ *set ?*Σ'. *length* σ' + *1* $\geq$ *length* Ψ_0 **by** *simp*
hence $\forall$ σ $\in$ *set ?*Σ. *length* σ + *1* $\geq$ *length* Ψ_0
using ‹$\forall$ σ $\in$ *set ?*Σ. $\exists$ σ' $\in$ *set ?*Σ'. *length* σ = *length* σ'›
by *fastforce*
thus *?thesis* **using** Ψ_0 **by** *fastforce*
qed
have *III*: $\forall$ σ $\in$ *set ?*Σ. *mset* σ $\subseteq$# *mset* Ξ
proof −
have *remdups* ($\mathfrak{V}$ Ξ) = $\mathfrak{V}$ Ξ
by (*simp add*: *core-optimal-pre-witness-distinct distinct-remdups-id*)
from Ψ_0(*1*) **have** *set* Ψ_0 $\subseteq$ *set* ($\mathfrak{V}$ Ξ)
by (*metis* (*no-types*, *lifting*) ‹*remdups* ($\mathfrak{V}$ Ξ) = $\mathfrak{V}$ Ξ›
mset-remdups-set-sub-iff
mset-remdups-subset-eq
subset-mset.dual-order.trans)
hence $\forall$ σ $\in$ *set* (*?T*$_\Sigma$ Ψ_0). *set* σ $\subseteq$ *set* ($\mathfrak{V}$ Ξ)
proof (*induct* Ψ_0)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ_0)
hence $\forall$ σ $\in$ *set* (*?T*$_\Sigma$ Ψ_0). *set* σ $\subseteq$ *set* ($\mathfrak{V}$ Ξ) **by** *auto*
obtain Δ δ **where** ψ = (Δ,δ) **by** *fastforce*
hence (Δ, δ) $\in$ *set* ($\mathfrak{V}$ Ξ) **using** *Cons* **by** *simp*
{
fix σ :: ('*a list* $\times$ '*a*) *list*
assume $\star$: σ $\in$ (#) (Δ, δ) ‘ *set* (*?T*$_\Sigma$ Ψ_0) $\cup$ (@) ($\mathfrak{V}$ Δ) ‘ *set* (*?T*$_\Sigma$ Ψ_0)
have *set* σ $\subseteq$ *set* ($\mathfrak{V}$ Ξ)
proof (*cases* σ $\in$ (#) (Δ, δ) ‘ *set* (*?T*$_\Sigma$ Ψ_0))
case *True*

then show *?thesis*
  using ‹∀ $\sigma \in$ *set* (*?*T_Σ Ψ_0). *set* $\sigma \subseteq$ *set* ($\mathfrak{V}$ Ξ)› ‹(Δ, δ) $\in$ *set* ($\mathfrak{V}$ Ξ)›
  by *fastforce*
next
  case *False*
  hence $\sigma \in$ (@) ($\mathfrak{V}$ Δ) ‘ *set* (*?*T_Σ Ψ_0) **using** ⋆ **by** *simp*
  moreover have *set* ($\mathfrak{V}$ Δ) $\subseteq$ *set* ($\mathfrak{V}$ Ξ)
    using *core-optimal-pre-witness-element-inclusion* ‹(Δ, δ) $\in$ *set* ($\mathfrak{V}$ Ξ)›
    by *fastforce*
  ultimately show *?thesis*
    using ‹∀ $\sigma \in$ *set* (*?*T_Σ Ψ_0). *set* $\sigma \subseteq$ *set* ($\mathfrak{V}$ Ξ)›
    by *force*
qed
}
hence ∀$\sigma\in$(#) (Δ, δ) ‘ *set* (*?*T_Σ Ψ_0) ∪ (@) ($\mathfrak{V}$ Δ) ‘ *set* (*?*T_Σ Ψ_0). *set* σ
$\subseteq$ *set* ($\mathfrak{V}$ Ξ)
  by *auto*
thus *?case* **using** ‹ψ = (Δ, δ)› **by** *simp*
qed
hence ∀ $\sigma \in$ *set* (*?*T_Σ Ψ_0). *mset* (*remdups* σ) ⊆# *mset* (*remdups* ($\mathfrak{V}$ Ξ))
  using *mset-remdups-set-sub-iff* **by** *blast*
hence ∀ $\sigma \in$ *set* *?*Σ. *mset* σ ⊆# *mset* (*map snd* ($\mathfrak{V}$ Ξ))
  using *map-monotonic* ‹*remdups* ($\mathfrak{V}$ Ξ) = $\mathfrak{V}$ Ξ›
  by *auto*
moreover have *map snd* ($\mathfrak{V}$ Ξ) = Ξ **by** (*induct* Ξ, *simp*+)
ultimately show *?thesis* **by** *simp*
qed
show *?thesis* **using** *I II III* **by** *fastforce*
qed
from *this* **obtain** Σ_0 **where** Σ_0:
  $\vdash$ (Ψ :→ φ) ↔ (⨆ (*map* ⨅ Σ_0) → φ)
  ∀ $\sigma \in$ *set* Σ_0. *mset* σ ⊆# *mset* Ξ ∧ *length* σ + *1* ≥ *length* Ψ
  by *blast*
moreover
have (Φ @ Ψ) :→ φ = Φ :→ (Ψ :→ φ) **by** (*induct* Φ, *simp*+)
hence $\vdash$ ((Φ @ Ψ) :→ φ) ↔ (⨅ Φ → (Ψ :→ φ))
  by (*simp add*: *list-curry-uncurry*)
moreover have $\vdash$ (Ψ :→ φ) ↔ (⨆ (*map* ⨅ Σ_0) → φ)
    → (Φ @ Ψ) :→ φ ↔ (⨅ Φ → Ψ :→ φ)
    → (Φ @ Ψ) :→ φ ↔ ((⨅ Φ ⊓ ⨆ (*map* ⨅ Σ_0)) → φ)
proof −
  let *?*φ = ⟨Ψ :→ φ⟩ ↔ (⟨⨆ (*map* ⨅ Σ_0)⟩ → ⟨φ⟩)
    → ⟨(Φ @ Ψ) :→ φ⟩ ↔ (⟨⨅ Φ⟩ → ⟨Ψ :→ φ⟩)
    → ⟨(Φ @ Ψ) :→ φ⟩ ↔ ((⟨⨅ Φ⟩ ⊓ ⟨⨆ (*map* ⨅ Σ_0)⟩) → ⟨φ⟩)
  have ∀$\mathfrak{M}$. $\mathfrak{M}$ $\models_{prop}$ *?*φ **by** *fastforce*
  hence $\vdash$ ⦇ *?*φ ⦈ **using** *propositional-semantics* **by** *blast*
  thus *?thesis* **by** *simp*
qed
moreover

let $?\Sigma = \mathit{map}\ ((@)\ \Phi)\ \Sigma_0$
have $\forall \varphi\ \psi\ \chi.\ \vdash (\varphi \to \psi) \to \chi \to \psi \lor \neg \vdash \chi \to \varphi$
by (*meson modus-ponens flip-hypothetical-syllogism*)
hence $\vdash ((\bigsqcap\ \Phi \sqcap \bigsqcup\ (\mathit{map} \bigsqcap\ \Sigma_0)) \to \varphi) \leftrightarrow (\bigsqcup\ (\mathit{map} \bigsqcap\ ?\Sigma) \to \varphi)$
using *append-dnf-distribute biconditional-def* **by** *fastforce*
ultimately have $\vdash (\Phi\ @\ \Psi) :\to \varphi \leftrightarrow (\bigsqcup\ (\mathit{map} \bigsqcap\ ?\Sigma) \to \varphi)$
using *modus-ponens biconditional-transitivity-rule*
by *blast*
moreover
{
fix σ
assume $\sigma \in \mathit{set}\ ?\Sigma$
from *this* **obtain** σ_0 **where** σ_0: $\sigma = \Phi\ @\ \sigma_0\ \sigma_0 \in \mathit{set}\ \Sigma_0$ **by** (*simp*, *blast*)
hence $\mathit{mset}\ \sigma_0 \subseteq\#\ \mathit{mset}\ \Xi$ **using** $\Sigma_0(2)$ **by** *blast*
hence $\mathit{mset}\ \sigma \subseteq\#\ \mathit{mset}\ (\Phi\ @\ \Xi)$ **using** $\sigma_0(1)$ **by** *simp*
hence $\mathit{mset}\ \sigma \subseteq\#\ \mathit{mset}\ \Gamma$ **using** *assms*(*1*) *assms*(*2*)
by (*simp*, *meson subset-mset.dual-order.trans subset-mset.le-diff-conv2*)
moreover
have $\mathit{length}\ \sigma + 1 \geq \mathit{length}\ (\Phi\ @\ \Psi)$ **using** $\Sigma_0(2)\ \sigma_0$ **by** *simp*
ultimately have $\mathit{mset}\ \sigma \subseteq\#\ \mathit{mset}\ \Gamma\ \mathit{length}\ \sigma + 1 \geq \mathit{length}\ (\Phi\ @\ \Psi)$ **by** *auto*
}
ultimately
show *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *unproving-core-optimal-witness*:
assumes $\neg \vdash \varphi$
shows $0 < (\|\ \Gamma\ \|_\varphi)$
$= (\exists\ \Sigma.\ \mathit{mset}\ (\mathit{map}\ \mathit{snd}\ \Sigma) \subseteq\#\ \mathit{mset}\ \Gamma\ \land$
$\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ \Sigma :\vdash \varphi\ \land$
$1 + (\|\ \mathit{map}\ (\mathit{uncurry}\ (\to))\ \Sigma\ @\ \Gamma \ominus \mathit{map}\ \mathit{snd}\ \Sigma\ \|_\varphi) = \|\ \Gamma\ \|_\varphi)$
proof (*rule iffI*)
assume $0 < \|\ \Gamma\ \|_\varphi$
from *this* **obtain** Ξ **where** Ξ: $\Xi \in \mathcal{C}\ \Gamma\ \varphi\ \mathit{length}\ \Xi < \mathit{length}\ \Gamma$
using $\langle \neg \vdash \varphi \rangle$
complement-core-size-def
core-size-intro
unproving-core-existence
by *fastforce*
from *this* **obtain** ψ **where** ψ: $\psi \in \mathit{set}\ (\Gamma \ominus \Xi)$
by (*metis* $\langle 0 < \|\ \Gamma\ \|_\varphi \rangle$
less-not-refl
list.exhaust
list.set-intros(*1*)
list.size(*3*)
complement-core-size-intro)
let $?\Sigma = \mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi)$
let $?\Sigma_A = \mathfrak{W}_\sqcup\ \varphi\ (\psi\ \#\ \Xi)$
let $?\Sigma_B = \mathfrak{W}_\to\ \varphi\ (\psi\ \#\ \Xi)$

```
have ◇: mset (ψ # Ξ) ⊆# mset Γ
        ψ # Ξ :⊢ φ
  using Ξ(1) ψ
        unproving-core-def
        list-deduction-theorem
        unproving-core-complement-deduction
        msub-list-subtract-elem-cons-msub [where Ξ=Ξ]
  by blast+
moreover have map snd ?Σ = ψ # Ξ by (induct Ξ, simp+)
ultimately have ?Σ_A :⊢ φ
                mset (map snd ?Σ) ⊆# mset Γ
  using core-optimal-witness-deduction
        list-deduction-def weak-biconditional-weaken
  by (metis+)
moreover
{
  let ?Γ' = ?Σ_B @ Γ ⊖ map snd ?Σ
  have A: length ?Σ_B = 1 + length Ξ
    by (induct Ξ, simp+)
  have B: ?Σ_B ∈ 𝒞 ?Γ' φ
  proof −
    have ¬ ?Σ_B :⊢ φ
      by (metis (no-types, lifting)
                Ξ(1) ‹?Σ_A :⊢ φ›
                modus-ponens list-deduction-def
                optimal-witness-split-identity
                unproving-core-def
                mem-Collect-eq)
    moreover have mset ?Σ_B ⊆# mset ?Γ'
      by simp
    hence ∀ Ψ. mset Ψ ⊆# mset ?Γ' ⟶ ¬ Ψ :⊢ φ ⟶ length Ψ ≤ length ?Σ_B
    proof −
      have ∀ Ψ ∈ 𝒞 ?Γ' φ. length Ψ = length ?Σ_B
      proof (rule ccontr)
        assume ¬ (∀ Ψ ∈ 𝒞 ?Γ' φ. length Ψ = length ?Σ_B)
        from this obtain Ψ where
          Ψ: Ψ ∈ 𝒞 ?Γ' φ
             length Ψ ≠ length ?Σ_B
          by blast
        have length Ψ ≥ length ?Σ_B
          using Ψ(1)
                ‹¬ ?Σ_B :⊢ φ›
                ‹mset ?Σ_B ⊆# mset ?Γ'›
          unfolding unproving-core-def
          by blast
        hence length Ψ > length ?Σ_B
          using Ψ(2)
          by linarith
        have length Ψ = length (Ψ ⊖ ?Σ_B) + length (Ψ ∩ ?Σ_B)
```

```
  (is length Ψ = length ?A + length ?B)
  by (metis (no-types, lifting)
          length-append
          list-diff-intersect-comp
          mset-append
          mset-eq-length)
{
  fix σ
  assume mset σ ⊆# mset Γ
        length σ + 1 ≥ length (?A @ ?B)
  hence length σ + 1 ≥ length Ψ
    using ‹length Ψ = length ?A + length ?B›
    by simp
  hence length σ + 1 > length ?Σ_B
    using ‹length Ψ > length ?Σ_B› by linarith
  hence length σ + 1 > length Ξ + 1
    using A by simp
  hence length σ > length Ξ by linarith
  have σ :⊢ φ
  proof (rule ccontr)
    assume ¬ σ :⊢ φ
    hence length σ ≤ length Ξ
      using ‹mset σ ⊆# mset Γ› Ξ(1)
      unfolding unproving-core-def
      by blast
    thus False using ‹length σ > length Ξ› by linarith
  qed
}
moreover
have mset Ψ ⊆# mset ?Γ′
    ¬ Ψ :⊢ φ
    ∀Φ. mset Φ ⊆# mset ?Γ′ ∧ ¬ Φ :⊢ φ ⟶ length Φ ≤ length Ψ
  using Ψ(1) unproving-core-def by blast+
hence mset ?A ⊆# mset (Γ ⊖ map snd ?Σ)
  by (simp add: add.commute subset-eq-diff-conv)
hence mset ?A ⊆# mset (Γ ⊖ (ψ # Ξ))
  using ‹map snd ?Σ = ψ # Ξ› by metis
moreover
have mset ?B ⊆# mset (𝔚→ φ (ψ # Ξ))
  using list-intersect-right-project by blast
ultimately obtain Σ where Σ: ⊢ ((?A @ ?B) :→ φ) ↔ (⨆ (map ⨅ Σ)
→ φ)
                      ∀σ∈set Σ. σ :⊢ φ
  using ◇ optimal-witness-list-intersect-biconditional
  by metis
hence ⊢ ⨆ (map ⨅ Σ) → φ
  using weak-disj-of-conj-equiv by blast
hence ?A @ ?B :⊢ φ
  using Σ(1) modus-ponens list-deduction-def weak-biconditional-weaken
```

