A Formalization Of The Dutch Book Argument

Matthew Doty

July 26, 2020

Contents

1 Implicational Intuitionistic Logic

theory *Implicational-Intuitionistic-Logic*
  imports *Main*
begin

This theory presents the implicational fragment of intuitionistic logic.

1.1 Axiomatization

Minimal logic is given by the following Hilbert-style axiom system:

class *Minimal-Logic* =
  fixes *deduction* :: $'a \Rightarrow bool$ ($\vdash$ - [*60*] *55*)
  fixes *implication* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** $\rightarrow$ *70*)
  assumes *Axiom-1*: $\vdash \varphi \rightarrow \psi \rightarrow \varphi$
  assumes *Axiom-2*: $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
  assumes *Modus-Ponens*: $\vdash \varphi \rightarrow \psi \Longrightarrow \vdash \varphi \Longrightarrow \vdash \psi$

A convenience class to have is *Minimal-Logic* extended with a single named constant, intended to be *falsum*. Other classes extending this class will provide rules for how this constant interacts with other terms.

class *Minimal-Logic-With-Falsum* = *Minimal-Logic* +
  fixes *falsum* :: $'a$ ($\bot$)

1.2 Common Rules

lemma (**in** *Minimal-Logic*) *trivial-implication*: $\vdash \varphi \rightarrow \varphi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *flip-implication*: $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow \psi \rightarrow \varphi \rightarrow \chi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *hypothetical-syllogism*: $\vdash (\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

lemma (**in** *Minimal-Logic*) *flip-hypothetical-syllogism*:
  shows $\vdash (\psi \rightarrow \varphi) \rightarrow (\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi)$
  using *Modus-Ponens flip-implication hypothetical-syllogism* **by** *blast*

lemma (**in** *Minimal-Logic*) *implication-absorption*: $\vdash (\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \psi$
  by (*meson Axiom-1 Axiom-2 Modus-Ponens*)

1.3 Lists of Assumptions

1.3.1 List Implication

Implication given a list of assumptions can be expressed recursively

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

1.3.2 Definition of Deduction

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

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

1.3.3 Interpretation as Minimal Logic

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

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

lemma (**in** *Minimal-Logic*) *list-implication-Axiom-1*: $\vdash \varphi \rightarrow \Gamma :\rightarrow \varphi$
by (*induct* Γ, (*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens*)+)

lemma (**in** *Minimal-Logic*) *list-implication-Axiom-2*: $\vdash \Gamma :\rightarrow (\varphi \rightarrow \psi) \rightarrow \Gamma :\rightarrow \varphi \rightarrow \Gamma :\rightarrow \psi$
by (*induct* Γ, (*simp*, *meson Axiom-1 Axiom-2 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 minimal logic, where a list of assumptions Γ plays the role of a *background theory* of $(:\vdash)$.

context *Minimal-Logic* **begin**
interpretation *List-Deduction-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \Gamma :\vdash \varphi\ (\rightarrow)$
proof qed (*meson list-deduction-def*
Axiom-1
Axiom-2
Modus-Ponens
list-implication-Axiom-1
list-implication-Axiom-2)+
end

The following *weakening* rule can also be derived.

lemma (**in** *Minimal-Logic*) *list-deduction-weaken*: $\vdash \varphi \Longrightarrow \Gamma :\vdash \varphi$
unfolding *list-deduction-def*
using *Modus-Ponens list-implication-Axiom-1*
by *blast*

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

lemma (**in** *Minimal-Logic*) *list-deduction-base-theory* [*simp*]: $[] :\vdash \varphi \equiv \vdash \varphi$

unfolding *list-deduction-def*
by *simp*

lemma (**in** *Minimal-Logic*) *list-deduction-modus-ponens*: $\Gamma :\vdash \varphi \to \psi \Longrightarrow \Gamma :\vdash \varphi \Longrightarrow \Gamma :\vdash \psi$
unfolding *list-deduction-def*
using *Modus-Ponens list-implication-Axiom-2*
by *blast*

1.4 The Deduction Theorem

One result in the meta-theory of minimal 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** *Minimal-Logic*) *list-flip-implication1*: $\vdash (\varphi \# \Gamma) :\to \chi \to \Gamma :\to (\varphi \to \chi)$
by (*induct* Γ,
(*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens flip-implication hypothetical-syllogism*)+)

lemma (**in** *Minimal-Logic*) *list-flip-implication2*: $\vdash \Gamma :\to (\varphi \to \chi) \to (\varphi \# \Gamma) :\to \chi$
by (*induct* Γ,
(*simp*, *meson Axiom-1 Axiom-2 Modus-Ponens flip-implication hypothetical-syllogism*)+)

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

theorem (**in** *Minimal-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*)

1.5 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 minimal logic.

The lemma $\vdash ?\Gamma :\to (?\varphi \to ?\chi) \to (?\varphi \# ?\Gamma) :\to ?\chi$ presents a means of *introducing* assumptions into a list of assumptions when those assumptions have arrived at 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** *Minimal-Logic*) *list-implication-removeAll*:
$\vdash \Gamma :\to \psi \to (\mathit{removeAll}\ \varphi\ \Gamma) :\to (\varphi \to \psi)$
proof −
have $\forall\ \psi.\ \vdash \Gamma :\to \psi \to (\mathit{removeAll}\ \varphi\ \Gamma) :\to (\varphi \to \psi)$
proof(*induct* Γ)
case *Nil*
then show *?case* **by** (*simp*, *meson Axiom-1*)
next
case (*Cons* χ Γ)
assume *inductive-hypothesis*: $\forall\ \psi.\ \vdash \Gamma :\to \psi \to \mathit{removeAll}\ \varphi\ \Gamma :\to (\varphi \to \psi)$
moreover {
assume $\varphi \neq \chi$
with *inductive-hypothesis*
have $\forall\ \psi.\ \vdash (\chi\ \#\ \Gamma) :\to \psi \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
by (*simp*, *meson Modus-Ponens hypothetical-syllogism*)
}
moreover {
fix ψ
assume *φ-equals-χ*: $\varphi = \chi$
moreover with *inductive-hypothesis*
have $\vdash \Gamma :\to (\chi \to \psi) \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \chi \to \psi)$ **by** *simp*
hence $\vdash \Gamma :\to (\chi \to \psi) \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
by (*metis calculation Modus-Ponens implication-absorption list-flip-implication1 list-flip-implication2 list-implication.simps(2)*)
ultimately have $\vdash (\chi\ \#\ \Gamma) :\to \psi \to \mathit{removeAll}\ \varphi\ (\chi\ \#\ \Gamma) :\to (\varphi \to \psi)$
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** *Minimal-Logic*) *list-implication-monotonic*:
$\mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \Longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$
proof −
assume $\mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma$
moreover have $\forall\ \Sigma\ \varphi.\ \mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$
proof(*induct* Γ)
case *Nil*
then show *?case*
by (*metis list-implication.simps(1) list-implication-Axiom-1 set-empty subset-empty*)
next
case (*Cons* ψ Γ)
assume *inductive-hypothesis*: $\forall \Sigma\ \varphi.\ \mathit{set}\ \Sigma \subseteq \mathit{set}\ \Gamma \longrightarrow \vdash \Sigma :\to \varphi \to \Gamma :\to \varphi$

```
{
  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-1 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 Minimal-Logic) list-deduction-monotonic:
  set Σ ⊆ set Γ ⟹ Σ :⊢ φ ⟹ Γ :⊢ φ
  unfolding list-deduction-def
  using Modus-Ponens list-implication-monotonic
  by blast
```

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** *Minimal-Logic*) *alternate-list-deduction-theorem*:
  $(\varphi \ \# \ \Gamma) :\vdash \psi = (\mathit{removeAll}\ \varphi\ \Gamma) :\vdash \varphi \to \psi$
  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)

1.7 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 $[\![\mathit{set}\ ?\Sigma \subseteq \mathit{set}\ ?\Gamma;\ ?\Sigma :\vdash\ ?\varphi]\!] \Longrightarrow\ ?\Gamma :\vdash\ ?\varphi$ among the other rules provided.

lemma (**in** *Minimal-Logic*) *list-deduction-reflection*: $\varphi \in \mathit{set}\ \Gamma \Longrightarrow \Gamma :\vdash \varphi$
  by (*metis list-deduction-def*
    insert-subset
    list.simps(15)
    list-deduction-monotonic
    list-implication.simps(2)
    list-implication-Axiom-1
    order-refl)

1.8 The Cut Rule

Cut is a rule commonly presented in sequent calculi, dating back to Gerhard Gentzen's "Investigations in Logical Deduction" (1934) TODO: Cite me

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) (TODO: Cite me), removing the rule can often lead to very inefficient proof systems.

Here the rule is presented just as a meta theorem.

theorem (**in** *Minimal-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 Minimal-Logic) strong-list-deduction-cut-rule:
  (Φ @ Γ) :⊢ ψ ⟹ ∀ φ ∈ set Φ. Δ :⊢ φ ⟹ Γ @ Δ :⊢ ψ
proof −
  have ∀ ψ. (Φ @ Γ :⊢ ψ ⟶ (∀ φ ∈ set Φ. Δ :⊢ φ) ⟶ Γ @ Δ :⊢ ψ)
    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:
         ∀ ψ. Φ @ Γ :⊢ ψ ⟶ (∀φ∈set Φ. Δ :⊢ φ) ⟶ Γ @ Δ :⊢ ψ
      {
        fix ψ χ
        assume (χ # Φ) @ Γ :⊢ ψ
        hence A: Φ @ Γ :⊢ χ → ψ  using list-deduction-theorem by auto
        assume ∀φ ∈ set (χ # Φ). Δ :⊢ φ
        hence B: ∀ φ ∈ set Φ. Δ :⊢ φ
          and C: Δ :⊢ χ by auto
        from A B have Γ @ Δ :⊢ χ → ψ 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 (Φ @ Γ) :⊢ ψ
  moreover assume ∀ φ ∈ set Φ. Δ :⊢ φ
  ultimately show ?thesis by blast
qed
```

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 (:⊢).

1.10 Definition of Deduction

Just as deduction from a list $(:\vdash)$ can be defined in terms of $(:\rightarrow)$, deduction from a *set* of assumptions can be expressed in terms of $(:\vdash)$.

definition (**in** *Minimal-Logic*) *set-deduction* :: $'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ *bool* (**infix** $\Vdash$ *60*)
where
$\Gamma \Vdash \varphi \equiv \exists\ \Psi.\ set(\Psi) \subseteq \Gamma \wedge \Psi :\vdash \varphi$

1.10.1 Interpretation as Minimal Logic

As in the case of $(:\vdash)$, the relation $(\Vdash)$ may be interpreted as a *proves* predicate for a fixed set of assumptions Γ.

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

lemma (**in** *Minimal-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** *Minimal-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** *Minimal-Logic*) *set-deduction-modus-ponens*: $\Gamma \Vdash \varphi \rightarrow \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$
proof −
assume $\Gamma \Vdash \varphi \rightarrow \psi$
then obtain Φ **where** *A*: *set* $\Phi \subseteq \Gamma$ **and** *B*: $\Phi :\vdash \varphi \rightarrow \psi$
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 *Minimal-Logic* **begin**
interpretation *Set-Deduction-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \Gamma \Vdash \varphi$ $(\rightarrow)$
proof
fix φ ψ
show $\Gamma \Vdash \varphi \rightarrow \psi \rightarrow \varphi$ **by** (*metis Axiom-1 set-deduction-weaken*)
next
fix φ ψ χ
show $\Gamma \Vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$ **by** (*metis Axiom-2 set-deduction-weaken*)

next
  fix φ ψ
  show $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$ **using** *set-deduction-modus-ponens* **by** *metis*
qed
end

1.11 The Deduction Theorem

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

theorem (**in** *Minimal-Logic*) *set-deduction-theorem*: *insert* φ $\Gamma \Vdash \psi = \Gamma \Vdash \varphi \to \psi$
proof –
 have $\Gamma \Vdash \varphi \to \psi \Longrightarrow$ *insert* φ $\Gamma \Vdash \psi$
  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** $\Phi :\vdash \psi$
   using *set-deduction-def* **by** *auto*
  hence *set* (*removeAll* φ Φ) $\subseteq \Gamma$ **by** *auto*
  moreover from ‹$\Phi :\vdash \psi$› **have** *removeAll* φ $\Phi :\vdash \varphi \to \psi$
   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

1.12 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*)

1.13 The Deduction Theorem Revisited

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

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

1.14 Reflection

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

theorem (**in** *Minimal-Logic*) *set-deduction-reflection*: $\varphi \in \Gamma \Longrightarrow \Gamma \Vdash \varphi$
  by (*metis Set.set-insert*
        list-implication.simps(*1*)
        list-implication-Axiom-1
        set-deduction-theorem
        set-deduction-weaken)

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** *Minimal-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** *Minimal-Logic*) *finite-set-deduction-list-deduction*:
  finite $\Phi \Longrightarrow$
   $\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow$
   $\exists\,\Psi.$ *set* $\Psi \subseteq \Delta \wedge (\forall\,\varphi \in \Phi.\ \Psi :\vdash \varphi)$
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 $(\Psi_1 @ \Psi_2) \subseteq \Delta$ **and**
    $\forall\,\varphi \in \Phi.\ \Psi_1 :\vdash \varphi$ **and**
    $\Psi_2 :\vdash \chi$
    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 $[\![\mathit{finite}\ ?\Phi; \forall \varphi \in ?\Phi.\ ?\Delta \Vdash \varphi]\!] \Longrightarrow \exists \Psi.\ \mathit{set}\ \Psi \subseteq\ ?\Delta \wedge (\forall \varphi \in ?\Phi.\ \Psi :\vdash \varphi)$ the strengthened form of the cut rule can be given.

theorem (**in** *Minimal-Logic*) *strong-set-deduction-cut-rule*:
$\Phi \cup \Gamma \Vdash \psi \Longrightarrow \forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi \Longrightarrow \Gamma \cup \Delta \Vdash \psi$
proof −
assume $\Phi \cup \Gamma \Vdash \psi$
then obtain Σ **where**
A: *set* $\Sigma\ \subseteq \Phi \cup \Gamma$ **and**
B: $\Sigma :\vdash \psi$
using *set-deduction-def*
by *auto*+
obtain $\Phi'\ \Gamma'$ **where**
C: *set* Φ' = *set* $\Sigma \cap \Phi$ **and**
D: *set* Γ' = *set* $\Sigma \cap \Gamma$
by (*metis inf-sup-aci*(*1*) *inter-set-filter*)+
then have *set* $(\Phi' \mathbin{@} \Gamma')$ = *set* Σ **using** *A* **by** *auto*
hence *E*: $\Phi' \mathbin{@} \Gamma' :\vdash \psi$ **using** *B list-deduction-monotonic* **by** *blast*
assume $\forall\ \varphi \in \Phi.\ \Delta \Vdash \varphi$
hence $\forall\ \varphi \in$ *set* $\Phi'.\ \Delta \Vdash \varphi$ **using** *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* $(\Gamma' \mathbin{@} \Delta') \subseteq \Gamma \cup \Delta$ **and** $\Gamma' \mathbin{@} \Delta' :\vdash \psi$ **by** *auto*
thus *?thesis* **using** *set-deduction-def* **by** *blast*
qed

1.16 Maximally Consistent Sets For Minimal Logic

definition (**in** *Minimal-Logic*)
Formula-Consistent :: $'a \Rightarrow 'a$ *set* $\Rightarrow$ *bool* (*-−Consistent -* [*100*] *100*)
where [*simp*]: φ*−Consistent* $\Gamma \equiv$ ~ $(\Gamma \Vdash \varphi)$

lemma (**in** *Minimal-Logic*) *Formula-Consistent-Extension*:
assumes φ*−Consistent* Γ
shows (φ*−Consistent insert* ψ Γ) $\vee$ (φ*−Consistent insert* $(\psi \to \varphi)$ Γ)
proof −
{
assume ~ 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

definition (**in** *Minimal-Logic*)
Formula-Maximally-Consistent-Set
:: $'a \Rightarrow {}'a$ *set* $\Rightarrow$ *bool* (*-−MCS* - [*100*] *100*)
where
[*simp*]: φ*−MCS* $\Gamma \equiv (\varphi$*−Consistent* $\Gamma) \wedge (\forall\ \psi.\ \psi \in \Gamma \vee (\psi \to \varphi) \in \Gamma)$

theorem (**in** *Minimal-Logic*) *Formula-Maximally-Consistent-Extension*:
assumes φ*−Consistent* Γ
shows $\exists\ \Omega.\ (\varphi$*−MCS* $\Omega) \wedge \Gamma \subseteq \Omega$
proof −
let *?*Γ*-Extensions* = $\{\Sigma.\ (\varphi$*−Consistent* $\Sigma) \wedge \Gamma \subseteq \Sigma\}$
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 $\mathcal{C}$: $\forall\ \Sigma \in \mathcal{C}.\ \Gamma \subseteq \Sigma\ \forall\ \Sigma \in \mathcal{C}.\ \varphi$*−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 $\Gamma \subseteq$ *?*Ω **by** (*simp add*: $\mathcal{C}$(*1*) *less-eq-Sup*)
moreover have φ*−Consistent ?*Ω
proof −
{
assume ~ φ*−Consistent ?*Ω
then obtain ω **where** ω: *finite* $\omega\ \omega \subseteq$ *?*Ω ~ φ*−Consistent* ω
unfolding *Formula-Consistent-def*
set-deduction-def
by *auto*
from ω(*1*) ω(*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* $\psi\ \omega$)
from *this* **obtain** $\Sigma_1\ \Sigma_2$ **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-𝒞
            unfolding subset.chain-def
            by blast
          hence (insert ψ ω) ⊆ Σ₁ ∨ (insert ψ ω) ⊆ Σ₂
            using Σ₁ Σ₂ by blast
          thus ?case using Σ₁ Σ₂ by blast
        qed
        hence ∃ Σ ∈ 𝒞. (φ−Consistent Σ) ∧ ~ (φ−Consistent Σ)
          using 𝒞(2) ω(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: Ω ∈ ?Γ-Extensions
                        ∀Σ ∈ ?Γ-Extensions. Ω ⊆ Σ ⟶ Σ = Ω by auto+
{
  fix ψ
  have (φ−Consistent insert ψ Ω) ∨ (φ−Consistent insert (ψ → φ) Ω)
       Γ ⊆ insert ψ Ω
       Γ ⊆ insert (ψ → φ) Ω
    using Ω(1) Formula-Consistent-Extension Formula-Consistent-def
    by auto
  hence insert ψ Ω ∈ ?Γ-Extensions
        ∨ insert (ψ → φ) Ω ∈ ?Γ-Extensions
    by blast
  hence ψ ∈ Ω ∨ (ψ → φ) ∈ Ω using Ω(2) by blast
}
thus ?thesis
  using Ω(1)
  unfolding Formula-Maximally-Consistent-Set-def
  by blast
qed

lemma (in Minimal-Logic) Formula-Maximally-Consistent-Set-reflection:
  φ−MCS Γ ⟹ ψ ∈ Γ = Γ ⊩ ψ
proof −
  assume φ−MCS Γ
  {
    assume Γ ⊩ ψ
    moreover from ‹φ−MCS Γ› have ψ ∈ Γ ∨ (ψ → φ) ∈ Γ ~ Γ ⊩ φ
      unfolding Formula-Maximally-Consistent-Set-def Formula-Consistent-def
      by auto
    ultimately have ψ ∈ Γ
```

    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** *Minimal-Logic*)
  Formula-Maximally-Consistent-Set-implication-elimination:
  assumes $\varphi{-}MCS\ \Omega$
  shows $(\psi \rightarrow \chi) \in \Omega \Longrightarrow \psi \in \Omega \Longrightarrow \chi \in \Omega$
  using *assms*
        Formula-Maximally-Consistent-Set-reflection
        set-deduction-modus-ponens
  by *blast*

end

2 Combinatory Logic

theory *Combinators*
  imports *./Implicational-Intuitionistic-Logic*
begin

2.1 Definitions

Combinatory logic, following Curry (TODO: citeme), can be formulated as follows.

datatype *Var* = *Var nat* ($\mathcal{X}$)

datatype *SKComb* =
    Var-Comb Var (⟨-⟩ [*100*] *100*)
  | *S-Comb* (*S*)
  | *K-Comb* (*K*)
  | *Comb-App SKComb SKComb* (**infixl** · *75*)

Note that in addition to *S* and *K* combinators, *SKComb* provides terms for *variables*. This is helpful when studying λ-abstraction embedding.

2.2 Typing

The fragment of the *SKComb* types without *Var-Comb* terms can be given *simple types*:

datatype *'a Simple-Type* =
    Atom 'a (⦃ - ⦄ [*100*] *100*)
  | *To 'a Simple-Type 'a Simple-Type* (**infixr** ⇒ *70*)

inductive
Simply-Typed-SKComb
  $:: SKComb \Rightarrow {}'a\ Simple\text{-}Type \Rightarrow bool$ (**infix** $:: 65$)
where
  $S\text{-}type : S :: (\varphi \Rightarrow \psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \chi$
| $K\text{-}type : K :: \varphi \Rightarrow \psi \Rightarrow \varphi$
| $Application\text{-}type : E_1 :: \varphi \Rightarrow \psi \Longrightarrow E_2 :: \varphi \Longrightarrow E_1 \cdot E_2 :: \psi$

2.3 Lambda Abstraction

Here a simple embedding of the λ-calculus into combinator logic is presented.

The SKI embedding below is originally due to David Turner [1].

Abstraction over combinators where the abstracted variable is not free are simplified using the K combinator.

primrec *free-variables-in-SKComb* $:: SKComb \Rightarrow Var\ set$ ($free_{SK}$)
  where
  $free_{SK}\ (\langle x \rangle) = \{x\}$
| $free_{SK}\ S = \{\}$
| $free_{SK}\ K = \{\}$
| $free_{SK}\ (E_1 \cdot E_2) = (free_{SK}\ E_1) \cup (free_{SK}\ E_2)$

primrec *Turner-Abstraction*
  $:: Var \Rightarrow SKComb \Rightarrow SKComb$ ($\boldsymbol{\lambda}$-. - $[90,90]\ 90$)
  where
  $abst\text{-}S$: $\boldsymbol{\lambda} x.\ S = K \cdot S$
| $abst\text{-}K$: $\boldsymbol{\lambda} x.\ K = K \cdot K$
| $abst\text{-}var$: $\boldsymbol{\lambda} x.\ \langle y \rangle = (if\ x = y\ then\ S \cdot K \cdot K\ else\ K \cdot \langle y \rangle)$
| $abst\text{-}app$:
  $\boldsymbol{\lambda}\ x.\ (E_1 \cdot E_2) = (if\ (x \in free_{SK}\ (E_1 \cdot E_2))$
      $then\ S \cdot (\boldsymbol{\lambda}\ x.\ E_1) \cdot (\boldsymbol{\lambda}\ x.\ E_2)$
      $else\ K \cdot (E_1 \cdot E_2))$

2.4 Common Combinators

This section presents various common combinators. Some combinators are simple enough to express in using S and K, however others are more easily expressed using λ-abstraction. TODO: Cite Haskell Curry's PhD thesis.

A useful lemma is the type of the *identity* combinator, designated by I in the literature.

lemma *Identity-type*: $S \cdot K \cdot K :: \varphi \Rightarrow \varphi$
  using *K-type S-type Application-type* **by** *blast*

Another significant combinator is the combinator, which corresponds to `flip` in Haskell.

lemma *C-type*:
  $\boldsymbol{\lambda}\ \mathcal{X}\ 1.\ \boldsymbol{\lambda}\ \mathcal{X}\ 2.\ \boldsymbol{\lambda}\ \mathcal{X}\ 3.\ (\langle\mathcal{X}\ 1\rangle \cdot \langle\mathcal{X}\ 3\rangle \cdot \langle\mathcal{X}\ 2\rangle)$
    $:: (\varphi \Rightarrow \psi \Rightarrow \chi) \Rightarrow \psi \Rightarrow \varphi \Rightarrow \chi$
  by (*simp*, *meson Identity-type Simply-Typed-SKComb.simps*)

Haskell also has a function `(.)`, which is referred to as the B combinator.

lemma *B-type*: $S \cdot (K \cdot S) \cdot K :: (\psi \Rightarrow \chi) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \chi$
  by (*meson Simply-Typed-SKComb.simps*)

The final combinator given is the B combinator.

lemma *W-type*:
  $\boldsymbol{\lambda}\ \mathcal{X}\ 1.\ \boldsymbol{\lambda}\ \mathcal{X}\ 2.\ (\langle\mathcal{X}\ 1\rangle \cdot \langle\mathcal{X}\ 2\rangle \cdot \langle\mathcal{X}\ 2\rangle) :: (\varphi \Rightarrow \varphi \Rightarrow \chi) \Rightarrow \varphi \Rightarrow \chi$
  by (*simp*, *meson Identity-type Simply-Typed-SKComb.simps*)

2.5 The Curry Howard Correspondence

The (polymorphic) typing for a combinator X is given by the relation $X ::$ φ.

Combinator types form an instance of minimal logic.

interpretation *Combinator-Minimal-Logic*: *Minimal-Logic* $\lambda\ \varphi.\ \exists\ X.\ X :: \varphi\ (\Rightarrow)$
proof qed (*meson Simply-Typed-SKComb.intros*)+

The minimal logic generated by combinator logic is *free* in the following sense: If $X :: \varphi$ holds for some combinator X then φ may be interpreted as logical consequence in any given minimal logic instance.

The fact that any valid type in combinator logic may be interpreted in minimal logic is a form of the *Curry-Howard correspondence*. TODO: Cite

primrec (**in** *Minimal-Logic*) *Simple-Type-interpretation*
    $:: 'a\ \mathit{Simple\text{-}Type} \Rightarrow 'a\ ((\!|\ \text{-}\ |\!))\ [50])$ **where**
  $(\!|\ \mathit{Atom}\ p\ |\!) = p$
  $|\ (\!|\ \varphi \Rightarrow \psi\ |\!) = (\!|\ \varphi\ |\!) \to (\!|\ \psi\ |\!)$

lemma (**in** *Minimal-Logic*) *Curry-Howard-correspondence*:
  $X :: \varphi \Longrightarrow \vdash (\!|\ \varphi\ |\!)$
  by (*induct rule*: *Simply-Typed-SKComb.induct*,
    (*simp add*: *Axiom-1 Axiom-2 Modus-Ponens*)+)

end

3 Kripke Semantics For Intuitionistic Logic

theory *Kripke-Semantics*
  imports *Main*
    ./Combinators
begin

record $('a, 'b)$ *Kripke-Model* =
  $R :: 'a \Rightarrow 'a \Rightarrow bool$
  $V :: 'a \Rightarrow 'b \Rightarrow bool$

primrec
Intuitionistic-Kripke-Semantics
  $:: ('a, 'b)$ *Kripke-Model* $\Rightarrow 'a \Rightarrow 'b$ *Simple-Type* $\Rightarrow bool$
    $(- - \models - [60,60,60]\ 60)$
  where
    $(\mathfrak{M}\ x \models \{\!\!\{\ v\ \}\!\!\}) = (\exists\ w.\ (R\ \mathfrak{M})^{**}\ w\ x \wedge (V\ \mathfrak{M})\ w\ v)$
  $|\ (\mathfrak{M}\ x \models \varphi \Rrightarrow \psi) = (\forall\ y.\ (R\ \mathfrak{M})^{**}\ x\ y \longrightarrow \mathfrak{M}\ y \models \varphi \longrightarrow \mathfrak{M}\ y \models \psi)$

lemma *Kripke-model-monotone*:
  $(R\ \mathfrak{M})^{**}\ x\ y \Longrightarrow \mathfrak{M}\ x \models \varphi \Longrightarrow \mathfrak{M}\ y \models \varphi$
  by (*induct* φ *arbitrary*: y; *simp*)
    (*meson rtranclp-trans*)+

lemma *Kripke-models-impl-flatten*:
  $\mathfrak{M}\ x \models \varphi \Rrightarrow \psi \Rrightarrow \chi =$
    $(\forall\ y.\ (R\ \mathfrak{M})^{**}\ x\ y \longrightarrow \mathfrak{M}\ y \models \varphi \longrightarrow \mathfrak{M}\ y \models \psi \longrightarrow \mathfrak{M}\ y \models \chi)$
  by (*rule iffI* ; *simp*)
    (*meson Kripke-model-monotone rtranclp-trans*)

lemma *Kripke-models-K*:
  $\mathfrak{M}\ x \models \varphi \Rrightarrow \psi \Rrightarrow \varphi$
  by (*meson Kripke-models-impl-flatten*)

lemma *Kripke-models-S*:
  $\mathfrak{M}\ x \models (\varphi \Rrightarrow \psi \Rrightarrow \chi) \Rrightarrow (\varphi \Rrightarrow \psi) \Rrightarrow \varphi \Rrightarrow \chi$
  by (*simp*, *meson rtranclp.rtrancl-refl rtranclp-trans*)

lemma *Kripke-models-Modus-Ponens*:
  $\mathfrak{M}\ x \models \varphi \Rrightarrow \psi \Longrightarrow \mathfrak{M}\ x \models \varphi \Longrightarrow \mathfrak{M}\ x \models \psi$
  by *auto*

theorem *Combinator-Typing-Kripke-Soundness*:
  $X :: \varphi \Longrightarrow \mathfrak{M}\ x \models \varphi$
  by (*induct rule*: *Simply-Typed-SKComb.induct*)
    (*meson Kripke-models-S*, *meson Kripke-models-K*, *auto*)

lemma *Combinator-Typing-Kripke-Soundness-alt*:
  $\exists\ X\ .\ X :: \varphi \Longrightarrow \forall\ \mathfrak{M}\ x.\ \mathfrak{M}\ x \models \varphi$
  by (*meson Combinator-Typing-Kripke-Soundness*)

lemma *Kripke-Cont-Monad*:
  assumes $a \neq b$

and $p \neq q$
and $\mathfrak{M} = (\!| R = (\lambda\ x\ y.\ x = a \wedge y = b),\ V = (\lambda\ x\ y.\ x = b \wedge y = p)\ |\!)$
shows $\neg\ \mathfrak{M}\ a \models ((\{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}) \Rightarrow \{\!\!\{\ q\ \}\!\!\}) \Rightarrow \{\!\!\{\ p\ \}\!\!\}$
proof –
have $\neg\ \mathfrak{M}\ b \models \{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}$
$\neg\ \mathfrak{M}\ a \models \{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}$
unfolding *assms(3)*
using *assms(1) assms(2)* **by** *auto*
hence $\forall\ x.\ (R\ \mathfrak{M})^{**}\ a\ x \longrightarrow \neg\ \mathfrak{M}\ x \models \{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}$
unfolding *assms(3)*
by (*simp*, *metis* (*mono-tags*, *lifting*) *rtranclp.simps*)
hence $\mathfrak{M}\ a \models (\{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}) \Rightarrow \{\!\!\{\ q\ \}\!\!\}$
by *fastforce*
moreover have $\neg\ \mathfrak{M}\ a \models \{\!\!\{\ p\ \}\!\!\}$
unfolding *assms(3)*
using *assms(1) converse-rtranclpE* **by** *fastforce*
ultimately show *?thesis*
by (*meson Kripke-models-Modus-Ponens*)
qed

lemma *no-extract*:
assumes $p \neq q$
shows $\nexists\ X\ .\ X :: ((\{\!\!\{\ p\ \}\!\!\} \Rightarrow \{\!\!\{\ q\ \}\!\!\}) \Rightarrow \{\!\!\{\ q\ \}\!\!\}) \Rightarrow \{\!\!\{\ p\ \}\!\!\}$
using *assms*
by (*metis*
Combinator-Typing-Kripke-Soundness
Kripke-Cont-Monad)

end

4 Classical Propositional Logic

theory *Classical-Propositional-Logic*
imports *../Intuitionistic/Implicational/Implicational-Intuitionistic-Logic*
begin

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

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

4.1 Axiomatization

Classical propositional logic is given by the following Hilbert-style axiom system:

class *Classical-Propositional-Logic* = *Minimal-Logic-With-Falsum* +

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-Propositional-Logic* +
assumes *consistency*: $\neg \vdash \bot$

4.2 Common Rules

lemma (**in** *Classical-Propositional-Logic*) *Ex-Falso-Quodlibet*: $\vdash \bot \to \varphi$
using *Axiom-1 Double-Negation Modus-Ponens hypothetical-syllogism*
by *blast*

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *Double-Negation-converse*:
$\vdash \varphi \to (\varphi \to \bot) \to \bot$
by (*meson Axiom-1 Modus-Ponens flip-implication*)

lemma (**in** *Classical-Propositional-Logic*) *The-Principle-of-Pseudo-Scotus*:
$\vdash (\varphi \to \bot) \to \varphi \to \psi$
using *Ex-Falso-Quodlibet Modus-Ponens hypothetical-syllogism* **by** *blast*

lemma (**in** *Classical-Propositional-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 *The-Principle-of-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 $[\varphi \to \bot, (\varphi \to \psi) \to \varphi]$:⊢ $\bot$
by (*meson list.set-intros*(*1*)
list-deduction-modus-ponens
list-deduction-reflection)
hence $[(\varphi \to \psi) \to \varphi]$:⊢ $(\varphi \to \bot) \to \bot$
using *list-deduction-theorem* **by** *blast*
hence $[(\varphi \to \psi) \to \varphi]$:⊢ φ
using *Double-Negation*
list-deduction-modus-ponens
list-deduction-weaken
by *blast*
thus *?thesis*
using *list-deduction-def*
by *auto*
qed

lemma (**in** *Classical-Propositional-Logic*) *excluded-middle-elimination*:
$\vdash (\varphi \to \psi) \to ((\varphi \to \bot) \to \psi) \to \psi$
proof −
let *?*Γ = $[\psi \to \bot, \varphi \to \psi, (\varphi \to \bot) \to \psi]$
have *?*Γ :⊢ $(\varphi \to \bot) \to \psi$
*?*Γ :⊢ $\psi \to \bot$
by (*simp add*: *list-deduction-reflection*)+
hence *?*Γ :⊢ $(\varphi \to \bot) \to \bot$
by (*meson flip-hypothetical-syllogism*
list-deduction-base-theory
list-deduction-monotonic
list-deduction-theorem
set-subset-Cons)
hence *?*Γ :⊢ φ
using *Double-Negation*
list-deduction-modus-ponens
list-deduction-weaken
by *blast*
hence *?*Γ :⊢ ψ
by (*meson list.set-intros*(*1*)
list-deduction-modus-ponens
list-deduction-reflection
set-subset-Cons subsetCE)
hence $[\varphi \to \psi, (\varphi \to \bot) \to \psi]$:⊢ ψ
using *Peirces-law*
list-deduction-modus-ponens
list-deduction-theorem
list-deduction-weaken
by *blast*
thus *?thesis*
unfolding *list-deduction-def*
by *simp*

qed

4.3 Maximally Consistent Sets For Classical Logic

definition (**in** *Classical-Propositional-Logic*)
  Consistent :: *'a set* ⇒ *bool* **where**
    [*simp*]: *Consistent* Γ ≡ ⊥−*Consistent* Γ

definition (**in** *Classical-Propositional-Logic*)
  Maximally-Consistent-Set :: *'a set* ⇒ *bool* (*MCS*) **where**
    [*simp*]: *MCS* Γ ≡ ⊥−*MCS* Γ

lemma (**in** *Classical-Propositional-Logic*)
  Formula-Maximal-Consistent-Set-negation: φ−*MCS* Γ ⟹ $\varphi \to \bot \in \Gamma$
proof −
  assume φ−*MCS* Γ
  {
    assume $\varphi \to \bot \notin \Gamma$
    hence $(\varphi \to \bot) \to \varphi \in \Gamma$
      using ‹φ−*MCS* Γ›
      unfolding *Formula-Maximally-Consistent-Set-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 ‹φ−*MCS* Γ›
      unfolding *Formula-Maximally-Consistent-Set-def*
              Formula-Consistent-def
      by *simp*
  }
  thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *Formula-Maximal-Consistency*:
  $(\exists \varphi.\ \varphi$−*MCS* Γ) = *MCS* Γ
proof −
  {
    fix φ
    have φ−*MCS* Γ ⟹ *MCS* Γ
    proof −
      assume φ−*MCS* Γ
      have *Consistent* Γ
        using ‹φ−*MCS* Γ›

Ex-Falso-Quodlibet [**where** φ=φ]
set-deduction-weaken [**where** Γ=Γ]
set-deduction-modus-ponens
unfolding *Formula-Maximally-Consistent-Set-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 ‹φ−*MCS* Γ›
unfolding *Formula-Maximally-Consistent-Set-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 ‹φ−*MCS* Γ›
Formula-Maximal-Consistent-Set-negation
set-deduction-reflection
by *simp*
hence $\Gamma \Vdash (\psi \to \bot) \to \bot$
using *calculation*
hypothetical-syllogism
[**where** φ=$\psi \to \bot$ **and** ψ=φ **and** χ=$\bot$]
set-deduction-weaken
[**where** Γ=Γ]
set-deduction-modus-ponens
by *metis*
hence $\Gamma \Vdash \psi$
using *Double-Negation*
[**where** φ=ψ]
set-deduction-weaken
[**where** Γ=Γ]
set-deduction-modus-ponens
by *metis*
thus *?thesis*
using ‹φ−*MCS* Γ›
Formula-Maximally-Consistent-Set-reflection
by *blast*
qed
}
ultimately show *?thesis*
unfolding *Maximally-Consistent-Set-def*
Formula-Maximally-Consistent-Set-def
Formula-Consistent-def

```
              Consistent-def
      by blast
    qed
  }
  thus ?thesis
    unfolding Maximally-Consistent-Set-def
    by metis
qed

lemma (in Classical-Propositional-Logic)
  Formula-Maximally-Consistent-Set-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-Maximal-Consistent-Set-negation
              Formula-Maximally-Consistent-Set-implication-elimination
              Formula-Maximally-Consistent-Set-reflection
              The-Principle-of-Pseudo-Scotus set-deduction-weaken)
      then have ∀χ ψ. insert χ Γ ⊩ ψ ∨ χ → φ ∉ Γ
        by (meson assms
              Axiom-1
              Formula-Maximally-Consistent-Set-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
              Formula-Maximally-Consistent-Set-reflection
              set-deduction-theorem)
    }
    moreover {
      assume χ ∈ Γ
      hence ψ → χ ∈ Γ
        by (metis assms
              calculation
              insert-absorb
              Formula-Maximally-Consistent-Set-reflection
              set-deduction-theorem)
    }
    ultimately have ψ → χ ∈ Γ using hypothesis by blast
  }
```

thus *?thesis*
  using *assms*
    Formula-Maximally-Consistent-Set-implication-elimination
  by *metis*
qed

end

5 Classical Propositional Calculus Soundness And Completeness

theory *Classical-Propositional-Completeness*
  imports *Classical-Propositional-Logic*
begin

5.1 Syntax

datatype *'a Classical-Propositional-Formula* =
  Falsum (⊥)
  | *Proposition 'a* (⟨ - ⟩ [*45*])
  | *Implication 'a Classical-Propositional-Formula*
      'a Classical-Propositional-Formula (**infixr** → *70*)

5.2 Propositional Calculus

named-theorems *Classical-Propositional-Calculus Rules for the Propositional Calculus*

inductive *Classical-Propositional-Calculus* ::
  'a Classical-Propositional-Formula ⇒ *bool* ($\vdash_{prop}$ - [*60*] *55*)
  where
    Axiom-1 [*Classical-Propositional-Calculus*]:
      $\vdash_{prop} \varphi \to \psi \to \varphi$
  | *Axiom-2* [*Classical-Propositional-Calculus*]:
      $\vdash_{prop} (\varphi \to \psi \to \chi) \to (\varphi \to \psi) \to \varphi \to \chi$
  | *Double-Negation* [*Classical-Propositional-Calculus*]:
      $\vdash_{prop} ((\varphi \to \bot) \to \bot) \to \varphi$
  | *Modus-Ponens* [*Classical-Propositional-Calculus*]:
      $\vdash_{prop} \varphi \to \psi \Longrightarrow \vdash_{prop} \varphi \Longrightarrow \vdash_{prop} \psi$

instantiation *Classical-Propositional-Formula* :: (*type*) *Classical-Propositional-Logic*
begin
definition [*simp*]: $\bot = \bot$
definition [*simp*]: $\vdash \varphi = \vdash_{prop} \varphi$
definition [*simp*]: $\varphi \to \psi = \varphi \to \psi$
instance by *standard* (*simp add*: *Classical-Propositional-Calculus*)+
end

5.3 Propositional Semantics

primrec *Classical-Propositional-Semantics* ::
  $'a\ set \Rightarrow 'a\ Classical\text{-}Propositional\text{-}Formula \Rightarrow bool$
  (**infix** $\models_{prop}$ *65*)
  where
    $\mathfrak{M} \models_{prop} Proposition\ p = (p \in \mathfrak{M})$
  $|\ \mathfrak{M} \models_{prop} \varphi \to \psi = (\mathfrak{M} \models_{prop} \varphi \longrightarrow \mathfrak{M} \models_{prop} \psi)$
  $|\ \mathfrak{M} \models_{prop} \bot = False$

theorem *Classical-Propositional-Calculus-Soundness*:
  $\vdash_{prop} \varphi \Longrightarrow \mathfrak{M} \models_{prop} \varphi$
  by (*induct rule*: *Classical-Propositional-Calculus.induct*, *simp+*)

5.4 Propositional Soundness and Completeness

definition *Strong-Classical-Propositional-Deduction* ::
  $'a\ Classical\text{-}Propositional\text{-}Formula\ set \Rightarrow 'a\ Classical\text{-}Propositional\text{-}Formula \Rightarrow bool$
  (**infix** $\Vdash_{prop}$ *65*)
  where
    [*simp*]: $\Gamma \Vdash_{prop} \varphi \equiv \Gamma \Vdash \varphi$

definition *Strong-Classical-Propositional-Models* ::
  $'a\ Classical\text{-}Propositional\text{-}Formula\ set \Rightarrow 'a\ Classical\text{-}Propositional\text{-}Formula \Rightarrow bool$
  (**infix** $\Vvdash_{prop}$ *65*)
  where
    [*simp*]: $\Gamma \models\!\!\models_{prop} \varphi \equiv \forall\ \mathfrak{M}.(\forall\ \gamma \in \Gamma.\ \mathfrak{M} \models_{prop} \gamma) \longrightarrow \mathfrak{M} \models_{prop} \varphi$

definition *Theory-Propositions* ::
  $'a\ Classical\text{-}Propositional\text{-}Formula\ set \Rightarrow 'a\ set$ ($\{\!|$ - $|\!\}$ [*50*])
  where
    [*simp*]: $\{\!|\ \Gamma\ |\!\} = \{p\ .\ \Gamma \Vdash_{prop} Proposition\ p\}$

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-implication
Classical-Propositional-Semantics.simps(2)
implication-Classical-Propositional-Formula-def
set-deduction-modus-ponens
set-deduction-reflection)
qed

theorem *Classical-Propositional-Calculus-Strong-Soundness-And-Completeness*:
$\Gamma \Vdash_{prop} \varphi \equiv \Gamma \models_{prop} \varphi$
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** Γ'(*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 Γ'(*2*) **have** $\mathfrak{M} \models_{prop} \varphi$ **by** *blast*
}
thus $\Gamma \models_{prop} \varphi$
using *Strong-Classical-Propositional-Models-def* **by** *blast*
qed
have *completeness*: $\Gamma \models_{prop} \varphi \Longrightarrow \Gamma \Vdash_{prop} \varphi$
proof (*erule contrapos-pp*)
assume $\sim \Gamma \Vdash_{prop} \varphi$
hence $\exists\ \mathfrak{M}.\ (\forall\ \gamma \in \Gamma.\ \mathfrak{M} \models_{prop} \gamma) \wedge \sim \mathfrak{M} \models_{prop} \varphi$
proof −
from $\langle \sim \Gamma \Vdash_{prop} \varphi \rangle$ **obtain** Ω **where** Ω: $\Gamma \subseteq \Omega$ φ*−MCS* Ω
by (*meson Formula-Consistent-def*
Formula-Maximally-Consistent-Extension
Strong-Classical-Propositional-Deduction-def)
hence $(\varphi \rightarrow \bot) \in \Omega$
using *Formula-Maximal-Consistent-Set-negation* **by** *blast*
hence $\sim \{\!\!\{\ \Omega\ \}\!\!\} \models_{prop} \varphi$
using Ω
Formula-Consistent-def
Formula-Maximal-Consistency

Formula-Maximally-Consistent-Set-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 $\sim\ \Gamma \Vdash_{prop} \varphi$
unfolding *Strong-Classical-Propositional-Models-def*
by *simp*
qed
from *soundness completeness* **show** $\Gamma \Vdash_{prop} \varphi \equiv \Gamma \Vdash_{prop} \varphi$
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-Models-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

primrec (**in** *Classical-Propositional-Logic*) *Classical-Propositional-Formula-embedding*
:: *'a Classical-Propositional-Formula* $\Rightarrow$ *'a* ((| - |) [*50*]) **where**
$(\!|\ \langle\ p\ \rangle\ |\!) = p$
$|\ (\!|\ \varphi \rightarrow \psi\ |\!) = (\!|\ \varphi\ |\!) \rightarrow (\!|\ \psi\ |\!)$
$|\ (\!|\ \bot\ |\!) = \bot$

theorem (**in** *Classical-Propositional-Logic*) *propositional-calculus*:
$\vdash_{prop} \varphi \Longrightarrow \vdash (\!|\ \varphi\ |\!)$
by (*induct rule*: *Classical-Propositional-Calculus.induct*,
(*simp add*: *Axiom-1 Axiom-2 Double-Negation Modus-Ponens*)+)

theorem (**in** *Classical-Propositional-Logic*) *propositional-semantics*:
$\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} \varphi \Longrightarrow \vdash (\!|\ \varphi\ |\!)$
by (*simp add*: *Classical-Propositional-Calculus-Soundness-And-Completeness propositional-calculus*)

end
theory *List-Utilities*
  imports ~~/*src*/*HOL*/*Library*/*Permutation*
begin

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

5.5 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
```

5.6 List Mapping

```
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* <~~> (*remove1* (*f b*) (*map f B*))
B <~~> *b* # *remove1 b B*
by (*metis* ‹*a* # *A* <~~> *map f B*› *perm-remove-perm remove-hd*,
meson b(*1*) *perm-remove*)
hence *A* <~~> (*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′ <~~> (*remove1 b B*)
using *Cons.hyps* **by** *blast*
with *b* **have** *a* # *A* = *map f* (*b* # *B′*)
by *simp*
moreover have *B* <~~> *b* # *B′*
by (*meson B′*(*2*) *b*(*1*) *cons-perm-eq perm.trans perm-remove perm-sym*)
ultimately have ∃ *B′*. *a* # *A* = *map f B′* ∧ *B′* <~~> *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* Φ ⊆# *mset* (*map f* Γ)

shows $\exists\ \Phi'.\ \mathit{mset}\ \Phi' \subseteq\#\ \mathit{mset}\ \Gamma \wedge \Phi = (\mathit{map}\ f\ \Phi')$
proof $-$
have $\forall\ \Phi.\ \mathit{mset}\ \Phi \subseteq\#\ \mathit{mset}\ (\mathit{map}\ f\ \Gamma) \longrightarrow (\exists\ \Phi'.\ \mathit{mset}\ \Phi' \subseteq\#\ \mathit{mset}\ \Gamma \wedge \Phi = (\mathit{map}\ f\ \Phi'))$
proof (*induct* Γ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\gamma\ \Gamma$)
{
fix Φ
assume $\mathit{mset}\ \Phi \subseteq\#\ \mathit{mset}\ (\mathit{map}\ f\ (\gamma\ \#\ \Gamma))$
have $\exists\Phi'.\ \mathit{mset}\ \Phi' \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma) \wedge \Phi = \mathit{map}\ f\ \Phi'$
proof *cases*
assume $f\ \gamma \in \mathit{set}\ \Phi$
hence $f\ \gamma\ \#\ (\mathit{remove1}\ (f\ \gamma)\ \Phi) <\sim\sim> \Phi$
by (*simp add*: *perm-remove perm-sym*)
with ‹$\mathit{mset}\ \Phi \subseteq\#\ \mathit{mset}\ (\mathit{map}\ f\ (\gamma\ \#\ \Gamma))$›
have $\mathit{mset}\ (\mathit{remove1}\ (f\ \gamma)\ \Phi) \subseteq\#\ \mathit{mset}\ (\mathit{map}\ f\ \Gamma)$
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':
$\mathit{mset}\ \Phi' \subseteq\#\ \mathit{mset}\ \Gamma$
$\mathit{remove1}\ (f\ \gamma)\ \Phi = \mathit{map}\ f\ \Phi'$
by *blast*
hence $\mathit{mset}\ (\gamma\ \#\ \Phi') \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma)$
and $f\ \gamma\ \#\ (\mathit{remove1}\ (f\ \gamma)\ \Phi) = \mathit{map}\ f\ (\gamma\ \#\ \Phi')$
by *simp+*
hence $\Phi <\sim\sim> \mathit{map}\ f\ (\gamma\ \#\ \Phi')$
using ‹$f\ \gamma \in \mathit{set}\ \Phi$› *perm-remove* **by** *force*
from *this* **obtain** Φ'' **where** Φ'':
$\Phi = \mathit{map}\ f\ \Phi''$
$\Phi'' <\sim\sim> \gamma\ \#\ \Phi'$
using *perm-map-perm-list-exists*
by *blast*
hence $\mathit{mset}\ \Phi'' \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma)$
by (*metis* ‹$\mathit{mset}\ (\gamma\ \#\ \Phi') \subseteq\#\ \mathit{mset}\ (\gamma\ \#\ \Gamma)$› *mset-eq-perm*)
thus *?thesis* **using** Φ'' **by** *blast*
next
assume $f\ \gamma \notin \mathit{set}\ \Phi$
have $\mathit{mset}\ \Phi - \{\#f\ \gamma\#\} = \mathit{mset}\ \Phi$
by (*metis* (*no-types*) ‹$f\ \gamma \notin \mathit{set}\ \Phi$› *diff-single-trivial set-mset-mset*)
moreover have $\mathit{mset}\ (\mathit{map}\ f\ (\gamma\ \#\ \Gamma)) = \mathit{add\text{-}mset}\ (f\ \gamma)\ (\mathit{image\text{-}mset}\ f\ (\mathit{mset}\ \Gamma))$
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

5.7 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

5.8 Permutations

lemma *perm-count-list*:
assumes Φ <~~> Ψ
shows *count-list* Φ φ = *count-list* Ψ φ
proof –
have $\forall \Psi.\ \Phi$ <~~> $\Psi \longrightarrow$ *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 from ‹χ = φ› ‹χ ∈ set Ψ› have count-list (remove1 χ Ψ) φ +
1 = count-list Ψ φ
          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 from ‹χ ≠ φ› have count-list (remove1 χ Ψ) φ = count-list Ψ 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 φ = ψ
```

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

5.9 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* Φ
by (*induct* Φ, *simp*, *auto*)

lemma *duplicates-alt-def*:
duplicates xs = {*x*. *count-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-list xs x*}
then show *?case*
proof *cases*
assume *x* $\in$ *set xs*
hence *count-list* (*x* # *xs*) *x* $\geq$ *2*

by (*simp*, *induct xs*, *simp*, *simp*, *blast*)
hence {*y*. *2* ≤ *count-list* (*x* # *xs*) *y*} = *insert x* {*y*. *2* ≤ *count-list xs y*}
by (*simp*, *blast*)
thus *?thesis* **using** *inductive-hypothesis* ‹*x* ∈ *set xs*›
by *simp*
next
assume *x* ∉ *set xs*
hence {*y*. *2* ≤ *count-list* (*x* # *xs*) *y*} = {*y*. *2* ≤ *count-list xs y*}
by (*simp*, *auto*)
thus *?thesis* **using** *inductive-hypothesis* ‹*x* ∉ *set xs*›
by *simp*
qed
qed

5.10 List Subtraction

primrec *listSubtract* :: *'a list* ⇒ *'a list* ⇒ *'a list* (**infixl** ⊖ *70*)
where
xs ⊖ [] = *xs*
| *xs* ⊖ (*y* # *ys*) = (*remove1 y* (*xs* ⊖ *ys*))

lemma *listSubtract-mset-homomorphism* [*simp*]:
mset (*A* ⊖ *B*) = *mset A* − *mset B*
by (*induct B*, *simp*, *simp*)

lemma *listSubtract-empty* [*simp*]:
[] ⊖ Φ = []
by (*induct* Φ, *simp*, *simp*)

lemma *listSubtract-remove1-cons-perm*:
Φ ⊖ (φ # Λ) <~~> (*remove1* φ Φ) ⊖ Λ
by (*induct* Λ, *simp*, *simp*, *metis perm-remove-perm remove1-commute*)

lemma *listSubtract-cons*:
assumes φ ∉ *set* Λ
shows (φ # Φ) ⊖ Λ = φ # (Φ ⊖ Λ)
using *assms*
by (*induct* Λ, *simp*, *simp*, *blast*)

lemma *listSubtract-cons-absorb*:
assumes *count-list* Φ φ ≥ *count-list* Λ φ
shows φ # (Φ ⊖ Λ) <~~> (φ # Φ) ⊖ Λ
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** *listSubtract-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 listSubtract-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 listSubtract-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 listSubtract-cons-remove1-perm:
  assumes φ ∈ set Λ
  shows (φ # Φ) ⊖ Λ <~~> Φ ⊖ (remove1 φ Λ)
  using assms listSubtract-remove1-perm by fastforce

lemma listSubtract-removeAll-perm:
  assumes count-list Φ φ ≤ count-list Λ φ
  shows Φ ⊖ Λ <~~> (removeAll φ Φ) ⊖ (removeAll φ Λ)
proof −
```

```
  have ∀ Λ. count-list Φ φ ≤ count-list Λ φ ⟶ Φ ⊖ Λ <~~> (removeAll φ Φ) ⊖ (removeAll φ Λ)
  proof (induct Φ)
    case Nil
    thus ?case by auto
  next
    case (Cons ξ Φ)
    {
      fix Λ
      assume count-list (ξ # Φ) φ ≤ count-list Λ φ
      hence Φ ⊖ Λ <~~> (removeAll φ Φ) ⊖ (removeAll φ Λ)
      by (metis Cons.hyps count-list.simps(2) dual-order.trans le-add-same-cancel1 zero-le-one)
      have (ξ # Φ) ⊖ Λ <~~> (removeAll φ (ξ # Φ)) ⊖ (removeAll φ Λ)
      proof cases
        assume ξ = φ
        hence count-list Φ φ < count-list Λ φ
          using ‹count-list (ξ # Φ) φ ≤ count-list Λ φ›
          by auto
         hence count-list Φ φ ≤ count-list (remove1 φ Λ) φ by (induct Λ, simp, auto)
        hence Φ ⊖ (remove1 φ Λ) <~~> removeAll φ Φ ⊖ removeAll φ (remove1 φ Λ)
          using Cons.hyps by blast
        hence Φ ⊖ (remove1 φ Λ) <~~> removeAll φ Φ ⊖ removeAll φ Λ
          by (simp add: filter-remove1 removeAll-filter-not-eq)
        moreover have φ ∈ set Λ and φ ∈ set (φ # Φ)
          using ‹ξ = φ›
                ‹count-list (ξ # Φ) φ ≤ count-list Λ φ›
                gr-implies-not0
          by fastforce+
        hence (φ # Φ) ⊖ Λ <~~> (remove1 φ (φ # Φ)) ⊖ (remove1 φ Λ)
          by (meson listSubtract-remove1-perm)
        hence (φ # Φ) ⊖ Λ <~~> Φ ⊖ (remove1 φ Λ) by simp
        ultimately show ?thesis using ‹ξ = φ› 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: listSubtract-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) $\ominus$ (*removeAll* φ (*remove1* ξ Λ)) <~~>
(*removeAll* φ Φ) $\ominus$ (*remove1* ξ (*removeAll* φ Λ))
by (*induct* Λ, *simp*, *simp add*: *filter-remove1 removeAll-filter-not-eq*)
hence (*removeAll* φ Φ) $\ominus$ (*removeAll* φ (*remove1* ξ Λ)) <~~>
(*removeAll* φ (ξ # Φ)) $\ominus$ (*removeAll* φ Λ)
by (*simp add*: ‹$\xi \in$ *set* Λ›
filter-remove1
listSubtract-cons-remove1-perm
perm-sym
removeAll-filter-not-eq)
ultimately show *?thesis* **by** *blast*
next
assume $\xi \notin$ *set* Λ
hence (ξ # Φ) $\ominus$ Λ <~~> ξ # ($\Phi \ominus \Lambda$)
by (*simp add*: *listSubtract-cons-absorb perm-sym*)
hence (ξ # Φ) $\ominus$ Λ <~~> ξ # ((*removeAll* φ Φ) $\ominus$ (*removeAll* φ Λ))
using ‹$\Phi \ominus \Lambda$ <~~> *removeAll* φ Φ $\ominus$ *removeAll* φ Λ› **by** *blast*
hence (ξ # Φ) $\ominus$ Λ <~~> (ξ # (*removeAll* φ Φ)) $\ominus$ (*removeAll* φ Λ)
by (*simp add*: ‹$\xi \notin$ *set* Λ› *listSubtract-cons*)
thus *?thesis* **using** ‹$\xi \neq \varphi$› **by** *auto*
qed
qed
}
then show *?case* **by** *auto*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma *listSubtract-permute*:
assumes Φ <~~> Ψ
shows $\Phi \ominus \Lambda$ <~~> $\Psi \ominus \Lambda$
proof −
from ‹Φ <~~> Ψ› **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-listSubtract-intro*:
assumes A <~~> B @ C
shows $A \ominus C$ <~~> B
proof −
from ‹A <~~> 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 *listSubtract-concat*:
assumes $\Psi \in$ *set* $\mathcal{L}$
shows *concat* $(\mathcal{L} \ominus [\Psi])$ <~~> $($*concat* $\mathcal{L}) \ominus \Psi$
using *assms*
by (*simp*,
meson append-perm-listSubtract-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* $\Phi \longrightarrow (\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow \Phi.\ f\ \varphi')$
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
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow ((\textit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow (\textit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
using *Cons.hyps* **by** *blast*
moreover have (*remove1* ψ $\Phi) \ominus \Psi$ <~~> $\Phi \ominus (\psi\ \#\ \Psi)$
by (*meson listSubtract-remove1-cons-perm perm-sym*)
hence $(\sum \varphi' \leftarrow ((\textit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow (\Phi \ominus (\psi\ \#\ \Psi)).\ f\ \varphi')$
using *perm-list-summation* **by** *blast*
ultimately have
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus (\psi\ \#\ \Psi)).\ f\ \varphi') = (\sum \varphi' \leftarrow (\textit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
by *simp*
hence
$(\sum \psi' \leftarrow (\psi\ \#\ \Psi).\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus (\psi\ \#\ \Psi)).\ f\ \varphi') =$
$(\sum \varphi' \leftarrow (\psi\ \#\ (\textit{remove1}\ \psi\ \Phi)).\ f\ \varphi')$
by (*simp add*: *add.assoc*)
moreover have $\psi \in$ *set* Φ

```
        by (metis append-Cons hypothesis list.set-intros(1) mset-le-perm-append
perm-set-eq)
      hence (ψ # (remove1 ψ Φ)) <~~> Φ
        by (simp add: perm-remove perm-sym)
      hence (∑φ′← (ψ # (remove1 ψ Φ)). f φ′) = (∑φ′←Φ. f φ′)
        using perm-list-summation by blast
      ultimately have
       (∑ψ′← (ψ # Ψ). f ψ′) + (∑φ′←(Φ ⊖ (ψ # Ψ)). f φ′) = (∑φ′←Φ. f φ′)
        by simp
    }
    then show ?case
      by blast
  qed
  with assms show ?thesis by blast
qed

lemma listSubtract-set-difference-lower-bound:
  set Γ − set Φ ⊆ set (Γ ⊖ Φ)
  using subset-Diff-insert
  by (induct Φ, simp, fastforce)

lemma listSubtract-set-trivial-upper-bound:
  set (Γ ⊖ Φ) ⊆ set Γ
      by (induct Φ,
          simp,
          simp,
          meson dual-order.trans
                set-remove1-subset)

lemma listSubtract-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 $\varphi \in$ *set* Γ
by (*metis* ‹*mset* ($\Gamma \ominus$ (φ # Φ)) = *mset* (*remove1* φ $\Gamma \ominus \Phi$)›
‹*mset* (φ # Φ) $\subseteq$# *mset* Γ›
‹*mset* Φ $\subseteq$# *mset* (*remove1* φ Γ)›
add-diff-cancel-left'
add-right-cancel
eq-iff
impossible-Cons
listSubtract-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* Γ = m + *length* (φ # Φ) by *simp*
}
thus *?case* by *blast*
qed
thus *?thesis* using *assms* by *blast*
qed

lemma *listSubtract-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
listSubtract-mset-homomorphism
remove1-idem set-mset-mset)

lemma *listSubtract-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-listSubtract-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-listSubtract-mset-equivalence*:
assumes *mset* $B \subseteq\#$ *mset* A
shows *mset* ((*map f A*) $\ominus$ (*map f B*)) = *mset* (*map f* ($A \ominus B$))
using *assms*
by (*induct B*, *simp*, *simp add*: *image-mset-Diff*)

lemma *msub-listSubtract-elem-cons-msub*:
assumes *mset* $\Xi \subseteq\#$ *mset* Γ
and $\psi \in$ *set* $(\Gamma \ominus \Xi)$
shows *mset* $(\psi \# \Xi) \subseteq\#$ *mset* Γ
proof −
have $\forall$ Γ. *mset* $\Xi \subseteq\#$ *mset* Γ −−> $\psi \in$ *set* $(\Gamma \ominus \Xi)$ −−> *mset* $(\psi \# \Xi) \subseteq\#$
mset Γ
proof(*induct* Ξ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
{
fix Γ
assume *mset* $(\xi \# \Xi) \subseteq\#$ *mset* Γ
$\psi \in$ *set* $(\Gamma \ominus (\xi \# \Xi))$
hence $\xi \in$ *set* Γ
mset $\Xi \subseteq\#$ *mset* (*remove1* ξ Γ)
$\psi \in$ *set* ((*remove1* ξ Γ) $\ominus$ 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 listSubtract.simps(*1*)
listSubtract.simps(*2*)
listSubtract-monotonic
remove-hd,
simp, *metis listSubtract-remove1-cons-perm*
perm-set-eq)
with *Cons.hyps* **have** *mset* Γ = *mset* $(\xi \#$ (*remove1* ξ Γ))
mset $(\psi \# \Xi) \subseteq\#$ *mset* (*remove1* ξ Γ)
by (*simp*, *blast*)
hence *mset* $(\psi \# \xi \# \Xi) \subseteq\#$ *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

5.11 Tuple Lists

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

lemma *remove1-pairs-list-projections-snd*:
assumes $(\gamma,\sigma) \in\#$ *mset* Φ
shows *mset* (*map snd* (*remove1* (γ, σ) Φ)) = *mset* (*map snd* Φ) − $\{\#\ \sigma\ \#\}$
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 ?ΔΩ = map (λ (-, σ, γ). (γ, σ)) Ω
  let ?ψ = fst ψ
```

let *?σ = snd ψ*
from *Cons.prems* **have** *add-mset ?σ (image-mset snd (mset Ψ)) ⊆# mset Σ* **by** *simp*
then have *mset Σ − {#?σ#} − image-mset snd (mset Ψ) ≠ mset Σ − image-mset snd (mset Ψ)*
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 *?σ ∈# mset Σ − mset (map snd Ψ)*
using *diff-single-trivial* **by** *fastforce*
have *mset (map snd (ψ # Ψ)) ⊆# mset (map snd Δ)*
by (*meson Cons.prems ‹mset Σ ⊆# mset (map snd Δ)› subset-mset.dual-order.trans*)
then have *mset (map snd Δ) − mset (map snd (ψ # Ψ)) + ({#} + {#snd ψ#})*
= mset (map snd Δ) + ({#} + {#snd ψ#}) − add-mset (snd ψ) (mset (map snd Ψ))
by (*metis* (*no-types*) *list.simps(9) mset.simps(2) mset-subset-eq-multiset-union-diff-commute*)
then have *mset (map snd Δ) − mset (map snd (ψ # Ψ)) + ({#} + {#snd ψ#})*
= mset (map snd Δ) − mset (map snd Ψ)
by *auto*
hence *?σ ∈# mset (map snd Δ) − mset (map snd Ψ)*
using *add-mset-remove-trivial-eq* **by** *fastforce*
moreover have *snd ∘ (λ (ψ, σ, -). (ψ, σ)) = snd ∘ (λ (-, σ, γ). (γ, σ))* **by** *auto*
hence *map snd (?Δ_Ω) = map snd (map (λ (ψ, σ, -). (ψ, σ)) Ω)*
by *fastforce*
hence *map snd (?Δ_Ω) = map snd Ψ*
using *Ω(1)* **by** *simp*
ultimately have *?σ ∈# mset (map snd Δ) − mset (map snd ?Δ_Ω)*
by *simp*
hence *?σ ∈# image-mset snd (mset Δ − mset ?Δ_Ω)*
using *Ω(2)* **by** (*metis image-mset-Diff mset-map*)
hence *?σ ∈ snd ‘ set-mset (mset Δ − mset ?Δ_Ω)*
by (*metis in-image-mset*)
from *this* **obtain** *ϱ* **where** *ϱ*:
snd ϱ = ?σ ϱ ∈# mset Δ − mset ?Δ_Ω
using *imageE* **by** *auto*
from *this* **obtain** *γ* **where**
(γ, ?σ) = ϱ
by (*metis prod.collapse*)
with *ϱ(2)* **have** *γ: (γ, ?σ) ∈# mset Δ − mset ?Δ_Ω* **by** *auto*
let *?Ω = (?ψ, ?σ, γ) # Ω*
have *map (λ (ψ, σ, -). (ψ, σ)) ?Ω = ψ # Ψ*
using *Ω(1)* **by** *simp*
moreover
have *A: (γ, snd ψ) = (case (snd ψ, γ) of (a, c) ⇒ (c, a))*
by *auto*

have B: *mset* (*map* ($\lambda(b, a, c). (c, a)$) Ω) + {#*case* (*snd* ψ, γ) *of* $(a, c) \Rightarrow (c, a)$#}
  = *mset* (*map* ($\lambda(b, a, c). (c, a)$) ((*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* ($\lambda(b, a, c). (c, a)$) Ω) + *mm* (*mset* Δ) (*mset* (*map* ($\lambda(b, a, c). (c, a)$) Ω))
  by (*metis* Ω(*2*) *subset-mset.le-iff-add*)
then have *mset* (*map* (λ (-, σ, γ). (γ, σ)) *?*Ω) $\subseteq$# *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

5.12 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* ($\Phi \cap \Psi$) = *mset* Φ $\cap$# *mset* Ψ
proof –
have $\forall$ Φ. *mset* ($\Phi \cap \Psi$) = *mset* Φ $\cap$# *mset* Ψ
proof (*induct* Ψ)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons* ψ Ψ)
  {
    fix Φ
    have *mset* ($\Phi \cap \psi$ # Ψ) = *mset* Φ $\cap$# *mset* (ψ # Ψ)
      using *Cons.hyps*
      by (*cases* $\psi \in$ *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*]: [] $\cap$ Φ = [] **by** (*induct* Φ, *simp*+)

lemma *list-diff-intersect-comp*:

mset Φ = *mset* ($\Phi \ominus \Psi$) + *mset* ($\Phi \cap \Psi$)
by (*simp add*: *multiset-inter-def*)

lemma *list-intersect-left-project*: *mset* ($\Phi \cap \Psi$) $\subseteq\#$ *mset* Φ
by *simp*

lemma *list-intersect-right-project*: *mset* ($\Phi \cap \Psi$) $\subseteq\#$ *mset* Ψ
by *simp*

end

6 Classical Propositional Connectives

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

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

6.1 Verum

definition (**in** *Minimal-Logic-With-Falsum*) *verum* :: $'a$ ($\top$)
where
$\top = \bot \to \bot$

lemma (**in** *Minimal-Logic-With-Falsum*) *verum-tautology* [*simp*]: $\vdash \top$
by (*metis list-implication.simps*(*1*) *list-implication-Axiom-1 verum-def*)

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

lemma (**in** *Classical-Propositional-Logic*) *verum-embedding* [*simp*]:
$(\!| \top |\!) = \top$
unfolding *verum-def Minimal-Logic-With-Falsum-class.verum-def*
by *simp*

6.2 Conjunction

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

primrec (**in** *Classical-Propositional-Logic*) *Arbitrary-Conjunction* :: $'a$ *list* $\Rightarrow$ $'a$ ($\bigsqcap$)
where
$\bigsqcap\ [] = \top$

| $\bigsqcap$ $(\varphi \# \Phi) = \varphi \sqcap \bigsqcap \Phi$

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *conjunction-left-elimination*:
$\vdash (\varphi \sqcap \psi) \to \varphi$
by (*metis* (*full-types*) *Peirces-law*
The-Principle-of-Pseudo-Scotus
conjunction-def
list-deduction-base-theory
list-deduction-modus-ponens
list-deduction-theorem
list-deduction-weaken)

lemma (**in** *Classical-Propositional-Logic*) *conjunction-right-elimination*:
$\vdash (\varphi \sqcap \psi) \to \psi$
by (*metis* (*full-types*) *Axiom-1*
Contraposition
Modus-Ponens
conjunction-def
flip-hypothetical-syllogism
flip-implication)

lemma (**in** *Classical-Propositional-Logic*) *conjunction-embedding* [*simp*]:
$(\!| \varphi \sqcap \psi |\!) = (\!| \varphi |\!) \sqcap (\!| \psi |\!)$
unfolding *conjunction-def Classical-Propositional-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*

6.3 Biconditional

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

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *biconditional-left-elimination*:
$\vdash (\varphi \leftrightarrow \psi) \rightarrow \varphi \rightarrow \psi$
by (*simp add*: *biconditional-def conjunction-left-elimination*)

lemma (**in** *Classical-Propositional-Logic*) *biconditional-right-elimination*:
$\vdash (\varphi \leftrightarrow \psi) \rightarrow \psi \rightarrow \varphi$
by (*simp add*: *biconditional-def conjunction-right-elimination*)

lemma (**in** *Classical-Propositional-Logic*) *biconditional-embedding* [*simp*]:
$(\!| \varphi \leftrightarrow \psi |\!) = (\!| \varphi |\!) \leftrightarrow (\!| \psi |\!)$
unfolding *biconditional-def Classical-Propositional-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*)

6.4 Negation

definition (**in** *Minimal-Logic-With-Falsum*) *negation* :: $'a \Rightarrow 'a$ ($\sim$)
where
$\sim \varphi = \varphi \rightarrow \bot$

lemma (**in** *Minimal-Logic-With-Falsum*) *negation-introduction*:
$\vdash (\varphi \rightarrow \bot) \rightarrow \sim \varphi$
unfolding *negation-def*
by (*metis Axiom-1 Modus-Ponens implication-absorption*)

lemma (**in** *Minimal-Logic-With-Falsum*) *negation-elimination*:
$\vdash \sim \varphi \rightarrow (\varphi \rightarrow \bot)$
unfolding *negation-def*
by (*metis Axiom-1 Modus-Ponens implication-absorption*)

lemma (**in** *Classical-Propositional-Logic*) *negation-embedding* [*simp*]:
$(\!| \sim \varphi |\!) = \sim (\!| \varphi |\!)$
unfolding *negation-def Minimal-Logic-With-Falsum-class.negation-def*
by *simp*

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

6.5 Disjunction

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

primrec (**in** *Classical-Propositional-Logic*) *Arbitrary-Disjunction* :: $'a$ *list* ⇒ $'a$
$(\bigsqcup)$
  where
    $\bigsqcup\ [] = \bot$
  | $\bigsqcup\ (\varphi \# \Phi) = \varphi \sqcup \bigsqcup\ \Phi$

lemma (**in** *Classical-Propositional-Logic*) *disjunction-elimination*:
  $\vdash (\varphi \to \chi) \to (\psi \to \chi) \to (\varphi \sqcup \psi) \to \chi$
proof −
  let $?\Gamma = [\varphi \to \chi,\ \psi \to \chi,\ \varphi \sqcup \psi]$
  have $?\Gamma :\vdash (\varphi \to \bot) \to \chi$
    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-Propositional-Logic*) *disjunction-left-introduction*:
  $\vdash \varphi \to (\varphi \sqcup \psi)$
  unfolding *disjunction-def*
  by (*metis Modus-Ponens*
            The-Principle-of-Pseudo-Scotus
            flip-implication)

lemma (**in** *Classical-Propositional-Logic*) *disjunction-right-introduction*:
  $\vdash \psi \to (\varphi \sqcup \psi)$
  unfolding *disjunction-def*
  using *Axiom-1*
  by *simp*

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

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

6.6 Mutual Exclusion

primrec (**in** *Classical-Propositional-Logic*) *exclusive* :: $'a$ *list* $\Rightarrow$ $'a$ ($\coprod$)
  where
    $\coprod$ [] = $\top$
  | $\coprod$ (φ # Φ) = $\sim (\varphi \sqcap \bigsqcup \Phi) \sqcap \coprod \Phi$

6.7 Subtraction

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

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

6.8 Common Rules

6.8.1 Biconditional Equivalence Relation

lemma (**in** *Classical-Propositional-Logic*) *biconditional-reflection*:
  $\vdash \varphi \leftrightarrow \varphi$
  by (*meson Axiom-1 Modus-Ponens biconditional-introduction implication-absorption*)

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-Logic*) *biconditional-transitivity*:
  $\vdash (\varphi \leftrightarrow \psi) \to (\psi \leftrightarrow \chi) \to (\varphi \leftrightarrow \chi)$
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-Propositional-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*

6.8.2 Biconditional Weakening

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-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)

6.8.3 Conjunction Identities

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

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

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

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-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*: *The-Principle-of-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* Φ)
assume $\psi \in$ *set* Φ
have $\forall\ \varphi \in$ *set* $\Phi. \vdash \bigsqcap \Phi \leftrightarrow (\varphi \sqcap \bigsqcap$ (*removeAll* φ Φ))
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* χ Φ)
{
fix φ
assume $\varphi \in$ *set* $(\chi \# \Phi)$
have $\vdash \bigsqcap (\chi \# \Phi) \leftrightarrow (\varphi \sqcap \bigsqcap$ (*removeAll* φ $(\chi \# \Phi)$))
proof *cases*
assume $\varphi \in$ *set* Φ
hence $\vdash \bigsqcap \Phi \leftrightarrow (\varphi \sqcap \bigsqcap$ (*removeAll* φ Φ))
using *Cons.hyps* ‹$\varphi \in$ *set* Φ›
by *auto*
moreover
have $\vdash (\bigsqcap \Phi \leftrightarrow (\varphi \sqcap \bigsqcap$ (*removeAll* φ Φ))) $\rightarrow$
$(\chi \sqcap \bigsqcap \Phi) \leftrightarrow (\varphi \sqcap \chi \sqcap \bigsqcap$ (*removeAll* φ Φ))
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\bigsqcap \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcap$ (*removeAll* φ $\Phi)\rangle)) \rightarrow$
$(\langle\chi\rangle \sqcap \langle\bigsqcap \Phi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap \langle\chi\rangle \sqcap \langle\bigsqcap$ (*removeAll* φ
$\Phi)\rangle)$
by *auto*
hence $\vdash$ ⦇ $(\langle\bigsqcap \Phi\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcap$ (*removeAll* φ $\Phi)\rangle)) \rightarrow$
$(\langle\chi\rangle \sqcap \langle\bigsqcap \Phi\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap \langle\chi\rangle \sqcap \langle\bigsqcap$ (*removeAll* φ $\Phi)\rangle)$ ⦈
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