by *blast*
moreover have *set* (*?A* @ *?B*) = *set* Ψ
using *list-diff-intersect-comp union-code set-mset-mset* **by** *metis*
hence *?A* @ *?B* :$\vdash$ φ = Ψ :$\vdash$ φ
using *list-deduction-monotonic* **by** *blast*
ultimately have Ψ :$\vdash$ φ **by** *metis*
thus *False* **using** Ψ(*1*) **unfolding** *unproving-core-def* **by** *blast*
qed
moreover have $\exists$ Ψ. $\Psi \in \mathcal{C}$ *?*Γ' φ
using *assms unproving-core-existence* **by** *blast*
ultimately show *?thesis*
using *unproving-core-def*
by *fastforce*
qed
ultimately show *?thesis*
unfolding *unproving-core-def*
by *fastforce*
qed
have *C*: $\forall \Xi\ \Gamma\ \varphi.\ \Xi \in \mathcal{C}\ \Gamma\ \varphi \longrightarrow$ *length* $\Xi = \mid \Gamma \mid_\varphi$
using *core-size-intro* **by** *blast*
then have *D*: *length* $\Xi = \mid \Gamma \mid_\varphi$
using ‹$\Xi \in \mathcal{C}\ \Gamma\ \varphi$› **by** *blast*
have
$\forall (\Sigma$ ::'*a list*) Γ *n*. ($\neg$ *mset* $\Sigma \subseteq\#$ *mset* $\Gamma \vee$ *length* ($\Gamma \ominus \Sigma$) $\neq$ *n*) $\vee$ *length* Γ = *n* + *length* Σ
using *list-subtract-msub-eq* **by** *blast*
then have *E*: *length* Γ = *length* ($\Gamma \ominus$ *map snd* ($\mathfrak{W}\ \varphi$ (ψ # Ξ))) + *length* (ψ # Ξ)
using ‹*map snd* ($\mathfrak{W}\ \varphi$ (ψ # Ξ)) = ψ # Ξ› ‹*mset* (ψ # Ξ) $\subseteq\#$ *mset* Γ› **by** *presburger*
have *1* + *length* $\Xi = \mid \mathfrak{W}_\rightarrow\ \varphi$ (ψ # Ξ) @ $\Gamma \ominus$ *map snd* ($\mathfrak{W}\ \varphi$ (ψ # Ξ)) $\mid_\varphi$
using *C B A* **by** *presburger*
hence *1* + ($\|$ *map* (*uncurry* ($\rightarrow$)) *?*Σ @ $\Gamma \ominus$ *map snd ?*Σ $\|_\varphi$) = $\| \Gamma \|_\varphi$
using *D E* ‹*map snd* ($\mathfrak{W}\ \varphi$ (ψ # Ξ)) = ψ # Ξ› *complement-core-size-def* **by** *force*
}
ultimately
show $\exists$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma \wedge$
map (*uncurry* ($\sqcup$)) Σ :$\vdash$ $\varphi \wedge$
1 + ($\|$ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ $\|_\varphi$) = $\| \Gamma \|_\varphi$
by *metis*
next
assume $\exists$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma \wedge$
map (*uncurry* ($\sqcup$)) Σ :$\vdash$ $\varphi \wedge$
1 + ($\|$ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ $\|_\varphi$) = $\| \Gamma \|_\varphi$
thus *0* < $\| \Gamma \|_\varphi$
by *auto*
qed

```
primrec (in implication-logic) core-witness :: ('a × 'a) list ⇒ 'a list ⇒ ('a × 'a)
list (𝔘)
  where
    𝔘 - [] = []
  | 𝔘 Σ (ξ # Ξ) = (case find (λ σ. ξ = snd σ) Σ of
                     None ⇒ 𝔘 Σ Ξ
                   | Some σ ⇒ σ # (𝔘 (remove1 σ Σ) Ξ))

lemma (in implication-logic) core-witness-right-msub:
  mset (map snd (𝔘 Σ Ξ)) ⊆# mset Ξ
proof −
  have ∀ Σ. mset (map snd (𝔘 Σ Ξ)) ⊆# mset Ξ
  proof (induct Ξ)
    case Nil
    then show ?case by simp
  next
    case (Cons ξ Ξ)
    {
      fix Σ
      have mset (map snd (𝔘 Σ (ξ # Ξ))) ⊆# mset (ξ # Ξ)
      proof (cases find (λ σ. ξ = snd σ) Σ)
        case None
        then show ?thesis
          by (simp, metis Cons.hyps
                          add-mset-add-single
                          mset-map mset-subset-eq-add-left subset-mset.order-trans)
      next
        case (Some σ)
        note σ = this
        hence ξ = snd σ
          by (meson find-Some-predicate)
        moreover
        have σ ∈ set Σ
        using σ
        proof (induct Σ)
          case Nil
          then show ?case by simp
        next
          case (Cons σ' Σ)
          then show ?case
            by (cases ξ = snd σ', simp+)
        qed
        ultimately show ?thesis using σ Cons.hyps by simp
      qed
    }
    then show ?case by simp
  qed
  thus ?thesis by simp
qed
```

lemma (**in** *implication-logic*) *core-witness-left-msub*:
  mset ($\mathfrak{U}$ Σ Ξ) $\subseteq\#$ *mset* Σ
proof −
  have $\forall$ Σ. *mset* ($\mathfrak{U}$ Σ Ξ) $\subseteq\#$ *mset* Σ
  proof (*induct* Ξ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ξ Ξ)
    {
      fix Σ
      have *mset* ($\mathfrak{U}$ Σ (ξ # Ξ)) $\subseteq\#$ *mset* Σ
      proof (*cases find* (λ σ. ξ = *snd* σ) Σ)
        case *None*
        then show *?thesis* **using** *Cons.hyps* **by** *simp*
      next
        case (*Some* σ)
        note σ = *this*
        hence $\sigma \in$ *set* Σ
        proof (*induct* Σ)
          case *Nil*
          then show *?case* **by** *simp*
        next
          case (*Cons* σ' Σ)
          then show *?case*
            by (*cases* ξ = *snd* σ', *simp+*)
        qed
          moreover from *Cons.hyps* **have** *mset* ($\mathfrak{U}$ (*remove1* σ Σ) Ξ) $\subseteq\#$ *mset*
(*remove1* σ Σ)
          by *blast*
        hence *mset* ($\mathfrak{U}$ Σ (ξ # Ξ)) $\subseteq\#$ *mset* (σ # *remove1* σ Σ) **using** σ **by** *simp*
        ultimately show *?thesis* **by** *simp*
      qed
    }
    then show *?case* **by** *simp*
  qed
  thus *?thesis* **by** *simp*
qed

lemma (**in** *implication-logic*) *core-witness-right-projection*:
  mset (*map snd* ($\mathfrak{U}$ Σ Ξ)) = *mset* ((*map snd* Σ) $\cap$ Ξ)
proof −
  have $\forall$ Σ. *mset* (*map snd* ($\mathfrak{U}$ Σ Ξ)) = *mset* ((*map snd* Σ) $\cap$ Ξ)
  proof (*induct* Ξ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ξ Ξ)

{
  fix Σ
  have *mset* (*map snd* ($\mathfrak{U}$ Σ (ξ # Ξ))) = *mset* (*map snd* Σ $\cap$ ξ # Ξ)
  proof (*cases find* (λ σ. ξ = *snd* σ) Σ)
    case *None*
    hence $\xi \notin$ *set* (*map snd* Σ)
    proof (*induct* Σ)
      case *Nil*
      then show *?case* **by** *simp*
    next
      case (*Cons* σ Σ)
      have *find* ($\lambda\sigma$. ξ = *snd* σ) Σ = *None*
           $\xi \neq$ *snd* σ
        using *Cons.prems*
       by (*auto*, *metis Cons.prems find.simps(2) find-None-iff list.set-intros(1)*)
      then show *?case* **using** *Cons.hyps* **by** *simp*
    qed
    then show *?thesis* **using** *None Cons.hyps* **by** *simp*
  next
    case (*Some* σ)
    hence $\sigma \in$ *set* Σ ξ = *snd* σ
      by (*meson find-Some-predicate find-Some-set-membership*)+
    moreover
    from ‹$\sigma \in$ *set* Σ› **have** *mset* Σ = *mset* (σ # (*remove1* σ Σ))
      by *simp*
     hence *mset* (*map snd* Σ) = *mset* ((*snd* σ) # (*remove1* (*snd* σ) (*map snd*
Σ)))
           mset (*map snd* Σ) = *mset* (*map snd* (σ # (*remove1* σ Σ)))
      by (*simp add*: ‹$\sigma \in$ *set* Σ›, *metis map-monotonic subset-mset.eq-iff*)
    hence *mset* (*map snd* (*remove1* σ Σ)) = *mset* (*remove1* (*snd* σ) (*map snd*
Σ))
      by *simp*
    ultimately show *?thesis* **using** *Some Cons.hyps* **by** *simp*
  qed
}
then show *?case* **by** *simp*
qed
thus *?thesis* **by** *simp*
qed

lemma (**in** *classical-logic*) *witness-list-implication-rule*:
  $\vdash$ (*map* (*uncurry* ($\sqcup$)) Σ :$\rightarrow$ φ) $\rightarrow$ $\prod$ (*map* (λ (χ, ξ). ($\chi \rightarrow \xi$) $\rightarrow \varphi$) Σ) $\rightarrow \varphi$
proof (*induct* Σ)
  case *Nil*
  then show *?case* **using** *axiom-k* **by** *simp*
next
  case (*Cons* σ Σ)
  let *?*χ = *fst* σ

let $?\xi = snd\ \sigma$
let $?\Sigma_A = map\ (uncurry\ (\sqcup))\ \Sigma$
let $?\Sigma_B = map\ (\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)\ \Sigma$
assume $\vdash\ ?\Sigma_A :\rightarrow \varphi \rightarrow \bigsqcap\ ?\Sigma_B \rightarrow \varphi$
moreover have
$\vdash (?\Sigma_A :\rightarrow \varphi \rightarrow \bigsqcap\ ?\Sigma_B \rightarrow \varphi)$
$\rightarrow ((?\chi \sqcup ?\xi) \rightarrow ?\Sigma_A :\rightarrow \varphi) \rightarrow (((?\chi \rightarrow ?\xi) \rightarrow \varphi) \sqcap \bigsqcap\ ?\Sigma_B) \rightarrow \varphi$
proof $-$
let $?\varphi = (\langle ?\Sigma_A :\rightarrow \varphi\rangle \rightarrow \langle \bigsqcap\ ?\Sigma_B\rangle \rightarrow \langle\varphi\rangle)$
$\rightarrow ((((\langle ?\chi\rangle \sqcup \langle ?\xi\rangle) \rightarrow \langle ?\Sigma_A :\rightarrow \varphi\rangle) \rightarrow (((\langle ?\chi\rangle \rightarrow \langle ?\xi\rangle) \rightarrow \langle\varphi\rangle) \sqcap$
$\langle \bigsqcap\ ?\Sigma_B\rangle) \rightarrow \langle\varphi\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately have $\vdash ((?\chi \sqcup ?\xi) \rightarrow ?\Sigma_A :\rightarrow \varphi) \rightarrow (((?\chi \rightarrow ?\xi) \rightarrow \varphi) \sqcap \bigsqcap\ ?\Sigma_B)$
$\rightarrow \varphi$
using *modus-ponens* **by** *blast*
moreover
have $(\lambda\ \sigma.\ (fst\ \sigma \rightarrow snd\ \sigma) \rightarrow \varphi) = (\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)$
$uncurry\ (\sqcup) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$
by *fastforce+*
hence $(\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)\ \sigma = (?\chi \rightarrow ?\xi) \rightarrow \varphi$
$uncurry\ (\sqcup)\ \sigma = ?\chi \sqcup ?\xi$
by *metis+*
ultimately show *?case* **by** *simp*
qed

lemma (**in** *classical-logic*) *witness-core-size-increase*:
assumes $\neg \vdash \varphi$
and $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma$
and $map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi$
shows $(|\ \Gamma\ |_\varphi) < (|\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ |_\varphi)$
proof $-$
from $\langle\neg \vdash \varphi\rangle$ **obtain** Ξ **where** Ξ: $\Xi \in \mathcal{C}\ \Gamma\ \varphi$
using *unproving-core-existence* **by** *blast*
let $?\Sigma' = \Sigma \ominus \mathfrak{U}\ \Sigma\ \Xi$
let $?\Sigma\Xi' = map\ (uncurry\ (\sqcup))\ (\mathfrak{U}\ \Sigma\ \Xi)\ @\ map\ (uncurry\ (\rightarrow))\ (\mathfrak{U}\ \Sigma\ \Xi)$
have $mset\ \Sigma = mset\ (\mathfrak{U}\ \Sigma\ \Xi\ @\ ?\Sigma')$ **by** (*simp add*: *core-witness-left-msub*)
hence $set\ (map\ (uncurry\ (\sqcup))\ \Sigma) = set\ (map\ (uncurry\ (\sqcup))\ ((\mathfrak{U}\ \Sigma\ \Xi)\ @\ ?\Sigma'))$
by (*metis mset-map mset-eq-setD*)
hence $map\ (uncurry\ (\sqcup))\ ((\mathfrak{U}\ \Sigma\ \Xi)\ @\ ?\Sigma') :\vdash \varphi$
using *list-deduction-monotonic assms*(*3*)
by *blast*
hence $map\ (uncurry\ (\sqcup))\ (\mathfrak{U}\ \Sigma\ \Xi)\ @\ map\ (uncurry\ (\sqcup))\ ?\Sigma' :\vdash \varphi$ **by** *simp*
moreover
{
fix $\Phi\ \Psi$
have $((\Phi\ @\ \Psi) :\rightarrow \varphi) = (\Phi :\rightarrow (\Psi :\rightarrow \varphi))$

by (*induct* Φ, *simp*+)

hence $(\Phi \ @ \ \Psi) :\vdash \varphi = \Phi :\vdash (\Psi :\rightarrow \varphi)$

unfolding *list-deduction-def*

by (*induct* Φ, *simp*+)

}

ultimately have *map* (*uncurry* $(\sqcup)$) $(\mathfrak{U}\ \Sigma\ \Xi) :\vdash$ *map* (*uncurry* $(\sqcup)$) $?\Sigma' :\rightarrow \varphi$

by *simp*

moreover have *set* (*map* (*uncurry* $(\sqcup)$) $(\mathfrak{U}\ \Sigma\ \Xi)) \subseteq$ *set* $?\Sigma\Xi'$

by *simp*

ultimately have $?\Sigma\Xi' :\vdash$ *map* (*uncurry* $(\sqcup)$) $?\Sigma' :\rightarrow \varphi$

using *list-deduction-monotonic* **by** *blast*

hence $?\Sigma\Xi' :\vdash \prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi$

using *list-deduction-modus-ponens*
list-deduction-weaken
witness-list-implication-rule

by *blast*

hence $?\Sigma\Xi'$ \$$\vdash$ $[\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi]$

using *segmented-deduction-one-collapse* **by** *metis*

hence

$?\Sigma\Xi'$ @ (*map snd* $(\mathfrak{U}\ \Sigma\ \Xi)) \ominus$ (*map snd* $(\mathfrak{U}\ \Sigma\ \Xi)$)
\$$\vdash$ $[\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi]$

by *simp*

hence *map snd* $(\mathfrak{U}\ \Sigma\ \Xi)$ \$$\vdash$ $[\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi]$

using *segmented-witness-left-split* [**where** Γ=*map snd* $(\mathfrak{U}\ \Sigma\ \Xi)$
and Σ=$\mathfrak{U}\ \Sigma\ \Xi$]

by *fastforce*

hence *map snd* $(\mathfrak{U}\ \Sigma\ \Xi)$ \$$\vdash$ $[\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi]$

using *core-witness-right-projection* **by** *auto*

hence *map snd* $(\mathfrak{U}\ \Sigma\ \Xi) :\vdash \prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)\ ?\Sigma') \rightarrow \varphi$

using *segmented-deduction-one-collapse* **by** *blast*

hence $\star$:

map snd $(\mathfrak{U}\ \Sigma\ \Xi)$ @ $\Xi \ominus$ (*map snd* $\Sigma) :\vdash \prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \rightarrow \gamma) \rightarrow \varphi)$
$?\Sigma') \rightarrow \varphi$