```
    ultimately have ⊢ ⨅ (χ # Φ) ↔ (φ ⊓ χ ⊓ ⨅ (removeAll φ Φ))
      using Modus-Ponens by auto
    show ?thesis
    proof cases
      assume φ = χ
      moreover
      {
        fix φ
        have ⊢ (χ ⊓ φ) → (χ ⊓ χ ⊓ φ)
          unfolding conjunction-def
          by (meson Axiom-2
                    Double-Negation
                    Modus-Ponens
                    flip-hypothetical-syllogism
                    flip-implication)
      } note tautology = this
      from ‹⊢ ⨅ (χ # Φ) ↔ (φ ⊓ χ ⊓ ⨅ (removeAll φ Φ))›
           ‹φ = χ›
      have ⊢ (χ ⊓ ⨅ (removeAll χ Φ)) → (χ ⊓ ⨅ Φ)
        unfolding biconditional-def
        by (simp, metis tautology hypothetical-syllogism Modus-Ponens)
      moreover
      from ‹⊢ ⨅ (χ # Φ) ↔ (φ ⊓ χ ⊓ ⨅ (removeAll φ Φ))›
           ‹φ = χ›
      have ⊢ (χ ⊓ ⨅ Φ) → (χ ⊓ ⨅ (removeAll χ Φ))
        unfolding biconditional-def
        by (simp,
            metis conjunction-right-elimination
                  hypothetical-syllogism
                  Modus-Ponens)
      ultimately show ?thesis
        unfolding biconditional-def
        by simp
    next
      assume φ ≠ χ
      then show ?thesis
        using ‹⊢ ⨅ (χ # Φ) ↔ (φ ⊓ χ ⊓ ⨅ (removeAll φ Φ))›
        by simp
    qed
  next
    assume φ ∉ set Φ
    hence φ = χ χ ∉ set Φ
      using ‹φ ∈ set (χ # Φ)› 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* ‹$\psi \in \mathit{set}\ \Phi$›
by *blast*
moreover
from ‹$\psi \in \mathit{set}\ \Phi$› ‹$\mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi\ \#\ \Psi)$› *Cons.hyps*
have $\vdash \bigsqcap\ \Psi \to \bigsqcap\ (\mathit{removeAll}\ \psi\ \Phi)$
by (*simp add*: *subset-insert-iff insert-absorb*)
hence $\vdash \bigsqcap\ (\psi\ \#\ \Psi) \to (\psi \sqcap \bigsqcap\ (\mathit{removeAll}\ \psi\ \Phi))$
apply *simp*
unfolding *conjunction-def*
using *Modus-Ponens hypothetical-syllogism flip-hypothetical-syllogism*
by *meson*
ultimately show *?thesis*
apply *simp*
using *Modus-Ponens hypothetical-syllogism*
by *blast*
next
assume $\psi \notin \mathit{set}\ \Phi$
hence $\vdash \bigsqcap\ \Psi \to \bigsqcap\ \Phi$
using *Cons.hyps* ‹$\mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi\ \#\ \Psi)$›
by *auto*
then show *?thesis*
apply *simp*
unfolding *conjunction-def*
by (*metis Modus-Ponens*
conjunction-def
conjunction-right-elimination
hypothetical-syllogism)
qed
}
thus *?case* **by** *blast*
qed
thus $\mathit{set}\ \Phi \subseteq \mathit{set}\ \Psi \Longrightarrow \vdash \bigsqcap\ \Psi \to \bigsqcap\ \Phi$ **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *arbitrary-conjunction-remdups*:
$\vdash (\bigsqcap\ \Phi) \leftrightarrow \bigsqcap\ (\mathit{remdups}\ \Phi)$
by (*simp add*: *arbitrary-conjunction-antitone biconditional-def*)

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

lemma (**in** *Classical-Propositional-Logic*) *list-curry-uncurry*:
$\vdash (\Phi :\rightarrow \chi) \leftrightarrow (\bigsqcap \Phi \rightarrow \chi)$
proof (*induct* Φ)
case *Nil*
then show *?case*
apply *simp*
unfolding *biconditional-def*
conjunction-def
verum-def
using *Axiom-1*
Ex-Falso-Quodlibet
Modus-Ponens
conjunction-def
excluded-middle-elimination
set-deduction-base-theory
conjunction-set-deduction-equivalence
by *metis*
next
case (*Cons* φ Φ)
have $\vdash ((\varphi \# \Phi) :\rightarrow \chi) \leftrightarrow (\varphi \rightarrow (\Phi :\rightarrow \chi))$
by (*simp add*: *biconditional-reflection*)
with *Cons* **have** $\vdash ((\varphi \# \Phi) :\rightarrow \chi) \leftrightarrow (\varphi \rightarrow \bigsqcap \Phi \rightarrow \chi)$
by (*metis Modus-Ponens*
biconditional-def
hypothetical-syllogism
list-implication.simps(*2*)
weak-conjunction-deduction-equivalence)
with *curry-uncurry* [**where** $?\varphi=\varphi$
and $?\psi=\bigsqcap \Phi$
and $?\chi=\chi$]
show *?case*
unfolding *biconditional-def*
by (*simp*, *metis Modus-Ponens hypothetical-syllogism*)
qed

6.8.4 Disjunction Identities

lemma (**in** *Classical-Propositional-Logic*) *bivalence*:
$\vdash \sim \varphi \sqcup \varphi$
by (*simp add*: *Double-Negation disjunction-def negation-def*)

lemma (**in** *Classical-Propositional-Logic*) *implication-equivalence*:
$\vdash (\sim \varphi \sqcup \psi) \leftrightarrow (\varphi \rightarrow \psi)$
by (*metis Double-Negation-converse*
Modus-Ponens
biconditional-introduction
bivalence
disjunction-def

flip-hypothetical-syllogism
negation-def)

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-Logic*) *arbitrary-disjunction-monotone*:
set $\Phi \subseteq$ *set* $\Psi \Longrightarrow \vdash \bigsqcup\ \Phi \rightarrow \bigsqcup\ \Psi$
proof −
have $\forall\ \Phi.$ *set* $\Phi \subseteq$ *set* $\Psi \longrightarrow \vdash \bigsqcup\ \Phi \rightarrow \bigsqcup\ \Psi$
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 $\vdash \bigsqcup\ \Phi \rightarrow \bigsqcup\ (\psi\ \#\ \Psi)$
proof *cases*
assume $\psi \in$ *set* Φ
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* Φ
hence $\vdash \bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup\ ($*removeAll* $\varphi\ \Phi))$

```
        using Cons.hyps ‹φ ∈ set Φ›
        by auto
      moreover
      have ⊢ (⨆ Φ ↔ (φ ⊔ ⨆ (removeAll φ Φ))) →
            (χ ⊔ ⨆ Φ) ↔ (φ ⊔ χ ⊔ ⨆ (removeAll φ Φ))
      proof −
        have ∀ 𝔐. 𝔐 ⊨prop (⟨⨆ Φ⟩ ↔ (⟨φ⟩ ⊔ ⟨⨆ (removeAll φ Φ)⟩)) →
                    (⟨χ⟩ ⊔ ⟨⨆ Φ⟩) ↔ (⟨φ⟩ ⊔ ⟨χ⟩ ⊔ ⟨⨆ (removeAll φ
Φ)⟩)
          by auto
        hence ⊢ ⦇ (⟨⨆ Φ⟩ ↔ (⟨φ⟩ ⊔ ⟨⨆ (removeAll φ Φ)⟩)) →
                (⟨χ⟩ ⊔ ⟨⨆ Φ⟩) ↔ (⟨φ⟩ ⊔ ⟨χ⟩ ⊔ ⟨⨆ (removeAll φ Φ)⟩) ⦈
          using propositional-semantics by blast
        thus ?thesis by simp
      qed
      ultimately have ⊢ ⨆ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⨆ (removeAll φ Φ))
        using Modus-Ponens by auto
      show ?thesis
      proof cases
        assume φ = χ
        then show ?thesis
          using ‹⊢ ⨆ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⨆ (removeAll φ Φ))›
          unfolding biconditional-def
          by (simp add: disjunction-def,
                meson Axiom-1 Modus-Ponens flip-hypothetical-syllogism
implication-absorption)
      next
        assume φ ≠ χ
        then show ?thesis
          using ‹⊢ ⨆ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⨆ (removeAll φ Φ))›
          by simp
      qed
    next
      assume φ ∉ set Φ
      hence φ = χ χ ∉ set Φ
        using ‹φ ∈ set (χ # Φ)› by auto
      then show ?thesis
        using biconditional-reflection
        by simp
    qed
  }
  thus ?case by blast
qed
hence ⊢ ⨆ Φ → (ψ ⊔ ⨆ (removeAll ψ Φ))
  using Modus-Ponens biconditional-left-elimination ‹ψ ∈ set Φ› by blast
moreover
from ‹ψ ∈ set Φ› ‹set Φ ⊆ set (ψ # Ψ)› Cons.hyps
have ⊢ ⨆ (removeAll ψ Φ) → ⨆ Ψ
  by (simp add: subset-insert-iff insert-absorb)
```

hence $\vdash (\psi \sqcup \bigsqcup \ (removeAll \ \psi \ \Phi)) \to \bigsqcup \ (\psi \ \# \ \Psi)$
apply *simp*
unfolding *disjunction-def*
using *Modus-Ponens hypothetical-syllogism* **by** *blast*
ultimately show *?thesis*
apply *simp*
using *Modus-Ponens hypothetical-syllogism* **by** *blast*
next
assume $\psi \notin set \ \Phi$
hence $\vdash \bigsqcup \ \Phi \to \bigsqcup \ \Psi$
using *Cons.hyps* ‹$set \ \Phi \subseteq set \ (\psi \ \# \ \Psi)$›
by *auto*
then show *?thesis*
apply *simp*
unfolding *disjunction-def*
using *Axiom-1 Modus-Ponens flip-implication* **by** *blast*
qed
}
then show *?case* **by** *blast*
qed
thus $set \ \Phi \subseteq set \ \Psi \Longrightarrow \vdash \bigsqcup \ \Phi \to \bigsqcup \ \Psi$ **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *arbitrary-disjunction-remdups*:
$\vdash (\bigsqcup \ \Phi) \leftrightarrow \bigsqcup \ (remdups \ \Phi)$
by (*simp add*: *arbitrary-disjunction-monotone biconditional-def*)

6.8.5 Distribution Identities

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-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*
apply *simp*
using *Modus-Ponens hypothetical-syllogism*
by *blast*
ultimately show *?case*
by (*simp*, *metis biconditional-transitivity-rule*)
qed

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

lemma (**in** *Classical-Propositional-Logic*) *list-implication-distribution*:
$\vdash (\Phi :\to (\psi \sqcap \chi)) \leftrightarrow ((\Phi :\to \psi) \sqcap (\Phi :\to \chi))$
proof (*induct* Φ)
case *Nil*
then show *?case*
by (*simp add*: *biconditional-reflection*)
next
case (*Cons* φ Φ)

hence $\vdash (\varphi \# \Phi) :\rightarrow (\psi \sqcap \chi) \leftrightarrow (\varphi \rightarrow (\Phi :\rightarrow \psi \sqcap \Phi :\rightarrow \chi))$
  unfolding *biconditional-def*
  apply *simp*
  using *Modus-Ponens hypothetical-syllogism*
  by *blast*
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-Propositional-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-Propositional-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-Propositional-Logic*) *disjunction-arbitrary-distribution*:
$\vdash (\varphi \sqcup \bigsqcap\ \Psi) \leftrightarrow \bigsqcap\ [\varphi \sqcup \psi.\ \psi \leftarrow \Psi]$
proof (*induct* Ψ)
case *Nil*
then show *?case*
  unfolding *disjunction-def biconditional-def*
  using *Axiom-1 Modus-Ponens verum-tautology*
  by (*simp*, *blast*)
next
case (*Cons* ψ Ψ)
have $\vdash (\varphi \sqcup \bigsqcap\ (\psi \# \Psi)) \leftrightarrow ((\varphi \sqcup \psi) \sqcap (\varphi \sqcup \bigsqcap\ \Psi))$
  by (*simp add*: *disjunction-distribution*)
moreover
from *biconditional-conjunction-weaken-rule*
  Cons
have $\vdash ((\varphi \sqcup \psi) \sqcap \varphi \sqcup \bigsqcap\ \Psi) \leftrightarrow \bigsqcap\ (map\ (\lambda\ \chi\ .\ \varphi \sqcup \chi)\ (\psi \# \Psi))$
  by *simp*
ultimately show *?case*
  by (*metis biconditional-transitivity-rule*)
qed

lemma (**in** *Classical-Propositional-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-1
              Modus-Ponens
              list-implication-Axiom-1
              verum-tautology)
next
  case (*Cons* ψ Ψ)
  have $\vdash \Phi :\to \prod (\psi \# \Psi) \leftrightarrow (\Phi :\to \psi \sqcap \Phi :\to \prod \Psi)$
    using *list-implication-distribution*
    by *fastforce*
  moreover
  from *biconditional-conjunction-weaken-rule*
       Cons
  have $\vdash (\Phi :\to \psi \sqcap \Phi :\to \prod \Psi) \leftrightarrow \prod [\Phi :\to \psi.\ \psi \leftarrow (\psi \# \Psi)]$
    by *simp*
  ultimately show *?case*
    by (*metis biconditional-transitivity-rule*)
qed

lemma (**in** *Classical-Propositional-Logic*) *implication-arbitrary-distribution*:
  $\vdash (\varphi \to \prod \Psi) \leftrightarrow \prod [\varphi \to \psi.\ \psi \leftarrow \Psi]$
  using *list-implication-arbitrary-distribution* [**where** $?\Phi=[\varphi]$]
  by *simp*

6.8.6 Negation

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-Logic*) *alt-double-negation-elimination* [*simp*]:
  $\Gamma \Vdash (\varphi \to \bot) \to \bot \equiv \Gamma \Vdash \varphi$
  using *double-negation-elimination*
  unfolding *negation-def*
  by *auto*

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

lemma (**in** *Classical-Propositional-Logic*) *alt-base-double-negation-elimination* [*simp*]:
$\vdash (\varphi \to \bot) \to \bot \equiv \vdash \varphi$
using *base-double-negation-elimination*
unfolding *negation-def*
by *auto*

6.9 Mutual Exclusion Identities

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

lemma (**in** *Classical-Propositional-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 $\vdash \sim (\psi \sqcap \bigsqcup\ (\varphi \# \Phi)) \leftrightarrow (\sim (\psi \sqcap \varphi) \sqcap \sim (\psi \sqcap \bigsqcup\ \Phi))$
by *simp*
hence $\Gamma \Vdash \sim (\psi \sqcap \bigsqcup\ (\varphi \# \Phi)) = (\Gamma \Vdash \sim (\psi \sqcap \varphi) \wedge (\forall \varphi{\in}set\ \Phi.\ \Gamma \Vdash \sim (\psi \sqcap$
$\varphi)))$
using *set-deduction-weaken* [**where** $\Gamma{=}\Gamma$]
conjunction-set-deduction-equivalence [**where** $\Gamma{=}\Gamma$]
Cons.hyps
biconditional-def
set-deduction-modus-ponens
by *metis*
thus $\Gamma \Vdash \sim (\psi \sqcap \bigsqcup\ (\varphi \# \Phi)) = (\forall \varphi{\in}set\ (\varphi \# \Phi).\ \Gamma \Vdash \sim (\psi \sqcap \varphi))$
by *simp*

qed

lemma (**in** *Classical-Propositional-Logic*) *exclusive-elimination1*:
  assumes $\Gamma \Vdash \coprod\ \Phi$
  shows $\forall\ \varphi \in$ *set* $\Phi.\ \forall\ \psi \in$ *set* $\Phi.\ (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$
  using *assms*
proof (*induct* Φ)
  case *Nil*
  thus *?case* **by** *auto*
next
  case (*Cons* χ Φ)
  assume $\Gamma \Vdash \coprod\ (\chi \# \Phi)$
  hence $\Gamma \Vdash \coprod\ \Phi$ **by** *simp*
  hence $\forall \varphi \in$*set* $\Phi.\ \forall \psi \in$*set* $\Phi.\ \varphi \neq \psi \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$ **using** *Cons.hyps* **by** *blast*
  moreover have $\Gamma \Vdash \sim (\chi \sqcap \bigsqcup\ \Phi)$
    using ‹$\Gamma \Vdash \coprod\ (\chi \# \Phi)$› *conjunction-set-deduction-equivalence* **by** *auto*
  hence $\forall\ \varphi \in$ *set* $\Phi.\ \Gamma \Vdash \sim (\chi \sqcap \varphi)$
    using *disjuction-exclusion-equivalence* **by** *auto*
  moreover {
    fix φ
    have $\vdash \sim (\chi \sqcap \varphi) \to \sim (\varphi \sqcap \chi)$
      unfolding *negation-def*
           conjunction-def
      using *Modus-Ponens flip-hypothetical-syllogism flip-implication* **by** *blast*
  }
  with ‹$\forall\ \varphi \in$ *set* $\Phi.\ \Gamma \Vdash \sim (\chi \sqcap \varphi)$› **have** $\forall\ \varphi \in$ *set* $\Phi.\ \Gamma \Vdash \sim (\varphi \sqcap \chi)$
    using *set-deduction-weaken* [**where** $\Gamma=\Gamma$]
        set-deduction-modus-ponens [**where** $\Gamma=\Gamma$]
    by *blast*
  ultimately show $\forall \varphi \in$ *set* $(\chi \# \Phi).\ \forall \psi \in$ *set* $(\chi \# \Phi).\ \varphi \neq \psi \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$
    by *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *exclusive-elimination2*:
  assumes $\Gamma \Vdash \coprod\ \Phi$
  shows $\forall\ \varphi \in$ *duplicates* $\Phi.\ \Gamma \Vdash \sim \varphi$
  using *assms*
proof (*induct* Φ)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons* φ Φ)
  assume $\Gamma \Vdash \coprod\ (\varphi \# \Phi)$
  hence $\Gamma \Vdash \coprod\ \Phi$ **by** *simp*
  hence $\forall \varphi \in$*duplicates* $\Phi.\ \Gamma \Vdash \sim \varphi$ **using** *Cons.hyps* **by** *auto*
  show *?case*
  proof *cases*

assume $\varphi \in$ *set* Φ
moreover {
fix $\varphi\ \psi\ \chi$
have $\vdash \sim (\varphi \sqcap (\psi \sqcup \chi)) \leftrightarrow (\sim (\varphi \sqcap \psi) \sqcap \sim (\varphi \sqcap \chi))$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\chi\rangle)) \leftrightarrow (\sim (\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcap \sim (\langle\varphi\rangle \sqcap \langle\chi\rangle))$
by *auto*
hence $\vdash (\!| \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\chi\rangle)) \leftrightarrow (\sim (\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcap \sim (\langle\varphi\rangle \sqcap \langle\chi\rangle)) |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence $\Gamma \Vdash \sim (\varphi \sqcap (\psi \sqcup \chi)) \equiv \Gamma \Vdash \sim (\varphi \sqcap \psi) \sqcap \sim (\varphi \sqcap \chi)$
using *set-deduction-weaken*
biconditional-weaken **by** *presburger*
}
moreover
have $\vdash \sim (\varphi \sqcap \varphi) \leftrightarrow \sim \varphi$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \sim (\langle\varphi\rangle \sqcap \langle\varphi\rangle) \leftrightarrow \sim \langle\varphi\rangle$
by *auto*
hence $\vdash (\!| \sim (\langle\varphi\rangle \sqcap \langle\varphi\rangle) \leftrightarrow \sim \langle\varphi\rangle |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence $\Gamma \Vdash \sim (\varphi \sqcap \varphi) \equiv \Gamma \Vdash \sim \varphi$
using *set-deduction-weaken*
biconditional-weaken **by** *presburger*
moreover have $\Gamma \Vdash \sim (\varphi \sqcap \bigsqcup\ \Phi)$ **using** ‹$\Gamma \Vdash \coprod\ (\varphi \ \#\ \Phi)$› **by** *simp*
ultimately have $\Gamma \Vdash \sim \varphi$ **by** (*induct* Φ, *simp*, *simp*, *blast*)
thus *?thesis* **using** ‹$\varphi \in$ *set* Φ› ‹$\forall\,\varphi\in$*duplicates* Φ. $\Gamma \Vdash \sim \varphi$› **by** *simp*
next
assume $\varphi \notin$ *set* Φ
hence *duplicates* $(\varphi \ \#\ \Phi) =$ *duplicates* Φ **by** *simp*
then show *?thesis* **using** ‹$\forall\,\varphi\in$*duplicates* Φ. $\Gamma \Vdash \sim \varphi$›
by *auto*
qed
qed

lemma (**in** *Classical-Propositional-Logic*) *exclusive-equivalence*:
$\Gamma \Vdash \coprod\ \Phi =$
$((\forall\,\varphi\in$*duplicates* $\Phi.\ \Gamma \Vdash \sim \varphi) \wedge (\forall\ \varphi \in$ *set* $\Phi.\ \forall\ \psi \in$ *set* $\Phi.\ (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)))$
proof −
{
assume $\forall\,\varphi\in$*duplicates* Φ. $\Gamma \Vdash \sim \varphi$
$\forall\ \varphi \in$ *set* Φ. $\forall\ \psi \in$ *set* Φ. $(\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$
hence $\Gamma \Vdash \coprod\ \Phi$
proof (*induct* Φ)

case *Nil*
then show *?case*
by (*simp add*: *set-deduction-weaken*)
next
case (*Cons* φ Φ)
assume *A*: $\forall\varphi{\in}$*duplicates* $(\varphi \mathbin{\#} \Phi)$. $\Gamma \Vdash \sim \varphi$
and *B*: $\forall\chi{\in}$*set* $(\varphi \mathbin{\#} \Phi)$. $\forall\psi{\in}$*set* $(\varphi \mathbin{\#} \Phi)$. $\chi \neq \psi \longrightarrow \Gamma \Vdash \sim (\chi \sqcap \psi)$
hence *C*: $\Gamma \Vdash \coprod \Phi$ **using** *Cons.hyps* **by** *simp*
then show *?case*
proof *cases*
assume $\varphi \in$ *duplicates* $(\varphi \mathbin{\#} \Phi)$
moreover from *this* **have** $\Gamma \Vdash \sim \varphi$ **using** *A* **by** *auto*
moreover have *duplicates* $\Phi \subseteq$ *set* Φ **by** (*induct* Φ, *simp*, *auto*)
ultimately have $\varphi \in$ *set* Φ **by** (*metis duplicates.simps*(*2*) *subsetCE*)
hence $\vdash \sim \varphi \leftrightarrow \sim(\varphi \sqcap \bigsqcup \Phi)$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Φ)
assume $\varphi \in$ *set* $(\psi \mathbin{\#} \Phi)$
then show $\vdash \sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup (\psi \mathbin{\#} \Phi))$
proof –
{
assume $\varphi = \psi$
hence *?thesis*
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle))$
using ‹$\varphi = \psi$› **by** *auto*
hence $\vdash (\!|\ \sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
moreover
{
assume $\varphi \neq \psi$
hence $\varphi \in$ *set* Φ **using** ‹$\varphi \in$ *set* $(\psi \mathbin{\#} \Phi)$› **by** *auto*
hence $\vdash \sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi)$ **using** *Cons.hyps* **by** *auto*
moreover have $\vdash (\sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi)) \to (\sim \varphi \leftrightarrow \sim (\varphi \sqcap (\psi \sqcup \bigsqcup$
$\Phi)))$
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle)) \to$
$(\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle)))$
by *auto*
hence $\vdash (\!|\ (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle)) \to (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle$
$\sqcup \langle\bigsqcup \Phi\rangle)))\ |\!)$
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 ‹Γ ⊩ ∼ φ› have Γ ⊩ ∼(φ ⊓ ⨆ Φ)
      using biconditional-weaken set-deduction-weaken by blast
    with ‹Γ ⊩ ∐ Φ› show ?thesis by simp
  next
    assume φ ∉ duplicates (φ # Φ)
    hence φ ∉ set Φ by auto
    with B have ∀ψ∈set Φ. Γ ⊩ ∼ (φ ⊓ ψ) by (simp, metis)
    hence Γ ⊩ ∼ (φ ⊓ ⨆ Φ)
      by (simp add: disjuction-exclusion-equivalence)
    with ‹Γ ⊩ ∐ Φ› show ?thesis by simp
  qed
 qed
}
thus ?thesis
  by (metis exclusive-elimination1 exclusive-elimination2)
qed


end
theory Logical-Probability
  imports ../../Logic/Classical/Classical-Propositional-Connectives
      ~~/src/HOL/Real
begin

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

TODO: Cite Hajek PROBABILITY, LOGIC, AND PROBABILITY LOGIC

```
class Logical-Probability = Classical-Propositional-Logic +
  fixes Pr :: 'a ⇒ real
  assumes Non-Negative: Pr φ ≥ 0
  assumes Unity: ⊢ φ ⟹ Pr φ = 1
  assumes Implicational-Additivity:
    ⊢ φ → ψ → ⊥ ⟹ Pr ((φ → ⊥) → ψ) = Pr φ + Pr ψ

lemma (in Logical-Probability) Additivity:
  assumes ⊢ ∼ (φ ⊓ ψ)
  shows Pr (φ ⊔ ψ) = Pr φ + Pr ψ
  using assms
  unfolding disjunction-def
        conjunction-def
        negation-def
  by (simp add: Implicational-Additivity)
```

lemma (**in** *Logical-Probability*) *Alternate-Additivity*:
assumes $\vdash \varphi \to \psi \to \bot$
shows $Pr\ (\varphi \sqcup \psi) = Pr\ \varphi + Pr\ \psi$
using *assms*
by (*metis Additivity*
Double-Negation-converse
Modus-Ponens
conjunction-def
negation-def)

lemma (**in** *Logical-Probability*) *complementation*:
$Pr\ (\sim \varphi) = 1 - Pr\ \varphi$
by (*metis Alternate-Additivity*
Unity
bivalence
negation-elimination
add.commute
add-diff-cancel-left')

lemma (**in** *Logical-Probability*) *unity-upper-bound*:
$Pr\ \varphi \leq 1$
by (*metis* (*no-types*) *diff-ge-0-iff-ge Non-Negative complementation*)

Alternate axiomatization of logical probability following Brian Weatherson in https://doi.org/10.1305/ndjfl/1082637807

class *Weatherson-Probability* = *Classical-Propositional-Logic* +
fixes $Pr :: 'a \Rightarrow real$
assumes *Thesis*: $Pr\ \top = 1$
assumes *Antithesis*: $Pr\ \bot = 0$
assumes *Monotonicity*: $\vdash \varphi \to \psi \Longrightarrow Pr\ \varphi \leq Pr\ \psi$
assumes *Sum-Rule*: $Pr\ \varphi + Pr\ \psi = Pr\ (\varphi \sqcap \psi) + Pr\ (\varphi \sqcup \psi)$

sublocale *Weatherson-Probability* $\subseteq$ *Logical-Probability*
proof
fix φ
have $\vdash \bot \to \varphi$
by (*simp add*: *Ex-Falso-Quodlibet*)
thus $0 \leq Pr\ \varphi$
using *Antithesis Monotonicity* **by** *fastforce*
next
fix φ
assume $\vdash \varphi$
thus $Pr\ \varphi = 1$
by (*metis Thesis*
Monotonicity
eq-iff
Axiom-1
Ex-Falso-Quodlibet
Modus-Ponens

verum-def)
next
fix φ ψ
assume $\vdash \varphi \to \psi \to \bot$
thus $Pr\ ((\varphi \to \bot) \to \psi) = Pr\ \varphi + Pr\ \psi$
by (*metis add.left-neutral*
eq-iff
Antithesis
Ex-Falso-Quodlibet
Monotonicity
Sum-Rule
conjunction-negation-identity
disjunction-def
negation-def
weak-biconditional-weaken)
qed

lemma (**in** *Logical-Probability*) *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*: *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** *Logical-Probability*) *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** *Logical-Probability*) *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-Propositional-Logic-class.subtraction-def*
Minimal-Logic-With-Falsum-class.negation-def
Classical-Propositional-Logic-class.biconditional-def
Classical-Propositional-Logic-class.conjunction-def
Classical-Propositional-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-Propositional-Logic-class.subtraction-def*
Minimal-Logic-With-Falsum-class.negation-def
Classical-Propositional-Logic-class.biconditional-def
Classical-Propositional-Logic-class.conjunction-def
Classical-Propositional-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 *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-Propositional-Logic-class.subtraction-def*
Minimal-Logic-With-Falsum-class.negation-def
Classical-Propositional-Logic-class.biconditional-def
Classical-Propositional-Logic-class.conjunction-def
Classical-Propositional-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 *Alternate-Additivity biconditional-equivalence calculation* **by** *auto*
ultimately show *?thesis*
by *simp*
qed

sublocale *Logical-Probability* $\subseteq$ *Weatherson-Probability*

proof
  show $Pr \top = 1$
    by (*simp add*: *Unity*)
next
  show $Pr \bot = 0$
    by (*metis add-cancel-left-right*
            Additivity
            Ex-Falso-Quodlibet
            Unity
            bivalence
            conjunction-right-elimination
            negation-def)
next
  fix φ ψ
  assume $\vdash \varphi \to \psi$
  thus $Pr\ \varphi \leq Pr\ \psi$
    using *monotonicity*
    by *auto*
next
  fix φ ψ
  show $Pr\ \varphi + Pr\ \psi = Pr\ (\varphi \sqcap \psi) + Pr\ (\varphi \sqcup \psi)$
    by (*metis sum-rule add.commute*)
qed

sublocale *Logical-Probability* $\subseteq$ *Consistent-Classical-Logic*
proof
  show $\neg \vdash \bot$ **using** *Unity Antithesis* **by** *auto*
qed

lemma (**in** *Logical-Probability*) *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-Propositional-Logic-class.subtraction-def*
            Minimal-Logic-With-Falsum-class.negation-def
            Classical-Propositional-Logic-class.biconditional-def
            Classical-Propositional-Logic-class.conjunction-def
            Classical-Propositional-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-Propositional-Logic-class.subtraction-def*
      Minimal-Logic-With-Falsum-class.negation-def
      Classical-Propositional-Logic-class.conjunction-def
      Classical-Propositional-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 *Additivity*
  by *auto*
qed

lemma (**in** *Logical-Probability*) *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*: *Non-Negative*)
  thus *?thesis* **by** *linarith*
qed

lemma (**in** *Logical-Probability*) *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*: *Antithesis*)
next
  case (*Cons* φ Φ)
  have $Pr\ (\bigsqcup\ (\varphi\ \#\ \Phi)) \leq Pr\ \varphi + Pr\ (\bigsqcup\ \Phi)$
    using *disjunction-sum-inequality*
    by *simp*
  with *Cons* **have** $Pr\ (\bigsqcup\ (\varphi\ \#\ \Phi)) \leq Pr\ \varphi + (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi)$ **by** *linarith*
  then show *?case* **by** *simp*
qed

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

lemma (**in** *Logical-Probability*) *arbitrary-disjunction-set-summation-inequality*:
  $Pr\ (\bigsqcup\ \Phi) \leq (\sum \varphi \in set\ \Phi.\ Pr\ \varphi)$
  by (*metis arbitrary-disjunction-list-summation-inequality*
    arbitrary-disjunction-remdups

biconditional-equivalence
sum.set-conv-list)

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

definition (**in** *Classical-Propositional-Logic*) *Logical-Probabilities* :: ($'a \Rightarrow real$) *set*
where *Logical-Probabilities* =
$\{Pr.\ class.Logical\text{-}Probability\ (\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr\ \}$

definition (**in** *Classical-Propositional-Logic*) *Dirac-Measures* :: ($'a \Rightarrow real$) *set*
where *Dirac-Measures* =
$\{\ Pr.\quad class.Logical\text{-}Probability\ (\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
$\wedge\ (\forall\, x.\ Pr\ x = 0 \vee Pr\ x = 1)\ \}$