(**is** $?\Xi_0 :\vdash$ -)

using *list-deduction-monotonic*

by (*metis* (*no-types*, *lifting*) *append-Nil2*
segmented-cancel
segmented-deduction.simps(*1*)
segmented-list-deduction-antitonic)

have *mset* Ξ = *mset* $(\Xi \ominus$ (*map snd* Σ)) + *mset* $(\Xi \cap$ (*map snd* Σ))

using *list-diff-intersect-comp* **by** *blast*

hence *mset* Ξ = *mset* ((*map snd* $\Sigma) \cap \Xi$) + *mset* $(\Xi \ominus$ (*map snd* Σ))

by (*metis subset-mset.inf-commute list-intersect-mset-homomorphism union-commute*)

hence *mset* Ξ = *mset* (*map snd* $(\mathfrak{U}\ \Sigma\ \Xi)$) + *mset* $(\Xi \ominus$ (*map snd* Σ))

using *core-witness-right-projection* **by** *simp*

hence *mset* Ξ = *mset* $?\Xi_0$

by *simp*

hence *set* Ξ = *set* $?\Xi_0$

by (*metis mset-eq-setD*)

have ¬ $?\Xi_0$:⊢ $\prod$ (*map* (λ (χ, γ). (χ → γ) → φ) $?\Sigma'$)
proof (*rule notI*)
  assume $?\Xi_0$:⊢ $\prod$ (*map* (λ (χ, γ). (χ → γ) → φ) $?\Sigma'$)
  hence $?\Xi_0$:⊢ φ
    using ⋆ *list-deduction-modus-ponens* **by** *blast*
  hence Ξ :⊢ φ
    using *list-deduction-monotonic* ‹*set* Ξ = *set* $?\Xi_0$› **by** *blast*
  thus *False*
    using Ξ *unproving-core-def* **by** *blast*
qed
moreover
have *mset* (*map snd* (𝔘 Σ Ξ)) ⊆# *mset* $?\Xi_0$
    mset (*map* (*uncurry* (→)) (𝔘 Σ Ξ) @ $?\Xi_0$ ⊖ *map snd* (𝔘 Σ Ξ))
   = *mset* (*map* (*uncurry* (→)) (𝔘 Σ Ξ) @ Ξ ⊖ (*map snd* Σ))
    (**is** - = *mset* $?\Xi_1$)
  by *auto*
hence $?\Xi_1 \preceq ?\Xi_0$
  by (*metis add.commute*
        witness-stronger-theory
        add-diff-cancel-right′
        list-subtract.simps(*1*)
        list-subtract-mset-homomorphism
        list-diff-intersect-comp
        list-intersect-right-project
        msub-stronger-theory-intro
        stronger-theory-combine
        stronger-theory-empty-list-intro
        self-append-conv)
ultimately have
  ¬ $?\Xi_1$:⊢ $\prod$ (*map* (λ (χ, γ). (χ → γ) → φ) $?\Sigma'$)
  using *stronger-theory-deduction-monotonic* **by** *blast*
from *this* **obtain** χ γ **where**
  (χ,γ) ∈ *set* $?\Sigma'$
  ¬ (χ → γ) # $?\Xi_1$:⊢ φ
  using *list-deduction-theorem*
  by *fastforce*
have *mset* (χ → γ # $?\Xi_1$) ⊆# *mset* (*map* (*uncurry* (→)) Σ @ Γ ⊖ *map snd* Σ)
proof −
  let *?A* = *map* (*uncurry* (→)) Σ
  let *?B* = *map* (*uncurry* (→)) (𝔘 Σ Ξ)
  have (χ,γ) ∈ (*set* Σ − *set* (𝔘 Σ Ξ))
  proof −
    from ‹(χ,γ) ∈ *set* $?\Sigma'$› **have** γ ∈# *mset* (*map snd* (Σ ⊖ 𝔘 Σ Ξ))
      by (*metis set-mset-mset image-eqI set-map snd-conv*)
    hence γ ∈# *mset* (*map snd* Σ ⊖ *map snd* (𝔘 Σ Ξ))
      by (*metis core-witness-left-msub map-list-subtract-mset-equivalence*)
    hence γ ∈# *mset* (*map snd* Σ ⊖ (*map snd* Σ ∩ Ξ))
      by (*metis core-witness-right-projection list-subtract-mset-homomorphism*)
    hence γ ∈# *mset* (*map snd* Σ ⊖ Ξ)

by (*metis add-diff-cancel-right′*
list-subtract-mset-homomorphism
list-diff-intersect-comp)
moreover from *assms*(*2*) **have** *mset* (*map snd* $\Sigma \ominus \Xi$) $\subseteq\#$ *mset* ($\Gamma \ominus \Xi$)
by (*simp*, *metis list-subtract-monotonic list-subtract-mset-homomorphism mset-map*)
ultimately have $\gamma \in\#$ *mset* ($\Gamma \ominus \Xi$)
by (*simp add*: *mset-subset-eqD*)
hence $\gamma \in$ *set* ($\Gamma \ominus \Xi$)
using *set-mset-mset* **by** *fastforce*
hence $\gamma \in$ *set* Γ − *set* Ξ
using Ξ **by** *simp*
hence $\gamma \notin$ *set* Ξ
by *blast*
hence $\forall$ Σ. (χ,γ) $\notin$ *set* ($\mathfrak{U}$ Σ Ξ)
proof (*induct* Ξ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
{
fix Σ
have (χ, γ) $\notin$ *set* ($\mathfrak{U}$ Σ (ξ # Ξ))
proof (*cases find* ($\lambda\sigma$. ξ = *snd* σ) Σ)
case *None*
then show *?thesis* **using** *Cons* **by** *simp*
next
case (*Some* σ)
moreover from *this* **have** *snd* $\sigma = \xi$
using *find-Some-predicate* **by** *fastforce*
with *Cons.prems* **have** $\sigma \neq (\chi,\gamma)$ **by** *fastforce*
ultimately show *?thesis* **using** *Cons* **by** *simp*
qed
}
then show *?case* **by** *blast*
qed
moreover from ‹(χ,γ) $\in$ *set ?*Σ'› **have** (χ,γ) $\in$ *set* Σ
by (*meson list-subtract-set-trivial-upper-bound subsetCE*)
ultimately show *?thesis* **by** *fastforce*
qed
with ‹(χ, γ) $\in$ *set ?*Σ'› **have** *mset* ((χ,γ) # $\mathfrak{U}$ Σ Ξ) $\subseteq\#$ *mset* Σ
by (*meson core-witness-left-msub msub-list-subtract-elem-cons-msub*)
hence *mset* ($\chi \to \gamma$ # *?B*) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) Σ)
by (*metis* (*no-types*, *lifting*) ‹(χ, γ) $\in$ *set ?*Σ'›
core-witness-left-msub
map-list-subtract-mset-equivalence
map-monotonic
mset-eq-setD msub-list-subtract-elem-cons-msub
pair-imageI

set-map
uncurry-def)
moreover
have *mset* $\Xi \subseteq\#$ *mset* Γ
using Ξ *unproving-core-def*
by *blast*
hence *mset* ($\Xi \ominus$ (*map snd* Σ)) $\subseteq\#$ *mset* ($\Gamma \ominus$ (*map snd* Σ))
using *list-subtract-monotonic* **by** *blast*
ultimately show *?thesis*
using *subset-mset.add-mono* **by** *fastforce*
qed
moreover have *length* $?\Xi_1$ = *length* $?\Xi_0$
by *simp*
hence *length* $?\Xi_1$ = *length* Ξ
using ‹*mset* Ξ = *mset* $?\Xi_0$› *mset-eq-length* **by** *fastforce*
hence *length* $((\chi \to \gamma)$ # $?\Xi_1)$ = *length* Ξ + *1*
by *simp*
hence *length* $((\chi \to \gamma)$ # $?\Xi_1) = (| \Gamma |_\varphi) + 1$
using Ξ
by (*simp add*: *core-size-intro*)
moreover from ‹$\neg \vdash \varphi$› **obtain** Ω **where** Ω: $\Omega \in \mathcal{C}$ (*map* (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ *map snd* Σ) φ
using *unproving-core-existence* **by** *blast*
ultimately have *length* $\Omega \geq (| \Gamma |_\varphi) + 1$
using *unproving-core-def*
by (*metis* (*no-types*, *lifting*) ‹$\neg\ \chi \to \gamma$ # $?\Xi_1$:⊢ φ› *mem-Collect-eq*)
thus *?thesis*
using Ω *core-size-intro* **by** *auto*
qed

lemma (**in** *classical-logic*) *unproving-core-stratified-deduction-lower-bound*:
assumes $\neg \vdash \varphi$
shows (Γ #⊢ n φ) = ($n \leq \| \Gamma \|_\varphi$)
proof −
have $\forall$ Γ. (Γ #⊢ n φ) = ($n \leq \| \Gamma \|_\varphi$)
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Γ
assume Γ #⊢ (*Suc n*) φ
from *this* **obtain** Σ **where** Σ:
mset (*map snd* Σ) $\subseteq\#$ *mset* Γ
map (*uncurry* $(\sqcup)$) Σ :⊢ φ
map (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ (*map snd* Σ) #⊢ n φ
by *fastforce*
let $?\Gamma'$ = *map* (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ (*map snd* Σ)

have *length* Γ = *length* $?\Gamma'$
  using $\Sigma(1)$ *list-subtract-msub-eq* **by** *fastforce*
hence $(\| \Gamma \|_\varphi) > (\| ?\Gamma' \|_\varphi)$
  by (*metis* $\Sigma(1)$ $\Sigma(2)$ $\langle\neg \vdash \varphi\rangle$
        witness-core-size-increase
        length-core-decomposition
        add-less-cancel-right
        nat-add-left-cancel-less)
with $\Sigma(3)$ *Suc.hyps* **have** *Suc n* $\leq \| \Gamma \|_\varphi$
  by *auto*
}
moreover
{
  fix Γ
  assume *Suc n* $\leq \| \Gamma \|_\varphi$
  from *this* **obtain** Σ **where** Σ:
    mset (*map snd* Σ) $\subseteq\#$ *mset* Γ
    map (*uncurry* $(\sqcup)$) Σ $:\vdash \varphi$
    1 + $(\|$ *map* (*uncurry* $(\rightarrow)$) Σ @ $\Gamma \ominus$ *map snd* $\Sigma \|_\varphi) = \| \Gamma \|_\varphi$
    (**is** *1* + $(\| ?\Gamma' \|_\varphi) = \| \Gamma \|_\varphi$)
    by (*metis Suc-le-D assms unproving-core-optimal-witness zero-less-Suc*)
  have $n \leq \| ?\Gamma' \|_\varphi$
    using $\Sigma(3)$ $\langle$*Suc n* $\leq \| \Gamma \|_\varphi\rangle$ **by** *linarith*
  hence $?\Gamma'$ $\#\vdash$ *n* φ **using** *Suc* **by** *blast*
  hence Γ $\#\vdash$ (*Suc n*) φ **using** $\Sigma(1)$ $\Sigma(2)$ **by** *fastforce*
}
ultimately show *?case* **by** *metis*
qed
thus *?thesis* **by** *auto*
qed

lemma (**in** *classical-logic*) *stratified-deduction-tautology-equiv*:
  $(\forall\ n.\ \Gamma\ \#\vdash\ n\ \varphi) = \vdash \varphi$
proof (*cases* $\vdash \varphi$)
  case *True*
  then show *?thesis*
    by (*simp add*: *stratified-deduction-tautology-weaken*)
next
  case *False*
  have $\neg\ \Gamma\ \#\vdash$ (*1* + *length* Γ) φ
  proof (*rule notI*)
    assume $\Gamma\ \#\vdash$ (*1* + *length* Γ) φ
    hence *1* + *length* $\Gamma \leq \| \Gamma \|_\varphi$
      using $\langle\neg \vdash \varphi\rangle$ *unproving-core-stratified-deduction-lower-bound* **by** *blast*
    hence *1* + *length* $\Gamma \leq$ *length* Γ
      using *complement-core-size-def* **by** *fastforce*
    thus *False* **by** *linarith*
  qed
  then show *?thesis*

using ‹$\neg \vdash \varphi$› **by** *blast*
qed

lemma (**in** *classical-logic*) *unproving-core-max-stratified-deduction*:
$\Gamma \ \#\vdash \ n \ \varphi = (\forall \ \Phi \in \mathcal{C} \ \Gamma \ \varphi. \ n \leq \mathit{length} \ (\Gamma \ominus \Phi))$
proof (*cases* $\vdash \varphi$)
case *True*
from ‹$\vdash \varphi$› **have** $\Gamma \ \#\vdash \ n \ \varphi$
using *stratified-deduction-tautology-weaken*
by *blast*
moreover from ‹$\vdash \varphi$› **have** $\mathcal{C} \ \Gamma \ \varphi = \{\}$
using *unproving-core-existence* **by** *auto*
hence $\forall \ \Phi \in \mathcal{C} \ \Gamma \ \varphi. \ n \leq \mathit{length} \ (\Gamma \ominus \Phi)$ **by** *blast*
ultimately show *?thesis* **by** *meson*
next
case *False*
from ‹$\neg \vdash \varphi$› **have** $(\Gamma \ \#\vdash \ n \ \varphi) = (n \leq \| \ \Gamma \ \|_\varphi)$
by (*simp add*: *unproving-core-stratified-deduction-lower-bound*)
moreover have $(n \leq \| \ \Gamma \ \|_\varphi) = (\forall \ \Phi \in \mathcal{C} \ \Gamma \ \varphi. \ n \leq \mathit{length} \ (\Gamma \ominus \Phi))$
proof (*rule iffI*)
assume $n \leq \| \ \Gamma \ \|_\varphi$
{
fix Φ
assume $\Phi \in \mathcal{C} \ \Gamma \ \varphi$
hence $n \leq \mathit{length} \ (\Gamma \ominus \Phi)$
using ‹$n \leq \| \ \Gamma \ \|_\varphi$› *complement-core-size-intro* **by** *auto*
}
thus $\forall \Phi \in \mathcal{C} \ \Gamma \ \varphi. \ n \leq \mathit{length} \ (\Gamma \ominus \Phi)$ **by** *blast*
next
assume $\forall \Phi \in \mathcal{C} \ \Gamma \ \varphi. \ n \leq \mathit{length} \ (\Gamma \ominus \Phi)$
with ‹$\neg \vdash \varphi$› **obtain** Φ **where**
$\Phi \in \mathcal{C} \ \Gamma \ \varphi$
$n \leq \mathit{length} \ (\Gamma \ominus \Phi)$
using *unproving-core-existence*
by *blast*
thus $n \leq \| \ \Gamma \ \|_\varphi$
by (*simp add*: *complement-core-size-intro*)
qed
ultimately show *?thesis* **by** *metis*
qed

lemma (**in** *probability-logic*) *list-probability-upper-bound*:
$(\sum \gamma \leftarrow \Gamma. \ \mathit{Pr} \ \gamma) \leq \mathit{real} \ (\mathit{length} \ \Gamma)$
proof (*induct* Γ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\gamma \ \Gamma$)
moreover have $\mathit{Pr} \ \gamma \leq \mathit{1}$ **using** *unity-upper-bound* **by** *blast*

```
  ultimately have Pr γ + (∑γ←Γ. Pr γ) ≤ 1 + real (length Γ) by linarith
  then show ?case by simp
qed
```

2.7 Completeness

```
theorem (in classical-logic) binary-limited-stratified-deduction-completeness:
  (∀ Pr ∈ dirac-measures. real n ∗ Pr φ ≤ (∑γ←Γ. Pr γ)) = ∼ Γ #⊢ n (∼ φ)
proof −
  {
    fix Pr :: 'a ⇒ real
    assume Pr ∈ dirac-measures
    from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding dirac-measures-def
      by auto
    assume ∼ Γ #⊢ n (∼ φ)
    moreover have replicate n (∼ φ) = ∼ (replicate n φ)
      by (induct n, auto)
    ultimately have ∼ Γ $⊢ ∼ (replicate n φ)
      using stratified-segmented-deduction-replicate by metis
    hence (∑φ←(replicate n φ). Pr φ) ≤ (∑γ←Γ. Pr γ)
      using segmented-deduction-summation-introduction
      by blast
    moreover have (∑φ←(replicate n φ). Pr φ) = real n ∗ Pr φ
      by (induct n, simp, simp add: semiring-normalization-rules(3))
    ultimately have real n ∗ Pr φ ≤ (∑γ←Γ. Pr γ)
      by simp
  }
  moreover
  {
    assume ¬ ∼ Γ #⊢ n (∼ φ)
    have ∃ Pr ∈ dirac-measures. real n ∗ Pr φ > (∑γ←Γ. Pr γ)
    proof −
      have ∃Φ. Φ ∈ C (∼ Γ) (∼ φ)
        using
          ‹¬ ∼ Γ #⊢ n (∼ φ)›
          unproving-core-existence
          stratified-deduction-tautology-weaken
        by blast
      from this obtain Φ where Φ:
        (∼ Φ) ∈ C (∼ Γ) (∼ φ)
        mset Φ ⊆# mset Γ
        unfolding map-negation-def
        by (metis
              (mono-tags, lifting)
              unproving-core-def
              mem-Collect-eq
              mset-sub-map-list-exists)
      hence ¬ ⊢ φ → ⨆ Φ
```

using
biconditional-weaken
list-deduction-def
map-negation-list-implication
set-deduction-base-theory
unproving-core-def
by *blast*
from *this* **obtain** Ω **where** Ω: *MCS* Ω $\varphi \in \Omega$ $\bigsqcup$ $\Phi \notin \Omega$
by (*meson*
insert-subset
formula-consistent-def
formula-maximal-consistency
formula-maximally-consistent-extension
formula-maximally-consistent-set-def-def
set-deduction-base-theory
set-deduction-reflection
set-deduction-theorem)
let *?Pr* $= \lambda\ \chi.$ *if* $\chi{\in}\Omega$ *then* (*1* :: *real*) *else 0*
from Ω **have** *?Pr* $\in$ *dirac-measures*
using *MCS-dirac-measure* **by** *blast*
moreover
from *this* **interpret** *probability-logic* $(\lambda\ \varphi. \vdash \varphi)$ $(\rightarrow)$ $\bot$ *?Pr*
unfolding *dirac-measures-def*
by *auto*
have $\forall\ \varphi \in$ *set* Φ. *?Pr* φ = *0*
using $\Phi(1)$ $\Omega(1)$ $\Omega(3)$ *arbitrary-disjunction-exclusion-MCS* **by** *auto*
with $\Phi(2)$ **have** $(\sum\gamma{\leftarrow}\Gamma.$ *?Pr* $\gamma) = (\sum\gamma{\leftarrow}(\Gamma \ominus \Phi).$ *?Pr* $\gamma)$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
then show *?case*
proof –
obtain ω :: *'a* **where**
ω: $\neg$ *mset* $\Phi \subseteq\#$ *mset* Γ
$\vee\ \omega \in$ *set* $\Phi \wedge \omega \in \Omega$
$\vee\ (\sum\gamma{\leftarrow}\Gamma.$ *?Pr* $\gamma) = (\sum\gamma{\leftarrow}\Gamma \ominus \Phi.$ *?Pr* $\gamma)$
using *Cons.hyps* **by** *fastforce*
have *A*:
$\forall (f$ :: *'a* $\Rightarrow$ *real*) (Γ ::*'a list*) Φ.
$\neg$ *mset* $\Phi \subseteq\#$ *mset* Γ
$\vee$ *sum-list* $((\sum\varphi{\leftarrow}\Phi.\ f\ \varphi)$ # *map f* $(\Gamma \ominus \Phi)) = (\sum\gamma{\leftarrow}\Gamma.\ f\ \gamma)$
using *listSubstract-multisubset-list-summation* **by** *auto*
have *B*: $\forall$ *rs*. *sum-list* ((*0*::*real*) # *rs*) = *sum-list rs*
by *auto*
have *C*: $\forall$ *r rs*. (*0*::*real*) = *r* $\vee$ *sum-list* (*r* # *rs*) $\neq$ *sum-list rs*
by *simp*
have *D*: $\forall f$. *sum-list* (*sum-list* (*map f* (φ # Φ)) # *map f* ($\Gamma \ominus$ (φ # Φ)))

$= (\textit{sum-list}\ (\textit{map}\ f\ \Gamma)::\textit{real})$
using *A Cons.prems(1)* **by** *blast*
have *E*: $\textit{mset}\ \Phi \subseteq\#\ \textit{mset}\ \Gamma$
using *Cons.prems(1) subset-mset.dual-order.trans* **by** *force*
then have *F*: $\forall f.\ (0::\textit{real}) = \textit{sum-list}\ (\textit{map}\ f\ \Phi)$
$\vee\ \textit{sum-list}\ (\textit{map}\ f\ \Gamma) \neq \textit{sum-list}\ (\textit{map}\ f\ (\Gamma \ominus \Phi))$
using *C A* **by** (*metis* (*no-types*))
then have *G*: $(\sum\varphi'\leftarrow(\varphi\ \#\ \Phi).\ ?Pr\ \varphi') = 0 \vee \omega \in \Omega$
using *E* ω *Cons.prems(2)* **by** *auto*
have *H*: $\forall\Gamma\ r::\textit{real}.\ r = (\sum\gamma\leftarrow\Gamma.\ ?Pr\ \gamma)$
$\vee\ \omega \in \textit{set}\ \Phi$
$\vee\ r \neq (\sum\gamma\leftarrow(\varphi\ \#\ \Gamma).\ ?Pr\ \gamma)$
using *Cons.prems(2)* **by** *auto*
have $(1::\textit{real}) \neq 0$ **by** *linarith*
moreover
{ assume $\omega \notin \textit{set}\ \Phi$
then have $\omega \notin \Omega \vee (\sum\gamma\leftarrow\Gamma.\ ?Pr\ \gamma) = (\sum\gamma\leftarrow\Gamma \ominus (\varphi\ \#\ \Phi).\ ?Pr\ \gamma)$
using *H F E D B* ω **by** (*metis* (*no-types*) *sum-list.Cons*) **}**
ultimately have *?thesis*
using *G D B* **by** (*metis Cons.prems(2) list.set-intros(2)*)
then show *?thesis*
by *linarith*
qed
qed
hence $(\sum\gamma\leftarrow\Gamma.\ ?Pr\ \gamma) \leq \textit{real}\ (\textit{length}\ (\Gamma \ominus \Phi))$
using *list-probability-upper-bound*
by *auto*
moreover
have $\textit{length}\ (\sim\Gamma \ominus \sim\Phi) < n$
by (*metis not-le* $\Phi(1)$ $\langle\neg\ (\sim\Gamma)\ \#\!\vdash\ n\ (\sim\varphi)\rangle$
unproving-core-max-stratified-deduction
unproving-list-subtract-length-equiv)
hence $\textit{real}\ (\textit{length}\ (\sim\Gamma \ominus \sim\Phi)) < \textit{real}\ n$
by *simp*
with $\Omega(2)$ **have** $\textit{real}\ (\textit{length}\ (\sim\Gamma \ominus \sim\Phi)) < \textit{real}\ n * ?Pr\ \varphi$
by *simp*
moreover
have $(\sim(\Gamma \ominus \Phi)) \rightleftharpoons (\sim\Gamma \ominus \sim\Phi)$
unfolding *map-negation-def*
by (*metis* $\Phi(2)$ *map-list-subtract-mset-equivalence mset-eq-perm*)
with *perm-length* **have** $\textit{length}\ (\Gamma \ominus \Phi) = \textit{length}\ (\sim\Gamma \ominus \sim\Phi)$
by *fastforce*
hence $\textit{real}\ (\textit{length}\ (\Gamma \ominus \Phi)) = \textit{real}\ (\textit{length}\ (\sim\Gamma \ominus \sim\Phi))$
by *simp*
ultimately show *?thesis*
by *force*
qed
}
ultimately show *?thesis* **by** *fastforce*

qed

lemma (**in** *classical-logic*) *binary-segmented-deduction-completeness*:
  $(\forall\ Pr \in \textit{dirac-measures}.\ (\sum \varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum \gamma\leftarrow\Gamma.\ Pr\ \gamma)) = \sim \Gamma\ \$\vdash\ \sim \Phi$
proof −
  {
    fix $Pr :: 'a \Rightarrow \textit{real}$
    assume $Pr \in \textit{dirac-measures}$
    from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
      unfolding *dirac-measures-def*
      by *auto*
    assume $\sim \Gamma\ \$\vdash\ \sim \Phi$
    hence $(\sum \varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum \gamma\leftarrow\Gamma.\ Pr\ \gamma)$
      using *segmented-deduction-summation-introduction*
      by *blast*
  }
  moreover
  {
    assume $\neg \sim \Gamma\ \$\vdash\ \sim \Phi$
    have $\exists\ Pr \in \textit{dirac-measures}.\ (\sum \varphi\leftarrow\Phi.\ Pr\ \varphi) > (\sum \gamma\leftarrow\Gamma.\ Pr\ \gamma)$
    proof −
      from $\langle\neg \sim \Gamma\ \$\vdash\ \sim \Phi\rangle$ **have** $\neg \sim (\sim \Phi)\ @ \sim \Gamma\ \#\vdash\ (\textit{length}\ (\sim \Phi))\ \bot$
        using *segmented-stratified-falsum-equiv* **by** *blast*
      moreover
      have $\sim (\sim \Phi)\ @ \sim \Gamma\ \#\vdash\ (\textit{length}\ (\sim \Phi))\ \bot = \sim (\sim \Phi)\ @ \sim \Gamma\ \#\vdash\ (\textit{length}$
$\Phi)\ \bot$
        by (*induct* Φ, *auto*)
      moreover have $\vdash \sim \top \rightarrow \bot$
        by (*simp add*: *negation-def*)
      ultimately have $\neg \sim (\sim \Phi\ @\ \Gamma)\ \#\vdash\ (\textit{length}\ \Phi)\ (\sim \top)$
        using *stratified-deduction-implication* **by** *fastforce*
      from *this* **obtain** *Pr* **where** *Pr*:
        $Pr \in \textit{dirac-measures}$
        $\textit{real}\ (\textit{length}\ \Phi) * Pr\ \top > (\sum \gamma\leftarrow (\sim \Phi\ @\ \Gamma).\ Pr\ \gamma)$
        using *binary-limited-stratified-deduction-completeness*
        by *fastforce*
      from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
        unfolding *dirac-measures-def*
        by *auto*
      from *Pr(2)* **have** $\textit{real}\ (\textit{length}\ \Phi) > (\sum \gamma\leftarrow \sim \Phi.\ Pr\ \gamma) + (\sum \gamma\leftarrow \Gamma.\ Pr\ \gamma)$
        by (*simp add*: *probability-unity*)
      moreover have $(\sum \gamma\leftarrow \sim \Phi.\ Pr\ \gamma) = \textit{real}\ (\textit{length}\ \Phi) - (\sum \gamma\leftarrow \Phi.\ Pr\ \gamma)$
        using *complementation*
        by (*induct* Φ, *auto*)
      ultimately show *?thesis*
        using *Pr(1)* **by** *auto*
    qed
  }
  ultimately show *?thesis* **by** *fastforce*

qed

theorem (**in** *classical-logic*) *segmented-deduction-completeness*:
$(\forall\ Pr \in probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)) = \sim \Gamma\ \$\vdash\ \sim \Phi$
proof −
{
fix $Pr :: 'a \Rightarrow real$
assume $Pr \in probabilities$
from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash\ \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *probabilities-def*
by *auto*
assume $\sim \Gamma\ \$\vdash\ \sim \Phi$
hence $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *segmented-deduction-summation-introduction*
by *blast*
}
thus *?thesis*
using *dirac-measures-subset binary-segmented-deduction-completeness*
by *fastforce*
qed

theorem (**in** *classical-logic*) *weakly-additive-completeness-collapse*:
$(\forall\ Pr \in probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in dirac\text{-}measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
by (*simp add*: *binary-segmented-deduction-completeness*
segmented-deduction-completeness)

lemma (**in** *classical-logic*) *stronger-theory-double-negation-right*:
$\Phi \preceq\ \sim (\sim \Phi)$
by (*induct* Φ, *simp*, *simp add*: *double-negation negation-def stronger-theory-left-right-cons*)

lemma (**in** *classical-logic*) *stronger-theory-double-negation-left*:
$\sim (\sim \Phi) \preceq \Phi$
by (*induct* Φ,
simp,
simp add: *double-negation-converse negation-def stronger-theory-left-right-cons*)

lemma (**in** *classical-logic*) *segmented-left-commute*:
$(\Phi\ @\ \Psi)\ \$\vdash\ \Xi = (\Psi\ @\ \Phi)\ \$\vdash\ \Xi$
proof −
have $(\Phi\ @\ \Psi) \preceq (\Psi\ @\ \Phi)\ (\Psi\ @\ \Phi) \preceq (\Phi\ @\ \Psi)$
using *stronger-theory-reflexive stronger-theory-right-permutation perm-append-swap*
by *blast+*
thus *?thesis*
using *segmented-stronger-theory-left-monotonic*
by *blast*
qed

lemma (**in** *classical-logic*) *stratified-deduction-completeness*:

(∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)) = (∼ Γ @ Φ) #⊢ (*length* Φ) ⊥
proof −
have (∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ))
= ∼ (∼ Φ) @ ∼ Γ #⊢ (*length* (∼ Φ)) ⊥
using *binary-segmented-deduction-completeness segmented-stratified-falsum-equiv* **by** *blast*
also have ... = ∼ (∼ Φ) @ ∼ Γ #⊢ (*length* Φ) ⊥ **by** (*induct* Φ, *auto*)
also have ... = ∼ Γ @ ∼ (∼ Φ) #⊢ (*length* Φ) ⊥
by (*simp add*: *segmented-left-commute stratified-segmented-deduction-replicate*)
also have ... = ∼ Γ @ Φ #⊢ (*length* Φ) ⊥
by (*meson segmented-cancel*
segmented-stronger-theory-intro
segmented-transitive
stratified-segmented-deduction-replicate
stronger-theory-double-negation-left
stronger-theory-double-negation-right)
finally show *?thesis* **by** *blast*
qed

lemma (**in** *classical-logic*) *complement-core-completeness*:
(∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)) = (*length* Φ ≤ ‖ ∼ Γ @ Φ $\|_\perp$)
proof (*cases* ⊢ ⊥)
case *True*
hence $\mathcal{C}$ (∼ Γ @ Φ) ⊥ = {}
using *unproving-core-existence* **by** *auto*
hence *length* (∼ Γ @ Φ) = ‖ ∼ Γ @ Φ $\|_\perp$
unfolding *complement-core-size-def core-size-def* **by** *presburger*
then show *?thesis*
using *True stratified-deduction-completeness stratified-deduction-tautology-weaken*
by *auto*
next
case *False*
then show *?thesis*
using *stratified-deduction-completeness unproving-core-stratified-deduction-lower-bound*
by *blast*
qed

lemma (**in** *classical-logic*) *binary-core-partial-completeness*:
(∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)) = ((| ∼ Γ @ Φ $|_\perp$) ≤ *length* Γ)
proof −
{
fix *Pr* :: *'a* ⇒ *real*
obtain ϱ :: *'a list* ⇒ *'a list* ⇒ *'a* ⇒ *real* **where**
(∀ Φ Γ. ϱ Φ Γ ∈ *dirac-measures* ∧ ¬ ($\sum\varphi\leftarrow\Phi$. ($\varrho$ Φ Γ) φ) ≤ ($\sum\gamma\leftarrow\Gamma$. ($\varrho$ Φ Γ) γ)
∨ *length* Φ ≤ ‖ ∼ Γ @ Φ $\|_\perp$)