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Measures-subset*:
Dirac-Measures $\subseteq$ *Logical-Probabilities*
unfolding *Logical-Probabilities-def Dirac-Measures-def*
by *fastforce*

lemma (**in** *Classical-Propositional-Logic*) *MCS-Dirac-Measure*:
assumes *MCS* Ω
shows $(\lambda\ \chi.\ if\ \chi{\in}\Omega\ then\ (1 :: real)\ else\ 0) \in$ *Dirac-Measures*
(**is** $?Pr \in$ *Dirac-Measures*)
proof −
have $class.Logical\text{-}Probability\ (\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ ?Pr$
proof (*standard*, *simp*,
meson assms
Formula-Maximally-Consistent-Set-reflection
Maximally-Consistent-Set-def
set-deduction-weaken)
fix $\varphi\ \psi$
assume $\vdash \varphi \rightarrow \psi \rightarrow \bot$
hence $\vdash \sim (\varphi \sqcap \psi)$
by (*simp add*: *conjunction-def negation-def*)
hence $\varphi \sqcap \psi \notin \Omega$
by (*metis assms*
Formula-Consistent-def
Formula-Maximally-Consistent-Set-def
Maximally-Consistent-Set-def
conjunction-def
conjunction-negation-identity
set-deduction-modus-ponens
set-deduction-reflection
set-deduction-weaken
weak-biconditional-weaken)

```
  hence φ ∉ Ω ∨ ψ ∉ Ω
    using assms
          Formula-Maximally-Consistent-Set-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-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

lemma (**in** *Classical-Propositional-Logic*) *arbitrary-disjunction-exclusion-MCS*:
  assumes *MCS* Ω
  shows $\bigsqcup\ \Psi \notin \Omega \equiv \forall\ \psi \in set\ \Psi.\ \psi \notin \Omega$
proof (*induct* Ψ)
  case *Nil*
  then show *?case*
    using *assms*
      Formula-Consistent-def
      Formula-Maximally-Consistent-Set-def
      Maximally-Consistent-Set-def
      set-deduction-reflection
    by (*simp*, *blast*)
next
  case (*Cons* ψ Ψ)
  have $\bigsqcup\ (\psi\ \#\ \Psi) \notin \Omega = (\psi \notin \Omega \wedge \bigsqcup\ \Psi \notin \Omega)$
    by (*simp add*: *disjunction-def*,
      meson assms
        Formula-Consistent-def
        Formula-Maximally-Consistent-Set-def
        Formula-Maximally-Consistent-Set-implication
        Maximally-Consistent-Set-def
        set-deduction-reflection)
  thus *?case* **using** *Cons.hyps* **by** *simp*
qed

end
theory *Suppes-Theorem*
  imports *Logical-Probability*
begin

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

lemma (**in** *Classical-Propositional-Logic*) *Dirac-List-Summation-Completeness*:
  $(\forall\ \delta \in Dirac\text{-}Measures.\ \delta\ \varphi \leq (\sum\psi\leftarrow\Psi.\ \delta\ \psi)) = \vdash \varphi \rightarrow \bigsqcup\ \Psi$
proof −
  {
    fix $\delta :: 'a \Rightarrow real$
    assume $\delta \in$ *Dirac-Measures*
    from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ \delta$
      unfolding *Dirac-Measures-def*
      by *auto*
    assume $\vdash \varphi \rightarrow \bigsqcup\ \Psi$

```
    hence δ φ ≤ (∑ψ←Ψ. δ ψ)
      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
               set-deduction-base-theory
               set-deduction-reflection
               set-deduction-theorem)
    hence∀ ψ ∈ set Ψ. ψ ∉ Ω
      using arbitrary-disjunction-exclusion-MCS by blast
    let ?δ = λ χ. if χ∈Ω then (1 :: real) else 0
    from ‹∀ ψ ∈ set Ψ. ψ ∉ Ω› have (∑ψ←Ψ. ?δ ψ) = 0
      by (induct Ψ, simp, simp)
    hence ¬ ?δ φ ≤ (∑ψ←Ψ. ?δ ψ)
      by (simp add: Ω(2))
    hence
      ∃ δ ∈ Dirac-Measures. ¬ (δ φ ≤ (∑ψ←Ψ. δ ψ))
      using Ω(1) MCS-Dirac-Measure by auto
  }
  ultimately show ?thesis by blast
qed

theorem (in Classical-Propositional-Logic) List-Summation-Completeness:
  (∀ Pr ∈ Logical-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 Logical-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 ∈ Logical-Probabilities
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Logical-Probabilities-def
      by auto
    show Pr φ ≤ (∑ψ←Ψ. Pr ψ)
```

    using ‹*?rhs*› *implication-list-summation-inequality*
    by *simp*
  qed
qed

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Set-Summation-Completeness*:
  $(\forall\ \delta \in \mathit{Dirac\text{-}Measures}.\ \delta\ \varphi \leq (\sum \psi \in \mathit{set}\ \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
  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-Propositional-Logic*) *Set-Summation-Completeness*:
  $(\forall\ \delta \in \mathit{Logical\text{-}Probabilities}.\ \delta\ \varphi \leq (\sum \psi \in \mathit{set}\ \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
  by (*metis Dirac-List-Summation-Completeness*
        Dirac-Set-Summation-Completeness
        List-Summation-Completeness
        sum.set-conv-list)

lemma (**in** *Logical-Probability*) *exclusive-sum-list-identity*:
  assumes $\vdash \coprod\ \Phi$
  shows $\mathit{Pr}\ (\bigsqcup\ \Phi) = (\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi)$
  using *assms*
proof (*induct* Φ)
  case *Nil*
  then show *?case*
    by (*simp add*: *Antithesis*)
next
  case (*Cons* φ Φ)
  assume $\vdash \coprod\ (\varphi \# \Phi)$
  hence $\vdash \sim (\varphi \sqcap \bigsqcup\ \Phi) \vdash \coprod\ \Phi$ **by** *simp+*
  hence $\mathit{Pr}\ (\bigsqcup(\varphi \# \Phi)) = \mathit{Pr}\ \varphi + \mathit{Pr}\ (\bigsqcup\ \Phi)$
      $\mathit{Pr}\ (\bigsqcup\ \Phi) = (\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi)$ **using** *Cons.hyps Additivity* **by** *auto*
  hence $\mathit{Pr}\ (\bigsqcup(\varphi \# \Phi)) = \mathit{Pr}\ \varphi + (\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi)$ **by** *auto*
  thus *?case* **by** *simp*
qed

lemma *sum-list-monotone*:
  fixes $f :: 'a \Rightarrow \mathit{real}$
  assumes $\forall\ x.\ f\ x \geq 0$
    and $\mathit{set}\ \Phi \subseteq \mathit{set}\ \Psi$
    and $\mathit{distinct}\ \Phi$
   shows $(\sum \varphi \leftarrow \Phi.\ f\ \varphi) \leq (\sum \psi \leftarrow \Psi.\ f\ \psi)$
  using *assms*
proof −
  assume $\forall\ x.\ f\ x \geq 0$

have $\forall\,\Phi.\ \mathit{set}\ \Phi \subseteq \mathit{set}\ \Psi \longrightarrow \mathit{distinct}\ \Phi \longrightarrow (\sum\varphi{\leftarrow}\Phi.\ f\ \varphi) \leq (\sum\psi{\leftarrow}\Psi.\ f\ \psi)$
proof (*induct* Ψ)
 case *Nil*
 then show *?case* **by** *simp*
next
 case (*Cons* ψ Ψ)
 {
  fix Φ
  assume $\mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi\ \#\ \Psi)$
   and *distinct* Φ
  have $(\sum\varphi{\leftarrow}\Phi.\ f\ \varphi) \leq (\sum\psi'{\leftarrow}(\psi\ \#\ \Psi).\ f\ \psi')$
  proof −
   {
    assume $\psi \notin \mathit{set}\ \Phi$
    with ‹$\mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi\ \#\ \Psi)$› **have** $\mathit{set}\ \Phi \subseteq \mathit{set}\ \Psi$ **by** *auto*
    hence $(\sum\varphi{\leftarrow}\Phi.\ f\ \varphi) \leq (\sum\psi{\leftarrow}\Psi.\ f\ \psi)$
     using *Cons.hyps* ‹*distinct* Φ› **by** *auto*
    moreover have $f\ \psi \geq 0$ **using** ‹$\forall\ x.\ f\,x \geq 0$› **by** *metis*
    ultimately have *?thesis* **by** *simp*
   }
   moreover
   {
    assume $\psi \in \mathit{set}\ \Phi$
    from ‹$\psi \in \mathit{set}\ \Phi$› **have** $\mathit{set}\ \Phi = \mathit{insert}\ \psi\ (\mathit{set}\ (\mathit{removeAll}\ \psi\ \Phi))$
     by *auto*
    with ‹$\mathit{set}\ \Phi \subseteq \mathit{set}\ (\psi\ \#\ \Psi)$› **have** $\mathit{set}\ (\mathit{removeAll}\ \psi\ \Phi) \subseteq \mathit{set}\ \Psi$
     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 $(\sum\varphi{\leftarrow}(\mathit{removeAll}\ \psi\ \Phi).\ f\ \varphi) \leq (\sum\psi{\leftarrow}\Psi.\ f\ \psi)$
     using *Cons.hyps*
     by *simp*
    moreover from ‹$\psi \in \mathit{set}\ \Phi$› ‹*distinct* Φ›
    have $(\sum\varphi{\leftarrow}\Phi.\ f\ \varphi) = f\ \psi + (\sum\varphi{\leftarrow}(\mathit{removeAll}\ \psi\ \Phi).\ f\ \varphi)$
     using *distinct-remove1-removeAll sum-list-map-remove1* **by** *fastforce*
    ultimately have *?thesis* **using** ‹$\forall\ x.\ f\,x \geq 0$›
     by *simp*
   }
   ultimately show *?thesis* **by** *blast*
  qed
 }
 thus *?case* **by** *blast*
qed
moreover assume $\mathit{set}\ \Phi \subseteq \mathit{set}\ \Psi$ **and** *distinct* Φ
ultimately show *?thesis* **by** *blast*
qed

lemma *count-remove-all-sum-list*:
 fixes $f :: {'a} \Rightarrow \mathit{real}$

shows *real* (*count-list xs x*) $*$ *f x* + ($\sum x'\leftarrow$(*removeAll x xs*). *f x'*) = ($\sum x\leftarrow xs$.
f x)
by (*induct xs*, *simp*, *simp*,
metis (*no-types*, *hide-lams*)
semiring-normalization-rules(*3*)
add.commute
add.left-commute)

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Exclusive-Implication-Completeness*:
($\forall$ $\delta \in$ *Dirac-Measures*. ($\sum \varphi\leftarrow\Phi$. δ φ) $\leq \delta$ ψ) = ($\vdash \coprod$ Φ $\wedge$ $\vdash \bigsqcup$ $\Phi \rightarrow \psi$)
proof −
{
fix δ
assume $\delta \in$ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* (λ φ. $\vdash \varphi$) ($\rightarrow$) $\bot$ δ
unfolding *Dirac-Measures-def*
by *simp*
assume $\vdash \coprod$ Φ $\vdash \bigsqcup$ $\Phi \rightarrow \psi$
hence ($\sum \varphi\leftarrow\Phi$. δ φ) $\leq \delta$ ψ
using *exclusive-sum-list-identity monotonicity* **by** *fastforce*
}
moreover
{
assume $\neg \vdash \coprod$ Φ
hence ($\exists$ $\varphi \in$ *set* Φ. $\exists$ $\psi \in$ *set* Φ. $\varphi \neq \psi \wedge \neg \vdash \sim (\varphi \sqcap \psi)$) $\vee$ ($\exists$ $\varphi \in$ *duplicates*
Φ. $\neg \vdash \sim \varphi$)
using *exclusive-equivalence set-deduction-base-theory* **by** *blast*
hence $\neg$ ($\forall$ $\delta \in$ *Dirac-Measures*. ($\sum \varphi\leftarrow\Phi$. δ φ) $\leq \delta$ ψ)
proof (*elim disjE*)
assume $\exists$ $\varphi \in$ *set* Φ. $\exists$ $\chi \in$ *set* Φ. $\varphi \neq \chi \wedge \neg \vdash \sim (\varphi \sqcap \chi)$
from *this* **obtain** φ **and** χ
where $\varphi\chi$*-properties*: $\varphi \in$ *set* Φ $\chi \in$ *set* Φ $\varphi \neq \chi$ $\neg \vdash \sim (\varphi \sqcap \chi)$
by *blast*
from *this* **obtain** Ω **where** Ω: *MCS* Ω $\sim (\varphi \sqcap \chi) \notin \Omega$
by (*meson insert-subset*
Formula-Consistent-def
Formula-Maximal-Consistency
Formula-Maximally-Consistent-Extension
Formula-Maximally-Consistent-Set-def
set-deduction-base-theory
set-deduction-reflection
set-deduction-theorem)
let *?*δ = λ χ. *if* $\chi\in\Omega$ *then* (*1* :: *real*) *else 0*
from Ω **have** $\varphi \in \Omega$ $\chi \in \Omega$
by (*metis Formula-Maximally-Consistent-Set-implication*
Maximally-Consistent-Set-def
conjunction-def
negation-def)+
with $\varphi\chi$*-properties* **have** ($\sum \varphi\leftarrow[\varphi, \chi]$. *?*$\delta$ φ) = *2*

set $[\varphi, \chi] \subseteq$ *set* Φ
distinct $[\varphi, \chi]$
$\forall \varphi.$ *?δ* $\varphi \geq 0$
by *simp+*
hence $(\sum \varphi \leftarrow \Phi.$ *?δ* $\varphi) \geq 2$ **using** *sum-list-monotone* **by** *metis*
hence $\neg (\sum \varphi \leftarrow \Phi.$ *?δ* $\varphi) \leq$ *?δ* (ψ) **by** *auto*
thus *?thesis*
using $\Omega(1)$ *MCS-Dirac-Measure*
by *auto*
next
assume $\exists\ \varphi \in$ *duplicates* $\Phi.\ \neg \vdash \sim \varphi$
from *this* **obtain** φ **where** φ: $\varphi \in$ *duplicates* $\Phi\ \neg \vdash \sim \varphi$
using *exclusive-equivalence* [**where** $\Gamma=\{\}$] *set-deduction-base-theory*
by *blast*
from φ **obtain** Ω **where** Ω: *MCS* $\Omega \sim \varphi \notin \Omega$
by (*meson insert-subset*
Formula-Consistent-def
Formula-Maximal-Consistency
Formula-Maximally-Consistent-Extension
Formula-Maximally-Consistent-Set-def
set-deduction-base-theory
set-deduction-reflection
set-deduction-theorem)
hence $\varphi \in \Omega$
using *negation-def* **by** *auto*
let *?δ* $= \lambda\ \chi.$ *if* $\chi \in \Omega$ *then* $(1 :: real)$ *else 0*
from φ **have** *count-list* $\Phi\ \varphi \geq 2$ **using** *duplicates-alt-def* [**where** $xs=\Phi$]
by *blast*
hence *real* (*count-list* $\Phi\ \varphi) *$ *?δ* $\varphi \geq 2$ **using** ‹$\varphi \in \Omega$› **by** *simp*
moreover
{
fix Ψ
have $(\sum \varphi \leftarrow \Psi.$ *?δ* $\varphi) \geq 0$ **by** (*induct* Ψ, *simp*, *simp*)
}
moreover have $(0::real) \leq (\sum a \leftarrow$ *removeAll* $\varphi\ \Phi.$ *if* $a \in \Omega$ *then 1 else 0*)
using ‹$\bigwedge\Psi.\ 0 \leq (\sum \varphi \leftarrow \Psi.$ *if* $\varphi \in \Omega$ *then 1 else 0*)› **by** *presburger*
ultimately have *real* (*count-list* $\Phi\ \varphi) *$ *?δ* $\varphi + (\sum\ \varphi \leftarrow$ (*removeAll* $\varphi\ \Phi$).
?δ $\varphi) \geq 2$
using ‹$2 \leq$ *real* (*count-list* $\Phi\ \varphi) *$ (*if* $\varphi \in \Omega$ *then 1 else 0*)› **by** *linarith*
hence $(\sum \varphi \leftarrow \Phi.$ *?δ* $\varphi) \geq 2$ **by** (*metis count-remove-all-sum-list*)
hence $\neg (\sum \varphi \leftarrow \Phi.$ *?δ* $\varphi) \leq$ *?δ* (ψ) **by** *auto*
thus *?thesis*
using $\Omega(1)$ *MCS-Dirac-Measure*
by *auto*
qed
}
moreover
{
assume $\neg \vdash \bigsqcup\ \Phi \to \psi$

```
    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
               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-Propositional-Logic) Exclusive-Implication-Completeness:
  (∀ Pr ∈ Logical-Probabilities. (∑φ←Φ. Pr φ) ≤ Pr ψ) = (⊢ ∐ Φ ∧ ⊢ ⨆ Φ
→ ψ)
  (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 $Pr \in$ *Logical-Probabilities*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *Logical-Probabilities-def*
by *simp*
show $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq Pr\ \psi$
using ‹*?rhs*›
exclusive-sum-list-identity
monotonicity
by *fastforce*
qed
qed

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Inequality-Completeness*:
$(\forall\ \delta \in Dirac\text{-}Measures.\ \delta\ \varphi \leq \delta\ \psi) = \vdash \varphi \rightarrow \psi$
proof −
have $\vdash \coprod\ [\varphi]$
by (*simp add*: *conjunction-right-elimination negation-def*)
hence $(\vdash \coprod\ [\varphi] \wedge\ \vdash \bigsqcup\ [\varphi] \rightarrow \psi) = \vdash \varphi \rightarrow \psi$
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** $\Phi=[\varphi]$]
by *auto*
qed

theorem (**in** *Classical-Propositional-Logic*) *Inequality-Completeness*:
$(\forall\ Pr \in Logical\text{-}Probabilities.\ Pr\ \varphi \leq Pr\ \psi) = \vdash \varphi \rightarrow \psi$
proof −
have $\vdash \coprod\ [\varphi]$
by (*simp add*: *conjunction-right-elimination negation-def*)
hence $(\vdash \coprod\ [\varphi] \wedge\ \vdash \bigsqcup\ [\varphi] \rightarrow \psi) = \vdash \varphi \rightarrow \psi$
by (*metis Arbitrary-Disjunction.simps*(*1*)
Arbitrary-Disjunction.simps(*2*)
disjunction-def implication-equivalence
negation-def
weak-biconditional-weaken)
thus *?thesis*
using *Exclusive-Implication-Completeness* [**where** $\Phi=[\varphi]$]
by *simp*
qed

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *Exclusive-List-Summation-Completeness*:
$(\forall\ Pr \in \textit{Logical-Probabilities}.\ Pr\ (\bigsqcup\ \Phi) = (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi)) = \vdash \coprod\ \Phi$
by (*metis antisym-conv*
Exclusive-Implication-Completeness
List-Summation-Completeness
trivial-implication)

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Exclusive-Set-Summation-Completeness*:
$(\forall\ \delta \in \textit{Dirac-Measures}.\ \delta\ (\bigsqcup\ \Phi) = (\sum \varphi \in \textit{set}\ \Phi.\ \delta\ \varphi)) = \vdash \coprod\ (\textit{remdups}\ \Phi)$
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-Propositional-Logic*) *Exclusive-Set-Summation-Completeness*:
$(\forall\ Pr \in \textit{Logical-Probabilities}.\ Pr\ (\bigsqcup\ \Phi) = (\sum \varphi \in \textit{set}\ \Phi.\ Pr\ \varphi)) = \vdash \coprod\ (\textit{remdups}$
$\Phi)$
by (*metis* (*mono-tags*, *hide-lams*)
eq-iff
Exclusive-Implication-Completeness
Set-Summation-Completeness
trivial-implication
set-remdups
sum.set-conv-list)

lemma (**in** *Logical-Probability*) *exclusive-list-set-inequality*:
assumes $\vdash \coprod\ \Phi$
shows $(\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) = (\sum \varphi{\in}\textit{set}\ \Phi.\ Pr\ \varphi)$
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 $(\forall \varphi \in \textit{duplicates}\ \Phi.\ \vdash \sim \varphi)$
$\land (\forall\ \varphi \in \textit{set}\ \Phi.\ \forall\ \psi \in \textit{set}\ \Phi.\ (\varphi \neq \psi) \longrightarrow \vdash \sim (\varphi \sqcap \psi))$
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
theory *Logical-Probability-Completeness*
imports *Logical-Probability*
begin

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

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

abbreviation (**in** *Classical-Propositional-Logic*) *map-negation* :: $'a\ list \Rightarrow 'a\ list$ $(\boldsymbol{\sim})$
where $\boldsymbol{\sim}\ \Phi \equiv map \sim \Phi$

lemma (**in** *Classical-Propositional-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*
by (*simp add*: *biconditional-def negation-def The-Principle-of-Pseudo-Scotus*)
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 $\vdash$ $(\!|$ $((\langle\sim \Phi :\rightarrow \sim \varphi\rangle \leftrightarrow (\langle\varphi\rangle \rightarrow \langle\bigsqcup \Phi\rangle)) \rightarrow$
$(\sim \langle\psi\rangle \rightarrow \langle\sim \Phi :\rightarrow \sim \varphi\rangle) \leftrightarrow (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle))$ $|\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis*
by *simp*
qed
with *Cons* **show** *?case*
by (*metis list.simps*(*9*)
Arbitrary-Disjunction.simps(*2*)
Modus-Ponens
list-implication.simps(*2*))
qed

lemma (**in** *Classical-Propositional-Logic*) *conjunction-monotonic-identity*:
$\vdash (\varphi \rightarrow \psi) \rightarrow (\varphi \sqcap \chi) \rightarrow (\psi \sqcap \chi)$
unfolding *conjunction-def*
using *Modus-Ponens*
flip-hypothetical-syllogism
by *blast*

lemma (**in** *Classical-Propositional-Logic*) *conjunction-monotonic*:
assumes $\vdash \varphi \rightarrow \psi$
shows $\vdash (\varphi \sqcap \chi) \rightarrow (\psi \sqcap \chi)$
using *assms*
Modus-Ponens
conjunction-monotonic-identity
by *blast*

lemma (**in** *Classical-Propositional-Logic*) *disjunction-monotonic-identity*:
$\vdash (\varphi \rightarrow \psi) \rightarrow (\varphi \sqcup \chi) \rightarrow (\psi \sqcup \chi)$
unfolding *disjunction-def*
using *Modus-Ponens*
flip-hypothetical-syllogism
by *blast*

lemma (**in** *Classical-Propositional-Logic*) *disjunction-monotonic*:
assumes $\vdash \varphi \rightarrow \psi$
shows $\vdash (\varphi \sqcup \chi) \rightarrow (\psi \sqcup \chi)$
using *assms*
Modus-Ponens
disjunction-monotonic-identity
by *blast*

lemma (**in** *Classical-Propositional-Logic*) *conj-dnf-distribute*:
$\vdash \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ \Lambda) \leftrightarrow (\varphi \sqcap \bigsqcup\ (map \bigsqcap\ \Lambda))$
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 $(Cons\ \Psi\ \Lambda)$
assume $\vdash \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ \Lambda) \leftrightarrow (\varphi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Lambda))$
(**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 $?\varphi = (\langle ?A\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle ?B\rangle)) \rightarrow (((\langle\varphi\rangle \sqcap \langle\bigsqcap\ \Psi\rangle) \sqcup \langle ?A\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcap$
$\Psi\rangle \sqcup \langle ?B\rangle))$
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 $\vdash ((\varphi \sqcap \bigsqcap\ \Psi) \sqcup\ ?A) \leftrightarrow (\varphi \sqcap \bigsqcap\ \Psi \sqcup\ ?B)$
using *Modus-Ponens*
by *blast*
moreover
have $map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ \Lambda = map\ (\lambda\Psi.\ \varphi \sqcap \bigsqcap\ \Psi)\ \Lambda$ **by** *simp*
ultimately show *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *append-dnf-distribute*:
$\vdash \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \Psi.\ \Phi\ @\ \Psi))\ \Lambda) \leftrightarrow (\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Lambda))$
proof(*induct* Φ)
case *Nil*
have $\vdash \bigsqcup\ (map\ \bigsqcap\ \Lambda) \leftrightarrow (\top \sqcap \bigsqcup\ (map\ \bigsqcap\ \Lambda))$
(**is** $\vdash\ ?A \leftrightarrow (\top \sqcap\ ?A)$)
proof $-$
let $?\varphi = \langle ?A\rangle \leftrightarrow ((\bot \rightarrow \bot) \sqcap \langle ?A\rangle)$
have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *simp*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis*
unfolding *verum-def*
by *simp*
qed
then show *?case* **by** *simp*
next
case $(Cons\ \varphi\ \Phi)$
have $\vdash \bigsqcup\ (map\ (\bigsqcap \circ (@)\ \Phi)\ \Lambda) \leftrightarrow (\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Lambda))$
$= \vdash \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ \Phi)\ \Lambda)) \leftrightarrow (\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Lambda))$
by *simp*
with *Cons* **have** $\vdash \bigsqcup\ (map\ \bigsqcap\ (map\ (\lambda\ \Psi.\ \Phi\ @\ \Psi)\ \Lambda)) \leftrightarrow (\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap$

Λ))
  (**is** $\vdash \bigsqcup$ (*map* $\bigsqcap$ *?A*) $\leftrightarrow$ (*?B* $\sqcap$ *?C*))
  by *meson*
 moreover have $\vdash \bigsqcup$ (*map* $\bigsqcap$ *?A*) $\leftrightarrow$ (*?B* $\sqcap$ *?C*)
   $\to$ ($\bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) *?A*) $\leftrightarrow$ ($\varphi \sqcap \bigsqcup$ (*map* $\bigsqcap$ *?A*)))
   $\to \bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) *?A*) $\leftrightarrow$ (($\varphi \sqcap$ *?B*) $\sqcap$ *?C*)
 proof −
  let $?\varphi = \langle\bigsqcup$ (*map* $\bigsqcap$ *?A*)$\rangle \leftrightarrow (\langle ?B\rangle \sqcap \langle ?C\rangle)$
   $\to (\langle\bigsqcup$ (*map* ($\bigsqcap \circ (\lambda\ \varphi s.\ \varphi \# \varphi s)$) *?A*)$\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcup$ (*map* $\bigsqcap$ *?A*)$\rangle))$
   $\to \langle\bigsqcup$ (*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

primrec (**in** *Classical-Propositional-Logic*)
 segmented-deduction :: *'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *bool* (- \$⊢ - [*60*,*100*] *60*)
 where
  Γ \$⊢ [] = *True*
 | Γ \$⊢ ($\varphi \# \Phi$) = ($\exists\ \Psi$. *mset* (*map snd* Ψ) $\subseteq\#$ *mset* $\Gamma \wedge$
   map (*uncurry* ($\sqcup$)) Ψ :⊢ $\varphi \wedge$
   map (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ) \$⊢ Φ)

definition (**in** *Minimal-Logic*)
 stronger-theory-relation :: *'a list* $\Rightarrow$ *'a list* $\Rightarrow$ *bool* (**infix** $\preceq$ *100*)
 where
  $\Sigma \preceq \Gamma = (\exists\ \Phi.$ *map snd* $\Phi = \Sigma \wedge$
   mset (*map fst* Φ) $\subseteq\#$ *mset* $\Gamma \wedge$
   $(\forall\ (\gamma,\sigma) \in$ *set* $\Phi.\ \vdash \gamma \to \sigma))$

abbreviation (**in** *Minimal-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 Minimal-Logic) msub-stronger-theory-intro:
  assumes mset Σ ⊆# mset Γ
  shows Σ ≼ Γ
proof −
  let ?ΔΣ = map (λ x. (x,x)) Σ
  have map snd ?ΔΣ = Σ
    by (induct Σ, simp, simp)
  moreover have map fst ?ΔΣ = Σ
    by (induct Σ, simp, simp)
  hence mset (map fst ?ΔΣ) ⊆# mset Γ
    using assms by simp
  moreover have ∀ (γ,σ) ∈ set ?ΔΣ. ⊢ γ → σ
    by (induct Σ, simp, simp,
        metis list-implication.simps(1) list-implication-Axiom-1)
  ultimately show ?thesis using stronger-theory-relation-def by (simp, blast)
qed

lemma (in Minimal-Logic) stronger-theory-reflexive [simp]: Γ ≼ Γ
  using msub-stronger-theory-intro by auto

lemma (in Minimal-Logic) weakest-theory [simp]: [] ≼ Γ
  using msub-stronger-theory-intro by auto

lemma (in Minimal-Logic) stronger-theory-empty-list-intro [simp]:
  assumes Γ ≼ []
  shows Γ = []
  using assms stronger-theory-relation-def by simp

lemma (in Minimal-Logic) stronger-theory-right-permutation:
  assumes Γ <~~> Δ
      and Σ ≼ Γ
    shows Σ ≼ Δ
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 Minimal-Logic) stronger-theory-left-permutation:
  assumes Σ <~~> Δ
      and Σ ≼ Γ
    shows Δ ≼ Γ
proof −
  have ∀ Σ Γ. Σ <~~> Δ ⟶ Σ ≼ Γ ⟶ Δ ≼ Γ
  proof (induct Δ)
    case Nil
    then show ?case by simp
```

```
next
  case (Cons δ Δ)
  {
    fix Σ Γ
    assume Σ <~~> (δ # Δ) Σ ≼ Γ
    from this obtain Φ where Φ:
      map snd Φ = Σ
      mset (map fst Φ) ⊆# mset Γ
      ∀ (γ,δ) ∈ set Φ. ⊢ γ → δ
      using stronger-theory-relation-def by fastforce
    with ‹Σ <~~> (δ # Δ)› have δ ∈# mset (map snd Φ)
      by (simp add: perm-set-eq)
    from this obtain γ where γ: (γ, δ) ∈# mset Φ
      by (induct Φ, fastforce+)
    let ?Φ0 = remove1 (γ, δ) Φ
    let ?Σ0 = map snd ?Φ0
    from γ Φ(2) have mset (map fst ?Φ0) ⊆# mset (remove1 γ Γ)
      by (metis ex-mset
                listSubtract-monotonic
                listSubtract-mset-homomorphism
                mset-remove1
                remove1-pairs-list-projections-fst)
    moreover have mset ?Φ0 ⊆# mset Φ by simp
    with Φ(3) have ∀ (γ,δ) ∈ set ?Φ0. ⊢ γ → δ by fastforce
    ultimately have ?Σ0 ≼ remove1 γ Γ
      unfolding stronger-theory-relation-def by blast
    moreover have Δ <~~> (remove1 δ Σ) using ‹Σ <~~> (δ # Δ)›
      by (metis perm-remove-perm perm-sym remove-hd)
    moreover from γ Φ(1) have mset ?Σ0 = 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) $\preceq \Gamma$
using *stronger-theory-right-permutation* **by** *blast*
}
then show *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma (**in** *Minimal-Logic*) *stronger-theory-transitive*:
assumes $\Sigma \preceq \Delta$ **and** $\Delta \preceq \Gamma$
shows $\Sigma \preceq \Gamma$
proof –
have $\forall\ \Delta\ \Gamma.\ \Sigma \preceq \Delta \longrightarrow \Delta \preceq \Gamma \longrightarrow \Sigma \preceq \Gamma$
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) $\preceq \Delta$ $\Delta \preceq \Gamma$
from *this* **obtain** Φ **where** Φ:
map snd $\Phi = \sigma$ # Σ
mset (*map fst* Φ) $\subseteq$# *mset* Δ
$\forall$ (δ,σ) $\in$ *set* Φ. $\vdash \delta \to \sigma$
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let *?*δ = *fst* (*hd* Φ)
from Φ(*1*) **have** $\Phi \neq$ [] **by** (*induct* Φ, *simp+*)
hence *?*δ $\in$# *mset* (*map fst* Φ) **by** (*induct* Φ, *simp+*)
with Φ(*2*) **have** *?*δ $\in$# *mset* Δ **by** (*meson mset-subset-eqD*)
with ‹$\Phi \neq$ []› Φ(*2*) **have** *mset* (*map fst* (*remove1* (*hd* Φ) Φ)) $\subseteq$# *mset* (*remove1* *?*δ Δ)
by (*simp*,
metis diff-single-eq-union
hd-in-set
image-mset-add-mset
insert-subset-eq-iff
set-mset-mset)
moreover from ‹$\Phi \neq$ []› **have** *remove1* (*hd* Φ) Φ = *tl* Φ **by** (*induct* Φ, *simp+*)
moreover from Φ(*1*) **have** *map snd* (*tl* Φ) = Σ
by (*simp add*: *map-tl*)
moreover from Φ(*3*) **have** $\forall$ (δ,σ) $\in$ *set* (*tl* Φ). $\vdash \delta \to \sigma$
by (*simp add*: ‹$\Phi \neq$ []› *list.set-sel*(*2*))
ultimately have $\Sigma \preceq$ *remove1* *?*δ Δ
using *stronger-theory-relation-def* **by** *auto*

from ‹$?\delta \in\# $ *mset* Δ› **have** $?\delta \ \# \ ($*remove1* $?\delta \ \Delta) <\sim\sim> \Delta$
  by (*simp add*: *perm-remove perm-sym*)
with ‹$\Delta \preceq \Gamma$› **have** $(?\delta \ \# \ ($*remove1* $?\delta \ \Delta)) \preceq \Gamma$
  using *stronger-theory-left-permutation perm-sym* **by** *blast*
from *this* **obtain** Ψ **where** Ψ:
  map snd $\Psi = (?\delta \ \# \ ($*remove1* $?\delta \ \Delta))$
  mset (*map fst* Ψ) $\subseteq\#$ *mset* Γ
  $\forall \ (\gamma,\delta) \in$ *set* $\Psi. \vdash \gamma \to \delta$
  using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let $?\gamma = $ *fst* (*hd* Ψ)
from $\Psi(1)$ **have** $\Psi \neq []$ **by** (*induct* Ψ, *simp+*)
hence $?\gamma \in\#$ *mset* (*map fst* Ψ) **by** (*induct* Ψ, *simp+*)
with $\Psi(2)$ **have** $?\gamma \in\#$ *mset* Γ **by** (*meson mset-subset-eqD*)
with ‹$\Psi \neq []$› $\Psi(2)$ **have** *mset* (*map fst* (*remove1* (*hd* Ψ) Ψ)) $\subseteq\#$ *mset* (*remove1* $?\gamma \ \Gamma$)
  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) $\Psi =$ *tl* Ψ **by** (*induct* Ψ, *simp+*)
moreover from $\Psi(1)$ **have** *map snd* (*tl* Ψ) $=$ (*remove1* $?\delta \ \Delta$)
  by (*simp add*: *map-tl*)
moreover from $\Psi(3)$ **have** $\forall \ (\gamma,\delta) \in$ *set* (*tl* Ψ). $\vdash \gamma \to \delta$
  by (*simp add*: ‹$\Psi \neq []$› *list.set-sel*(*2*))
ultimately have *remove1* $?\delta \ \Delta \preceq$ *remove1* $?\gamma \ \Gamma$
  using *stronger-theory-relation-def* **by** *auto*
with ‹$\Sigma \preceq$ *remove1* $?\delta \ \Delta$› *Cons.hyps* **have** $\Sigma \preceq$ *remove1* $?\gamma \ \Gamma$
  by *blast*
from *this* **obtain** Ω_0 **where** Ω_0:
  map snd $\Omega_0 = \Sigma$
  mset (*map fst* Ω_0) $\subseteq\#$ *mset* (*remove1* $?\gamma \ \Gamma$)
  $\forall \ (\gamma,\sigma) \in$ *set* $\Omega_0. \vdash \gamma \to \sigma$
  using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let $?\Omega = (?\gamma, \sigma) \ \# \ \Omega_0$
from $\Omega_0(1)$ **have** *map snd* $?\Omega = \sigma \ \# \ \Sigma$ **by** *simp*
moreover from $\Omega_0(2)$ **have** *mset* (*map fst* $?\Omega$) $\subseteq\#$ *mset* ($?\gamma \ \#$ (*remove1* $?\gamma \ \Gamma$))
  by *simp*
moreover from $\Phi(1)$ $\Psi(1)$ **have** $\sigma =$ *snd* (*hd* Φ) $?\delta =$ *snd* (*hd* Ψ) **by** *fastforce+*
with $\Phi(3)$ $\Psi(3)$ ‹$\Phi \neq []$› ‹$\Psi \neq []$› *hd-in-set* **have** $\vdash ?\delta \to \sigma \vdash ?\gamma \to ?\delta$
  by *fastforce+*
hence $\vdash ?\gamma \to \sigma$ **using** *Modus-Ponens hypothetical-syllogism* **by** *blast*
with $\Omega_0(3)$ **have** $\forall \ (\gamma,\sigma) \in$ *set* $?\Omega. \vdash \gamma \to \sigma$
  by *auto*
ultimately have $(\sigma \ \# \ \Sigma) \preceq (?\gamma \ \#$ (*remove1* $?\gamma \ \Gamma$))

```
      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 Minimal-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
          listSubtract-monotonic
          listSubtract-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 $\Phi(2)$ **have** $\gamma \in\# \; mset\ \Gamma$
using *mset-subset-eqD* **by** *fastforce*
ultimately show $\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \wedge (remove1\ \sigma\ \Sigma) \preceq (remove1\ \gamma\ \Gamma)$ **by** *auto*
next
assume $\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \wedge (remove1\ \sigma\ \Sigma) \preceq (remove1\ \gamma\ \Gamma)$
from *this* **obtain** Φ γ **where** γ: $\gamma \in set\ \Gamma \vdash \gamma \to \sigma$
and Φ: $map\ snd\ \Phi = (remove1\ \sigma\ \Sigma)$
$mset\ (map\ fst\ \Phi) \subseteq\# \; mset\ (remove1\ \gamma\ \Gamma)$
$\forall\ (\gamma,\sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma$
unfolding *stronger-theory-relation-def* **by** *blast*
let $?\Phi = (\gamma,\ \sigma)\ \#\ \Phi$
from $\Phi(1)$ **have** $map\ snd\ ?\Phi = \sigma\ \#\ (remove1\ \sigma\ \Sigma)$ **by** *simp*
moreover from $\Phi(2)$ $\gamma(1)$ **have** $mset\ (map\ fst\ ?\Phi) \subseteq\# \; mset\ \Gamma$
by (*simp add*: *insert-subset-eq-iff*)
moreover from $\Phi(3)$ $\gamma(2)$ **have** $\forall\ (\gamma,\sigma) \in set\ ?\Phi.\ \vdash \gamma \to \sigma$
by *auto*
ultimately have $(\sigma\ \#\ (remove1\ \sigma\ \Sigma)) \preceq \Gamma$
unfolding *stronger-theory-relation-def* **by** *metis*
moreover from *assms* **have** $\sigma\ \#\ (remove1\ \sigma\ \Sigma)\ <^{\sim\sim}>\ \Sigma$
by (*simp add*: *perm-remove perm-sym*)
ultimately show $\Sigma \preceq \Gamma$
using *stronger-theory-left-permutation* **by** *blast*
qed

lemma (**in** *Minimal-Logic*) *stronger-theory-cons-witness*:
$(\sigma\ \#\ \Sigma) \preceq \Gamma = (\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \wedge \Sigma \preceq (remove1\ \gamma\ \Gamma))$
proof −
have $\sigma \in\# \; mset\ (\sigma\ \#\ \Sigma)$ **by** *simp*
hence $(\sigma\ \#\ \Sigma) \preceq \Gamma = (\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \wedge (remove1\ \sigma\ (\sigma\ \#\ \Sigma)) \preceq (remove1\ \gamma\ \Gamma))$
by (*meson list.set-intros*(*1*) *stronger-theory-witness*)
thus *?thesis* **by** *simp*
qed

lemma (**in** *Minimal-Logic*) *stronger-theory-left-cons*:
assumes $(\sigma\ \#\ \Sigma) \preceq \Gamma$
shows $\Sigma \preceq \Gamma$
proof −
from *assms* **obtain** Φ **where** Φ:
$map\ snd\ \Phi = \sigma\ \#\ \Sigma$
$mset\ (map\ fst\ \Phi) \subseteq\# \; mset\ \Gamma$
$\forall\ (\delta,\sigma) \in set\ \Phi.\ \vdash \delta \to \sigma$
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let $?\Phi' = remove1\ (hd\ \Phi)\ \Phi$
from $\Phi(1)$ **have** $map\ snd\ ?\Phi' = \Sigma$ **by** (*induct* Φ, *simp+*)
moreover from $\Phi(2)$ **have** $mset\ (map\ fst\ ?\Phi') \subseteq\# \; mset\ \Gamma$
by (*metis diff-subset-eq-self*
listSubtract.simps(*1*)

listSubtract.simps(*2*)
listSubtract-mset-homomorphism
map-monotonic
subset-mset.dual-order.trans)
moreover from $\Phi(3)$ **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** *Minimal-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*
listSubtract.simps(*2*)
listSubtract-mset-homomorphism
mset-zero-iff remove1.simps(*1*))
with $\Phi(1)$ $\Phi(3)$ **show** *?thesis*
unfolding *stronger-theory-relation-def*
by *auto*
qed

lemma (**in** *Minimal-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** *Minimal-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 $\exists\,\Phi.\ mset\ (map\ snd\ \Phi) = mset\ \Sigma\ \wedge$
$mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ (\gamma\ \#\ \Gamma)\ \wedge$
$(\forall\,(\gamma, \sigma){\in}set\ \Phi.\ \vdash \gamma \to \sigma)$
from *this* **obtain** Φ **where** Φ:
$mset\ (map\ snd\ \Phi) = mset\ \Sigma$
$mset\ (map\ fst\ \Phi) \subseteq\#\ mset\ (\gamma\ \#\ \Gamma)$
$\forall\,(\gamma, \sigma){\in}set\ \Phi.\ \vdash \gamma \to \sigma$
by *metis*
show $\Sigma \preceq (\gamma\ \#\ \Gamma)$
proof (*cases* $\exists\ \sigma.\ (\gamma, \sigma) \in set\ \Phi$)
assume $\exists\ \sigma.\ (\gamma, \sigma) \in set\ \Phi$
from *this* **obtain** σ **where** σ: $(\gamma, \sigma) \in set\ \Phi$ **by** *auto*
let $?\Phi = remove1\ (\gamma, \sigma)\ \Phi$
from σ **have** $mset\ (map\ snd\ ?\Phi) = mset\ (remove1\ \sigma\ \Sigma)$
using Φ(*1*) *remove1-pairs-list-projections-snd* **by** *force+*
moreover
from σ **have** $mset\ (map\ fst\ ?\Phi) = mset\ (remove1\ \gamma\ (map\ fst\ \Phi))$
using Φ(*1*) *remove1-pairs-list-projections-fst* **by** *force+*
with Φ(*2*) **have** $mset\ (map\ fst\ ?\Phi) \subseteq\#\ mset\ \Gamma$