$\wedge$ ($\forall\,\Phi\ \Gamma$. *length* $\Phi \leq (\|\sim \Gamma\ @\ \Phi\ \|_\bot)$
$\longrightarrow (\forall\, Pr \in$ *dirac-measures*. $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)))$
using *complement-core-completeness* **by** *moura*
moreover have $\forall\,\Gamma\ \varphi\ n$. *length* $\Gamma - n \leq (\|\ \Gamma\ \|_\varphi) \vee (|\ \Gamma\ |_\varphi) - n \neq 0$
by (*metis add-diff-cancel-right'*
cancel-ab-semigroup-add-class.diff-right-commute
diff-is-0-eq length-core-decomposition)
moreover have $\forall\ \Gamma\ \Phi\ n$. *length* $(\Gamma\ @\ \Phi) - n \leq$ *length* $\Gamma \vee$ *length* $\Phi - n \neq 0$
by *force*
ultimately have
$(Pr \in$ *dirac-measures* $\longrightarrow (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
$\wedge\ (|\sim \Gamma\ @\ \Phi\ |_\bot) \leq$ *length* $(\sim \Gamma)$
$\vee\quad \neg\ (|\sim \Gamma\ @\ \Phi\ |_\bot) \leq$ *length* $(\sim \Gamma)$
$\wedge\ (\exists\, Pr.\ Pr \in$ *dirac-measures* $\wedge\ \neg\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
by (*metis* (*no-types*) *add-diff-cancel-left'*
add-diff-cancel-right'
diff-is-0-eq length-append
length-core-decomposition)
}
then show *?thesis* **by** *auto*
qed

lemma (**in** *classical-logic*) *nat-binary-probability*:
$\forall\ Pr \in$ *dirac-measures*. $\exists\, n :: nat.\ real\ n = (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi)$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\varphi\ \Phi$)
{
fix $Pr :: {}'a \Rightarrow real$
assume $Pr \in$ *dirac-measures*
from *Cons this* **obtain** n **where** $real\ n = (\sum \varphi' \leftarrow \Phi.\ Pr\ \varphi')$ **by** *fastforce*
hence $\star$: $(\sum \varphi' \leftarrow \Phi.\ Pr\ \varphi') = real\ n$ **by** *simp*
have $\exists\ n.\ real\ n = (\sum \varphi' \leftarrow (\varphi\ \#\ \Phi).\ Pr\ \varphi')$
proof (*cases* $Pr\ \varphi = 1$)
case *True*
then show *?thesis*
by (*simp add*: $\star$, *metis of-nat-Suc*)
next
case *False*
hence $Pr\ \varphi = 0$ **using** ‹$Pr \in$ *dirac-measures*› *dirac-measures-def* **by** *auto*
then show *?thesis* **using** $\star$
by *simp*
qed
}
thus *?case* **by** *blast*
qed

lemma (**in** *classical-logic*) *dirac-ceiling*:
  $\forall$ *Pr* $\in$ *dirac-measures*.
    $((\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
      $= ((\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
proof −
  {
    fix *Pr*
    assume *Pr* $\in$ *dirac-measures*
    have $((\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
        $= ((\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
    proof (*rule iffI*)
      assume *assm*: $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
      show $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
      proof (*rule ccontr*)
        assume $\neg\ (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
        moreover
        obtain x :: *int*
          and y :: *int*
          and z :: *int*
          where *xyz*: $x = (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi)$
                $y = \lceil c \rceil$
                $z = (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
          using *nat-binary-probability*
          by (*metis* ‹*Pr* $\in$ *dirac-measures*› *of-int-of-nat-eq*)
        ultimately have $x + y - 1 \geq z$ **by** *linarith*
        hence $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c > (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$ **using** *xyz* **by** *linarith*
        thus *False* **using** *assm* **by** *simp*
      qed
    next
      assume $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
      thus $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma)$
        by *linarith*
    qed
  }
  thus *?thesis* **by** *blast*
qed

lemma (**in** *probability-logic*) *probability-replicate-verum*:
  fixes n :: *nat*
  shows $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + n = (\sum \varphi{\leftarrow}(replicate\ n\ \top)\ @\ \Phi.\ Pr\ \varphi)$
  using *probability-unity*
  by (*induct n*, *auto*)

2.7.1 Collapse Theorem For Probability Logic

lemma (**in** *classical-logic*) *dirac-collapse*:
  $(\forall\ Pr \in probabilities.\ (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + c \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
    $= (\forall\ Pr \in dirac\text{-}measures.\ (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
proof

assume ∀ *Pr* ∈ *probabilities*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
hence ∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using *dirac-measures-subset* **by** *fastforce*
thus ∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using *dirac-ceiling* **by** *blast*
next
assume *assm*: ∀ *Pr* ∈ *dirac-measures*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
show ∀ *Pr* ∈ *probabilities*. ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
proof (*cases c* ≥ *0*)
case *True*
from *this* **obtain** *n* :: *nat* **where** *real n* = $\lceil c\rceil$
by (*metis* (*full-types*)
antisym-conv
ceiling-le-zero
ceiling-zero
nat-0-iff
nat-eq-iff2
of-nat-nat)
{
fix *Pr*
assume *Pr* ∈ *dirac-measures*
from *this* **interpret** *probability-logic* (λ φ. ⊢ φ) (→) ⊥ *Pr*
unfolding *dirac-measures-def*
by *auto*
have ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using *assm* ‹*Pr* ∈ *dirac-measures*› **by** *blast*
hence ($\sum\varphi\leftarrow$(*replicate n* ⊤) @ Φ. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using ‹*real n* = $\lceil c\rceil$›
probability-replicate-verum [**where** Φ=Φ **and** *n*=*n*]
by *metis*
}
hence ∀ *Pr* ∈ *dirac-measures*.
($\sum\varphi\leftarrow$(*replicate n* ⊤) @ Φ. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
by *blast*
hence †: ∀ *Pr* ∈ *probabilities*.
($\sum\varphi\leftarrow$(*replicate n* ⊤) @ Φ. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using *weakly-additive-completeness-collapse* **by** *blast*
{
fix *Pr*
assume *Pr* ∈ *probabilities*
from *this* **interpret** *probability-logic* (λ φ. ⊢ φ) (→) ⊥ *Pr*
unfolding *probabilities-def*
by *auto*
have ($\sum\varphi\leftarrow$(*replicate n* ⊤) @ Φ. *Pr* φ) ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using † ‹*Pr* ∈ *probabilities*› **by** *blast*
hence ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* ≤ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using ‹*real n* = $\lceil c\rceil$›
probability-replicate-verum [**where** Φ=Φ **and** *n*=*n*]

```
      by linarith
    }
    then show ?thesis by blast
  next
    case False
    hence ⌈c⌉ ≤ 0 by auto
    from this obtain n :: nat where real n = − ⌈c⌉
      by (metis neg-0-le-iff-le of-nat-nat)
    {
      fix Pr
      assume Pr ∈ dirac-measures
      from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ Pr
        unfolding dirac-measures-def
        by auto
      have (∑φ←Φ. Pr φ) + ⌈c⌉ ≤ (∑γ←Γ. Pr γ)
        using assm ‹Pr ∈ dirac-measures› by blast
      hence (∑φ←Φ. Pr φ) ≤ (∑γ←(replicate n ⊤) @ Γ. Pr γ)
        using ‹real n = − ⌈c⌉›
              probability-replicate-verum [where Φ=Γ and n=n]
        by linarith
    }
    hence ∀ Pr ∈ dirac-measures.
              (∑φ←Φ. Pr φ) ≤ (∑γ←(replicate n ⊤) @ Γ. Pr γ)
      by blast
    hence ‡: ∀ Pr ∈ probabilities.
              (∑φ←Φ. Pr φ) ≤ (∑γ←(replicate n ⊤) @ Γ. Pr γ)
      using weakly-additive-completeness-collapse by blast
    {
      fix Pr
      assume Pr ∈ probabilities
      from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ Pr
        unfolding probabilities-def
        by auto
      have (∑φ←Φ. Pr φ) ≤ (∑γ←(replicate n ⊤) @ Γ. Pr γ)
        using ‡ ‹Pr ∈ probabilities› by blast
      hence (∑φ←Φ. Pr φ) + c ≤ (∑γ←Γ. Pr γ)
        using ‹real n = − ⌈c⌉›
              probability-replicate-verum [where Φ=Γ and n=n]
        by linarith
    }
    then show ?thesis by blast
  qed
qed

lemma (in classical-logic) dirac-strict-floor:
  ∀ Pr ∈ dirac-measures.
      ((∑φ←Φ. Pr φ) + c < (∑γ←Γ. Pr γ))
        = ((∑φ←Φ. Pr φ) + ⌊c⌋ + 1 ≤ (∑γ←Γ. Pr γ))
proof
```

fix Pr :: $'a \Rightarrow real$
let $?Pr' = (\lambda\ \varphi.\ \lfloor\ Pr\ \varphi\ \rfloor)$ :: $'a \Rightarrow int$
assume $Pr \in dirac\text{-}measures$
hence $\forall\ \varphi.\ Pr\ \varphi = ?Pr'\ \varphi$
unfolding *dirac-measures-def*
by (*metis* (*mono-tags*, *lifting*)
mem-Collect-eq
of-int-0
of-int-1
of-int-floor-cancel)
hence A: $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) = (\sum\varphi\leftarrow\Phi.\ ?Pr'\ \varphi)$
by (*induct* Φ, *auto*)
have B: $(\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma) = (\sum\gamma\leftarrow\Gamma.\ ?Pr'\ \gamma)$
using $\langle\forall\ \varphi.\ Pr\ \varphi = ?Pr'\ \varphi\rangle$ **by** (*induct* Γ, *auto*)
have $((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= ((\sum\varphi\leftarrow\Phi.\ ?Pr'\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ ?Pr'\ \gamma))$
unfolding A B **by** *auto*
also have $\ldots = ((\sum\varphi\leftarrow\Phi.\ ?Pr'\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ ?Pr'\ \gamma))$
by *linarith*
finally show $((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)) =$
$((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
using A B **by** *linarith*
qed

lemma (**in** *classical-logic*) *strict-dirac-collapse*:
$(\forall\ Pr \in probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in dirac\text{-}measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
assume $\forall\ Pr \in probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
hence $\forall\ Pr \in dirac\text{-}measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *dirac-measures-subset* **by** *blast*
thus $\forall\ Pr \in dirac\text{-}measures.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
using *dirac-strict-floor* **by** *blast*
next
assume $\forall\ Pr \in dirac\text{-}measures.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
moreover have $\lfloor c\rfloor + 1 = \lceil\ (\lfloor c\rfloor + 1)$ :: $real\rceil$
by *simp*
ultimately have $\star$:
$\forall\ Pr \in probabilities.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
using *dirac-collapse* [*of* Φ $\lfloor c\rfloor + 1$ Γ]
by *auto*
show $\forall\ Pr \in probabilities.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
fix Pr :: $'a \Rightarrow real$
assume $Pr \in probabilities$
hence $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using $\star$ **by** *auto*
thus $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$

by *linarith*
qed
qed

lemma (**in** *classical-logic*) *unproving-core-verum-extract*:
assumes $\neg \vdash \varphi$
shows $(\mid$ *replicate* $n \top @ \Phi \mid_{\varphi}) = n + (\mid \Phi \mid_{\varphi})$
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Φ
obtain Σ **where** $\Sigma \in \mathcal{C}\ (\top\ \#\ \Phi)\ \varphi$
using *assms unproving-core-existence* **by** *fastforce*
hence $\top \in$ *set* Σ
by (*metis* (*no-types*, *lifting*)
list.set-intros(*1*)
list-deduction-modus-ponens
list-deduction-weaken
unproving-core-complement-equiv
unproving-core-def
verum-tautology
mem-Collect-eq)
hence $\neg$ (*remove1* $\top\ \Sigma$:$\vdash \varphi$)
by (*meson* $\langle\Sigma \in \mathcal{C}\ (\top\ \#\ \Phi)\ \varphi\rangle$
list.set-intros(*1*)
axiom-k
list-deduction-modus-ponens
list-deduction-monotonic
list-deduction-weaken
unproving-core-complement-equiv
set-remove1-subset)
moreover
have *mset* $\Sigma \subseteq\#$ *mset* $(\top\ \#\ \Phi)$
using $\langle\Sigma \in \mathcal{C}\ (\top\ \#\ \Phi)\ \varphi\rangle$ *unproving-core-def* **by** *blast*
hence *mset* (*remove1* $\top\ \Sigma$) $\subseteq\#$ *mset* Φ
using *subset-eq-diff-conv* **by** *fastforce*
ultimately have $(\mid \Phi \mid_{\varphi}) \geq$ *length* (*remove1* $\top\ \Sigma$)
by (*metis* (*no-types*, *lifting*)
core-size-intro
list-deduction-weaken
unproving-core-def
unproving-core-existence
mem-Collect-eq)
hence $(\mid \Phi \mid_{\varphi}) + 1 \geq$ *length* Σ
by (*simp add*: $\langle\top \in$ *set* $\Sigma\rangle$ *length-remove1*)
moreover have $(\mid \Phi \mid_{\varphi}) <$ *length* Σ

proof (*rule ccontr*)
assume $\neg$ $(\mid \Phi \mid_\varphi) <$ *length* Σ
hence $(\mid \Phi \mid_\varphi) \geq$ *length* Σ **by** *linarith*
from *this* **obtain** Δ **where** $\Delta \in \mathcal{C}\ \Phi\ \varphi$ *length* $\Delta \geq$ *length* Σ
using *assms core-size-intro unproving-core-existence* **by** *fastforce*
hence $\neg$ $(\top \# \Delta) :\vdash \varphi$
using *list-deduction-modus-ponens*
list-deduction-theorem
list-deduction-weaken
unproving-core-def
verum-tautology
by *blast*
moreover have *mset* $(\top \# \Delta) \subseteq\#$ *mset* $(\top \# \Phi)$
using ‹$\Delta \in \mathcal{C}\ \Phi\ \varphi$› *unproving-core-def* **by** *auto*
ultimately have *length* $\Sigma \geq$ *length* $(\top \# \Delta)$
using ‹$\Sigma \in \mathcal{C}\ (\top \# \Phi)\ \varphi$› *unproving-core-def* **by** *blast*
hence *length* $\Delta \geq$ *length* $(\top \# \Delta)$
using ‹*length* $\Sigma \leq$ *length* Δ› *dual-order.trans* **by** *blast*
thus *False* **by** *simp*
qed
ultimately have $(\mid \top \# \Phi \mid_\varphi) = (1 + \mid \Phi \mid_\varphi)$
by (*metis Suc-eq-plus1 Suc-le-eq* ‹$\Sigma \in \mathcal{C}\ (\top \# \Phi)\ \varphi$› *add.commute le-antisym core-size-intro*)
}
thus *?case* **using** *Suc* **by** *simp*
qed

lemma (**in** *classical-logic*) *unproving-core-neg-verum-elim*:
$(\mid$ *replicate* $n\ (\sim \top)$ @ $\Phi \mid_\varphi) = (\mid \Phi \mid_\varphi)$
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Φ
have $(\mid (\sim \top) \# \Phi \mid_\varphi) = (\mid \Phi \mid_\varphi)$
proof (*cases* $\vdash \varphi$)
case *True*
then show *?thesis*
unfolding *core-size-def unproving-core-def*
by (*simp add*: *list-deduction-weaken*)
next
case *False*
from *this* **obtain** Σ **where** $\Sigma \in \mathcal{C}\ ((\sim \top) \# \Phi)\ \varphi$
using *unproving-core-existence* **by** *fastforce*
have $[(\sim \top)] :\vdash \varphi$
by (*metis modus-ponens*

Peirces-law
pseudo-scotus
list-deduction-theorem
list-deduction-weaken
negation-def
verum-def)
hence $\sim \top \notin$ *set* Σ
by (*meson* ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ›
list.set-intros(*1*)
list-deduction-base-theory
list-deduction-theorem
list-deduction-weaken
unproving-core-complement-equiv)
hence *remove1* ($\sim \top$) $\Sigma = \Sigma$
by (*simp add*: *remove1-idem*)
moreover have *mset* $\Sigma \subseteq$# *mset* (($\sim \top$) # Φ)
using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *unproving-core-def* **by** *blast*
ultimately have *mset* $\Sigma \subseteq$# *mset* Φ
by (*metis add-mset-add-single mset.simps*(*2*) *mset-remove1 subset-eq-diff-conv*)
moreover have $\neg$ (Σ :$\vdash$ φ)
using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *unproving-core-def* **by** *blast*
ultimately have (| Φ $|_\varphi$) $\geq$ *length* Σ
by (*metis* (*no-types*, *lifting*)
core-size-intro
list-deduction-weaken
unproving-core-def
unproving-core-existence
mem-Collect-eq)
hence (| Φ $|_\varphi$) $\geq$ (| ($\sim \top$) # Φ $|_\varphi$)
using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *core-size-intro* **by** *auto*
moreover
have (| Φ $|_\varphi$) $\leq$ (| ($\sim \top$) # Φ $|_\varphi$)
proof –
obtain Δ **where** $\Delta \in \mathcal{C}$ Φ φ
using *False unproving-core-existence* **by** *blast*
hence
$\neg$ Δ :$\vdash$ φ
mset $\Delta \subseteq$# *mset* (($\sim \top$) # Φ)
unfolding *unproving-core-def*
by (*simp*,
metis (*mono-tags*, *lifting*)
Diff-eq-empty-iff-mset
list-subtract.simps(*2*)
list-subtract-mset-homomorphism
unproving-core-def
mem-Collect-eq
mset-zero-iff
remove1.simps(*1*))
hence *length* $\Delta \leq$ *length* Σ

using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *unproving-core-def* **by** *blast*
thus *?thesis*
using ‹$\Delta \in \mathcal{C}\ \Phi\ \varphi$› ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *core-size-intro* **by** *auto*
qed
ultimately show *?thesis*
using *le-antisym* **by** *blast*
qed
}
thus *?case* **using** *Suc* **by** *simp*
qed