```
      by (simp add: subset-eq-diff-conv)
    moreover from Φ(3) have ∀(γ, σ)∈set ?Φ. ⊢ γ → σ
      by fastforce
    ultimately have remove1 σ Σ ≼ Γ using Cons by blast
    from this obtain Ψ where Ψ:
      map snd Ψ = remove1 σ Σ
      mset (map fst Ψ) ⊆# mset Γ
      ∀(γ, σ)∈set Ψ. ⊢ γ → σ
      unfolding stronger-theory-relation-def
      by blast
    let ?Ψ = (γ, σ) # Ψ
    from Ψ have map snd ?Ψ = σ # (remove1 σ Σ)
             mset (map fst ?Ψ) ⊆# mset (γ # Γ)
      by simp+
    moreover from Φ(3) σ have ⊢ γ → σ by auto
    with Ψ(3) have ∀(γ, σ)∈set ?Ψ. ⊢ γ → σ by auto
    ultimately have (σ # (remove1 σ Σ)) ≼ (γ # Γ)
      unfolding stronger-theory-relation-def
      by metis
    moreover
    have σ ∈ set Σ
      by (metis Φ(1) σ set-mset-mset set-zip-rightD zip-map-fst-snd)
    hence Σ <~~> σ # (remove1 σ Σ)
       by (simp add: perm-remove)
    hence Σ ≼ (σ # (remove1 σ Σ))
      using stronger-theory-reflexive
           stronger-theory-right-permutation
      by blast
    ultimately show ?thesis
      using stronger-theory-transitive
      by blast
  next
    assume ∄σ. (γ, σ) ∈ set Φ
    hence γ ∉ set (map fst Φ) by fastforce
    with Φ(2) have mset (map fst Φ) ⊆# mset Γ
      by (metis diff-single-trivial
               in-multiset-in-set
               insert-DiffM2
               mset-remove1
               remove-hd
               subset-eq-diff-conv)
    hence Σ ≼ Γ
      using Cons Φ(1) Φ(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 Minimal-Logic) stronger-theory-deduction-monotonic:
  assumes Σ ≼ Γ
      and Σ :⊢ φ
    shows Γ :⊢ φ
using assms
proof −
  have ∀ φ. Σ ≼ Γ ⟶ Σ :⊢ φ ⟶ Γ :⊢ φ
  proof (induct Σ)
    case Nil
    then show ?case
      by (simp add: list-deduction-weaken)
  next
    case (Cons σ Σ)
    {
      fix φ
      assume (σ # Σ) ≼ Γ (σ # Σ) :⊢ φ
      hence Σ :⊢ σ → φ Σ ≼ Γ
        using list-deduction-theorem
              stronger-theory-left-cons
        by (blast, metis)
      with Cons have Γ :⊢ σ → φ 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-Propositional-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› listSubtract-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-Propositional-Logic) segmented-stronger-theory-intro:
  assumes Γ ⪰ Σ
  shows Γ $⊢ Σ
proof −
  have ∀ Γ. Σ ⪯ Γ ⟶ Γ $⊢ Σ
  proof (induct Σ)
    case Nil
    then show ?case by fastforce
  next
    case (Cons σ Σ)
    {
      fix Γ
      assume (σ # Σ) ⪯ Γ
      from this obtain γ where γ: γ ∈ set Γ ⊢ γ → σ Σ ⪯ (remove1 γ Γ)
        using stronger-theory-cons-witness by blast
```

let *?Φ* = [(γ,γ)]
from γ *Cons* **have** (*remove1* γ Γ) $⊢ Σ **by** *blast*
moreover have *mset* (*remove1* γ Γ) ⊆# *mset* (*map* (*uncurry* (→)) *?Φ* @ Γ ⊖ (*map snd ?Φ*))
by *simp*
ultimately have *map* (*uncurry* (→)) *?Φ* @ Γ ⊖ (*map snd ?Φ*) $⊢ Σ
using *segmented-msub-left-monotonic* **by** *blast*
moreover have *map* (*uncurry* (⊔)) *?Φ* :⊢ σ
by (*simp*, *metis* γ(*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 γ(*1*) **have** *mset* (*map snd ?Φ*) ⊆# *mset* Γ **by** *simp*
ultimately have Γ $⊢ (σ # Σ)
using *segmented-deduction.simps*(*2*) **by** *blast*
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *witness-weaker-theory*:
assumes *mset* (*map snd* Σ) ⊆# *mset* Γ
shows *map* (*uncurry* (⊔)) Σ ⪯ Γ
proof −
have ∀ Γ. *mset* (*map snd* Σ) ⊆# *mset* Γ ⟶ *map* (*uncurry* (⊔)) Σ ⪯ Γ
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
{
fix Γ
assume *mset* (*map snd* (σ # Σ)) ⊆# *mset* Γ
hence *mset* (*map snd* Σ) ⊆# *mset* (*remove1* (*snd* σ) Γ)
by (*simp add*: *insert-subset-eq-iff*)
with *Cons* **have** *map* (*uncurry* (⊔)) Σ ⪯ *remove1* (*snd* σ) Γ **by** *blast*
moreover have *uncurry* (⊔) = (λ σ. *fst* σ ⊔ *snd* σ) **by** *fastforce*
hence *uncurry* (⊔) σ = *fst* σ ⊔ *snd* σ **by** *simp*
moreover have ⊢ *snd* σ → (*fst* σ ⊔ *snd* σ)
unfolding *disjunction-def*
by (*simp add*: *Axiom-1*)
ultimately have *map* (*uncurry* (⊔)) (σ # Σ) ⪯ (*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)) ⊆# mset Γ›
        by (simp, meson insert-DiffM mset-subset-eq-insertD)
      ultimately have map (uncurry (⊔)) (σ # Σ) ≼ 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-Propositional-Logic) segmented-deduction-one-collapse:
  Γ $⊢ [φ] = Γ :⊢ φ
proof (rule iffI)
  assume Γ $⊢ [φ]
  from this obtain Σ where
    Σ: mset (map snd Σ) ⊆# mset Γ
       map (uncurry (⊔)) Σ :⊢ φ
    by auto
  hence map (uncurry (⊔)) Σ ≼ Γ
    using witness-weaker-theory by blast
  thus Γ :⊢ φ using Σ(2)
    using stronger-theory-deduction-monotonic by blast
next
  assume Γ :⊢ φ
  let ?Σ = map (λ γ. (⊥, γ)) Γ
  have Γ ≼ map (uncurry (⊔)) ?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 Minimal-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) ⊆# mset Δ
          using Σ(2) mset.simps(1) subset-eq-diff-conv by force
        moreover from Σ(3) have ∀ (δ,φ) ∈ set ?Σ. ⊢ δ → φ by auto
        ultimately have remove1 φ Φ ≼ Δ
          unfolding stronger-theory-relation-alt-def by blast
        hence (remove1 φ Φ @ Ψ) ≼ (Δ @ Γ) using Cons by auto
        from this obtain Ω where Ω:
          map snd Ω = (remove1 φ Φ) @ Ψ
          mset (map fst Ω) ⊆# mset (Δ @ Γ)
          ∀ (α,β) ∈ set Ω. ⊢ α → β
          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) ⊆# mset ((δ # Δ) @ Γ)
          using Ω(2) by simp
        moreover have ⊢ δ → φ
```

```
        using Σ(3) φ by blast
      hence ∀ (α,β) ∈ set ?Ω. ⊢ α → β using Ω(3) by auto
      ultimately have (φ # remove1 φ Φ @ Ψ) ≼ ((δ # Δ) @ Γ)
        by (metis stronger-theory-relation-def)
      moreover have φ ∈ set Φ
        using Σ(1) φ by force
      hence (φ # remove1 φ Φ) <~~> Φ
        by (simp add: perm-remove perm-sym)
      hence (φ # remove1 φ Φ @ Ψ) <~~> Φ @ Ψ
        by (metis append-Cons perm-append2)
      ultimately show ?thesis
        using stronger-theory-left-permutation by blast
    next
      assume ∄φ. (δ, φ) ∈ set Σ
      hence δ ∉ set (map fst Σ)
            mset Δ + add-mset δ (mset []) = mset (δ # Δ)
        by auto
      hence mset (map fst Σ) ⊆# mset Δ
        by (metis (no-types) ‹mset (map fst Σ) ⊆# mset (δ # Δ)›
                          diff-single-trivial
                          mset.simps(1)
                          set-mset-mset
                          subset-eq-diff-conv)
      with Σ(1) Σ(3) have Φ ≼ Δ
        unfolding stronger-theory-relation-def
        by blast
      hence (Φ @ Ψ) ≼ (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-Propositional-Logic) segmented-empty-deduction:
 [] $⊢ Φ = (∀ φ ∈ set Φ. ⊢ φ)
 by (induct Φ, simp, rule iffI, fastforce+)

lemma (in Classical-Propositional-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 ?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** $\forall\ (\varphi,\chi) \in set\ ?\Phi'.\ \vdash \varphi \to \chi$ **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 $map\ (uncurry\ (\sqcup))\ ?\Theta \mathrel{:}\vdash \varphi$
using $\Psi(2)$
stronger-theory-deduction-monotonic
[**where** $\Sigma{=}map\ (uncurry\ (\sqcup))\ \Psi$
and $\Gamma{=}map\ (uncurry\ (\sqcup))\ ?\Theta$
and $\varphi{=}\varphi$]
by *metis*
moreover have
$(map\ (uncurry\ (\to))\ \Psi\ @\ \Sigma \ominus (map\ snd\ \Psi)) \preceq$
$(map\ (uncurry\ (\to))\ ?\Theta\ @\ \Gamma \ominus (map\ snd\ ?\Theta))$
proof –
have $map\ (uncurry\ (\to))\ \Psi \preceq map\ (uncurry\ (\to))\ ?\Theta$
proof –
let $?\Phi = map\ (\lambda\ (\psi, \sigma, \gamma).\ (\psi \to \gamma,\ \psi \to \sigma))\ \Omega$
have $map\ snd\ ?\Phi = map\ (uncurry\ (\to))\ \Psi$
using $\Omega(1)$ **by** *fastforce*
moreover have $map\ fst\ ?\Phi = map\ (uncurry\ (\to))\ ?\Theta$
by *fastforce*
hence $mset\ (map\ fst\ ?\Phi) \subseteq\#\ mset\ (map\ (uncurry\ (\to))\ ?\Theta)$
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)›
                listSubtract-monotonic
                map-listSubtract-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-listSubtract-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-Propositional-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
    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
```

```
    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 (⊔) = (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
    }
    thus ?case by blast
  qed
```

with $\Psi(2)$ $\Delta(2)$ **have** $\sim$ (*map* (*uncurry* (\)) *?*Ψ) :$\vdash$ φ
using *stronger-theory-deduction-monotonic* **by** *blast*
moreover
have (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\sim$ $\Gamma \ominus$ *map snd* Ψ) $\preceq$
$\sim$ (*map* (*uncurry* ($\sqcap$)) *?*Ψ @ $\Gamma \ominus$ (*map fst ?*Ψ))
proof −
from $\Delta(1)$ **have** *mset* ($\sim$ $\Gamma \ominus \sim$ Δ) = *mset* ($\sim$ ($\Gamma \ominus \Delta$))
by (*simp add*: *image-mset-Diff*)
hence *mset* ($\sim$ $\Gamma \ominus$ *map snd* Ψ) = *mset* ($\sim$ ($\Gamma \ominus$ *map fst ?*Ψ))
using $\Psi(1)$ $\Delta(2)$ ‹*map fst ?*$\Psi = \Delta$› **by** *simp*
hence ($\sim$ $\Gamma \ominus$ *map snd* Ψ) $\preceq$ $\sim$ ($\Gamma \ominus$ *map fst ?*Ψ)
by (*simp add*: *msub-stronger-theory-intro*)
moreover have $\forall$ Δ. *map snd* Ψ = $\sim$ $\Delta \longrightarrow$
map (*uncurry* ($\rightarrow$)) $\Psi \preceq \sim$ (*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: $\sim \gamma$ = *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* ($\rightarrow$)) $\Psi \preceq \sim$ (*map* (*uncurry* ($\sqcap$)) (*zip* (*tl* Δ) (*map fst* Ψ)))
by *simp*
moreover
{
fix ψ γ
have $\vdash \sim(\gamma \sqcap \psi) \rightarrow (\psi \rightarrow \sim \gamma)$
unfolding *disjunction-def*
conjunction-def
negation-def
by (*meson Modus-Ponens*
flip-implication
hypothetical-syllogism)
} **note** *tautology* = *this*
have (*uncurry* ($\rightarrow$)) = (λ ψ. (*fst* ψ) $\rightarrow$ (*snd* ψ))
by *fastforce*
with γ **have** *uncurry* ($\rightarrow$) ψ = *?*$\psi \rightarrow \sim \gamma$
by *simp*
with *tautology* **have** $\vdash \sim(\gamma \sqcap ?\psi) \rightarrow$ (*uncurry* ($\rightarrow$)) ψ
by *simp*
ultimately have *map* (*uncurry* ($\rightarrow$)) (ψ # Ψ) $\preceq$
$\sim$ (*map* (*uncurry* ($\sqcap$)) ((*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* $(\sim (\Gamma \ominus \mathit{map\ fst}\ \Psi)) = \mathit{mset}\ (\sim \Gamma \ominus \mathit{map\ snd}\ ?\Psi)$
using $\Psi(\mathit{1})$
by (*simp add*: *image-mset-Diff multiset.map-comp*)
hence $\sim (\Gamma \ominus \mathit{map\ fst}\ \Psi) \preceq (\sim \Gamma \ominus \mathit{map\ snd}\ ?\Psi)$
using *stronger-theory-reflexive*
stronger-theory-right-permutation
mset-eq-perm
by *blast*
ultimately show *?thesis*
using *stronger-theory-combine*
by *simp*
qed
hence $\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ ?\Psi\ @ \sim \Gamma \ominus \mathit{map\ snd}\ ?\Psi\ \$\vdash \Phi$
using $\Psi(\mathit{3})$ *segmented-stronger-theory-left-monotonic* **by** *blast*
ultimately show $\sim \Gamma\ \$\vdash (\varphi \# \Phi)$
using *segmented-deduction.simps*$(\mathit{2})$ **by** *blast*
qed

lemma (**in** *Logical-Probability*) *segmented-deduction-summation-introduction*:
assumes $\sim \Gamma\ \$\vdash \sim \Phi$
shows $(\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ \mathit{Pr}\ \gamma)$
proof –
have $\forall\ \Gamma.\ \sim \Gamma\ \$\vdash \sim \Phi \longrightarrow (\sum \varphi \leftarrow \Phi.\ \mathit{Pr}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ \mathit{Pr}\ \gamma)$
proof (*induct* Φ)
case *Nil*
then show *?case*
by (*simp*, *metis* (*full-types*) *ex-map-conv Non-Negative sum-list-nonneg*)
next
case (*Cons* φ Φ)
{
fix Γ
assume $\sim \Gamma\ \$\vdash \sim (\varphi \# \Phi)$
hence $\sim \Gamma\ \$\vdash (\sim \varphi \# \sim \Phi)$ **by** *simp*
from *this* **obtain** Ψ **where** Ψ:
$\mathit{mset}\ (\mathit{map\ fst}\ \Psi) \subseteq\#\ \mathit{mset}\ \Gamma$
$\sim (\mathit{map}\ (\mathit{uncurry}\ (\backslash))\ \Psi) :\vdash \sim \varphi$
$\sim (\mathit{map}\ (\mathit{uncurry}\ (\sqcap))\ \Psi\ @\ \Gamma \ominus (\mathit{map\ fst}\ \Psi))\ \$\vdash \sim \Phi$
using *negated-segmented-deduction* **by** *blast*
let $?\Gamma = \Gamma \ominus (\mathit{map\ fst}\ \Psi)$
let $?\Psi_1 = \mathit{map}\ (\mathit{uncurry}\ (\backslash))\ \Psi$
let $?\Psi_2 = \mathit{map}\ (\mathit{uncurry}\ (\sqcap))\ \Psi$
have $(\sum \varphi' \leftarrow \Phi.\ \mathit{Pr}\ \varphi') \leq (\sum \varphi \leftarrow (?\Psi_2\ @\ ?\Gamma).\ \mathit{Pr}\ \varphi)$
using *Cons* $\Psi(\mathit{3})$ **by** *blast*
moreover
have $\mathit{Pr}\ \varphi \leq (\sum \varphi \leftarrow ?\Psi_1.\ \mathit{Pr}\ \varphi)$
using $\Psi(\mathit{2})$
biconditional-weaken
list-deduction-def

map-negation-list-implication
set-deduction-base-theory
implication-list-summation-inequality
by *blast*
ultimately have $(\sum\varphi'\leftarrow(\varphi\ \#\ \Phi).\ Pr\ \varphi') \leq (\sum\gamma\ \leftarrow(?\Psi_1\ @\ ?\Psi_2\ @\ ?\Gamma).\ Pr$
$\gamma)$
by *simp*
moreover have $(\sum\varphi'\leftarrow(?\Psi_1\ @\ ?\Psi_2).\ Pr\ \varphi') = (\sum\gamma\leftarrow(map\ fst\ \Psi).\ Pr\ \gamma)$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
let $?\Psi_1 = map\ (uncurry\ (\backslash))\ \Psi$
let $?\Psi_2 = map\ (uncurry\ (\sqcap))\ \Psi$
let $?\psi_1 = uncurry\ (\backslash)\ \psi$
let $?\psi_2 = uncurry\ (\sqcap)\ \psi$
assume $(\sum\varphi'\leftarrow(?\Psi_1\ @\ ?\Psi_2).\ Pr\ \varphi') = (\sum\gamma\leftarrow(map\ fst\ \Psi).\ Pr\ \gamma)$
moreover
{
let $?\gamma = fst\ \psi$
let $?\psi = snd\ \psi$
have $uncurry\ (\backslash) = (\lambda\ \psi.\ (fst\ \psi) \setminus (snd\ \psi))$
$uncurry\ (\sqcap) = (\lambda\ \psi.\ (fst\ \psi) \sqcap (snd\ \psi))$
by *fastforce+*
moreover have $Pr\ ?\gamma = Pr\ (?\gamma \setminus ?\psi) + Pr\ (?\gamma \sqcap ?\psi)$
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** *Minimal-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** *Minimal-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** *Minimal-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* $(\rightarrow)$) $(\mathfrak{A}\ \Psi\ (\delta\ \#\ \Delta))$) =
mset (*map snd* $(\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))$)
proof (*cases find* $(\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None$)
case *True*
then show *?thesis* **using** *Cons* **by** *simp*
next
case *False*
from *this* **obtain** ψ **where**
find $(\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
uncurry $(\rightarrow)\ \psi = snd\ \delta$
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 Minimal-Logic) secondComponent-right-empty [simp]:
  𝔅 [] Δ = []
  by (induct Δ, simp+)

lemma (in Minimal-Logic) firstComponent-msub:
  mset (𝔄 Ψ Δ) ⊆# mset Ψ
proof −
  have ∀ Ψ. mset (𝔄 Ψ Δ) ⊆# mset Ψ
  proof(induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have mset (𝔄 Ψ (δ # Δ)) ⊆# mset Ψ
      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
          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 Minimal-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 Minimal-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 (𝔅 Ψ (δ # Δ))) ⊆# mset (map (uncurry (→)) Ψ)
    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 (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = Some ψ
        by auto
      hence 𝔅 Ψ (δ # Δ) = δ # (𝔅 (remove1 ψ Ψ) Δ)
        using ψ by fastforce
      with Cons have mset (map snd (𝔅 Ψ (δ # Δ))) ⊆#
                  mset ((snd δ) # map (uncurry (→)) (remove1 ψ Ψ))
        by (simp, metis mset-map mset-remove1)
      moreover from ψ have snd δ = (uncurry (→)) ψ
        using find-Some-predicate by fastforce
      ultimately have mset (map snd (𝔅 Ψ (δ # Δ))) ⊆#
                   mset (map (uncurry (→)) (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 Minimal-Logic) secondComponent-diff-msub:
  assumes mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
Ψ))
  shows mset (map snd (Δ ⊖ (𝔅 Ψ Δ))) ⊆# mset (Γ ⊖ (map snd Ψ))
proof −
  have ∀ Ψ Γ. mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map
snd Ψ)) ⟶
              mset (map snd (Δ ⊖ (𝔅 Ψ Δ))) ⊆# mset (Γ ⊖ (map snd Ψ))
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ Γ
      assume ◇: mset (map snd (δ # Δ)) ⊆# mset (map (uncurry (→)) Ψ @ Γ
⊖ map snd Ψ)
     have mset (map snd ((δ # Δ) ⊖ 𝔅 Ψ (δ # Δ))) ⊆# mset (Γ ⊖ map snd Ψ)
      proof (cases find (λ ψ. (uncurry (→)) 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
                          listSubtract.simps(2)
                          listSubtract-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* (→)) Ψ)
by (*meson secondComponent-snd-projection-msub mset-eq-perm mset-le-perm-append*)
have *F*: ∀ *a m ma*. (*add-mset* (*a*::′*a*) *m* ⊆# *ma*) = (*a* ∈# *ma* ∧ *m* ⊆# *ma*
− {#*a*#})
using *insert-subset-eq-iff* **by** *blast*
then have *snd* δ ∈# *mset* (*map snd* (𝔅 Ψ (δ # Δ))
@ *diff* (*map* (*uncurry* (→)) Ψ) (*map snd* (𝔅 Ψ (δ #
Δ))))
+ *mset* (Γ ⊖ *map snd* Ψ)
using *E* ◇ **by** *force*
then have *snd* δ ∈# *mset* (Γ ⊖ *map snd* Ψ)
using *A E* **by** (*metis* (*no-types*) *in-multiset-in-set union-iff*)
then have *G*: *add-mset* (*snd* δ) (*mset* (*map snd* (Δ ⊖ 𝔅 Ψ Δ))) ⊆# *mset*
(Γ ⊖ *map snd* Ψ)
using *B F* **by** *force*
have *H*: ∀ *ps psa f*. ¬ *mset* (*ps*::(′*a* × ′*a*) *list*) ⊆# *mset psa* ∨
mset ((*map f psa*::′*a list*) ⊖ *map f ps*) = *mset* (*map f* (*psa*
⊖ *ps*))
using *map-listSubtract-mset-equivalence* **by** *blast*
have *snd* δ ∉# *mset* (*map snd* (𝔅 Ψ (δ # Δ)))
+ *mset* (*diff* (*map* (*uncurry* (→)) Ψ) (*map snd* (𝔅 Ψ (δ # Δ))))
using *A E* **by** *auto*
then have *add-mset* (*snd* δ) (*mset* (*map snd* (Δ ⊖ 𝔅 Ψ Δ)))

$= mset\ (map\ snd\ (\delta \mathbin{\#} \Delta) \ominus map\ snd\ (\mathfrak{B}\ \Psi\ (\delta \mathbin{\#} \Delta)))$
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 $snd\ \delta = uncurry\ (\rightarrow)\ \psi$
$\psi \in set\ \Psi$
$mset\ ((\delta \mathbin{\#} \Delta) \ominus \mathfrak{B}\ \Psi\ (\delta \mathbin{\#} \Delta)) =$
$mset\ (\Delta \ominus \mathfrak{B}\ ?\Psi'\ \Delta)$
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 $(find\ (\lambda p.\ uncurry\ (\rightarrow)\ p = snd\ \delta)\ \Psi \neq None$
$\vee\ (\forall p.\ p \notin set\ \Psi \vee uncurry\ (\rightarrow)\ p \neq snd\ \delta))$
$\wedge\ (find\ (\lambda p.\ uncurry\ (\rightarrow)\ p = snd\ \delta)\ \Psi = None$
$\vee\ search\ \Psi\ (\lambda p.\ uncurry\ (\rightarrow)\ p = snd\ \delta) \in set\ \Psi$
$\wedge\ uncurry\ (\rightarrow)\ (search\ \Psi\ (\lambda p.\ uncurry\ (\rightarrow)\ p = snd\ \delta)) = snd\ \delta)$
by *blast*
hence $snd\ \delta \in set\ (map\ (uncurry\ (\rightarrow))\ \Psi)$
by (*metis* (*no-types*) *False image-eqI image-set*)
moreover
have A: $add\text{-}mset\ (uncurry\ (\rightarrow)\ \psi)\ (image\text{-}mset\ snd\ (mset\ \Delta))$
$= image\text{-}mset\ snd\ (add\text{-}mset\ \delta\ (mset\ \Delta))$
by (*simp add*: ‹$snd\ \delta = uncurry\ (\rightarrow)\ \psi$›)
have B: $\{\#snd\ \delta\#\} \subseteq\# image\text{-}mset\ (uncurry\ (\rightarrow))\ (mset\ \Psi)$
using ‹$snd\ \delta \in set\ (map\ (uncurry\ (\rightarrow))\ \Psi)$› **by** *force*
have $image\text{-}mset\ (uncurry\ (\rightarrow))\ (mset\ \Psi) - \{\#snd\ \delta\#\}$
$= image\text{-}mset\ (uncurry\ (\rightarrow))\ (mset\ (remove1\ \psi\ \Psi))$
by (*simp add*: ‹$\psi \in set\ \Psi$› ‹$snd\ \delta = uncurry\ (\rightarrow)\ \psi$› *image-mset-Diff*)
then have $mset\ (map\ snd\ (\Delta \ominus \mathfrak{B}\ (remove1\ \psi\ \Psi)\ \Delta))$
$\subseteq\# mset\ (remove1\ (snd\ \psi)\ \Gamma \ominus map\ snd\ (remove1\ \psi\ \Psi))$
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-Propositional-Logic)
 mergeWitness :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (𝔍)
 where
  𝔍 Ψ [] = Ψ
 | 𝔍 Ψ (δ # Δ) =
     (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of
        None ⇒ δ # 𝔍 Ψ Δ
      | Some ψ ⇒ (fst δ ⊓ fst ψ, snd ψ) # (𝔍 (remove1 ψ Ψ) Δ))

lemma (in Classical-Propositional-Logic) mergeWitness-right-empty [simp]:
 𝔍 [] Δ = Δ
 by (induct Δ, simp+)

lemma (in Classical-Propositional-Logic) secondComponent-mergeWitness-snd-projection:
 mset (map snd Ψ @ map snd (Δ ⊖ (𝔅 Ψ Δ))) = mset (map snd (𝔍 Ψ Δ))
proof −
 have ∀ Ψ. mset (map snd Ψ @ map snd (Δ ⊖ (𝔅 Ψ Δ))) = mset (map snd (𝔍
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 (𝔍 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* ($\lambda\psi$. *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ =
Some ψ
by *auto*
moreover have $\psi \in$ *set* Ψ
by (*meson* ψ *find-Some-set-membership*)
moreover
let *?*Ψ' = *remove1* ψ Ψ
from *Cons* **have**
mset (*map snd ?*Ψ' @ *map snd* ($\Delta \ominus \mathfrak{B}$ *?*Ψ' Δ)) =
mset (*map snd* ($\mathfrak{J}$ *?*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-Propositional-Logic*) *secondComponent-mergeWitness-stronger-theory*:
(*map* (*uncurry* ($\rightarrow$)) Δ @ *map* (*uncurry* ($\rightarrow$)) $\Psi \ominus$ *map snd* ($\mathfrak{B}$ Ψ Δ)) $\preceq$
map (*uncurry* ($\rightarrow$)) ($\mathfrak{J}$ Ψ Δ)
proof –
have $\forall$ Ψ. (*map* (*uncurry* ($\rightarrow$)) Δ @
map (*uncurry* ($\rightarrow$)) $\Psi \ominus$ *map snd* ($\mathfrak{B}$ Ψ Δ)) $\preceq$
map (*uncurry* ($\rightarrow$)) ($\mathfrak{J}$ Ψ Δ)
proof (*induct* Δ)
case *Nil*
then show *?case*
by *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have $\vdash$ (*uncurry* ($\rightarrow$)) $\delta \rightarrow$ (*uncurry* ($\rightarrow$)) δ
using *Axiom-1 Modus-Ponens implication-absorption* **by** *blast*

have
$(\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Psi \ominus \mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))) \preceq$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta))$
proof $(\mathit{cases}\ \mathit{find}\ (\lambda\ \psi.\ (\mathit{uncurry}\ (\rightarrow))\ \psi = \mathit{snd}\ \delta)\ \Psi = \mathit{None})$
case *True*
thus *?thesis*
using *Cons*
$\langle\vdash (\mathit{uncurry}\ (\rightarrow))\ \delta \rightarrow (\mathit{uncurry}\ (\rightarrow))\ \delta\rangle$
by (*simp*, *metis stronger-theory-left-right-cons*)
next
case *False*
from *this* **obtain** ψ **where** ψ: $\mathit{find}\ (\lambda\psi.\ \mathit{uncurry}\ (\rightarrow)\ \psi = \mathit{snd}\ \delta)\ \Psi =$
$\mathit{Some}\ \psi$
by *auto*
from ψ **have** $\mathit{snd}\ \delta = \mathit{uncurry}\ (\rightarrow)\ \psi$
using *find-Some-predicate* **by** *fastforce*
from ψ $\langle \mathit{snd}\ \delta = \mathit{uncurry}\ (\rightarrow)\ \psi\rangle$ **have**
$\mathit{mset}\ (\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Psi \ominus \mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))) =$
$\mathit{mset}\ (\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathit{remove1}\ \psi\ \Psi) \ominus$
$\mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta))$
by (*simp add*: *find-Some-set-membership image-mset-Diff*)
hence
$(\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Psi \ominus \mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))) \preceq$
$(\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathit{remove1}\ \psi\ \Psi) \ominus \mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta))$
by (*simp add*: *msub-stronger-theory-intro*)
with *Cons* $\langle\vdash (\mathit{uncurry}\ (\rightarrow))\ \delta \rightarrow (\mathit{uncurry}\ (\rightarrow))\ \delta\rangle$ **have**
$(\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\delta\ \#\ \Delta)\ @$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Psi \ominus \mathit{map}\ \mathit{snd}\ (\mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)))$
$\preceq ((\mathit{uncurry}\ (\rightarrow))\ \delta\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathfrak{J}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta))$
using *stronger-theory-left-right-cons*
stronger-theory-transitive
by *fastforce*
moreover
let $?\alpha = \mathit{fst}\ \delta$
let $?\beta = \mathit{fst}\ \psi$
let $?\gamma = \mathit{snd}\ \psi$
have $\mathit{uncurry}\ (\rightarrow) = (\lambda\ \delta.\ \mathit{fst}\ \delta \rightarrow \mathit{snd}\ \delta)$ **by** *fastforce*
with ψ **have** $(\mathit{uncurry}\ (\rightarrow))\ \delta = ?\alpha \rightarrow ?\beta \rightarrow ?\gamma$
using *find-Some-predicate* **by** *fastforce*
hence $\vdash ((?\alpha \sqcap ?\beta) \rightarrow ?\gamma) \rightarrow (\mathit{uncurry}\ (\rightarrow))\ \delta$
using *biconditional-def curry-uncurry* **by** *auto*
with ψ **have**
$((\mathit{uncurry}\ (\rightarrow))\ \delta\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathfrak{J}\ (\mathit{remove1}\ \psi\ \Psi)\ \Delta)) \preceq$
$\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ (\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta))$