2.8 MaxSAT Completeness For Probability Inequality Identities

lemma (**in** *consistent-classical-logic*) *binary-inequality-elim*:
assumes $\forall$ *Pr* $\in$ *dirac-measures*.
($\sum\varphi\leftarrow\Phi$. *Pr* φ) + (c :: *real*) $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
shows (*MaxSAT* ($\boldsymbol{\sim}$ Γ @ Φ) + (c :: *real*) $\leq$ *length* Γ)
proof (*cases* $c \geq 0$)
case *True*
from *this* **obtain** n :: *nat* **where** *real* $n = \lceil c \rceil$
by (*metis ceiling-mono ceiling-zero of-nat-nat*)
{
fix *Pr*
assume *Pr* $\in$ *dirac-measures*
from *this* **interpret** *probability-logic* ($\lambda\ \varphi.\ \vdash \varphi$) ($\rightarrow$) $\perp$ *Pr*
unfolding *dirac-measures-def*
by *auto*
have ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $n \leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
by (*metis assms* ‹*Pr* $\in$ *dirac-measures*› ‹*real* $n = \lceil c \rceil$› *dirac-ceiling*)
hence ($\sum\varphi\leftarrow$(*replicate* n $\top$) @ Φ. *Pr* φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
using *probability-replicate-verum* [**where** Φ=Φ **and** n=n]
by *metis*
}
hence (| $\boldsymbol{\sim}$ Γ @ *replicate* n $\top$ @ Φ $|_\perp$) $\leq$ *length* Γ
using *binary-core-partial-completeness* **by** *blast*
moreover have *mset* ($\boldsymbol{\sim}$ Γ @ *replicate* n $\top$ @ Φ) = *mset* (*replicate* n $\top$ @ $\boldsymbol{\sim}$ Γ @ Φ)
by *simp*
ultimately have (| *replicate* n $\top$ @ $\boldsymbol{\sim}$ Γ @ Φ $|_\perp$) $\leq$ *length* Γ
unfolding *core-size-def unproving-core-def*
by *metis*
hence (| $\boldsymbol{\sim}$ Γ @ Φ $|_\perp$) + $\lceil c \rceil$ $\leq$ *length* Γ
using ‹*real* $n = \lceil c \rceil$› *consistency unproving-core-verum-extract*
by *auto*
then show *?thesis* **by** *linarith*
next
case *False*

hence $\lceil c \rceil \leq \mathit{0}$ **by** *auto*
from *this* **obtain** n :: *nat* **where** *real* $n = - \lceil c \rceil$
  by (*metis neg-0-le-iff-le of-nat-nat*)
{
  fix *Pr*
  assume $Pr \in$ *dirac-measures*
  from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
    unfolding *dirac-measures-def*
    by *auto*
  have $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
    using *assms* ‹$Pr \in$ *dirac-measures*› *dirac-ceiling*
    by *blast*
  hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma) + n$
    using ‹*real* $n = - \lceil c \rceil$› **by** *linarith*
  hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow (replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
    using *probability-replicate-verum* [**where** Φ=Γ **and** n=n]
    by *metis*
}
hence $(|\sim (replicate\ n\ \top\ @\ \Gamma)\ @\ \Phi\ |_{\bot}) \leq length\ (replicate\ n\ \top\ @\ \Gamma)$
  using *binary-core-partial-completeness* [**where** Φ=Φ **and** Γ=*replicate* $n\ \top$ @
Γ]
  by *metis*
hence $(|\sim \Gamma\ @\ \Phi\ |_{\bot}) \leq n + length\ \Gamma$
  by (*simp add*: *unproving-core-neg-verum-elim*)
then show *?thesis* **using** ‹*real* $n = - \lceil c \rceil$› **by** *linarith*
qed

lemma (**in** *classical-logic*) *binary-inequality-intro*:
assumes *MaxSAT* $(\sim \Gamma\ @\ \Phi) + (c :: real) \leq length\ \Gamma$
shows $\forall\ Pr \in$ *dirac-measures*. $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + (c :: real) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
proof (*cases* $\vdash \bot$)
assume $\vdash \bot$
{
  fix *Pr*
  assume $Pr \in$ *dirac-measures*
  from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
    unfolding *dirac-measures-def*
    by *auto*
  have *False*
    using ‹$\vdash \bot$› *consistency* **by** *blast*
}
then show *?thesis* **by** *blast*
next
assume $\neg \vdash \bot$
then show *?thesis*
proof (*cases* $c \geq \mathit{0}$)
  assume $c \geq \mathit{0}$
  from *this* **obtain** n :: *nat* **where** *real* $n = \lceil c \rceil$
    by (*metis ceiling-mono ceiling-zero of-nat-nat*)

hence $n + (\| \sim \Gamma \mathbin{@} \Phi \|_{\bot}) \leq \mathit{length}\ \Gamma$
using *assms* **by** *linarith*
hence $(\| \mathit{replicate}\ n\ \top \mathbin{@} \sim \Gamma \mathbin{@} \Phi \|_{\bot}) \leq \mathit{length}\ \Gamma$
by (*simp add*: $\langle \neg \vdash \bot \rangle$ *unproving-core-verum-extract*)
moreover have $\mathit{mset}\ (\mathit{replicate}\ n\ \top \mathbin{@} \sim \Gamma \mathbin{@} \Phi) = \mathit{mset}\ (\sim \Gamma \mathbin{@} \mathit{replicate}\ n$
$\top \mathbin{@} \Phi)$
by *simp*
ultimately have $(\| \sim \Gamma \mathbin{@} \mathit{replicate}\ n\ \top \mathbin{@} \Phi \|_{\bot}) \leq \mathit{length}\ \Gamma$
unfolding *core-size-def unproving-core-def*
by *metis*
hence $\forall\ \mathit{Pr} \in \mathit{dirac\text{-}measures}.\ (\sum \varphi \leftarrow (\mathit{replicate}\ n\ \top) \mathbin{@} \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.$
$\mathit{Pr}\ \gamma)$
using *binary-core-partial-completeness* **by** *blast*
{
fix *Pr*
assume $\mathit{Pr} \in \mathit{dirac\text{-}measures}$
from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ \mathit{Pr}$
unfolding *dirac-measures-def*
by *auto*
have $(\sum \varphi \leftarrow (\mathit{replicate}\ n\ \top) \mathbin{@} \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ \mathit{Pr}\ \gamma)$
using $\langle \mathit{Pr} \in \mathit{dirac\text{-}measures} \rangle$
$\langle \forall\ \mathit{Pr} \in \mathit{dirac\text{-}measures}.\ (\sum \varphi \leftarrow (\mathit{replicate}\ n\ \top) \mathbin{@} \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.$
$\mathit{Pr}\ \gamma) \rangle$
by *blast*
hence $(\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) + n \leq (\sum \gamma \leftarrow \Gamma.\ \mathit{Pr}\ \gamma)$
by (*simp add*: *probability-replicate-verum*)
hence $(\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) + c \leq (\sum \gamma \leftarrow \Gamma.\ \mathit{Pr}\ \gamma)$
using $\langle \mathit{real}\ n = \mathit{real\text{-}of\text{-}int}\ \lceil c \rceil \rangle$ **by** *linarith*
}
then show *?thesis* **by** *blast*
next
assume $\neg\ (c \geq 0)$
hence $\lceil c \rceil \leq 0$ **by** *auto*
from *this* **obtain** $n :: \mathit{nat}$ **where** $\mathit{real}\ n = -\ \lceil c \rceil$
by (*metis neg-0-le-iff-le of-nat-nat*)
hence $(\| \sim \Gamma \mathbin{@} \Phi \|_{\bot}) \leq n + \mathit{length}\ \Gamma$
using *assms* **by** *linarith*
hence $(\| \sim (\mathit{replicate}\ n\ \top \mathbin{@} \Gamma) \mathbin{@} \Phi \|_{\bot}) \leq \mathit{length}\ (\mathit{replicate}\ n\ \top \mathbin{@} \Gamma)$
by (*simp add*: *unproving-core-neg-verum-elim*)
hence $\forall\ \mathit{Pr} \in \mathit{dirac\text{-}measures}.$
$(\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow (\mathit{replicate}\ n\ \top) \mathbin{@} \Gamma.\ \mathit{Pr}\ \gamma)$
using *binary-core-partial-completeness* **by** *blast*
{
fix *Pr*
assume $\mathit{Pr} \in \mathit{dirac\text{-}measures}$
from *this* **interpret** *probability-logic* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ \mathit{Pr}$
unfolding *dirac-measures-def*
by *auto*
have $(\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow (\mathit{replicate}\ n\ \top) \mathbin{@} \Gamma.\ \mathit{Pr}\ \gamma)$

using ‹*Pr* $\in$ *dirac-measures*›
‹$\forall$ *Pr* $\in$ *dirac-measures*.
$(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow (replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$›
by *blast*
hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
using ‹*real* $n = - \lceil c \rceil$› *probability-replicate-verum* **by** *auto*
hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + c \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
by *linarith*
}
then show *?thesis* **by** *blast*
qed
qed

lemma (**in** *consistent-classical-logic*) *binary-inequality-equiv*:
$(\forall\ Pr \in dirac\text{-}measures.\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + (c :: real) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
$= (MaxSAT\ (\sim \Gamma\ @\ \Phi) + (c :: real) \leq length\ \Gamma)$
using *binary-inequality-elim binary-inequality-intro consistency* **by** *auto*

lemma (**in** *consistent-classical-logic*) *probability-inequality-equiv*:
$(\forall\ Pr \in probabilities.\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + c \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
$= (MaxSAT\ (\sim \Gamma\ @\ \Phi) + (c :: real) \leq length\ \Gamma)$
unfolding *dirac-collapse*
using *binary-inequality-equiv dirac-ceiling* **by** *blast*

end

Chapter 3

Dutch Book Theorem

The first completeness theorem for inequalities for probability logic to be investigated is due to Patrick Suppes.

theory *Dutch-Book*
  imports
    ../../Logic/Classical/Classical-Connectives
    Probability-Logic-Inequality-Completeness
    HOL.Real
begin

3.1 Fixed Odds Markets

record *'p bet-offer* =
  bet :: *'p*
  amount :: *real*

record *'p book* =
  buys :: (*'p bet-offer*) *list*
  sells :: (*'p bet-offer*) *list*

definition *payoff* :: ($'p \Rightarrow bool$) $\Rightarrow$ $'p$ *book* $\Rightarrow$ *real* (π) **where**
  [*simp*]: $\pi\ s\ b \equiv (\sum\ i \leftarrow sells\ b.\ (if\ s\ (bet\ i)\ then\ 1\ else\ 0) - amount\ i)$
  $\quad + (\sum\ i \leftarrow buys\ b.\ amount\ i - (if\ s\ (bet\ i)\ then\ 1\ else\ 0))$

definition *settle-bet* :: ($'p \Rightarrow bool$) $\Rightarrow$ $'p \Rightarrow$ *real* **where**
  settle-bet $s\ \varphi \equiv if\ (s\ \varphi)\ then\ 1\ else\ 0$

lemma *payoff-alt-def1*:
  $\pi\ s\ book = (\sum\ i \leftarrow sells\ book.\ settle\text{-}bet\ s\ (bet\ i) - amount\ i)$
  $\quad + (\sum\ i \leftarrow buys\ book.\ amount\ i - settle\text{-}bet\ s\ (bet\ i))$
  unfolding *settle-bet-def*
  by *simp*

definition *settle* :: ($'p \Rightarrow bool$) $\Rightarrow$ $'p$ *bet-offer list* $\Rightarrow$ *real* **where**
settle s bets $\equiv \sum$ $b \leftarrow bets.$ *settle-bet s* (*bet b*)

definition *total-amount* :: ($'p$ *bet-offer*) *list* $\Rightarrow$ *real* **where**
total-amount offers $\equiv \sum$ $i \leftarrow offers.$ *amount i*

lemma *payoff-alt-def2*:
π *s book* = *settle s* (*sells book*)
$-$ *settle s* (*buys book*)
$+$ *total-amount* (*buys book*)
$-$ *total-amount* (*sells book*)
unfolding *payoff-alt-def1 total-amount-def settle-def*
by (*simp add*: *sum-list-subtractf*)

definition (**in** *classical-logic*) *possibility* :: ($'a \Rightarrow bool$) $\Rightarrow$ *bool* **where**
[*simp*]: *possibility p* $\equiv$
$\neg$ (p $\bot$)
$\wedge$ ($\forall$ φ. $\vdash \varphi \longrightarrow p\ \varphi$)
$\wedge$ ($\forall$ $\varphi\ \psi$. p ($\varphi \rightarrow \psi$) $\longrightarrow p\ \varphi \longrightarrow p\ \psi$)
$\wedge$ ($\forall$ φ . $p\ \varphi \vee p$ ($\varphi \rightarrow \bot$))

definition (**in** *classical-logic*) *possibilities* :: ($'a \Rightarrow bool$) *set* **where**
[*simp*]: *possibilities* = $\{p.$ *possibility p*$\}$

lemma (**in** *classical-logic*) *possibility-negation*:
assumes *possibility p*
shows p ($\varphi \rightarrow \bot$) = ($\neg$ $p\ \varphi$)
proof
assume p ($\varphi \rightarrow \bot$)
show $\neg$ $p\ \varphi$
proof
assume $p\ \varphi$
have $\vdash \varphi \rightarrow (\varphi \rightarrow \bot) \rightarrow \bot$
by (*simp add*: *double-negation-converse*)
hence p (($\varphi \rightarrow \bot$) $\rightarrow \bot$)
using ‹$p\ \varphi$› ‹*possibility p*› **by** *auto*
thus *False* **using** ‹p ($\varphi \rightarrow \bot$)› ‹*possibility p*› **by** *auto*
qed
next
show $\neg$ $p\ \varphi \Longrightarrow p$ ($\varphi \rightarrow \bot$) **using** ‹*possibility p*› **by** *fastforce*
qed

lemma (**in** *classical-logic*) *possibilities-logical-closure*:
assumes *possibility p*
and $\{x.\ p\ x\} \Vdash \varphi$
shows $p\ \varphi$
proof $-$

```
  {
    fix Γ
    assume set Γ ⊆ Collect p
    hence ∀ φ. Γ :⊢ φ ⟶ p φ
    proof (induct Γ)
      case Nil
      have ∀φ. ⊢ φ ⟶ p φ
        using ‹possibility p› by auto
      then show ?case
        using list-deduction-base-theory by blast
    next
      case (Cons γ Γ)
      hence p γ
        by simp
      have ∀ φ. Γ :⊢ γ → φ ⟶ p (γ → φ)
        using Cons.hyps Cons.prems by auto
      then show ?case
        by (meson ‹p γ› ‹possibility p› list-deduction-theorem possibility-def)
    qed
  }
  thus ?thesis
    using ‹Collect p ⊩ φ› set-deduction-def by auto
qed

lemma (in classical-logic) possibilities-are-MCS:
  assumes possibility p
  shows MCS {x. p x}
  using assms
  by (metis
        (mono-tags, lifting)
        formula-consistent-def
        formula-maximally-consistent-set-def-def
        maximally-consistent-set-def
        possibilities-logical-closure
        possibility-def
        mem-Collect-eq)

lemma (in classical-logic) MCSs-are-possibilities:
  assumes MCS s
  shows possibility (λ x. x ∈ s)
proof −
  have ⊥ ∉ s
    using ‹MCS s›
          formula-consistent-def
          formula-maximally-consistent-set-def-def
          maximally-consistent-set-def
          set-deduction-reflection
    by blast
  moreover have ∀ φ. ⊢ φ ⟶ φ ∈ s
```

using ⟨*MCS s*⟩
formula-maximally-consistent-set-def-reflection
maximally-consistent-set-def
set-deduction-weaken
by *blast*
moreover have $\forall\ \varphi\ \psi.\ (\varphi \to \psi) \in s \longrightarrow \varphi \in s \longrightarrow \psi \in s$
using ⟨*MCS s*⟩
formula-maximal-consistency
formula-maximally-consistent-set-def-implication
by *blast*
moreover have $\forall\ \varphi.\ \varphi \in s \vee (\varphi \to \bot) \in s$
using *assms*
formula-maximally-consistent-set-def-implication
maximally-consistent-set-def
by *blast*
ultimately show *?thesis* **by** *simp*
qed

definition (**in** *classical-logic*) *negate-bets* $(\text{-}^{\sim})$ **where**
$bets^{\sim} = [b\ (\!|\ bet := \sim (bet\ b)\ |\!).\ b \leftarrow bets]$

lemma (**in** *classical-logic*) *possibility-payoff*:
assumes *possibility p*
shows $\pi\ p\ (\!|\ buys = buys',\ sells = sells'\ |\!)$
$= settle\ p\ (buys'^{\sim}\ @\ sells') + total\text{-}amount\ buys' - total\text{-}amount\ sells' -$
$length\ buys'$
proof (*induct buys'*)
case *Nil*
then show *?case*
unfolding *payoff-alt-def2*
negate-bets-def
total-amount-def
settle-def
settle-bet-def
by *simp*
next
case (*Cons b buys'*)
have $p\ (\sim (bet\ b)) = (\neg\ (p\ (bet\ b)))$ **using** *assms negation-def* **by** *auto*
moreover have $total\text{-}amount\ ((b\ \#\ buys')\ @\ sells')$
$= amount\ b + total\text{-}amount\ buys' + total\text{-}amount\ sells'$
unfolding *total-amount-def*
by (*induct buys'*, *induct sells'*, *auto*)
ultimately show *?case*
using *Cons*
unfolding *payoff-alt-def2 negate-bets-def settle-def settle-bet-def*
by *simp*
qed

lemma (**in** *consistent-classical-logic*) *minimum-payoff-existence*:

$\exists!\, x.\ (\exists\ p \in possibilities.\ \pi\ p\ bets = x) \wedge (\forall\ q \in possibilities.\ x \leq \pi\ q\ bets)$
proof (*rule ex-ex1I*)
show $\exists\, x.\ (\exists\, p{\in}possibilities.\ \pi\ p\ bets = x) \wedge (\forall\, q{\in}possibilities.\ x \leq \pi\ q\ bets)$
proof (*rule ccontr*)
obtain *buys′ sells′* **where** $bets = (\!|\ buys = buys',\ sells = sells'\ |\!)$
by (*metis book.cases*)
assume $\nexists\, x.\ (\exists\ p \in possibilities.\ \pi\ p\ bets = x) \wedge (\forall\ q \in possibilities.\ x \leq \pi\ q$
$bets)$
hence $\forall\, x.\ (\exists\ p \in possibilities.\ \pi\ p\ bets = x) \longrightarrow (\exists\ q \in possibilities.\ \pi\ q\ bets$
$< x)$
by (*meson le-less-linear*)
hence $\star$: $\forall\, p \in possibilities.\ \exists\ q \in possibilities.\ \pi\ q\ bets < \pi\ p\ bets$
by *blast*
have $\Diamond$: $\forall\ p \in possibilities.\ \exists\ q \in possibilities.$
$settle\ q\ (buys'^{\sim}\ @\ sells') < settle\ p\ (buys'^{\sim}\ @\ sells')$
proof
fix p
assume $p \in possibilities$
from *this* **obtain** q **where** $q \in possibilities$ **and** $\pi\ q\ bets < \pi\ p\ bets$
using $\star$ **by** *blast*
hence
$settle\ q\ (buys'^{\sim}\ @\ sells') + total\text{-}amount\ buys' - total\text{-}amount\ sells' -$
$length\ buys'$
$< settle\ p\ (buys'^{\sim}\ @\ sells') + total\text{-}amount\ buys' - total\text{-}amount\ sells' -$
$length\ buys'$
by (*metis* ‹$\pi\ q\ bets < \pi\ p\ bets$›
‹$bets = (\!|buys = buys',\ sells = sells'|\!)$›
‹$p \in possibilities$›
possibilities-def
possibility-payoff
mem-Collect-eq)
hence $settle\ q\ (buys'^{\sim}\ @\ sells') < settle\ p\ (buys'^{\sim}\ @\ sells')$
by *simp*
thus $\exists\, q{\in}possibilities.\ settle\ q\ (buys'^{\sim}\ @\ sells') < settle\ p\ (buys'^{\sim}\ @\ sells')$
using ‹$q \in possibilities$› **by** *blast*
qed
{
fix $bets :: ('a\ bet\text{-}offer)\ list$
fix $s :: 'a \Rightarrow bool$
have $\exists\ n \in \mathbb{N}.\ settle\ s\ bets = real\ n$
unfolding *settle-def settle-bet-def*
by (*induct bets*, *auto*, *metis Nats-1 Nats-add Suc-eq-plus1-left of-nat-Suc*)
} **note** $\dagger = this$
{
fix $n :: nat$
have $(\exists\ p \in possibilities.\ settle\ p\ (buys'^{\sim}\ @\ sells') \leq n)$
$\longrightarrow (\exists\ q \in possibilities.\ settle\ q\ (buys'^{\sim}\ @\ sells') < 0)$ (**is** - $\longrightarrow$
?consequent)
proof (*induct n*)

```
      case 0
      {
        fix p :: 'a ⇒ bool
        assumep ∈ possibilities and settle p (buys'~ @ sells') ≤ 0
        from this obtain q where
          q ∈ possibilities
          settle q (buys'~ @ sells') < settle p (buys'~ @ sells')
          using ◇ by blast
        hence ?consequent
         by (metis † ‹settle p (buys'~ @ sells') ≤ 0› of-nat-0-eq-iff of-nat-le-0-iff)
      }
      then show ?case by auto
    next
      case (Suc n)
      {
        fix p :: 'a ⇒ bool
        assumep ∈ possibilities and settle p (buys'~ @ sells') ≤ Suc n
        from this obtain q₁ where
          q₁ ∈ possibilities
          settle q₁ (buys'~ @ sells') < Suc n
          by (metis ◇ antisym-conv not-less)
        from this obtain q₂ where
          q₂ ∈ possibilities
          settle q₂ (buys'~ @ sells') < n
          using ◇
       by (metis † add.commute nat-le-real-less nat-less-le of-nat-Suc of-nat-less-iff)
        hence ?consequent
          by (metis † Suc.hyps nat-less-le of-nat-le-iff of-nat-less-iff)
      }
      then show ?case by auto
    qed
  }
  hence ∄ p. p ∈ possibilities
    by (metis † not-less0 of-nat-0 of-nat-less-iff order-refl)
  moreover
  have ¬ {} ⊩ ⊥
    using consistency set-deduction-base-theory by auto
  from this obtain Γ where MCS Γ
    by (meson formula-consistent-def
            formula-maximal-consistency
            formula-maximally-consistent-extension)
  hence (λ γ. γ ∈ Γ) ∈ possibilities
    using MCSs-are-possibilities possibilities-def by blast
  ultimately show False
    by blast
 qed
next
 fix x y
 assume A: (∃ p ∈ possibilities. π p bets = x) ∧ (∀ q ∈ possibilities. x ≤ π q bets)
```

and B: $(\exists\, p \in possibilities.\ \pi\ p\ bets = y) \wedge (\forall\, q \in possibilities.\ y \leq \pi\ q\ bets)$
from *this* **obtain** p_x p_y **where**
$p_x \in possibilities$
$p_y \in possibilities$
$\pi\ p_x\ bets = x$
$\pi\ p_y\ bets = y$
by *blast*
with A B **have** $x \leq y$ $y \leq x$
by *blast+*
thus $x = y$ **by** *linarith*
qed

definition (**in** *consistent-classical-logic*)
minimum-payoff :: $'a\ book \Rightarrow real$ (π_{min}) **where**
$\pi_{min}\ b \equiv THE\ x.\ (\exists\ p \in possibilities.\ \pi\ p\ b = x)$
$\wedge\ (\forall\ q \in possibilities.\ x \leq \pi\ q\ b)$

lemma (**in** *classical-logic*) *possibility-payoff-dual*:
assumes *possibility p*
shows $\pi\ p\ (\!|\ buys = buys',\ sells = sells'\ |\!)$
$= -\ settle\ p\ (sells'^{\sim}\ @\ buys')$
$+\ total\text{-}amount\ buys' + length\ sells' - total\text{-}amount\ sells'$
proof (*induct sells'*)
case *Nil*
then show *?case*
unfolding *payoff-alt-def2*
negate-bets-def
total-amount-def
settle-def
by *simp*
next
case (*Cons sell' sells'*)
have $p\ (\sim (bet\ sell')) = (\neg\ (p\ (bet\ sell')))$
using *assms negation-def* **by** *auto*
moreover have
$total\text{-}amount\ ((sell'\ \#\ sells')\ @\ buys')$
$= amount\ sell' + total\text{-}amount\ sells' + total\text{-}amount\ buys'$
unfolding *total-amount-def*
by (*induct buys'*, *induct sells'*, *auto*)
ultimately show *?case*
using *Cons*
unfolding *payoff-alt-def2 negate-bets-def settle-def settle-bet-def*
by *simp*
qed

lemma *settle-alt-def*:
$settle\ q\ bets = length\ [\varphi \leftarrow [\ bet\ b\ .\ b \leftarrow bets\ ]\ .\ q\ \varphi]$
unfolding *settle-def settle-bet-def*
by (*induct bets*, *simp+*)

3.2 Dutch Book Theorems

3.2.1 MaxSAT Dutch Book

theorem (**in** *consistent-classical-logic*) *dutch-book-maxsat*:
  ($k \leq \pi_{min}$ (| *buys* = *buys'*, *sells* = *sells'* |))
  = (*MaxSAT* [*bet b* . *b* ← *sells'*$^\sim$ @ *buys'*] + (k :: *real*)
    ≤ *total-amount buys'* + *length sells'* − *total-amount sells'*)
 (**is** ($k \leq \pi_{min}$ *?bets*) = (*MaxSAT ?props* + k ≤ *total-amount* - + - − -))
proof
 assume $k \leq \pi_{min}$ *?bets*
 let *?P* = λ *x* . (∃ *p* ∈ *possibilities*. π *p ?bets* = *x*)
          ∧ (∀ *q* ∈ *possibilities*. *x* ≤ π *q ?bets*)
 obtain *p* **where**
   possibility p **and**
   ∀ *q* ∈ *possibilities*. π *p ?bets* ≤ π *q ?bets*
  using ‹$k \leq \pi_{min}$ *?bets*›
    minimum-payoff-existence [*of ?bets*]
  by (*metis possibilities-def mem-Collect-eq*)
 hence *?P* (π *p ?bets*)
  using *possibilities-def* **by** *blast*
 hence π_{min} *?bets* = π *p ?bets*
  unfolding *minimum-payoff-def*
  using *minimum-payoff-existence* [*of ?bets*]
    the1-equality [**where** *P* = *?P* **and** *a* = π *p ?bets*]
  by *blast*

 let *?Φ* = [φ ← *?props*. *p* φ]

 have *mset ?Φ* ⊆# *mset ?props*
  by(*induct ?props*,
   auto,
   simp add: *subset-mset.add-mono*)
 moreover
 have ¬ (*?Φ* :⊢ ⊥)
 proof −
  have *set ?Φ* ⊆ {*x*. *p x*}
   by *auto*
  hence ¬ (*set ?Φ* ⊩ ⊥)
   by (*meson* ‹*possibility p*›
      possibilities-are-MCS [*of p*]
      formula-consistent-def
      formula-maximally-consistent-set-def-def
      maximally-consistent-set-def
      list-deduction-monotonic
      set-deduction-def)
  thus *?thesis*
   using *set-deduction-def* **by** *blast*
 qed
 moreover

```
{
  fix Ψ
  assume mset Ψ ⊆# mset ?props and ¬ Ψ :⊢ ⊥
  from this obtain Ω_Ψ where MCS Ω_Ψ and set Ψ ⊆ Ω_Ψ
    by (meson formula-consistent-def
              formula-maximal-consistency
              formula-maximally-consistent-extension
              list-deduction-monotonic
              set-deduction-def)
  let ?q = λφ . φ ∈ Ω_Ψ
  have possibility ?q
    using ‹MCS Ω_Ψ› MCSs-are-possibilities by blast
  hence π p ?bets ≤ π ?q ?bets
    using ‹∀ q∈possibilities. π p ?bets ≤ π q ?bets›
          possibilities-def
    by blast
  let ?c = total-amount buys' + length sells' − total-amount sells'
  have − settle p (sells'~ @ buys') + ?c ≤ − settle ?q (sells'~ @ buys') + ?c
    using ‹π p ?bets ≤ π ?q ?bets›
          ‹possibility p›
          possibility-payoff-dual [of p buys' sells']
          ‹possibility ?q›
          possibility-payoff-dual [of ?q buys' sells']
    by linarith
  hence settle ?q (sells'~ @ buys') ≤ settle p (sells'~ @ buys')
    by linarith
  let ?Ψ' = [φ ← ?props. ?q φ]
  have length ?Ψ' ≤ length ?Φ
    using ‹settle ?q (sells'~ @ buys') ≤ settle p (sells'~ @ buys')›
    unfolding settle-alt-def
    by simp
  moreover
  have length Ψ ≤ length ?Ψ'
  proof −
    have mset [ψ ← Ψ. ?q ψ] ⊆# mset ?Ψ'
    proof −
      {
        fix props :: 'a list
        have ∀ Ψ. ∀ Ω. mset Ψ ⊆# mset props ⟶
                        mset [ψ ← Ψ. ψ ∈ Ω] ⊆# mset [φ ← props. φ ∈ Ω]
          by (simp add: multiset-filter-mono)
      }
      thus ?thesis
        using ‹mset Ψ ⊆# mset ?props› by blast
    qed
    hence length [ψ ← Ψ. ?q ψ] ≤ length ?Ψ'
     by (metis (no-types, lifting) length-sub-mset mset-eq-length nat-less-le not-le)
    moreover have length Ψ = length [ψ ← Ψ. ?q ψ]
      using ‹set Ψ ⊆ Ω_Ψ›
```

```
        by (induct Ψ, simp+)
      ultimately show ?thesis by linarith
    qed
    ultimately have length Ψ ≤ length ?Φ by linarith
  }
  ultimately have ?Φ ∈ 𝒞 ?props ⊥
    unfolding unproving-core-def
    by blast
  hence MaxSAT ?props = length ?Φ
    using core-size-intro by presburger
  hence MaxSAT ?props = settle p (sells'~ @ buys')
    unfolding settle-alt-def
    by simp
  thus MaxSAT ?props + k ≤ total-amount buys' + length sells' − total-amount
sells'
    using possibility-payoff-dual [of p buys' sells']
          ‹k ≤ π_min ?bets›
          ‹π_min ?bets = π p ?bets›
          ‹possibility p›
    by linarith
next
  let ?c = total-amount buys' + length sells' − total-amount sells'
  assume MaxSAT ?props + k ≤ ?c
  from this obtain Φ where Φ ∈ 𝒞 ?props ⊥ and length Φ + k ≤ ?c
    using consistency core-size-intro unproving-core-existence by fastforce
  hence ¬ Φ :⊢ ⊥
    using unproving-core-def by blast
  from this obtain Ω_Φ where MCS Ω_Φ and set Φ ⊆ Ω_Φ
    by (meson formula-consistent-def
              formula-maximal-consistency
              formula-maximally-consistent-extension
              list-deduction-monotonic
              set-deduction-def)
  let ?p = λφ . φ ∈ Ω_Φ
  have possibility ?p
    using ‹MCS Ω_Φ› MCSs-are-possibilities by blast
  have mset Φ ⊆# mset ?props
    using ‹Φ ∈ 𝒞 ?props ⊥› unproving-core-def by blast
  have mset Φ ⊆# mset [ b ← ?props. ?p b]
    by (metis ‹mset Φ ⊆# mset ?props›
              ‹set Φ ⊆ Ω_Φ›
              filter-True
              mset-filter
              multiset-filter-mono
              subset-code(1))
  have mset Φ = mset [ b ← ?props. ?p b]
  proof (rule ccontr)
    assume mset Φ ≠ mset [ b ← ?props. ?p b]
    hence length Φ < length [ b ← ?props. ?p b]
```

using ‹*mset* $\Phi \subseteq\#$ *mset* [$b \leftarrow$ *?props*. *?p* b]› *length-sub-mset not-less* **by** *blast*

moreover

have $\neg$ [$b \leftarrow$ *?props*. *?p* b] $:\vdash \bot$

by (*metis IntE*

‹*MCS* Ω_Φ›

inter-set-filter

formula-consistent-def

formula-maximally-consistent-set-def-def

maximally-consistent-set-def

set-deduction-def

subsetI)

hence *length* [$b \leftarrow$ *?props*. *?p* b] $\leq$ *length* Φ

by (*metis* (*mono-tags*, *lifting*)