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-Propositional-Logic*) *mergeWitness-msub-intro*:
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* ($\mathfrak{J}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
proof −
have $\forall\, \Psi\ \Gamma$. *mset* (*map snd* Ψ) $\subseteq\#$ *mset* $\Gamma \longrightarrow$
mset (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\rightarrow)$) Ψ @ $\Gamma \ominus$ (*map snd* Ψ)) $\longrightarrow$
mset (*map snd* ($\mathfrak{J}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
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 $\Diamond$: *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* ($\lambda\, \psi$. (*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 $\Diamond$(*2*) **have** *snd* $\delta \in\#$ *mset* ($\Gamma \ominus$ *map snd* Ψ)
using *mset-subset-eq-insertD* **by** *fastforce*
with $\Diamond$(*1*) **have** *mset* (*map snd* Ψ) $\subseteq\#$ *mset* (*remove1* (*snd* δ) Γ)
by (*metis listSubtract-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* (Γ $\ominus$ *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)))
$\subseteq\#$ {#$x \to y$. (x, y) $\in\#$ *mset* Ψ#}
using $\diamondsuit$(*2*) **by** (*simp*, *metis add-mset-diff-bothsides*
listSubtract-mset-homomorphism
mset-map subset-eq-diff-conv)
hence *mset* (*map snd* Δ)
$\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) 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
listSubtract-mset-homomorphism
multi-drop-mem-not-eq)
with ‹*find* (λ ψ. (*uncurry* ($\to$)) ψ = *snd* δ) Ψ = *None*›
show *?thesis*
by *simp*
next
case *False*
from *this* **obtain** ψ **where** ψ:
find ($\lambda\psi$. *uncurry* ($\to$) ψ = *snd* δ) Ψ = *Some* ψ
by *fastforce*
let *?*χ = *fst* ψ
let *?*γ = *snd* ψ
have *uncurry* ($\to$) = (λ ψ. *fst* $\psi \to$ *snd* ψ)
by *fastforce*
moreover
from *this* **have** *uncurry* ($\to$) ψ = *?*$\chi \to$ *?*γ **by** *fastforce*
with ψ **have** *A*: (*?*χ, *?*γ) $\in$ *set* Ψ
and *B*: *snd* δ = *?*$\chi \to$ *?*γ
using *find-Some-predicate*
by (*simp add*: *find-Some-set-membership*, *fastforce*)
let *?*Ψ' = *remove1* (*?*χ, *?*γ) Ψ
from *B* $\diamondsuit$(*2*) **have**
mset (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ *map snd* Ψ) $-$ {# *?*$\chi \to$ *?*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*
   listSubtract-remove1-perm
   mset-eq-perm
   prod.sel(*2*)
   set-map)
with *A* **have** *mset* (Γ ⊖ *map snd* Ψ) = *mset*((*remove1 ?*γ Γ) ⊖ (*map snd* *?*Ψ′))
 by (*metis remove1-pairs-list-projections-snd*
   in-multiset-in-set
   listSubtract-mset-homomorphism
   mset-remove1)
ultimately have *mset* (*map snd* Δ) ⊆#
   mset (*map* (*uncurry* (→)) *?*Ψ′ @ (*remove1 ?*γ Γ) ⊖ *map snd* *?*Ψ′)
 by *simp*
hence *mset* (*map snd* (𝔍 *?*Ψ′ Δ)) ⊆# *mset* (*remove1 ?*γ Γ)
 using *Cons* ◇(*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 ◇(*1*) *A* **have** *mset* (*map snd* (𝔍 *?*Ψ′ Δ)) + {# *?*γ #} ⊆# *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* $\delta \sqcap$ *?χ*, *?γ*) # ($\mathfrak{J}$ *?Ψ′* Δ))) ⊆# *mset* Γ
by *simp*
moreover from *ψ* **have**
$\mathfrak{J}$ Ψ (δ # Δ) = (*fst* δ ⊓ *?χ*, *?γ*) # ($\mathfrak{J}$ *?Ψ′* Δ)
by *simp*
ultimately show *?thesis* **by** *simp*
qed
}
thus *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *right-mergeWitness-stronger-theory*:
map (*uncurry* (⊔)) Δ ⪯ *map* (*uncurry* (⊔)) ($\mathfrak{J}$ Ψ Δ)
proof −
have ∀ Ψ. *map* (*uncurry* (⊔)) Δ ⪯ *map* (*uncurry* (⊔)) ($\mathfrak{J}$ Ψ Δ)
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *map* (*uncurry* (⊔)) (δ # Δ) ⪯ *map* (*uncurry* (⊔)) ($\mathfrak{J}$ Ψ (δ # Δ))
proof (*cases find* (λ ψ. (*uncurry* (→)) ψ = *snd* δ) Ψ = *None*)
case *True*
hence $\mathfrak{J}$ Ψ (δ # Δ) = δ # $\mathfrak{J}$ Ψ Δ
by *simp*
moreover have ⊢ (*uncurry* (⊔)) δ → (*uncurry* (⊔)) δ
by (*metis Axiom-1 Axiom-2 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* $(\rightarrow) = (\lambda\ \psi.\ \mathit{fst}\ \psi \rightarrow \mathit{snd}\ \psi)$
uncurry $(\sqcup) = (\lambda\ \delta.\ \mathit{fst}\ \delta \sqcup \mathit{snd}\ \delta)$
by *fastforce+*
hence *uncurry* $(\sqcup)\ \delta = ?\mu \sqcup (?\chi \rightarrow ?\gamma)$
using ψ *find-Some-predicate*
by *fastforce*
moreover
{
fix $\mu\ \chi\ \gamma$
have $\vdash ((\mu \sqcap \chi) \sqcup \gamma) \rightarrow (\mu \sqcup (\chi \rightarrow \gamma))$
proof −
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\mu\rangle \sqcap \langle\chi\rangle) \sqcup \langle\gamma\rangle) \rightarrow (\langle\mu\rangle \sqcup (\langle\chi\rangle \rightarrow \langle\gamma\rangle)))$
by *fastforce*
hence $\vdash (\!|\ (((\langle\mu\rangle \sqcap \langle\chi\rangle) \sqcup \langle\gamma\rangle) \rightarrow (\langle\mu\rangle \sqcup (\langle\chi\rangle \rightarrow \langle\gamma\rangle)))\ |\!)$
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-Propositional-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* $\delta\ \Delta$)
{
fix Ψ
have *map* (*uncurry* $(\sqcup)$) $\Psi \preceq$ *map* (*uncurry* $(\sqcup)$) $(\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta))$
proof (*cases find* $(\lambda\ \psi.\ (\mathit{uncurry}\ (\rightarrow))\ \psi = \mathit{snd}\ \delta)\ \Psi = \mathit{None}$)
case *True*
then show *?thesis*
using *Cons stronger-theory-right-cons*
by *auto*
next
case *False*

from *this* **obtain** ψ **where** ψ:
  $find\ (\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
  by *fastforce*
let $?\chi = fst\ \psi$
let $?\gamma = snd\ \psi$
let $?\mu = fst\ \delta$
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
  $map\ (uncurry\ (\sqcup))\ (\psi\ \#\ (remove1\ \psi\ \Psi)) \preceq$
   $map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta))$
  using *Cons* ψ *stronger-theory-left-right-cons*
  by *simp*
moreover from ψ **have** $\psi \in set\ \Psi$
  by (*simp add*: *find-Some-set-membership*)
hence $mset\ (map\ (uncurry\ (\sqcup))\ (\psi\ \#\ (remove1\ \psi\ \Psi))) =$
    $mset\ (map\ (uncurry\ (\sqcup))\ \Psi)$
  by (*metis insert-DiffM*
      mset.simps(2)
      mset-map
      mset-remove1
      set-mset-mset)
hence $map\ (uncurry\ (\sqcup))\ \Psi \preceq map\ (uncurry\ (\sqcup))\ (\psi\ \#\ (remove1\ \psi\ \Psi))$
  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-Propositional-Logic*) *mergeWitness-segmented-deduction-intro*:
assumes *mset* (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
and *map* (*uncurry* ($\to$)) Δ @ (*map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ *map snd* Ψ) $\ominus$ *map snd* Δ \$$\vdash$ Φ
(**is** *?*Γ_0 \$$\vdash$ Φ)
shows *map* (*uncurry* ($\to$)) ($\mathfrak{J}$ Ψ Δ) @ $\Gamma \ominus$ *map snd* ($\mathfrak{J}$ Ψ Δ) \$$\vdash$ Φ
(**is** *?*Γ \$$\vdash$ Φ)
proof –
let *?*Σ = $\mathfrak{B}$ Ψ Δ
let *?A* = *map* (*uncurry* ($\to$)) Δ
let *?B* = *map* (*uncurry* ($\to$)) Ψ
let *?C* = *map snd ?*Σ
let *?D* = $\Gamma \ominus$ (*map snd* Ψ)
let *?E* = *map snd* ($\Delta \ominus$ *?*Σ)
have Σ: *mset ?*Σ $\subseteq\#$ *mset* Δ
mset ?C $\subseteq\#$ *mset ?B*
mset ?E $\subseteq\#$ *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* $\to$ *y*. (*x*, *y*) $\in\#$ *mset* Ψ#}
+ (*mset* Γ – (*image-mset snd* (*mset* Ψ)
+ *image-mset snd* (*mset* Δ – *mset* ($\mathfrak{B}$ Ψ Δ)))))
= {#*x* $\to$ *y*. (*x*, *y*) $\in\#$ *mset* Ψ#} + (*mset* Γ – *image-mset snd* (*mset* Ψ))
by (*simp add*: *union-commute*)
with *calculation* **have** *mset ?*Γ_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* ($\to$)) ($\mathfrak{J}$ 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
listSubtract-mset-homomorphism
msub-stronger-theory-intro
map-listSubtract-mset-containment
map-listSubtract-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-Propositional-Logic*) *segmented-formula-right-split*:
$\Gamma \ \$\vdash (\varphi \# \Phi) = \Gamma \ \$\vdash (\psi \sqcup \varphi \# \psi \to \varphi \# \Phi)$
proof (*rule iffI*)
assume $\Gamma \ \$\vdash (\varphi \# \Phi)$
from *this* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq\#$ *mset* Γ
map (*uncurry* $(\sqcup)$) $\Psi :\vdash \varphi$
(*map* (*uncurry* $(\to)$) $\Psi\ @\ \Gamma \ominus$ (*map snd* Ψ)) $\$\vdash \Phi$
by *auto*
let $?\Psi_1 =$ *zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi$) (*map snd* Ψ)
let $?\Gamma_1 =$ *map* (*uncurry* $(\to)$) $?\Psi_1\ @\ \Gamma \ominus$ (*map snd* $?\Psi_1$)
let $?\Psi_2 =$ *zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \to \chi)\ \Psi$) (*map* (*uncurry* $(\to)$) $?\Psi_1$)
let $?\Gamma_2 =$ *map* (*uncurry* $(\to)$) $?\Psi_2\ @\ ?\Gamma_1 \ominus$ (*map snd* $?\Psi_2$)
have *map* (*uncurry* $(\to)$) $\Psi \preceq$ *map* (*uncurry* $(\to)$) $?\Psi_2$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Ψ)
let $?\chi =$ *fst* δ
let $?\gamma =$ *snd* δ
let $?\Psi_1 =$ *zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi$) (*map snd* Ψ)
let $?\Psi_2 =$ *zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \to \chi)\ \Psi$) (*map* (*uncurry* $(\to)$) $?\Psi_1$)
let $?T_1 = \lambda\ \Psi.$ *map* (*uncurry* $(\to)$) (*zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi$) (*map snd* Ψ))
let $?T_2 = \lambda\ \Psi.$ *map* (*uncurry* $(\to)$) (*zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \to \chi)\ \Psi$) ($?T_1\ \Psi$))
{
fix $\delta :: 'a \times 'a$
have $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi) = (\lambda\ \delta.\ \psi \sqcup (fst\ \delta))$
$(\lambda\ (\chi,\gamma).\ \psi \to \chi) = (\lambda\ \delta.\ \psi \to (fst\ \delta))$

by *fastforce*+
note *functional-identities* = *this*
have $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \delta = \psi \sqcup (\mathit{fst}\ \delta)$
$(\lambda\ (\chi,\gamma).\ \psi \to \chi)\ \delta = \psi \to (\mathit{fst}\ \delta)$
by (*simp add*: *functional-identities*)+
}
hence $?T_2\ (\delta\ \#\ \Psi) = ((\psi \to\ ?\chi) \to (\psi \sqcup\ ?\chi) \to\ ?\gamma)\ \#\ (\mathit{map}\ (\mathit{uncurry}\ (\to))\ ?\Psi_2)$
by *simp*
moreover have $\mathit{map}\ (\mathit{uncurry}\ (\to))\ (\delta\ \#\ \Psi) = (?\chi \to\ ?\gamma)\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\to))\ \Psi$
by (*simp add*: *case-prod-beta*)
moreover
{
fix $\chi\ \psi\ \gamma$
have $\vdash ((\psi \to \chi) \to (\psi \sqcup \chi) \to \gamma) \leftrightarrow (\chi \to \gamma)$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\psi\rangle \to \langle\chi\rangle) \to (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \leftrightarrow (\langle\chi\rangle \to \langle\gamma\rangle))$
by *fastforce*
hence $\vdash (\!|\ (((\langle\psi\rangle \to \langle\chi\rangle) \to (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \leftrightarrow (\langle\chi\rangle \to \langle\gamma\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
hence *identity*: $\vdash ((\psi \to\ ?\chi) \to (\psi \sqcup\ ?\chi) \to\ ?\gamma) \to (?\chi \to\ ?\gamma)$
using *biconditional-def* **by** *auto*
assume $\mathit{map}\ (\mathit{uncurry}\ (\to))\ \Psi \preceq \mathit{map}\ (\mathit{uncurry}\ (\to))\ ?\Psi_2$
with *identity* **have** $((?\chi \to\ ?\gamma)\ \#\ \mathit{map}\ (\mathit{uncurry}\ (\to))\ \Psi) \preceq$
$(((\psi \to\ ?\chi) \to (\psi \sqcup\ ?\chi) \to\ ?\gamma)\ \#\ (\mathit{map}\ (\mathit{uncurry}\ (\to))\ ?\Psi_2))$
using *stronger-theory-left-right-cons* **by** *blast*
ultimately show *?case* **by** *simp*
qed
hence $(\mathit{map}\ (\mathit{uncurry}\ (\to))\ \Psi\ @\ \Gamma \ominus (\mathit{map}\ \mathit{snd}\ \Psi)) \preceq$
$((\mathit{map}\ (\mathit{uncurry}\ (\to))\ ?\Psi_2)\ @\ \Gamma \ominus (\mathit{map}\ \mathit{snd}\ \Psi))$
using *stronger-theory-combine stronger-theory-reflexive* **by** *blast*
moreover have $\mathit{mset}\ ?\Gamma_2 = \mathit{mset}\ ((\mathit{map}\ (\mathit{uncurry}\ (\to))\ ?\Psi_2)\ @\ \Gamma \ominus (\mathit{map}\ \mathit{snd}\ ?\Psi_1))$
by *simp*
ultimately have $(\mathit{map}\ (\mathit{uncurry}\ (\to))\ \Psi\ @\ \Gamma \ominus (\mathit{map}\ \mathit{snd}\ \Psi)) \preceq\ ?\Gamma_2$
by (*simp add*: *stronger-theory-relation-def*)
hence $?\Gamma_2\ \$\!\vdash \Phi$
using $\Psi(3)$ *segmented-stronger-theory-left-monotonic* **by** *blast*
moreover
have $(\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ ?\Psi_2) :\vdash \psi \to \varphi$
proof −
let $?\Gamma = \mathit{map}\ (\lambda\ (\chi,\ \gamma).\ (\psi \to \chi) \sqcup (\psi \sqcup \chi) \to \gamma)\ \Psi$
let $?\Sigma = \mathit{map}\ (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \Psi$
have $\mathit{map}\ (\mathit{uncurry}\ (\sqcup))\ ?\Psi_2 =\ ?\Gamma$
proof (*induct* Ψ)

case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* χ Ψ)
have (λ φ. (*case* φ *of* $(\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup$ (*case* φ *of* $(\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow$ *snd* φ) =
(λ φ. (*case* φ *of* $(\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma$))
by *fastforce*
hence (*case* χ *of* $(\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup$ (*case* χ *of* $(\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow$ *snd* χ =
(*case* χ *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* δ Ψ)
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* δ
let $?\gamma =$ *snd* δ
have (λ δ. (*case* δ *of* $(\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma$)) =
(λ δ. $\psi \rightarrow$ *fst* $\delta \sqcup (\psi \sqcup$ *fst* $\delta) \rightarrow$ *snd* δ)
(λ δ. (*case* δ *of* $(\chi, \gamma) \Rightarrow \psi \rightarrow (\chi \sqcup \gamma)$)) = ($\lambda$ δ. $\psi \rightarrow$ (*fst* $\delta \sqcup$ *snd* δ))
by *fastforce+*
hence $?\alpha = (\psi \rightarrow ?\chi) \sqcup (\psi \sqcup ?\chi) \rightarrow ?\gamma$
$?\beta = \psi \rightarrow (?\chi \sqcup ?\gamma)$
by *metis+*
moreover
{
fix ψ χ γ
have $\vdash ((\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma) \rightarrow (\psi \rightarrow (\chi \sqcup \gamma))$
proof −
have $\forall$ $\mathfrak{M}$. $\mathfrak{M} \models_{prop} (((\langle\psi\rangle \rightarrow \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \rightarrow \langle\gamma\rangle) \rightarrow (\langle\psi\rangle \rightarrow (\langle\chi\rangle \sqcup \langle\gamma\rangle))$
by *fastforce*
hence $\vdash (\!|\ (((\langle\psi\rangle \rightarrow \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \rightarrow \langle\gamma\rangle) \rightarrow (\langle\psi\rangle \rightarrow (\langle\chi\rangle \sqcup \langle\gamma\rangle))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
ultimately have $\vdash ?\alpha \rightarrow ?\beta$ **by** *simp*
thus *?case*
using *Cons*
stronger-theory-left-right-cons
by *simp*

```
  qed
  moreover have ∀ φ. (map (uncurry (⊔)) Ψ) :⊢ φ ⟶ ?Σ :⊢ ψ → φ
  proof (induct Ψ)
    case Nil
    then show ?case
      using Axiom-1 Modus-Ponens
      by fastforce
  next
    case (Cons δ Ψ)
    let ?δ′ = (λ (χ, γ). (ψ → (χ ⊔ γ))) δ
    let ?Σ = map (λ (χ, γ). (ψ → (χ ⊔ γ))) Ψ
    let ?Σ′ = map (λ (χ, γ). (ψ → (χ ⊔ γ))) (δ # Ψ)
    {
      fix φ
      assume map (uncurry (⊔)) (δ # Ψ) :⊢ φ
      hence map (uncurry (⊔)) Ψ :⊢ (uncurry (⊔)) δ → φ
        using list-deduction-theorem
        by simp
      hence ?Σ :⊢ ψ → (uncurry (⊔)) δ → φ
        using Cons
        by blast
      moreover
      {
        fix α β γ
        have ⊢ (α → β → γ) → ((α → β) → α → γ)
          using Axiom-2 by auto
      }
      ultimately have ?Σ :⊢ (ψ → (uncurry (⊔)) δ) → ψ → φ
        using list-deduction-weaken [where ?Γ=?Σ]
              list-deduction-modus-ponens [where ?Γ=?Σ]
        by metis
      moreover
      have (λ δ. ψ → (uncurry (⊔)) δ) = (λ δ. (λ (χ, γ). (ψ → (χ ⊔ γ))) δ)
        by fastforce
      ultimately have ?Σ :⊢ (λ (χ, γ). (ψ → (χ ⊔ γ))) δ → ψ → φ
        by metis
      hence ?Σ′ :⊢ ψ → φ
        using list-deduction-theorem
        by simp
    }
    then show ?case by simp
  qed
  with Ψ(2) have ?Σ :⊢ ψ → φ
    by blast
  ultimately show ?thesis
    using stronger-theory-deduction-monotonic by auto
qed
moreover have mset (map snd ?Ψ₂) ⊆# mset ?Γ₁ by simp
ultimately have ?Γ₁ $⊢ (ψ → φ # Φ) using segmented-deduction.simps(2) by
```

blast

moreover have $\vdash$ (*map* (*uncurry* ($\sqcup$)) Ψ $:\rightarrow$ φ) $\rightarrow$ (*map* (*uncurry* ($\sqcup$)) *?*Ψ_1) $:\rightarrow$ ($\psi \sqcup \varphi$)

proof (*induct* Ψ)

case *Nil*

then show *?case*

unfolding *disjunction-def*

using *Axiom-1 Modus-Ponens*

by *fastforce*

next

case (*Cons* ν Ψ)

let *?*Δ = *map* (*uncurry* ($\sqcup$)) Ψ

let *?*Δ' = *map* (*uncurry* ($\sqcup$)) (ν # Ψ)

let *?*Σ = *map* (*uncurry* ($\sqcup$)) (*zip* (*map* (λ (χ,γ). $\psi \sqcup \chi$) Ψ) (*map snd* Ψ))

let *?*Σ' = *map* (*uncurry* ($\sqcup$)) (*zip* (*map* (λ (χ,γ). $\psi \sqcup \chi$) (ν # Ψ)) (*map snd* (ν # Ψ)))

have $\vdash$ (*?*Δ' $:\rightarrow$ φ) $\rightarrow$ (*uncurry* ($\sqcup$)) ν $\rightarrow$ *?*Δ $:\rightarrow$ φ

by (*simp*, *metis Axiom-1 Axiom-2 Modus-Ponens*)

with *Cons* **have** $\vdash$ (*?*Δ' $:\rightarrow$ φ) $\rightarrow$ (*uncurry* ($\sqcup$)) ν $\rightarrow$ *?*Σ $:\rightarrow$ ($\psi \sqcup \varphi$)

using *hypothetical-syllogism Modus-Ponens*

by *blast*

hence (*?*Δ' $:\rightarrow$ φ) # ((*uncurry* ($\sqcup$)) ν) # *?*Σ $:\vdash$ $\psi \sqcup \varphi$

by (*simp add*: *list-deduction-def*)

moreover have *set* ((*?*Δ' $:\rightarrow$ φ) # ((*uncurry* ($\sqcup$)) ν) # *?*Σ) =
set (((*uncurry* ($\sqcup$)) ν) # (*?*Δ' $:\rightarrow$ φ) # *?*Σ)

by *fastforce*

ultimately have ((*uncurry* ($\sqcup$)) ν) # (*?*Δ' $:\rightarrow$ φ) # *?*Σ $:\vdash$ $\psi \sqcup \varphi$

using *list-deduction-monotonic* **by** *blast*

hence (*?*Δ' $:\rightarrow$ φ) # *?*Σ $:\vdash$ ((*uncurry* ($\sqcup$)) ν) $\rightarrow$ ($\psi \sqcup \varphi$)

using *list-deduction-theorem*

by *simp*

moreover

let *?*χ = *fst* ν

let *?*γ = *snd* ν

have (λ ν . (*uncurry* ($\sqcup$)) ν) = (λ ν. *fst* ν $\sqcup$ *snd* ν)

by *fastforce*

hence (*uncurry* ($\sqcup$)) ν = *?*χ $\sqcup$ *?*γ **by** *simp*

ultimately have (*?*Δ' $:\rightarrow$ φ) # *?*Σ $:\vdash$ (*?*χ $\sqcup$ *?*γ) $\rightarrow$ ($\psi \sqcup \varphi$) **by** *simp*

moreover

{

fix α β δ γ

have $\vdash$ (($\beta \sqcup \alpha$) $\rightarrow$ ($\gamma \sqcup \delta$)) $\rightarrow$ (($\gamma \sqcup \beta$) $\sqcup$ α) $\rightarrow$ ($\gamma \sqcup \delta$)

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* ν
\# *map* (*uncurry* $(\sqcup)$) (*zip* (*map* $(\lambda(\text{-}, a).\ \psi \sqcup a)\ \Psi$) (*map snd* Ψ)) $:\vdash (\psi \sqcup$ *fst* $\nu) \sqcup$ *snd* ν
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 $(\lambda\ \nu.\ \psi \sqcup$ *fst* $\nu) = (\lambda\ (\chi, \gamma).\ \psi \sqcup \chi)$
by *fastforce*
hence $\psi \sqcup$ *fst* $\nu = (\lambda\ (\chi, \gamma).\ \psi \sqcup \chi)\ \nu$
by *metis*
hence $((\psi \sqcup ?\chi) \sqcup ?\gamma)$ # $?\Sigma = ?\Sigma'$
by *simp*
ultimately have $(?\Delta' :\rightarrow \varphi)$ # $?\Sigma' :\vdash \psi \sqcup \varphi$ **by** *simp*
then show *?case* **by** (*simp add*: *list-deduction-def*)
qed
with $\Psi(2)$ **have** *map* (*uncurry* $(\sqcup)$) $?\Psi_1 :\vdash (\psi \sqcup \varphi)$
unfolding *list-deduction-def*
using *Modus-Ponens*
by *blast*
moreover have *mset* (*map snd* $?\Psi_1$) $\subseteq$# *mset* Γ **using** $\Psi(1)$ **by** *simp*
ultimately show Γ \$$\vdash$ $(\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # $\Phi)$
using *segmented-deduction.simps*(*2*) **by** *blast*
next
assume Γ \$$\vdash$ $(\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # $\Phi)$
from *this* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq$# *mset* Γ
map (*uncurry* $(\sqcup)$) $\Psi :\vdash \psi \sqcup \varphi$
map (*uncurry* $(\rightarrow)$) Ψ @ $\Gamma \ominus$ (*map snd* Ψ) \$$\vdash$ $(\psi \rightarrow \varphi$ # $\Phi)$
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 = ℑ 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 (→)) ?Ω @ Γ ⊖ (map snd ?Ω) $⊢ Φ
    using Δ(1) Δ(3) Ψ(1) mergeWitness-segmented-deduction-intro by blast
  ultimately show Γ $⊢ (φ # Φ)
    using segmented-deduction.simps(2) by blast
qed

primrec (in Minimal-Logic)
  XWitness :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (𝔛)
  where
    𝔛 Ψ [] = []
  | 𝔛 Ψ (δ # Δ) =
      (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of
            None ⇒ δ # 𝔛 Ψ Δ
          | Some ψ ⇒ (fst ψ → fst δ, snd ψ) # (𝔛 (remove1 ψ Ψ) Δ))

primrec (in Minimal-Logic)
  XComponent :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (𝔛•)
  where
    𝔛• Ψ [] = []
  | 𝔛• Ψ (δ # Δ) =
      (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of
```

$None \Rightarrow \mathfrak{X}_\bullet\ \Psi\ \Delta$
$\mid Some\ \psi \Rightarrow (fst\ \psi \to fst\ \delta,\ snd\ \psi)\ \#\ (\mathfrak{X}_\bullet\ (remove1\ \psi\ \Psi)\ \Delta))$

primrec (**in** *Minimal-Logic*)
YWitness :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{Y})$
where
$\mathfrak{Y}\ \Psi\ []= \Psi$
$\mid \mathfrak{Y}\ \Psi\ (\delta\ \#\ \Delta) =$
$(case\ find\ (\lambda\ \psi.\ (uncurry\ (\to))\ \psi = snd\ \delta)\ \Psi\ of$
$None \Rightarrow \mathfrak{Y}\ \Psi\ \Delta$
$\mid Some\ \psi \Rightarrow (fst\ \psi,\ (fst\ \psi \to fst\ \delta) \to snd\ \psi)\ \#$
$(\mathfrak{Y}\ (remove1\ \psi\ \Psi)\ \Delta))$

primrec (**in** *Minimal-Logic*)
YComponent :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{Y}_\bullet)$
where
$\mathfrak{Y}_\bullet\ \Psi\ [] = []$
$\mid \mathfrak{Y}_\bullet\ \Psi\ (\delta\ \#\ \Delta) =$
$(case\ find\ (\lambda\ \psi.\ (uncurry\ (\to))\ \psi = snd\ \delta)\ \Psi\ of$
$None \Rightarrow \mathfrak{Y}_\bullet\ \Psi\ \Delta$
$\mid Some\ \psi \Rightarrow (fst\ \psi,\ (fst\ \psi \to fst\ \delta) \to snd\ \psi)\ \#$
$(\mathfrak{Y}_\bullet\ (remove1\ \psi\ \Psi)\ \Delta))$

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

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

lemma (**in** *Minimal-Logic*) *XWitness-map-snd-decomposition*:
$mset\ (map\ snd\ (\mathfrak{X}\ \Psi\ \Delta)) = mset\ (map\ snd\ ((\mathfrak{A}\ \Psi\ \Delta)\ @\ (\Delta \ominus (\mathfrak{B}\ \Psi\ \Delta))))$
proof −
have $\forall\Psi.\ mset\ (map\ snd\ (\mathfrak{X}\ \Psi\ \Delta)) = mset\ (map\ snd\ ((\mathfrak{A}\ \Psi\ \Delta)\ @\ (\Delta \ominus (\mathfrak{B}\ \Psi\ \Delta))))$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have $mset\ (map\ snd\ (\mathfrak{X}\ \Psi\ (\delta\ \#\ \Delta)))$
$= mset\ (map\ snd\ (\mathfrak{A}\ \Psi\ (\delta\ \#\ \Delta)\ @\ (\delta\ \#\ \Delta) \ominus \mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta)))$
using *Cons*
by (*cases find* $(\lambda\ \psi.\ (uncurry\ (\to))\ \psi = snd\ \delta)\ \Psi = None$,
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** *Minimal-Logic*) *YWitness-map-snd-decomposition*:
mset (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* ((Ψ ⊖ ($\mathfrak{A}$ Ψ Δ)) @ ($\mathfrak{Y}_\bullet$ Ψ Δ)))
proof −
have ∀ Ψ. *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* ((Ψ ⊖ ($\mathfrak{A}$ 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* (Ψ ⊖ $\mathfrak{A}$ Ψ (δ # Δ) @ $\mathfrak{Y}_\bullet$ Ψ (δ # Δ)))
using *Cons*
by (*cases find* (λ ψ. (*uncurry* (→)) ψ = *snd* δ) Ψ = *None*, *fastforce*+)
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *Minimal-Logic*) *XWitness-msub*:
assumes *mset* (*map snd* Ψ) ⊆# *mset* Γ
and *mset* (*map snd* Δ) ⊆# *mset* (*map* (*uncurry* (→)) Ψ @ Γ ⊖ (*map snd* Ψ))
shows *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ)) ⊆# *mset* Γ
proof −
have *mset* (*map snd* (Δ ⊖ ($\mathfrak{B}$ Ψ Δ))) ⊆# *mset* (Γ ⊖ (*map snd* Ψ))
using *assms secondComponent-diff-msub* **by** *blast*
moreover have *mset* (*map snd* ($\mathfrak{A}$ Ψ Δ)) ⊆# *mset* (*map snd* Ψ)
using *firstComponent-msub*
by (*simp add*: *image-mset-subseteq-mono*)
moreover have *mset* ((*map snd* Ψ) @ (Γ ⊖ *map snd* Ψ)) = *mset* Γ
using *assms*(*1*)
by *simp*
moreover have *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* (Δ