‹$\Phi \in \mathcal{C}$ *?props* $\bot$›

unproving-core-def [*of ?props* $\bot$]

mem-Collect-eq

mset-filter

multiset-filter-subset)

ultimately show *False*

using *not-le* **by** *blast*

qed

hence *length* Φ = *settle ?p* (*sells*$'^{\sim}$ @ *buys*$'$)

unfolding *settle-alt-def*

using *mset-eq-length* **by** *fastforce*

hence $k \leq$ *settle ?p* (*sells*$'^{\sim}$ @ *buys*$'$)

$+$ *total-amount buys*$'$ $+$ *length sells*$'$ $-$ *total-amount sells*$'$

using ‹*length* $\Phi + k \leq$ *?c*› **by** *linarith*

hence $k \leq \pi$ *?p ?bets*

using ‹*possibility ?p*›

possibility-payoff-dual [*of ?p buys*$'$ *sells*$'$]

‹*length* $\Phi + k \leq$ *?c*›

‹*length* Φ = *settle ?p* (*sells*$'^{\sim}$ @ *buys*$'$)›

by *linarith*

have $\forall$ $q \in$ *possibilities*. π *?p ?bets* $\leq \pi$ q *?bets*

proof

fix q

assume $q \in$ *possibilities*

hence $\neg$ [$b \leftarrow$ *?props*. q b] $:\vdash \bot$

unfolding *possibilities-def*

by (*metis filter-set*

possibilities-logical-closure

possibility-def

set-deduction-def

mem-Collect-eq

member-filter

subsetI)

hence *length* [$b \leftarrow$ *?props*. q b] $\leq$ *length* Φ

by (*metis* (*mono-tags*, *lifting*)

‹Φ ∈ 𝒞 *?props* ⊥›
unproving-core-def
mem-Collect-eq
mset-filter
multiset-filter-subset)
hence
− *settle ?p* (*sells*$'^{\sim}$ @ *buys*$'$) + *total-amount buys*$'$ + *length sells*$'$ − *total-amount sells*$'$
≤ − *settle q* (*sells*$'^{\sim}$ @ *buys*$'$) + *total-amount buys*$'$ + *length sells*$'$ − *total-amount sells*$'$
using ‹*length* Φ = *settle ?p* (*sells*$'^{\sim}$ @ *buys*$'$)›
settle-alt-def [*of q sells*$'^{\sim}$ @ *buys*$'$]
by *linarith*
thus π *?p ?bets* ≤ π *q ?bets*
using *possibility-payoff-dual* [*of ?p buys*$'$ *sells*$'$]
possibility-payoff-dual [*of q buys*$'$ *sells*$'$]
‹*possibility ?p*›
‹*q* ∈ *possibilities*›
unfolding *possibilities-def*
by (*metis mem-Collect-eq*)
qed
have π_{min} *?bets* = π *?p ?bets*
unfolding *minimum-payoff-def*
proof
show (∃ *p*∈*possibilities*. π *p ?bets* = π *?p ?bets*) ∧ (∀ *q*∈*possibilities*. π *?p ?bets* ≤ π *q ?bets*)
using ‹∀ *q* ∈ *possibilities*. π *?p ?bets* ≤ π *q ?bets*›
‹*possibility ?p*›
unfolding *possibilities-def*
by *blast*
next
fix *n*
assume ⋆: (∃ *p*∈*possibilities*. π *p ?bets* = *n*) ∧ (∀ *q*∈*possibilities*. *n* ≤ π *q ?bets*)
from *this* **obtain** *p* **where** π *p ?bets* = *n* **and** *possibility p*
using *possibilities-def* **by** *blast*
hence π *p ?bets* ≤ π *?p ?bets*
using ⋆ ‹*possibility ?p*›
unfolding *possibilities-def*
by *blast*
moreover have π *?p ?bets* ≤ π *p ?bets*
using ‹∀ *q* ∈ *possibilities*. π *?p ?bets* ≤ π *q ?bets*›
‹*possibility p*›
unfolding *possibilities-def*
by *blast*
ultimately show *n* = π *?p ?bets* **using** ‹π *p ?bets* = *n*› **by** *linarith*
qed
thus *k* ≤ π_{min} *?bets*
using ‹*k* ≤ π *?p ?bets*›
by *auto*

qed

3.2.2 Probability Dutch Book

lemma (**in** *consistent-classical-logic*) *nonstrict-dutch-book*:
$(k \leq \pi_{min}$ (| *buys* = *buys'*, *sells* = *sells'* |))
= (∀ *Pr* ∈ *probabilities*.
$(\sum b \leftarrow buys'. \ Pr\ (bet\ b)) + total\text{-}amount\ sells' + k$
$\leq (\sum s \leftarrow sells'. \ Pr\ (bet\ s)) + total\text{-}amount\ buys')$
(**is** *?lhs* = -)
proof –
let *?tot-ss* = *total-amount sells'* **and** *?tot-bs* = *total-amount buys'*
have [*bet b* . *b* ← *sells'*$^{\sim}$ @ *buys'*] = ∼ [*bet s*. *s* ← *sells'*] @ [*bet b*. *b* ← *buys'*]
(**is** - = ∼ *?sell-φs* @ *?buy-φs*)
unfolding *negate-bets-def*
by (*induct sells'*, *simp*+)
hence *?lhs* = (*MaxSAT* (∼ *?sell-φs* @ *?buy-φs*) + *k* ≤ *?tot-bs* + *length sells'* − *?tot-ss*)
using *dutch-book-maxsat* [*of k buys' sells'*] **by** *auto*
also have ... = (*MaxSAT* (∼ *?sell-φs* @ *?buy-φs*) + (*?tot-ss* − *?tot-bs* + *k*) ≤ *length sells'*)
by *linarith*
also have ... = (*MaxSAT* (∼ *?sell-φs* @ *?buy-φs*) + (*?tot-ss* − *?tot-bs* + *k*) ≤ *length ?sell-φs*)
by *simp*
finally have *I*: *?lhs* = (∀ *Pr* ∈ *dirac-measures*.
$(\sum \varphi \leftarrow ?buy\text{-}\varphi s. \ Pr\ \varphi) + (?tot\text{-}ss - ?tot\text{-}bs + k) \leq (\sum \gamma \leftarrow ?sell\text{-}\varphi s. \ Pr\ \gamma))$
using *binary-inequality-equiv* [*of ?buy-φs ?tot-ss* − *?tot-bs* + *k ?sell-φs*]
by *blast*
moreover
{
fix *Pr* :: *'a* ⇒ *real*
have $(\sum \varphi \leftarrow ?buy\text{-}\varphi s. \ Pr\ \varphi) = (\sum b \leftarrow buys'. \ Pr\ (bet\ b))$
$(\sum \gamma \leftarrow ?sell\text{-}\varphi s. \ Pr\ \gamma) = (\sum s \leftarrow sells'. \ Pr\ (bet\ s))$
by (*simp add*: *comp-def*)+
hence $((\sum \varphi \leftarrow ?buy\text{-}\varphi s. \ Pr\ \varphi) + (?tot\text{-}ss - ?tot\text{-}bs + k) \leq (\sum \gamma \leftarrow ?sell\text{-}\varphi s. \ Pr\ \gamma))$
$= ((\sum b \leftarrow buys'. \ Pr\ (bet\ b)) + ?tot\text{-}ss + k \leq (\sum s \leftarrow sells'. \ Pr\ (bet\ s)) + ?tot\text{-}bs)$
by *linarith*
}
ultimately show *?thesis*
by (*meson dirac-measures-subset dirac-ceiling dirac-collapse subset-eq*)
qed

lemma (**in** *consistent-classical-logic*) *strict-dutch-book*:
$(k < \pi_{min}$ (| *buys* = *buys'*, *sells* = *sells'* |))
= (∀ *Pr* ∈ *probabilities*.
$(\sum b \leftarrow buys'. \ Pr\ (bet\ b)) + total\text{-}amount\ sells' + k$

< (∑ s←sells′. Pr (bet s)) + total-amount buys′)
(**is** ?lhs = ?rhs)
proof
assume ?lhs
from this **obtain** ε **where** 0 < ε k + ε ≤ π_{min} ⦇buys = buys′, sells = sells′⦈
using less-diff-eq **by** fastforce
hence ∀ Pr ∈ probabilities.
(∑ b←buys′. Pr (bet b)) + total-amount sells′ + (k + ε)
≤ (∑ s←sells′. Pr (bet s)) + total-amount buys′
using nonstrict-dutch-book [of k + ε buys′ sells′] **by** auto
thus ?rhs
using ‹0 < ε› **by** auto
next
have [bet b . b ← sells′~ @ buys′] = ∼ [bet s. s ← sells′] @ [bet b. b ← buys′]
(**is** - = ∼ ?sell-φs @ ?buy-φs)
unfolding negate-bets-def
by (induct sells′, simp+)
{
fix Pr :: ′a ⇒ real
have (∑ b←buys′. Pr (bet b)) = (∑ φ←?buy-φs. Pr φ)
(∑ b←sells′. Pr (bet b)) = (∑ φ←?sell-φs. Pr φ)
by (induct buys′, auto, induct sells′, auto)
}
note ⋆ = this
let ?tot-ss = total-amount sells′ **and** ?tot-bs = total-amount buys′
let ?c = ?tot-ss + k − ?tot-bs
assume ?rhs
have ∀ Pr ∈ probabilities. (∑ b←buys′. Pr (bet b)) + ?c < (∑ s←sells′. Pr (bet s))
using ‹?rhs› **by** fastforce
hence ∀ Pr ∈ probabilities. (∑ φ←?buy-φs. Pr φ) + ?c < (∑ φ←?sell-φs. Pr φ)
using ⋆ **by** auto
hence ∀ Pr ∈ dirac-measures. (∑ φ←?buy-φs. Pr φ) + (⌊?c⌋ + 1) ≤ (∑ φ←?sell-φs. Pr φ)
using strict-dirac-collapse [of ?buy-φs ?c ?sell-φs]
by auto
hence MaxSAT (∼ ?sell-φs @ ?buy-φs) + (⌊?c⌋ + 1) ≤ length ?sell-φs
by (metis floor-add-int floor-mono floor-of-nat binary-inequality-equiv)
hence MaxSAT (∼ ?sell-φs @ ?buy-φs) + ?c < length ?sell-φs
by linarith
from this **obtain** ε :: real **where**
0 < ε
MaxSAT (∼ ?sell-φs @ ?buy-φs) + (k + ε) ≤ ?tot-bs + length sells′ − ?tot-ss
using less-diff-eq **by** fastforce
hence k + ε ≤ π_{min} ⦇buys = buys′, sells = sells′⦈
using ‹[bet b . b ← sells′~ @ buys′] = ∼ ?sell-φs @ ?buy-φs›
dutch-book-maxsat [of k + ε buys′ sells′]
by simp

thus *?lhs*
  using ‹*0* < *ε*› **by** *linarith*
qed

theorem (**in** *consistent-classical-logic*) *dutch-book*:
  $(0 < \pi_{min}\ ⦇\ buys = buys',\ sells = sells'\ ⦈)$
  $= (\forall\ Pr \in probabilities.$
    $(\sum b{\leftarrow}buys'.\ Pr\ (bet\ b)) + total\text{-}amount\ sells'$
   $< (\sum s{\leftarrow}sells'.\ Pr\ (bet\ s)) + total\text{-}amount\ buys')$
  by (*simp add*: *strict-dutch-book*)

end

Bibliography

[1] G. Birkhoff. Rings of sets. *Duke Mathematical Journal*, 3(3):443–454, Sept. 1937.

[2] P. Blackburn, M. de Rijke, and Y. Venema. Section 4.2 Canonical Models. In *Modal Logic*, pages 196–201. 2001.

[3] A. Bobenrieth. The Origins of the Use of the Argument of Trivialization in the Twentieth Century. *History and Philosophy of Logic*, 31(2):111–121, May 2010.

[4] G. Boole. Chapter XVI. On The Theory Of Probabilities. In *An Investigation of the Laws of Thought On Which Are Founded the Mathematical Theories of Logic and Probabilities*, pages 243–252. 1853.

[5] G. Boole. Chapter XVII. General Method In Probabilities. In *An Investigation of the Laws of Thought On Which Are Founded the Mathematical Theories of Logic and Probabilities*, pages 253–275. 1853.

[6] G. Boolos. Don't Eliminate Cut. *Journal of Philosophical Logic*, 13(4):373–378, 1984.

[7] T. S. Broderick and E. Schrödinger. Boolean Algebra and Probability Theory. *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*, 46:103–112, 1940.

[8] B. A. Davey and H. A. Priestley. Chapter 5. Representation: The finite case. In *Introduction to Lattices and Order*, pages 112–124. Cambridge University Press, Cambridge, UK ; New York, NY, 2nd ed edition, 2002.

[9] B. De Finetti. Sui passaggi al limite nel calcolo delle probabilità. *Reale Istituto Lombardo di Scienze e Lettere*, 63:1–12, 1930.

[10] M. Eberl. Buffon's needle problem. *Archive of Formal Proofs*, June 2017.

[11] B. R. Gaines. Fuzzy and probability uncertainty logics. *Information and Control*, 38(2):154–169, Aug. 1978.

[12] G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(1):176–210, Dec. 1935.

[13] G. Gerla. Inferences in probability logic. *Artificial Intelligence*, 70(1-2):33–52, Oct. 1994.

[14] T. Hailperin. Probability Logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, July 1984.

[15] T. Hailperin. *Boole's Logic and Probability: A Critical Exposition from the Standpoint of Contemporary Algebra, Logic and Probability Theory.* Number 85 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 2. ed, rev. and enl edition, 1986.

[16] T. Hailperin. *Sentential Probability Logic: Origins, Development, Current Status, and Technical Applications.* Lehigh University Press, Bethlehem : London ; Cranbury, N.J, 1996.

[17] A. Horn and A. Tarski. Measures in Boolean algebras. *Transactions of the American Mathematical Society*, 64(3):467–467, Mar. 1948.

[18] A. Kolmogoroff. Chapter 1. Die elementare Wahrscheinlichkeitsrechnung. In *Grundbegriffe der Wahrscheinlichkeitsrechnung*, number 2 in Ergebnisse der Mathematik und Ihrer Grenzgebiete, pages 1–12. Springer-Verlag Berlin Heidelberg, first edition, 1933.

[19] C. S. Peirce. On the Algebra of Logic: A Contribution to the Philosophy of Notation. *American Journal of Mathematics*, 7(2):180–196, Jan. 1885.

[20] N. Rescher. *Many-Valued Logic.* McGraw-Hill, New York, first edition, Jan. 1969.

[21] L. J. Savage. Difficulties in the Theory of Personal Probability. *Philosophy of Science*, 34(4):305–310, 1967.

[22] P. Suppes. Probabilistic Inference and the Concept of Total Evidence. In J. Hintikka and P. Suppes, editors, *Studies in Logic and the Foundations of Mathematics*, volume 43 of *Aspects of Inductive Logic*, pages 49–65. Elsevier, Jan. 1966.

[23] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2nd ed edition, 2000.

[24] A. Urquhart. Implicational Formulas in Intuitionistic Logic. *The Journal of Symbolic Logic*, 39(4):661–664, 1974.

[25] D. van Dalen. *Logic and Structure*. Universitext. Springer-Verlag, London, fifth edition, 2013.

[26] B. Weatherson. From Classical to Intuitionistic Probability. *Notre Dame Journal of Formal Logic*, 44(2):111–123, Apr. 2003.