```
⊖ 𝔅 Ψ Δ))
               = mset (map snd (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 Minimal-Logic) YComponent-msub:
  mset (map snd (𝔜• Ψ Δ)) ⊆# mset (map (uncurry (→)) (𝔛 Ψ Δ))
proof −
  have ∀ Ψ. mset (map snd (𝔜• Ψ Δ)) ⊆# mset (map (uncurry (→)) (𝔛 Ψ Δ))
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have mset (map snd (𝔜• Ψ (δ # Δ))) ⊆# mset (map (uncurry (→)) (𝔛 Ψ
(δ # Δ)))
        using Cons
        by (cases find (λ ψ. (uncurry (→)) 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 Minimal-Logic) YWitness-msub:
  assumes mset (map snd Ψ) ⊆# mset Γ
      and mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
Ψ))
    shows mset (map snd (𝔜 Ψ Δ)) ⊆#
          mset (map (uncurry (→)) (𝔛 Ψ Δ) @ Γ ⊖ map snd (𝔛 Ψ Δ))
proof −
  have A: image-mset snd (mset Ψ) ⊆# mset Γ using assms by simp
  have B: image-mset snd (mset (𝔄 Ψ Δ)) + image-mset snd (mset Δ − mset
(𝔅 Ψ Δ)) ⊆# mset Γ
    using A XWitness-map-snd-decomposition assms(2) XWitness-msub by auto
  have mset Γ − image-mset snd (mset Ψ) = mset (Γ ⊖ map snd Ψ)
    by simp
  then have C: mset (map snd (Δ ⊖ 𝔅 Ψ Δ)) + image-mset snd (mset Ψ) ⊆#
mset Γ
   using A by (metis (full-types) assms(2) secondComponent-diff-msub subset-mset.le-diff-conv2)
```

have *image-mset snd* (*mset* ($\Psi \ominus \mathfrak{A}\ \Psi\ \Delta$)) + *image-mset snd* (*mset* ($\mathfrak{A}\ \Psi\ \Delta$))
= *image-mset snd* (*mset* Ψ)
by (*metis* (*no-types*) *image-mset-union*
listSubtract-mset-homomorphism
firstComponent-msub
subset-mset.diff-add)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}\ \Psi\ \Delta$))
+ (*image-mset snd* (*mset* ($\mathfrak{A}\ \Psi\ \Delta$)) + *image-mset snd* (*mset* Δ − *mset*
($\mathfrak{B}\ \Psi\ \Delta$)))
= *mset* (*map snd* ($\Delta \ominus \mathfrak{B}\ \Psi\ \Delta$)) + *image-mset snd* (*mset* Ψ)
by (*simp add*: *union-commute*)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}\ \Psi\ \Delta$))
$\subseteq$# *mset* Γ − (*image-mset snd* (*mset* ($\mathfrak{A}\ \Psi\ \Delta$)) + *image-mset snd* (*mset*
Δ − *mset* ($\mathfrak{B}\ \Psi\ \Delta$)))
by (*metis* (*no-types*) *B C subset-mset.le-diff-conv2*)
hence *mset* (*map snd* ($\Psi \ominus \mathfrak{A}\ \Psi\ \Delta$)) $\subseteq$# *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{X}\ \Psi\ \Delta$))
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-Propositional-Logic*) *XWitness-right-stronger-theory*:
map (*uncurry* ($\sqcup$)) $\Delta \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{X}\ \Psi\ \Delta$)
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 (⊔)) Δ ≼ map (uncurry (⊔)) (𝔛 ?Ψ' Δ)
        using Cons by simp
      moreover
      have (uncurry (⊔)) = (λ δ. fst δ ⊔ snd δ) by fastforce
      hence (uncurry (⊔)) δ = ?γ ⊔ (?α → ?β) using ψ(3) by fastforce
      moreover
      {
        fix α β γ
        have ⊢ (α → γ ⊔ β) → (γ ⊔ (α → β))
        proof -
          let ?φ = (⟨α⟩ → ⟨γ⟩ ⊔ ⟨β⟩) → (⟨γ⟩ ⊔ (⟨α⟩ → ⟨β⟩))
          have ∀𝔐. 𝔐 ⊨prop ?φ by fastforce
          hence ⊢ (| ?φ |) using propositional-semantics by blast
          thus ?thesis by simp
        qed
      }
      hence ⊢ (?α → ?γ ⊔ ?β) → (?γ ⊔ (?α → ?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-Propositional-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* ($\lambda\psi$. *uncurry* ($\to$) ψ = *snd* δ) Ψ = *Some* ψ
$\psi \in$ *set* Ψ
(*uncurry* ($\sqcup$)) ψ = *fst* ψ $\sqcup$ *snd* ψ
using *find-Some-set-membership*
by *fastforce*
let *?*φ = *fst* ψ $\sqcup$ (*fst* ψ $\to$ *fst* δ) $\to$ *snd* ψ
let *?*Ψ' = *remove1* ψ Ψ
have *map* (*uncurry* ($\sqcup$)) *?*Ψ' $\preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Y}$ *?*Ψ' Δ)
using *Cons* **by** *simp*
moreover
{
fix α β γ
have $\vdash$ ($\alpha \sqcup (\alpha \to \gamma) \to \beta) \to (\alpha \sqcup \beta)$
proof −
let *?*φ = $(\langle\alpha\rangle \sqcup (\langle\alpha\rangle \to \langle\gamma\rangle) \to \langle\beta\rangle) \to (\langle\alpha\rangle \sqcup \langle\beta\rangle)$
have $\forall \mathfrak{M}$. $\mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ (| *?*φ |) **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
hence $\vdash$ *?*φ $\to$ (*uncurry* ($\sqcup$)) ψ **using** ψ(*3*) **by** *auto*
ultimately
have *map* (*uncurry* ($\sqcup$)) (ψ # *?*Ψ') $\preceq$ (*?*φ # *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Y}$ *?*Ψ' Δ))
by (*simp add*: *stronger-theory-left-right-cons*)
moreover
from ψ **have** *mset* (*map* (*uncurry* ($\sqcup$)) (ψ # *?*Ψ')) = *mset* (*map* (*uncurry* ($\sqcup$)) 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** *Minimal-Logic*) *XWitness-secondComponent-diff-decomposition*:
mset ($\mathfrak{X}$ Ψ Δ) = *mset* ($\mathfrak{X}_\bullet$ Ψ Δ @ $\Delta \ominus \mathfrak{B}$ Ψ Δ)
proof −
have $\forall$ Ψ. *mset* ($\mathfrak{X}$ Ψ Δ) = *mset* ($\mathfrak{X}_\bullet$ 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 (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 Minimal-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 (Ψ ⊖ 𝔄 Ψ (δ # Δ) @ 𝔜• Ψ (δ # Δ))
      using Cons
        by (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None, simp, fastforce)
    }
    then show ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in Minimal-Logic) YWitness-right-stronger-theory:
    map (uncurry (→)) Δ ≼ map (uncurry (→)) (𝔜 Ψ Δ ⊖ (Ψ ⊖ 𝔄 Ψ Δ) @ (Δ
⊖ 𝔅 Ψ Δ))
proof −
  let ?𝔣 = λΨ Δ. (Ψ ⊖ 𝔄 Ψ Δ)
  let ?𝔤 = λ Ψ Δ. (Δ ⊖ 𝔅 Ψ Δ)
  have ∀ Ψ. map (uncurry (→)) Δ ≼  map (uncurry (→)) (𝔜 Ψ Δ ⊖ ?𝔣 Ψ Δ @
?𝔤 Ψ Δ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    let ?δ = (uncurry (→)) 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 $(cases\ find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None)$
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
$\forall \Psi\ \Phi.\ \Psi \preceq \Phi$
$= (\exists \Sigma.\ map\ snd\ \Sigma = \Psi$
$\wedge\ mset\ (map\ fst\ \Sigma) \subseteq\#\ mset\ \Phi$
$\wedge\ (\forall \xi.\ \xi \notin set\ \Sigma \vee \vdash (uncurry\ (\rightarrow)\ \xi)))$
by (*simp add*: *Ball-def-raw stronger-theory-relation-def*)
moreover have
$((uncurry\ (\rightarrow)\ \delta)\ \#\ map\ (uncurry\ (\rightarrow))\ \Delta)$
$\preceq ((uncurry\ (\rightarrow)\ \delta)\ \#\ map\ (uncurry\ (\rightarrow))\ (\mathfrak{Y}\ \Psi\ \Delta \ominus (?\mathfrak{f}\ \Psi\ \Delta))$
$@\ map\ (uncurry\ (\rightarrow))\ (?\mathfrak{g}\ \Psi\ \Delta))$
using *calculation* **by** *auto*
ultimately show *?thesis*
by (*simp*, *metis union-mset-add-mset-right*)
next
case *False*
from *this* **obtain** ψ **where**
ψ: $find\ (\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
$uncurry\ (\rightarrow)\ \psi = snd\ \delta$
using *find-Some-predicate*
by *fastforce*
let $?\alpha = fst\ \psi$
let $?\beta = fst\ \delta$
let $?\gamma = snd\ \psi$
have $(\lambda\ \delta.\ fst\ \delta \rightarrow snd\ \delta) = uncurry\ (\rightarrow)$ **by** *fastforce*
hence $?\beta \rightarrow\ ?\alpha \rightarrow\ ?\gamma = uncurry\ (\rightarrow)\ \delta$ **using** $\psi(2)$ **by** *metis*
moreover
let $?A = \mathfrak{Y}\ (remove1\ \psi\ \Psi)\ \Delta$
let $?B = \mathfrak{A}\ (remove1\ \psi\ \Psi)\ \Delta$

let *?C* = $\mathfrak{B}$ (*remove1* ψ Ψ) Δ
let *?D* = *?A* $\ominus$ ((*remove1* ψ Ψ) $\ominus$ *?B*)
have *mset* ((*remove1* ψ Ψ) $\ominus$ *?B*) $\subseteq\#$ *mset* *?A*
  using *YWitness-firstComponent-diff-decomposition* **by** *simp*
hence *mset* (*map* (*uncurry* ($\to$))
  (((*?*α, (*?*α $\to$ *?*β) $\to$ *?*γ) # *?A*) $\ominus$ *remove1* ψ (Ψ $\ominus$ *?B*)
  @ (*remove1* δ ((δ # Δ) $\ominus$ *?C*))))
  = *mset* ((*?*α $\to$ (*?*α $\to$ *?*β) $\to$ *?*γ) # *map* (*uncurry* ($\to$)) (*?D* @ (Δ $\ominus$
?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 $\vdash$ (*?*α $\to$ (*?*α $\to$ *?*β) $\to$ *?*γ) $\to$ *?*β $\to$ *?*α $\to$ *?*γ
proof −
  let *?*Γ = [(*?*α $\to$ (*?*α $\to$ *?*β) $\to$ *?*γ), *?*β, *?*α]
  have *?*Γ :$\vdash$ *?*α $\to$ (*?*α $\to$ *?*β) $\to$ *?*γ
    ?Γ :$\vdash$ *?*α
    by (*simp add*: *list-deduction-reflection*)+
  hence *?*Γ :$\vdash$ (*?*α $\to$ *?*β) $\to$ *?*γ
    using *list-deduction-modus-ponens* **by** *blast*
  moreover have *?*Γ :$\vdash$ *?*β
    by (*simp add*: *list-deduction-reflection*)
  hence *?*Γ :$\vdash$ *?*α $\to$ *?*β
    using *Axiom-1* *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 $\to$ *?*α $\to$ *?*γ # *map* (*uncurry* ($\to$)) Δ) $\preceq$
  (*?*α $\to$ (*?*α $\to$ *?*β) $\to$ *?*γ # *map* (*uncurry* ($\to$)) (*?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** *Minimal-Logic*) *XComponent-YComponent-connection*:
  map (*uncurry* ($\to$)) ($\mathfrak{X}_\bullet$ Ψ Δ) = *map* *snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)
proof −
  have $\forall$ Ψ. *map* (*uncurry* ($\to$)) ($\mathfrak{X}_\bullet$ 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 (→)) (𝔛• Ψ (δ # Δ)) = map snd (𝔜• Ψ (δ # Δ))
        using Cons
        by (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None, simp, fastforce)
    }
    then show ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in Classical-Propositional-Logic) XWitness-YWitness-segmented-deduction-intro:
  assumes mset (map snd Ψ) ⊆# mset Γ
      and 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 (→)) (𝔜 Ψ Δ) @
                (map (uncurry (→)) (𝔛 Ψ Δ) @ Γ ⊖ map snd (𝔛 Ψ Δ)) ⊖
                 map snd (𝔜 Ψ Δ) $⊢ Φ
          (is ?Γ $⊢ Φ)
proof −
  let ?A = map (uncurry (→)) (𝔜 Ψ Δ)
  let ?B = map (uncurry (→)) (𝔛 Ψ Δ)
  let ?C = Ψ ⊖ 𝔄 Ψ Δ
  let ?D = map (uncurry (→)) ?C
  let ?E = Δ ⊖ 𝔅 Ψ Δ
  let ?F = map (uncurry (→)) ?E
  let ?G = map snd (𝔅 Ψ Δ)
  let ?H = map (uncurry (→)) (𝔛• Ψ Δ)
  let ?I = 𝔄 Ψ Δ
  let ?J = map snd (𝔛 Ψ Δ)
  let ?K = map snd (𝔜 Ψ Δ)
  have mset (map (uncurry (→)) (𝔜 Ψ Δ ⊖ ?C @ ?E)) = mset (?A ⊖ ?D @ ?F)
    by (simp add: YWitness-firstComponent-diff-decomposition)
  hence (map (uncurry (→)) Δ) ≼ (?A ⊖ ?D @ ?F)
    using YWitness-right-stronger-theory
          stronger-theory-relation-alt-def
    by (simp, metis (no-types, lifting))
  hence ?Γ₀ ≼ ((?A ⊖ ?D @ ?F) @ (map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ)
⊖ map snd Δ)
    using stronger-theory-combine stronger-theory-reflexive by blast
  moreover
```

have ♠: *mset ?G* $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
mset ($\mathfrak{B}$ Ψ Δ) $\subseteq$# *mset* Δ
mset (*map snd ?E*) $\subseteq$# *mset* ($\Gamma \ominus$ *map snd* Ψ)
mset (*map* (*uncurry* ($\rightarrow$)) $\Psi \ominus$ *?G*) = *mset ?D*
mset ?D $\subseteq$# *mset ?A*
mset (*map snd ?I*) $\subseteq$# *mset* (*map snd* Ψ)
mset (*map snd ?I*) $\subseteq$# *mset* Γ
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* Δ − *mset* ($\mathfrak{B}$ Ψ Δ) + *mset* ($\mathfrak{B}$ Ψ Δ) = *mset* Δ
by *simp*
hence ♡: {#$x \rightarrow y$. (x, y) ∈# *mset* Ψ#} + (*mset* Γ − *image-mset snd* (*mset*
Ψ))
− *image-mset snd* (*mset* Δ)
= {#$x \rightarrow y$. (x, y) ∈# *mset* Ψ#} + (*mset* Γ − *image-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 @ $\Gamma \ominus$ *map snd* Ψ) $\ominus$ *map snd* Δ)
= *mset* (*?D* @ ($\Gamma \ominus$ *?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* @ ($\Gamma \ominus$ *?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* ($\Psi \ominus$ *?I*)) $\subseteq$# *mset* ($\Gamma \ominus$ *?J*)
by (*simp add*: *XWitness-secondComponent-diff-decomposition*,
simp add: *YWitness-firstComponent-diff-decomposition*,
simp, *metis* (*no-types*, *lifting*)
$\heartsuit$(*2*) $\spadesuit$(*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* @ ($\Gamma \ominus$ *?J*) $\ominus$ *map snd ?C*)
= *mset* (*?A* @ (*?B* $\ominus$ *?H* @ $\Gamma \ominus$ *?J*) $\ominus$ *map snd ?C*)
mset ?H $\subseteq$# *mset ?B*
{#*x* $\to$ *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* @ ($\Gamma \ominus$ *?J*) $\ominus$ *map snd ?C*)
= *mset* (*?A* @ (*?B* @ $\Gamma \ominus$ *?J*) $\ominus$ (*?H* @ *map snd ?C*))
{#*x* $\to$ *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* @ ($\Gamma \ominus$ *?J*) $\ominus$ *map snd ?C*)
= *mset* (*?A* @ (*?B* @ $\Gamma \ominus$ *?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-Propositional-Logic*) *segmented-cons-cons-right-permute*:
assumes Γ \$$\vdash$ (φ # ψ # Φ)
shows Γ \$$\vdash$ (ψ # φ # Φ)
proof −
from *assms* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq$# *mset* Γ
map (*uncurry* ($\sqcup$)) Ψ :$\vdash$ φ
map (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ) \$$\vdash$ (ψ # Φ)
by *fastforce*
let *?*Γ_0 = *map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ)
from Ψ(*3*) **obtain** Δ **where** Δ:
mset (*map snd* Δ) $\subseteq$# *mset ?*Γ_0
map (*uncurry* ($\sqcup$)) Δ :$\vdash$ ψ

```
    (map (uncurry (→)) Δ @ ?Γ₀ ⊖ (map snd Δ)) $⊢ Φ
    using segmented-deduction.simps(2) by blast
  let ?Ψ' = 𝔛 Ψ Δ
  let ?Γ₁ = map (uncurry (→)) ?Ψ' @ Γ ⊖ (map snd ?Ψ')
  let ?Δ' = 𝔜 Ψ Δ
  have (map (uncurry (→)) ?Δ' @ ?Γ₁ ⊖ (map snd ?Δ')) $⊢ Φ
       map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) ?Δ'
    using Ψ(1) Δ(1) Δ(3)
          XWitness-YWitness-segmented-deduction-intro
          YWitness-left-stronger-theory
    by auto
  hence ?Γ₁ $⊢ (φ # Φ)
    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-Propositional-Logic) segmented-cons-remove1:
  assumes φ ∈ set Φ
    shows Γ $⊢ Φ = Γ $⊢ (φ # (remove1 φ Φ))
proof –
  from ‹φ ∈ set Φ›
  have ∀ Γ. Γ $⊢ Φ = Γ $⊢ (φ # (remove1 φ Φ))
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons χ Φ)
    {
      fix Γ
      have Γ $⊢ (χ # Φ) = Γ $⊢ (φ # (remove1 φ (χ # Φ)))
      proof (cases χ = φ)
        case True
        then show ?thesis by simp
      next
        case False
        hence φ ∈ set Φ
          using Cons.prems by simp
        with Cons.hyps have Γ $⊢ (χ # Φ) = Γ $⊢ (χ # φ # (remove1 φ Φ))
          by fastforce
        hence Γ $⊢ (χ # Φ) = Γ $⊢ (φ # χ # (remove1 φ Φ))
```

```
        using segmented-cons-cons-right-permute by blast
      then show ?thesis using ‹χ ≠ φ› by simp
    qed
  }
  then show ?case by blast
 qed
 thus ?thesis using assms by blast
qed

lemma (in Classical-Propositional-Logic) witness-stronger-theory:
 assumes mset (map snd Ψ) ⊆# mset Γ
 shows (map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)) ≼ Γ
proof −
 have ∀ Γ. mset (map snd Ψ) ⊆# mset Γ ⟶ (map (uncurry (→)) Ψ @ Γ ⊖
(map snd Ψ)) ≼ Γ
 proof (induct Ψ)
  case Nil
  then show ?case by simp
 next
  case (Cons ψ Ψ)
  let ?γ = snd ψ
  {
   fix Γ
   assume mset (map snd (ψ # Ψ)) ⊆# mset Γ
   hence mset (map snd Ψ) ⊆# mset (remove1 (snd ψ) Γ)
    by (simp add: insert-subset-eq-iff)
   with Cons have
    (map (uncurry (→)) Ψ @ (remove1 (snd ψ) Γ) ⊖ (map snd Ψ)) ≼ (remove1
?γ Γ)
    by blast
   hence (map (uncurry (→)) Ψ @ Γ ⊖ (map snd (ψ # Ψ))) ≼ (remove1 ?γ Γ)
    by (simp add: stronger-theory-relation-alt-def)
   moreover
   have (uncurry (→)) = (λ ψ. fst ψ → snd ψ)
    by fastforce
   hence ⊢ ?γ → uncurry (→) ψ
    using Axiom-1 by simp
   ultimately have
    (map (uncurry (→)) (ψ # Ψ) @ Γ ⊖ (map snd (ψ # Ψ))) ≼ (?γ # (remove1
?γ Γ))
    using stronger-theory-left-right-cons by auto
   hence (map (uncurry (→)) (ψ # Ψ) @ Γ ⊖ (map snd (ψ # Ψ))) ≼ Γ
    using stronger-theory-relation-alt-def
       ‹mset (map snd (ψ # Ψ)) ⊆# mset Γ›
       mset-subset-eqD
    by fastforce
  }
  then show ?case by blast
 qed
```

```
  thus ?thesis using assms by blast
qed

lemma (in Classical-Propositional-Logic) segmented-msub-weaken:
  assumes mset Ψ ⊆# mset Φ
      and Γ $⊢ Φ
    shows Γ $⊢ Ψ
proof −
  have ∀ Ψ Γ. mset Ψ ⊆# mset Φ ⟶ Γ $⊢ Φ ⟶ Γ $⊢ Ψ
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons φ Φ)
    {
      fix Ψ Γ
      assume mset Ψ ⊆# mset (φ # Φ)
             Γ $⊢ (φ # Φ)
      hence Γ $⊢ Φ
        using segmented-deduction.simps(2)
              segmented-stronger-theory-left-monotonic
              witness-stronger-theory
        by blast
      have Γ $⊢ Ψ
      proof (cases φ ∈ set Ψ)
        case True
        hence mset (remove1 φ Ψ) ⊆# mset Φ
          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-Propositional-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) $\subseteq$# *mset* Γ
map (*uncurry* ($\sqcup$)) Δ :$\vdash$ φ
(*map* (*uncurry* ($\rightarrow$)) Δ @ Γ $\ominus$ (*map snd* Δ)) \$$\vdash$ Φ
by *auto*
ultimately have (*map* (*uncurry* ($\rightarrow$)) Δ @ Γ $\ominus$ (*map snd* Δ)) \$$\vdash$ *remove1* ψ Ψ
using *Cons* **by** *blast*
moreover have *map* (*uncurry* ($\sqcup$)) Δ :$\vdash$ ψ
using Δ(*2*) Σ(*3*) ‹(φ,ψ) $\in$ *set* Σ›
list-deduction-weaken
list-deduction-modus-ponens
by *blast*
ultimately have ‹Γ \$$\vdash$ (ψ # (*remove1* ψ Ψ))›
using Δ(*1*) **by** *auto*
moreover from ‹(φ,ψ) $\in$ *set* Σ› Σ(*1*) **have** ψ $\in$ *set* Ψ
by *force*
hence *mset* Ψ $\subseteq$# *mset* (ψ # (*remove1* ψ Ψ))
by *auto*
ultimately show *?thesis* **using** *segmented-msub-weaken* **by** *blast*
next
case *False*
hence *mset* (*map fst* Σ) $\subseteq$# *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 Ψ $\preceq$ Φ
using Σ(*1*) Σ(*3*)
unfolding *stronger-theory-relation-def*
by *auto*
moreover from ‹Γ \$$\vdash$ (φ # Φ)› **have** Γ \$$\vdash$ Φ
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-Propositional-Logic*) *segmented-witness-right-split*:
assumes *mset* (*map snd* Ψ) $\subseteq$# *mset* Φ
shows Γ \$$\vdash$ (*map* (*uncurry* ($\sqcup$)) Ψ @ *map* (*uncurry* ($\rightarrow$)) Ψ @ Φ $\ominus$ (*map snd* Ψ)) = Γ \$$\vdash$ Φ

proof −
  have $\forall\ \Gamma\ \Phi.\ mset\ (map\ snd\ \Psi) \subseteq\#\ mset\ \Phi \longrightarrow$
     $\Gamma\ \$\vdash\ \Phi = \Gamma\ \$\vdash\ (map\ (uncurry\ (\sqcup))\ \Psi\ @\ map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \Phi \ominus (map$
$snd\ \Psi))$
  proof (*induct* Ψ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ψ Ψ)
    {
      fix $\Gamma\ \Phi$
      let $?\chi = fst\ \psi$
      let $?\varphi = snd\ \psi$
      let $?\Phi' = map\ (uncurry\ (\sqcup))\ (\psi\ \#\ \Psi)\ @$
             $map\ (uncurry\ (\rightarrow))\ (\psi\ \#\ \Psi)\ @$
             $\Phi \ominus map\ snd\ (\psi\ \#\ \Psi)$
      let $?\Phi_0 = map\ (uncurry\ (\sqcup))\ \Psi\ @$
             $map\ (uncurry\ (\rightarrow))\ \Psi\ @$
             $(remove1\ ?\varphi\ \Phi) \ominus map\ snd\ \Psi$
      assume $mset\ (map\ snd\ (\psi\ \#\ \Psi)) \subseteq\#\ mset\ \Phi$
      hence $mset\ (map\ snd\ \Psi) \subseteq\#\ mset\ (remove1\ ?\varphi\ \Phi)$
         $mset\ (?\varphi\ \#\ remove1\ ?\varphi\ \Phi) = mset\ \Phi$
        by (*simp add*: *insert-subset-eq-iff*)+
      hence $\Gamma\ \$\vdash\ \Phi = \Gamma\ \$\vdash\ (?\varphi\ \#\ remove1\ ?\varphi\ \Phi)$
         $\forall\ \Gamma.\ \Gamma\ \$\vdash\ (remove1\ ?\varphi\ \Phi) = \Gamma\ \$\vdash\ ?\Phi_0$
        by (*metis list.set-intros*(*1*) *segmented-cons-remove1 set-mset-mset*,
          metis Cons.hyps)
      moreover
      have $(uncurry\ (\sqcup)) = (\lambda\ \psi.\ fst\ \psi \sqcup snd\ \psi)$
         $(uncurry\ (\rightarrow)) = (\lambda\ \psi.\ fst\ \psi \rightarrow snd\ \psi)$
        by *fastforce*+
      hence $mset\ ?\Phi' \subseteq\#\ mset\ (?\chi \sqcup ?\varphi\ \#\ ?\chi \rightarrow ?\varphi\ \#\ ?\Phi_0)$
         $mset\ (?\chi \sqcup ?\varphi\ \#\ ?\chi \rightarrow ?\varphi\ \#\ ?\Phi_0) \subseteq\#\ mset\ ?\Phi'$
         (**is** $mset\ ?X \subseteq\#\ mset\ ?Y$)
        by *fastforce*+
      hence $\Gamma\ \$\vdash\ ?\Phi' = \Gamma\ \$\vdash\ (?\varphi\ \#\ ?\Phi_0)$
        using *segmented-formula-right-split*
          segmented-msub-weaken
        by *blast*
      ultimately have $\Gamma\ \$\vdash\ \Phi = \Gamma\ \$\vdash\ ?\Phi'$
        by *fastforce*
    }
    then show *?case* **by** *blast*
  qed
  with *assms* **show** *?thesis* **by** *blast*
qed

primrec (**in** *Classical-Propositional-Logic*)
  $submergeWitness :: ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{E})$

```
  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-Propositional-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 (⊔)) Σ ≼ map (uncurry (⊔)) (𝔈 Σ (δ # Δ))
      proof (cases find (λ σ. (uncurry (→)) σ = snd δ) Σ = None)
        case True
        then show ?thesis using Cons by simp
      next
        case False
        from this obtain σ where
          σ: find (λσ. uncurry (→) σ = snd δ) Σ = Some σ
             uncurry (→) σ = snd δ
             σ ∈ set Σ
          using find-Some-predicate find-Some-set-membership
          by fastforce
        {
          fix α β γ
          have ⊢ (α ⊔ (γ ⊓ α) ⊔ β) → (α ⊔ β)
          proof −
```

let $?\varphi = (\langle\alpha\rangle \sqcup (\langle\gamma\rangle \sqcap \langle\alpha\rangle) \sqcup \langle\beta\rangle) \to (\langle\alpha\rangle \sqcup \langle\beta\rangle)$
have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
note *tautology = this*
let $?\alpha = fst\ \sigma$
let $?\beta = snd\ \sigma$
let $?\gamma = fst\ \delta$
have $(uncurry\ (\sqcup)) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$ **by** *fastforce*
hence $(uncurry\ (\sqcup))\ \sigma =\ ?\alpha \sqcup\ ?\beta$ **by** *simp*
hence A: $\vdash (?\alpha \sqcup (?\gamma \sqcap ?\alpha) \sqcup ?\beta) \to (uncurry\ (\sqcup))\ \sigma$ **using** *tautology* **by** *simp*
moreover
have $map\ (uncurry\ (\sqcup))\ (remove1\ \sigma\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{E}\ (remove1\ \sigma\ \Sigma)\ \Delta)$
using *Cons* **by** *simp*
ultimately have A:
$map\ (uncurry\ (\sqcup))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
$\preceq (?\alpha \sqcup (?\gamma \sqcap ?\alpha) \sqcup ?\beta\ \#\ map\ (uncurry\ (\sqcup))\ (\mathfrak{E}\ (remove1\ \sigma\ \Sigma)\ \Delta))$
using *stronger-theory-left-right-cons* **by** *fastforce*
from $\sigma(3)$ **have** $mset\ \Sigma = mset\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
by *simp*
hence $mset\ (map\ (uncurry\ (\sqcup))\ \Sigma) = mset\ (map\ (uncurry\ (\sqcup))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)))$
by (*metis mset-map*)
hence B: $map\ (uncurry\ (\sqcup))\ \Sigma \preceq map\ (uncurry\ (\sqcup))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
by (*simp add*: *msub-stronger-theory-intro*)
have $(\ \ fst\ \sigma$
$\sqcup (fst\ \delta \sqcap fst\ \sigma)$
$\sqcup\ snd\ \sigma\ \#\ map\ (\lambda(x,\ y).\ x \sqcup y)\ (\mathfrak{E}\ (remove1\ \sigma\ \Sigma)\ \Delta)) \succeq map\ (\lambda(x,\ y).\ x \sqcup y)\ \Sigma$
by (*metis* (*no-types*, *hide-lams*) *A B stronger-theory-transitive uncurry-def*)
thus *?thesis* **using** $A\ B\ \sigma$ **by** *simp*
qed
}
then show *?case* **by** *auto*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *submergeWitness-msub*:
$mset\ (map\ snd\ (\mathfrak{E}\ \Sigma\ \Delta)) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta))$
proof –
have $\forall\ \Sigma.\ mset\ (map\ snd\ (\mathfrak{E}\ \Sigma\ \Delta)) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta))$
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-Propositional-Logic) submergeWitness-stronger-theory-right:
    map (uncurry (⊔)) Δ
  ≼ (map (uncurry (→)) (𝔈 Σ Δ) @ map (uncurry (⊔)) (𝔍 Σ Δ) ⊖ map snd (𝔈 Σ
Δ))
proof −
  have ∀ Σ. map (uncurry (⊔)) Δ
            ≼ (map (uncurry (→)) (𝔈 Σ Δ) @ map (uncurry (⊔)) (𝔍 Σ Δ)  ⊖ map
snd (𝔈 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 (→)) (𝔈 Σ (δ # Δ))
              @ map (uncurry (⊔)) (𝔍 Σ (δ # Δ))
                ⊖ map snd (𝔈 Σ (δ # Δ)))
      proof (cases find (λ σ. (uncurry (→)) σ = snd δ) Σ = None)
        case True
        from Cons obtain Φ where Φ:
          map snd Φ = map (uncurry (⊔)) Δ
```

mset (*map fst* Φ) $\subseteq\#$
mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{C}$ Σ Δ)
@ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{C}$ Σ Δ))
$\forall(\gamma, \sigma)\in$*set* Φ. $\vdash \gamma \rightarrow \sigma$
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* ($\mathfrak{C}$ Σ Δ)#}
+ ({#*x* $\sqcup$ *y*. (*x*, *y*) $\in\#$ *mset* ($\mathfrak{J}$ Σ Δ)#} $-$ *image-mset snd* (*mset* ($\mathfrak{C}$ Σ
Δ)))
by *simp*
have *image-mset snd* (*mset* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ Σ Δ)))
$\subseteq\#$ {#*x* $\rightarrow$ *y*. (*x*, *y*) $\in\#$ *mset* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ 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* ($\mathfrak{C}$ Σ Δ)))
using *subset-eq-diff-conv* **by** *blast*
hence
mset (*map fst ?*Φ') $\subseteq\#$
mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{C}$ Σ (δ # Δ))
@ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ (δ # Δ))
$\ominus$ *map snd* ($\mathfrak{C}$ Σ (δ # Δ)))

using *True* $\Phi(2)$
by *simp*
moreover have $\forall(\gamma, \sigma)\in set\ ?\Phi'. \vdash \gamma \rightarrow \sigma$
using $\Phi(3)$ *trivial-implication* **by** *auto*
ultimately show *?thesis*
unfolding *stronger-theory-relation-def*
by *blast*
next
case *False*
from *this* **obtain** σ **where**
σ: $find\ (\lambda\sigma.\ uncurry\ (\rightarrow)\ \sigma = snd\ \delta)\ \Sigma = Some\ \sigma$
$uncurry\ (\rightarrow)\ \sigma = snd\ \delta$
using *find-Some-predicate*
by *fastforce*
moreover from *Cons* **have**
$map\ (uncurry\ (\sqcup))\ \Delta \preceq$
$(map\ (uncurry\ (\rightarrow))\ (\mathfrak{E}\ (remove1\ \sigma\ \Sigma)\ \Delta)\ @$
$remove1\ ((fst\ \delta \sqcap fst\ \sigma) \sqcup snd\ \sigma)$
$(((fst\ \delta \sqcap fst\ \sigma) \sqcup snd\ \sigma\ \#\ map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ (remove1\ \sigma\ \Sigma)\ \Delta))$
$\ominus map\ snd\ (\mathfrak{E}\ (remove1\ \sigma\ \Sigma)\ \Delta)))$
unfolding *stronger-theory-relation-alt-def*
by *simp*
moreover
{
fix $\alpha\ \beta\ \gamma$
have $\vdash (\alpha \rightarrow ((\gamma \sqcap \alpha) \sqcup \beta)) \rightarrow (\gamma \sqcup (\alpha \rightarrow \beta))$
proof $-$
let $?\varphi = (\langle\alpha\rangle \rightarrow ((\langle\gamma\rangle \sqcap \langle\alpha\rangle) \sqcup \langle\beta\rangle)) \rightarrow (\langle\gamma\rangle \sqcup (\langle\alpha\rangle \rightarrow \langle\beta\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **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-Propositional-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-Propositional-Logic*)
recoverWitnessA :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{P})$
where
$\mathfrak{P}\ \Sigma\ []\ = \Sigma$
$|\ \mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta) =$
$(case\ find\ (\lambda\ \sigma.\ snd\ \sigma = (uncurry\ (\sqcup))\ \delta)\ \Sigma\ of$
$None \Rightarrow \mathfrak{P}\ \Sigma\ \Delta$
$|\ Some\ \sigma \Rightarrow (fst\ \sigma \sqcup fst\ \delta,\ snd\ \delta)\ \#\ (\mathfrak{P}\ (remove1\ \sigma\ \Sigma)\ \Delta))$

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

primrec (**in** *Classical-Propositional-Logic*)
recoverWitnessB :: $('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list \Rightarrow ('a \times 'a)\ list\ (\mathfrak{Q})$
where
$\mathfrak{Q}\ \Sigma\ []\ = []$
$|\ \mathfrak{Q}\ \Sigma\ (\delta\ \#\ \Delta) =$

$(case\ find\ (\lambda\ \sigma.\ (snd\ \sigma) = (uncurry\ (\sqcup))\ \delta)\ \Sigma\ of$
$None \Rightarrow \delta\ \#\ \mathfrak{Q}\ \Sigma\ \Delta$
$\mid Some\ \sigma \Rightarrow (fst\ \delta,\ (fst\ \sigma \sqcup fst\ \delta) \to snd\ \delta)\ \#\ (\mathfrak{Q}\ (remove1\ \sigma\ \Sigma)\ \Delta))$

lemma (**in** *Classical-Propositional-Logic*) *recoverWitnessA-left-stronger-theory*:
$map\ (uncurry\ (\sqcup))\ \Sigma \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ \Delta)$
proof −
have $\forall\ \Sigma.\ map\ (uncurry\ (\sqcup))\ \Sigma \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ \Delta)$
proof (*induct* Δ)
case *Nil*
{
fix Σ
have $map\ (uncurry\ (\sqcup))\ \Sigma \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ [])$
by(*induct* Σ, *simp+*)
}
then show *?case* **by** *auto*
next
case (*Cons* δ Δ)
{
fix Σ
have $map\ (uncurry\ (\sqcup))\ \Sigma \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta))$
proof ($cases\ find\ (\lambda\ \sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = None$)
case *True*
then show *?thesis* **using** *Cons* **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*
let $?\alpha = fst\ \sigma$
let $?\beta = fst\ \delta$
let $?\gamma = snd\ \delta$
have $uncurry\ (\sqcup) = (\lambda\delta.\ fst\ \delta \sqcup snd\ \delta)$ **by** *fastforce*
hence $\vdash ((?\alpha \sqcup ?\beta) \sqcup ?\gamma) \to uncurry\ (\sqcup)\ \sigma$
using $\sigma(2)$ *biconditional-def disjunction-associativity*
by *auto*
moreover
have $map\ (uncurry\ (\sqcup))\ (remove1\ \sigma\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ (remove1\ \sigma\ \Sigma)\ \Delta)$
using *Cons* **by** *simp*
ultimately have $map\ (uncurry\ (\sqcup))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta))$
using $\sigma(1)$
by (*simp*, *metis stronger-theory-left-right-cons*)
moreover

from $\sigma(3)$ **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 *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq$ *map* (*uncurry* ($\sqcup$)) (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-Propositional-Logic*) *recoverWitnessA-mset-equiv*:
assumes *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
shows *mset* (*map snd* ($\mathfrak{P}$ Σ Δ @ $\mathfrak{P}^C$ Σ Δ)) = *mset* (*map snd* Δ)
proof −
have $\forall$ Σ. *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
$\longrightarrow$ *mset* (*map snd* ($\mathfrak{P}$ Σ Δ @ $\mathfrak{P}^C$ 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* $(\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
$snd\ \sigma = uncurry\ (\sqcup)\ \delta$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
have *A*: $mset\ (map\ snd\ \Sigma)$
$\subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta) + add\text{-}mset\ (uncurry\ (\sqcup)\ \delta)\ (mset\ [])$
using $\star$ **by** *auto*
have $(fst\ \sigma,\ uncurry\ (\sqcup)\ \delta) \in\#\ mset\ \Sigma$
by (*metis* (*no-types*) $\sigma(2)$ $\sigma(3)$ *prod.collapse set-mset-mset*)
then have *B*: $mset\ (map\ snd\ (remove1\ (fst\ \sigma,\ uncurry\ (\sqcup)\ \delta)\ \Sigma))$
$= mset\ (map\ snd\ \Sigma) - \{\#uncurry\ (\sqcup)\ \delta\#\}$
by (*meson remove1-pairs-list-projections-snd*)
have $(fst\ \sigma,\ uncurry\ (\sqcup)\ \delta) = \sigma$
by (*metis* $\sigma(2)$ *prod.collapse*)
then have $mset\ (map\ snd\ \Sigma) - add\text{-}mset\ (uncurry\ (\sqcup)\ \delta)\ (mset\ [])$
$= mset\ (map\ snd\ (remove1\ \sigma\ \Sigma))$
using *B* **by** *simp*
hence $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
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-Propositional-Logic*) *recoverWitnessB-stronger-theory*:
assumes $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
shows $(map\ (uncurry\ (\to))\ \Sigma\ @\ map\ (uncurry\ (\sqcup))\ \Delta \ominus map\ snd\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ \Delta)$
proof –
have $\forall\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
$\longrightarrow (map\ (uncurry\ (\to))\ \Sigma\ @\ map\ (uncurry\ (\sqcup))\ \Delta \ominus map\ snd\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ \Delta)$
proof(*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix $\Sigma :: ('a \times 'a)\ list$

```
assume ⋆: mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) (δ # Δ))
have (map (uncurry (→)) Σ @ map (uncurry (⊔)) (δ # Δ) ⊖ map snd Σ)
    ≼ map (uncurry (⊔)) (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
  hence mset (map (uncurry (→)) Σ @ (map (uncurry (⊔)) (δ # Δ)) ⊖ map
snd Σ)
        = mset (uncurry (⊔) δ # map (uncurry (→)) Σ
            @ map (uncurry (⊔)) Δ ⊖ map snd Σ)
        mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
    using ⋆
    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 (→)) Σ @ map (uncurry (⊔)) Δ ⊖ map snd Σ)
     ≼ map (uncurry (⊔)) (𝔔 Σ Δ)
    using Cons
    by auto
  hence (uncurry (⊔) δ # map (uncurry (→)) Σ @ map (uncurry (⊔)) Δ ⊖
map snd Σ)
         ≼ map (uncurry (⊔)) (𝔔 Σ (δ # Δ))
    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 ?Γ = map (uncurry (→)) Σ @ (map (uncurry (⊔)) (δ # Δ)) ⊖ map
snd Σ
  from False obtain σ where
    σ: find (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
       snd σ = uncurry (⊔) δ
       σ ∈ set Σ
    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 $\sigma(2)$ ‹$uncurry\ (\rightarrow)\ \sigma = ?\alpha \rightarrow (?\beta \sqcup ?\gamma)$›
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** ♡: $?\Gamma_0 \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ (remove1\ \sigma\ \Sigma)\ \Delta)$ **by** *simp*
{
fix $\alpha\ \beta\ \gamma$
have $\vdash (\beta \sqcup (\alpha \sqcup \beta) \rightarrow \gamma) \rightarrow (\alpha \rightarrow (\beta \sqcup \gamma))$
proof −
let $?\varphi = (\langle\beta\rangle \sqcup (\langle\alpha\rangle \sqcup \langle\beta\rangle) \rightarrow \langle\gamma\rangle) \rightarrow (\langle\alpha\rangle \rightarrow (\langle\beta\rangle \sqcup \langle\gamma\rangle))$
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 $\vdash (?\beta \sqcup (?\alpha \sqcup ?\beta) \rightarrow ?\gamma) \rightarrow (?\alpha \rightarrow (?\beta \sqcup ?\gamma))$

by *simp*
hence $(?\alpha \rightarrow (?\beta \sqcup ?\gamma) \# ?\Gamma_0) \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ (\delta \# \Delta))$
using $\sigma(1)$ ♡
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-Propositional-Logic*) *recoverWitnessB-mset-equiv*:
assumes $mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
shows $mset\ (map\ snd\ (\mathfrak{Q}\ \Sigma\ \Delta))$
$= mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ \Delta)\ @\ map\ snd\ \Delta \ominus map\ snd\ (\mathfrak{P}\ \Sigma\ \Delta))$
proof −
have $\forall\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
$\longrightarrow\ mset\ (map\ snd\ (\mathfrak{Q}\ \Sigma\ \Delta)) = mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ \Delta)\ @$
$map\ snd\ (\mathfrak{P}^C\ \Sigma\ \Delta))$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix $\Sigma :: ('a \times 'a)\ list$
assume ⋆: $mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ (\delta \# \Delta))$
have $mset\ (map\ snd\ (\mathfrak{Q}\ \Sigma\ (\delta \# \Delta)))$
$= mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ (\delta \# \Delta))\ @\ map\ snd\ (\mathfrak{P}^C\ \Sigma\ (\delta \# \Delta)))$
proof $(cases\ find\ (\lambda\ \sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = None)$
case *True*
hence $uncurry\ (\sqcup)\ \delta \notin set\ (map\ snd\ \Sigma)$
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
then show *?case*
by $(cases\ (uncurry\ (\sqcup))\ \delta = snd\ \sigma,\ fastforce+)$
qed
moreover have $mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ \Delta) +$
$\{\#uncurry\ (\sqcup)\ \delta\#\}$
using ⋆ **by** *force*
ultimately have $mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
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 (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
           snd σ = uncurry (⊔) δ
           σ ∈ set Σ
        using find-Some-predicate
              find-Some-set-membership
        by fastforce
      hence (fst σ, uncurry (⊔) δ) ∈# mset Σ
        by (metis (full-types) prod.collapse set-mset-mset)
      then have mset (map snd (remove1 (fst σ, uncurry (⊔) δ) Σ))
               = mset (map snd Σ) − {#uncurry (⊔) δ#}
        by (meson remove1-pairs-list-projections-snd)
      moreover have
      mset (map snd Σ)
   ⊆# mset (map (uncurry (⊔)) Δ) + add-mset (uncurry (⊔) δ) (mset [])
        using ⋆ by force
      ultimately have mset (map snd (remove1 σ Σ))
          ⊆# mset (map (uncurry (⊔)) 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-Propositional-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 ($\lambda\sigma$. *snd* σ = *uncurry* ($\sqcup$) δ) Σ = *Some* σ
by *fastforce*
let *?*α = *fst* δ
let *?*β = *snd* δ
let *?*γ = *fst* σ
have *uncurry* ($\rightarrow$) = ($\lambda\delta$. *fst* $\delta \rightarrow$ *snd* δ) **by** *fastforce*
hence *uncurry* ($\rightarrow$) δ = *?*$\alpha \rightarrow$ *?*β **by** *auto*
moreover have $\vdash$ (*?*$\alpha \rightarrow$ (*?*$\gamma \sqcup$ *?*α) $\rightarrow$ *?*β) $\rightarrow$ *?*$\alpha \rightarrow$ *?*β
unfolding *disjunction-def*
using *Axiom-1 Axiom-2 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-Propositional-Logic*) *recoverWitnesses-mset-equiv*:
assumes *mset* (*map snd* Δ) $\subseteq\#$ *mset* Γ
and *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
shows *mset* ($\Gamma \ominus$ *map snd* Δ)
= *mset* ((*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ Σ Δ) @ $\Gamma \ominus$ *map snd* ($\mathfrak{P}$ Σ Δ)) $\ominus$ *map snd* ($\mathfrak{Q}$ Σ Δ))
proof −
have *mset* ($\Gamma \ominus$ *map snd* Δ) = *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{P}^C$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{P}$ Σ Δ))
using *assms*(*2*) *recoverWitnessA-mset-equiv*
by (*simp add*: *union-commute*)
moreover have $\forall$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
$\longrightarrow$ *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{P}^C$ Σ Δ))
= (*mset* ((*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ Σ Δ) @ Γ) $\ominus$ *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* $\delta \in$ *set* Γ
using *mset-subset-eqD* **by** *fastforce*
from *Cons.prems* **have** $\heartsuit$: *mset* (*map snd* Δ) $\subseteq\#$ *mset* Γ
using *subset-mset.dual-order.trans*
by *fastforce*
{

fix Σ :: $('a \times 'a)$ *list*
assume $\star$: *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) (δ # Δ))
have *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{P}^C$ Σ (δ # Δ)))
= *mset* ((*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ Σ (δ # Δ)) @ Γ) $\ominus$ *map snd* ($\mathfrak{Q}$ 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
moreover have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ) +
{#*uncurry* ($\sqcup$) δ#}
using $\star$ **by** *auto*
ultimately have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
by (*metis* (*full-types*) *diff-single-trivial in-multiset-in-set subset-eq-diff-conv*)
with *Cons.hyps* $\heartsuit$ **have** *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{P}^C$ Σ Δ))
= *mset* ((*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ Σ Δ) @ Γ) $\ominus$ *map snd*
($\mathfrak{Q}$ Σ Δ))
by *simp*
thus *?thesis* **using** *True* ⟨*snd* $\delta \in$ *set* Γ⟩ **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*
with $\star$ **have** *mset* (*map snd* (*remove1* σ Σ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$))
Δ)
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* σ Σ) Δ))
= *mset* ((*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{P}$ (*remove1* σ Σ) Δ) @ Γ)
$\ominus$ *map snd* ($\mathfrak{Q}$ (*remove1* σ Σ) Δ))
using $\heartsuit$ **by** *blast*
then show *?thesis* **using** σ **by** *simp*
qed
}

then show *?case* **by** *blast*
qed
moreover have *image-mset snd* (*mset* ($\mathfrak{P}^C$ Σ Δ)) = *mset* (*map snd* $\Delta \ominus$ *map snd* ($\mathfrak{P}$ Σ Δ))
using *assms*(*2*) *recoverWitnessA-mset-equiv*
by (*simp*, *metis* (*no-types*) *diff-union-cancelL listSubtract-mset-homomorphism mset-map*)
then have *mset* Γ − (*image-mset snd* (*mset* ($\mathfrak{P}^C$ Σ Δ)) + *image-mset snd* (*mset* ($\mathfrak{P}$ Σ Δ)))
= {#*x* → *y*. (*x*, *y*) ∈# *mset* ($\mathfrak{P}$ Σ Δ)#}
+ (*mset* Γ − *image-mset snd* (*mset* ($\mathfrak{P}$ Σ Δ))) − *image-mset snd* (*mset* ($\mathfrak{Q}$ Σ Δ))
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-Propositional-Logic*) *segmented-deduction-generalized-witness*:
Γ \$⊢ (Φ @ Ψ) = (∃ Σ. *mset* (*map snd* Σ) ⊆# *mset* Γ ∧
map (*uncurry* (⊔)) Σ \$⊢ Φ ∧
(*map* (*uncurry* (→)) Σ @ Γ ⊖ (*map snd* Σ)) \$⊢ Ψ)
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 ?Γ₀ $⊢ (Φ @ Ψ))
        by auto
      from this(3) obtain Δ where
        Δ: mset (map snd Δ) ⊆# mset ?Γ₀
           map (uncurry (⊔)) Δ $⊢ Φ
           map (uncurry (→)) Δ @ ?Γ₀ ⊖ (map snd Δ) $⊢ Ψ
        using Cons
        by auto
      let ?Σ' = 𝔍 Σ Δ
      have map (uncurry (⊔)) ?Σ' $⊢ (φ # Φ)
        using Δ(1) Δ(2) Σ(2) mergeWitness-cons-segmented-deduction by blast
      moreover have mset (map snd ?Σ') ⊆# mset Γ
        using Δ(1) Σ(1) mergeWitness-msub-intro by blast
      moreover have map (uncurry (→)) ?Σ' @ Γ ⊖ map snd ?Σ' $⊢ Ψ
        using Δ(1) Δ(3) mergeWitness-segmented-deduction-intro by blast
      ultimately show
        ∃Σ. mset (map snd Σ) ⊆# mset Γ ∧
             map (uncurry (⊔)) Σ $⊢ (φ # Φ) ∧
             map (uncurry (→)) Σ @ Γ ⊖ map snd Σ $⊢ Ψ
        by fast
    next
```

assume $\exists\,\Sigma.$ *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
map (*uncurry* ($\sqcup$)) Σ $\$\vdash$ (φ $\#$ Φ) $\wedge$
map (*uncurry* ($\rightarrow$)) Σ @ Γ $\ominus$ *map snd* Σ $\$\vdash$ Ψ
from *this* **obtain** Δ **where** Δ:
mset (*map snd* Δ) $\subseteq\#$ *mset* Γ
map (*uncurry* ($\sqcup$)) Δ $\$\vdash$ (φ $\#$ Φ)
map (*uncurry* ($\rightarrow$)) Δ @ Γ $\ominus$ *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* ($\rightarrow$)) $?\Omega$ @ Γ $\ominus$ *map snd* $?\Omega$
let $?\Gamma_1$ = *map* (*uncurry* ($\rightarrow$)) $?\Xi$ @ $?\Gamma_0$ $\ominus$ *map snd* $?\Xi$
have *mset* (Γ $\ominus$ *map snd* Δ) = *mset* ($?\Gamma_0$ $\ominus$ *map snd* $?\Xi$)
using Δ(*1*) Σ(*1*) *recoverWitnesses-mset-equiv* **by** *blast*
hence (Γ $\ominus$ *map snd* Δ) $\preceq$ ($?\Gamma_0$ $\ominus$ *map snd* $?\Xi$)
by (*simp add*: *msub-stronger-theory-intro*)
hence $?\Gamma_1$ $\$\vdash$ Ψ
using Δ(*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 Σ(*1*) Δ(*1*) *recoverWitnessB-mset-equiv*
by (*simp*,
metis listSubtract-monotonic
listSubtract-mset-homomorphism
mset-map)
moreover
have *map* (*uncurry* ($\sqcup$)) $?\Xi$ $\$\vdash$ Φ
using Σ(*1*) *recoverWitnessB-stronger-theory*
Σ(*3*) *segmented-stronger-theory-left-monotonic* **by** *blast*
ultimately have $?\Gamma_0$ $\$\vdash$ (Φ @ Ψ)
using *Cons* **by** *fast*
moreover
have *mset* (*map snd* $?\Omega$) $\subseteq\#$ *mset* (*map snd* Δ)
using Σ(*1*) *recoverWitnessA-mset-equiv*
by (*simp*, *metis mset-subset-eq-add-left*)
hence *mset* (*map snd* $?\Omega$) $\subseteq\#$ *mset* Γ **using** Δ(*1*) **by** *simp*
moreover
have *map* (*uncurry* ($\sqcup$)) $?\Omega$ $:\vdash$ φ
using Σ(*2*)
recoverWitnessA-left-stronger-theory
stronger-theory-deduction-monotonic

```
        by blast
      ultimately show Γ $⊢ ((φ # Φ) @ Ψ)
        by (simp, blast)
    qed
  }
  then show ?case by metis
 qed
 thus ?thesis by blast
qed

lemma (in Classical-Propositional-Logic) segmented-list-deduction-antitonic:
 assumes Γ $⊢ Ψ
    and Ψ :⊢ φ
  shows Γ :⊢ φ
proof −
 have ∀ Γ φ. Γ $⊢ Ψ ⟶ Ψ :⊢ φ ⟶ Γ :⊢ 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 $⊢ (ψ # Ψ)
       and ψ # Ψ :⊢ φ
    hence Ψ :⊢ ψ → φ
      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-Propositional-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 Ψ) ⊆# mset Γ
        map (uncurry (⊔)) Ψ $⊢ map (uncurry (⊔)) Σ
         map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ $⊢ (map (uncurry (→)) Σ @ Λ
⊖ map snd Σ)
        using segmented-deduction-generalized-witness
        by fastforce
      have map (uncurry (→)) Ψ @ Γ ⊖ map snd Ψ $⊢ Δ
        using Σ(3) Ψ(3) Cons
        by auto
      moreover
      have map (uncurry (⊔)) Ψ :⊢ δ
        using Ψ(2) Σ(2) segmented-list-deduction-antitonic
        by blast
      ultimately have Γ $⊢ (δ # Δ)
        using Ψ(1)
        by fastforce
    }
    then show ?case by auto
  qed
  with assms show ?thesis by simp
qed
```

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *segmented-witness-left-split* [*simp*]:
  assumes *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ
  shows (*map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ (*map snd* Σ)) $\$\vdash$ $\Phi = \Gamma \$\vdash \Phi$
proof −
  have $\forall$ Γ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma \longrightarrow$
    (*map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ (*map snd* Σ)) $\$\vdash \Phi =$ $\Gamma \$\vdash \Phi$
  proof (*induct* Σ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* σ Σ)
    {
      fix Γ
      let $?\chi = \mathit{fst}\ \sigma$
      let $?\gamma = \mathit{snd}\ \sigma$
      let $?\Gamma_0$ = *map* (*uncurry* ($\sqcup$)) Σ @ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* (σ $\#$ Σ)
      let $?\Gamma'$ = *map* (*uncurry* ($\sqcup$)) ($\sigma \# \Sigma$) @ *map* (*uncurry* ($\rightarrow$)) ($\sigma \# \Sigma$) @ Γ $\ominus$ *map snd* ($\sigma \# \Sigma$)
      assume *mset* (*map snd* ($\sigma \# \Sigma$)) $\subseteq\#$ *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 → y. (x, y) ∈# mset Σ#} + mset Γ − add-mset (snd σ)
(image-mset snd (mset Σ))
            = {#x → y. (x, y) ∈# mset Σ#} + (mset Γ − add-mset (snd σ)
(image-mset snd (mset Σ)))
        using A subset-mset.diff-add-assoc by blast
      hence {#x → y. (x, y) ∈# mset Σ#} + (mset Γ − image-mset snd (mset
Σ))
          = add-mset (snd σ) ({#x → y. (x, y) ∈# mset Σ#}
            + mset Γ − add-mset (snd σ) (image-mset snd (mset Σ)))
        using B by auto
      hence C:
        mset (map snd Σ) ⊆# mset Γ
        mset (map (uncurry (⊔)) Σ @ map (uncurry (→)) Σ @ Γ ⊖ map snd Σ)
       = mset (?γ # ?Γ0)
        using ‹mset (map snd (σ # Σ)) ⊆# mset Γ›
              subset-mset.dual-order.trans
        by (fastforce+)
      hence Γ $⊢ Φ = (?χ ⊔ ?γ # ?χ → ?γ # ?Γ0) $⊢ Φ
      proof −
        have ∀Γ Δ. ¬ mset (map snd Σ) ⊆# mset Γ
                ∨ ¬ Γ $⊢ Φ
                ∨ ¬ mset (map (uncurry (⊔)) Σ
                         @ map (uncurry (→)) Σ
                         @ Γ ⊖ 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-Propositional-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-Propositional-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-Propositional-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 Σ(2)
      by blast
  next
    case False
    from this obtain σ χ where
      σ: σ = (χ, δ)
         σ ∈ set Σ
      using find-Some-predicate
            find-Some-set-membership
      by fastforce
    let ?Σ′ = remove1 σ Σ
    let ?Σ_A = map (uncurry (⊔)) ?Σ′
    let ?Σ_B = map (uncurry (→)) ?Σ′
```

have *mset* Σ = *mset* ($?\Sigma'$ @ $[(\chi, \delta)]$)
mset Σ = *mset* ((χ, δ) # $?\Sigma'$)
using σ **by** *simp*+
hence *mset* (*map* (*uncurry* ($\sqcup$)) Σ) = *mset* (*map* (*uncurry* ($\sqcup$)) ($?\Sigma'$ @ $[(\chi, \delta)]$))
mset (*map snd* Σ) = *mset* (*map snd* ((χ, δ) # $?\Sigma'$))
mset (*map* (*uncurry* ($\rightarrow$)) Σ) = *mset* (*map* (*uncurry* ($\rightarrow$)) ((χ, δ) # $?\Sigma'$))
by (*metis mset-map*)+
hence *mset* (*map* (*uncurry* ($\sqcup$)) Σ) = *mset* ($?\Sigma_A$ @ $[\chi \sqcup \delta]$)
mset (*map* (*uncurry* ($\rightarrow$)) Σ @ ((δ # Δ) @ Γ) $\ominus$ *map snd* Σ)
= *mset* ($\chi \rightarrow \delta$ # $?\Sigma_B$ @ (Δ @ Γ) $\ominus$ *map snd* $?\Sigma'$)
by *simp*+
hence
$?\Sigma_A$ @ $[\chi \sqcup \delta]$ \$$\vdash$ (Δ @ Φ)
$\chi \rightarrow \delta$ # ($?\Sigma_B$ @ (Δ @ Γ) $\ominus$ *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*
The-Principle-of-Pseudo-Scotus
flip-hypothetical-syllogism
by *blast*
ultimately have ($?\Sigma_A$ @ $?\Sigma_B$ @ (Δ @ Γ) $\ominus$ *map snd* $?\Sigma'$) \$$\vdash$ (Δ @ Φ)
using *segmented-deduction-one-collapse*
list-deduction-theorem
list-deduction-modus-ponens
list-deduction-weaken
forward-direction
segmented-transitive
by *meson*
moreover
have δ = *snd* σ
snd $\sigma \in$ *set* (*map snd* Σ)
by (*simp add*: $\sigma(1)$, *simp add*: $\sigma(2)$)
with $\Sigma(1)$ **have** *mset* (*map snd* (*remove1* σ Σ)) $\subseteq$# *mset* (*remove1* δ ((δ # Δ) @ Γ))
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* σ Σ)) $\subseteq$# *mset* (Δ @ Γ) **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-Propositional-Logic) segmented-biconditional-cancel:
  assumes ⊢ γ ↔ φ
  shows (γ # Γ) $⊢ (φ # Φ) = Γ $⊢ Φ
proof −
  from assms have (γ # Φ) ≼ (φ # Φ) (φ # Φ) ≼ (γ # Φ)
    unfolding biconditional-def
    by (simp add: stronger-theory-left-right-cons)+
  hence (γ # Φ) $⊢ (φ # Φ)
        (φ # Φ) $⊢ (γ # Φ)
    using segmented-stronger-theory-intro by blast+
  moreover
  have Γ $⊢ Φ = (γ # Γ) $⊢ (γ # Φ)
    by (metis append-Cons append-Nil segmented-cancel)+
  ultimately
  have Γ $⊢ Φ ⟹ γ # Γ $⊢ (φ # Φ)
       γ # Γ $⊢ (φ # Φ) ⟹ Γ $⊢ Φ
    using segmented-transitive by blast+
  thus ?thesis by blast
qed

lemma (in Classical-Propositional-Logic) right-segmented-sub:
  assumes ⊢ φ ↔ ψ
  shows Γ $⊢ (φ # Φ) = Γ $⊢ (ψ # Φ)
proof −
  have Γ $⊢ (φ # Φ) = (ψ # Γ) $⊢ (ψ # φ # Φ)
    using segmented-cancel [where Δ=[ψ] and Γ=Γ and Φ=φ # Φ] by simp
  also have ... = (ψ # Γ) $⊢ (φ # ψ # Φ)
    using segmented-cons-cons-right-permute by blast
  also have ... = Γ $⊢ (ψ # Φ)
      using assms biconditional-symmetry-rule segmented-biconditional-cancel by
blast
  finally show ?thesis .
qed

lemma (in Classical-Propositional-Logic) left-segmented-sub:
  assumes ⊢ γ ↔ χ
  shows (γ # Γ) $⊢ Φ = (χ # Γ) $⊢ Φ
proof −
  have (γ # Γ) $⊢ Φ = (χ # γ # Γ) $⊢ (χ # Φ)
    using segmented-cancel [where Δ=[χ] and Γ=(γ # Γ) and Φ=Φ] by simp
  also have ... = (γ # χ # Γ) $⊢ (χ # Φ)
   by (metis segmented-msub-left-monotonic mset-eq-perm perm.swap subset-mset.dual-order.refl)
  also have ... = (χ # Γ) $⊢ Φ
```

using *assms biconditional-symmetry-rule segmented-biconditional-cancel* **by** *blast*
finally show *?thesis* **.**
qed

lemma (**in** *Classical-Propositional-Logic*) *right-segmented-sum-rule*:
$\Gamma \ \$\vdash (\alpha \ \# \ \beta \ \# \ \Phi) = \Gamma \ \$\vdash (\alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \Phi)$
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 $\Gamma \ \$\vdash (\alpha \ \# \ \beta \ \# \ \Phi) = \Gamma \ \$\vdash (\beta \sqcup \alpha \ \# \ \beta \to \alpha \ \# \ \beta \ \# \ \Phi)$
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 ... $= \Gamma \ \$\vdash (\beta \to \alpha \ \# \ \beta \ \# \ \alpha \sqcup \beta \ \# \ \Phi)$
by (*metis A segmented-msub-weaken subset-mset.dual-order.refl*)
also have ... $= \Gamma \ \$\vdash (\beta \to (\alpha \sqcap \beta) \ \# \ \beta \ \# \ \alpha \sqcup \beta \ \# \ \Phi)$
using *B right-segmented-sub* **by** *blast*
also have ... $= \Gamma \ \$\vdash (\beta \ \# \ \beta \to (\alpha \sqcap \beta) \ \# \ \alpha \sqcup \beta \ \# \ \Phi)$
using *segmented-cons-cons-right-permute* **by** *blast*
also have ... $= \Gamma \ \$\vdash (\beta \sqcup (\alpha \sqcap \beta) \ \# \ \beta \to (\alpha \sqcap \beta) \ \# \ \alpha \sqcup \beta \ \# \ \Phi)$
using *C right-segmented-sub* **by** *blast*
also have ... $= \Gamma \ \$\vdash (\alpha \sqcap \beta \ \# \ \alpha \sqcup \beta \ \# \ \Phi)$
using *segmented-formula-right-split* **by** *blast*
finally show *?thesis*
using *segmented-cons-cons-right-permute* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *left-segmented-sum-rule*:
$(\alpha \ \# \ \beta \ \# \ \Gamma) \ \$\vdash \Phi = (\alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \Gamma) \ \$\vdash \Phi$
proof −
have $\star$: *mset* $(\alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \alpha \ \# \ \beta \ \# \ \Gamma) =$ *mset* $(\alpha \ \# \ \beta \ \# \ \alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \Gamma)$ **by** *simp*
have $(\alpha \ \# \ \beta \ \# \ \Gamma) \ \$\vdash \Phi = (\alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \alpha \ \# \ \beta \ \# \ \Gamma) \ \$\vdash (\alpha \sqcup \beta \ \# \ \alpha \sqcap \beta \ \# \ \Phi)$

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) \$⊢ (α # β # Φ)
using *right-segmented-sum-rule* **by** *blast*
also have ... = (α # β # $\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) \$⊢ (α # β # Φ)
by (*metis* ⋆ *segmented-msub-left-monotonic subset-mset.dual-order.refl*)
also have ... = ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Γ) \$⊢ 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-Propositional-Logic*) *segmented-exchange*:
(γ # Γ) \$⊢ (φ # Φ) = ($\varphi \to \gamma$ # Γ) \$⊢ ($\gamma \to \varphi$ # Φ)
proof −
have (γ # Γ) \$⊢ (φ # Φ)
= ($\varphi \sqcup \gamma$ # $\varphi \to \gamma$ # Γ) \$⊢ ($\gamma \sqcup \varphi$ # $\gamma \to \varphi$ # 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-Propositional-Logic*) *segmented-negation-swap*:
Γ \$⊢ (φ # Φ) = ($\sim \varphi$ # Γ) \$⊢ ($\bot$ # Φ)
proof −
have Γ \$⊢ (φ # Φ) = ($\bot$ # Γ) \$⊢ ($\bot$ # φ # Φ)
by (*metis append-Cons append-Nil segmented-cancel*)
also have ... = ($\bot$ # Γ) \$⊢ (φ # $\bot$ # Φ)
using *segmented-cons-cons-right-permute* **by** *blast*
also have ... = ($\sim \varphi$ # Γ) \$⊢ ($\bot \to \varphi$ # $\bot$ # Φ)
unfolding *negation-def*
using *segmented-exchange*
by *blast*
also have ... = ($\sim \varphi$ # Γ) \$⊢ ($\bot$ # Φ)
using *Ex-Falso-Quodlibet*
segmented-tautology-right-cancel
by *blast*
finally show *?thesis* **.**
qed

primrec (**in** *Classical-Propositional-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 ∧

map (*uncurry* ($\sqcup$)) Ψ :$\vdash$ φ $\wedge$
map (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ) #$\vdash$ *n* φ)

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

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

lemma (**in** *Classical-Propositional-Logic*) *stratified-deduction-weaken*:
assumes $n \leq m$
and Γ #$\vdash$ *m* φ
shows Γ #$\vdash$ *n* φ
proof −
have Γ \$$\vdash$ *replicate m* φ
using *assms*(*2*) *stratified-segmented-deduction-replicate*
by *blast*
hence Γ \$$\vdash$ *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-Propositional-Logic*) *stratified-deduction-implication*:
  assumes $\vdash \varphi \to \psi$
    and Γ #$\vdash$ n φ
   shows Γ #$\vdash$ n ψ
proof −
  have *replicate* n ψ $\preceq$ *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-Propositional-Logic*) *segmented-stratified-falsum-equiv*:
  Γ \$$\vdash$ Φ = ($\sim$ Φ @ Γ) #$\vdash$ (*length* Φ) $\bot$
proof −
  have $\forall$ Γ Ψ. Γ \$$\vdash$ (Φ @ Ψ) = ($\sim$ Φ @ Γ) \$$\vdash$ (*replicate* (*length* Φ) $\bot$ @ Ψ)
  proof (*induct* Φ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* φ Φ)
    {
      fix Γ Ψ
      have Γ \$$\vdash$ ((φ # Φ) @ Ψ) = ($\sim$ φ # Γ) \$$\vdash$ ($\bot$ # Φ @ Ψ)
        using *segmented-negation-swap* **by** *auto*
      moreover have *mset* (Φ @ ($\bot$ # Ψ)) = *mset* ($\bot$ # Φ @ Ψ)
        by *simp*
      ultimately have Γ \$$\vdash$ ((φ # Φ) @ Ψ) = ($\sim$ φ # Γ) \$$\vdash$ (Φ @ ($\bot$ # Ψ))
        by (*metis segmented-msub-weaken subset-mset.order-refl*)
      hence Γ \$$\vdash$ ((φ # Φ) @ Ψ) = ($\sim$ Φ @ ($\sim$ φ # Γ)) \$$\vdash$ (*replicate* (*length* Φ)
$\bot$ @ ($\bot$ # Ψ))
        using *Cons*
        by *blast*
      moreover have *mset* ($\sim$ Φ @ ($\sim$ φ # Γ)) = *mset* ($\sim$ (φ # Φ) @ Γ)
                 mset (*replicate* (*length* Φ) $\bot$ @ ($\bot$ # Ψ))
                = *mset* (*replicate* (*length* (φ # Φ)) $\bot$ @ Ψ)
        by *simp+*
      ultimately have
        Γ \$$\vdash$ ((φ # Φ) @ Ψ) = $\sim$ (φ # Φ) @ Γ \$$\vdash$ (*replicate* (*length* (φ # Φ)) $\bot$
@ Ψ)
        by (*metis append.assoc*
              append-Cons
              append-Nil
              length-Cons

```
                  replicate-append-same
                  listSubtract.simps(1)
                  map-ident replicate-Suc
                  segmented-msub-left-monotonic
                  map-listSubtract-mset-containment)
    }
    then show ?case by blast
  qed
  thus ?thesis
    by (metis append-Nil2 stratified-segmented-deduction-replicate)
qed


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

lemma (in Minimal-Logic) unproving-core-finite:
  finite (𝒞 Γ φ)
proof −
  {
    fix Φ
    assume Φ ∈ 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 Minimal-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 Minimal-Logic) unproving-core-complement-deduction:
  assumes Φ ∈ 𝒞 Γ φ
      and ψ ∈ set (Γ ⊖ Φ)
    shows Φ :⊢ ψ → φ
proof (rule ccontr)
  assume ¬ Φ :⊢ ψ → φ
  hence ¬ (ψ # Φ) :⊢ φ
    by (simp add: list-deduction-theorem)
  moreover
  have mset Φ ⊆# mset Γ ψ ∈# mset (Γ ⊖ Φ)
    using assms
    unfolding unproving-core-def
    by (blast, meson in-multiset-in-set)
  hence mset (ψ # Φ) ⊆# 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** *Minimal-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 *listSubtract-set-trivial-upper-bound* [**where** Γ=Γ **and** Φ=Φ]
by *blast*
qed
next
show *set* $\Gamma -$ *set* $\Phi \subseteq$ *set* $(\Gamma \ominus \Phi)$
by (*simp add*: *listSubtract-set-difference-lower-bound*)
qed

lemma (**in** *Minimal-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** *Minimal-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** *Minimal-Logic*) *unproving-listSubtract-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 listSubtract-mset-homomorphism size-Diff-submset size-mset*)+
ultimately show *?thesis* **by** *metis*
qed

lemma (**in** *Minimal-Logic*) *unproving-core-max-list-deduction*:
$\Gamma :\vdash \varphi = (\forall\ \Phi \in \mathcal{C}\ \Gamma\ \varphi.\ 1 \leq$ *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* Γ − *length* Ω
by (*metis listSubtract-mset-homomorphism*
size-Diff-submset
size-mset)
show *?thesis*

```
  proof (cases Γ :⊢ φ)
    assume Γ :⊢ φ
    from Ω have mset Ω ⊂# mset Γ
      by (metis (no-types, lifting)
                Diff-cancel
                Diff-eq-empty-iff
                ‹Γ :⊢ φ›
                list-deduction-monotonic
                unproving-core-def
                mem-Collect-eq
                mset-eq-setD
                subset-mset.dual-order.not-eq-order-implies-strict)
    hence length Ω < length Γ
      using mset-subset-size by fastforce
    hence 1 ≤ length Γ − length Ω
      by (simp add: Suc-leI)
    with ◇ have 1 ≤ length (Γ ⊖ Ω)
      by simp
    with ‹Γ :⊢ φ› Ω show ?thesis
      by (metis unproving-listSubtract-length-equiv)
  next
    assume ¬ Γ :⊢ φ
    moreover have mset Γ ⊆# mset Γ
      by simp
    moreover have length Ω ≤ length Γ
      using ‹mset Ω ⊆# mset Γ› length-sub-mset mset-eq-length
      by fastforce
    ultimately have length Ω = length Γ
      using Ω
      unfolding unproving-core-def
      by (simp add: dual-order.antisym)
    hence 1 > length (Γ ⊖ Ω)
      using ◇
      by simp
    with ‹¬ Γ :⊢ φ› Ω show ?thesis
      by fastforce
  qed
qed

definition (in Minimal-Logic) core-size :: 'a list ⇒ 'a ⇒ nat ((| - |_ [45])
  where
    (| Γ |φ) = (if 𝒞 Γ φ = {} then 0 else Max { length Φ | Φ. Φ ∈ 𝒞 Γ φ })

abbreviation (in Minimal-Logic-With-Falsum) MaxSAT :: 'a list ⇒ nat
  where
    MaxSAT Γ ≡ | Γ |⊥

definition (in Minimal-Logic) complement-core-size :: 'a list ⇒ 'a ⇒ nat (‖ - ‖_
[45])
```

```
  where
    (‖ Γ ‖_φ) = length Γ − | Γ |_φ

lemma (in Minimal-Logic) core-size-intro:
  assumes Φ ∈ 𝒞 Γ φ
  shows length Φ = | Γ |_φ
proof −
  have ∀ n ∈ { length Ψ | Ψ. Ψ ∈ 𝒞 Γ φ }. n ≤ length Φ
       length Φ ∈ { length Ψ | Ψ. Ψ ∈ 𝒞 Γ φ }
    using assms unproving-core-def
    by auto
  moreover
  have finite { length Ψ | Ψ. Ψ ∈ 𝒞 Γ φ }
    using finite-imageI unproving-core-finite
    by simp
  ultimately have Max { length Ψ | Ψ. Ψ ∈ 𝒞 Γ φ } = length Φ
    using Max-eqI
    by blast
  thus ?thesis
    using assms core-size-def
    by auto
qed

lemma (in Minimal-Logic) complement-core-size-intro:
  assumes Φ ∈ 𝒞 Γ φ
  shows length (Γ ⊖ Φ) = ‖ Γ ‖_φ
proof −
  have mset Φ ⊆# mset Γ
    using assms
    unfolding unproving-core-def
    by auto
  moreover from this have length (Γ ⊖ Φ) = length Γ − length Φ
    by (metis listSubtract-mset-homomorphism size-Diff-submset size-mset)
  ultimately show ?thesis
    unfolding complement-core-size-def
    by (metis assms core-size-intro)
qed

lemma (in Minimal-Logic) length-core-decomposition:
  length Γ = (| Γ |_φ) + ‖ Γ ‖_φ
proof (cases 𝒞 Γ φ = {})
  case True
  then show ?thesis
    unfolding core-size-def
              complement-core-size-def
    by simp
next
  case False
  from this obtain Φ where Φ ∈ 𝒞 Γ φ
```

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* $\Gamma -$ *length* Φ
by (*metis listSubtract-mset-homomorphism size-Diff-submset size-mset*)
ultimately show *?thesis*
unfolding *complement-core-size-def*
using *listSubtract-msub-eq core-size-intro*
by *fastforce*
qed

primrec *core-optimal-pre-witness* :: $'a$ *list* $\Rightarrow$ $('a$ *list* $\times$ $'a)$ *list* $(\mathfrak{V})$
where
$\mathfrak{V}$ [] = []
| $\mathfrak{V}$ (ψ # Ψ) = (Ψ, ψ) # $\mathfrak{V}$ Ψ

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

lemma *core-optimal-pre-witness-length-iff-eq*:
$\forall$ $(\Delta,\delta) \in$ *set* $(\mathfrak{V}\ \Psi)$. $\forall$ $(\Sigma,\sigma) \in$ *set* $(\mathfrak{V}\ \Psi)$. (*length* $\Delta =$ *length* Σ) $= ((\Delta, \delta) = (\Sigma,\sigma))$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ)
{
fix Δ
fix δ
assume $(\Delta,\delta) \in$ *set* $(\mathfrak{V}\ (\psi\ \#\ \Psi))$

```
      and length Δ = length Ψ
    hence (Δ,δ) = (Ψ, ψ)
      by (simp add: core-optimal-pre-witness-nonelement)
  }
  hence ∀ (Δ,δ) ∈ set (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 ⊆# 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) ⊆# mset (remdups B)) = (set A ⊆ set B)
proof −
  have ∀ B. (mset (remdups A) ⊆# mset (remdups B)) = (set A ⊆ 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 ∧ (∀ x ∈ set X. x <
length X))
proof (rule iffI)
 assume mset X = mset [0..<length X]
 thus distinct X ∧ (∀x∈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 ∧ (∀x∈set X. x < length X)
 moreover
 {
  fix n
  have ∀ X. n = length X −→
            distinct X ∧ (∀x∈set X. x < length X) −→
            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: ∀x∈set X. x < length X
  have n ∈ set X
  proof (rule ccontr)
    assume n ∉ 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': ∀x∈set (tl X). x < length (tl X)
      by (metis A A' C ‹n ∉ 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* $\Sigma \subseteq\#$ *mset* ($\mathfrak{V}$ Ψ)
distinct Σ
by (*metis mset.simps*(*2*) *mset-subset-eq-insertD subset-mset-def*, *simp*)
with *Cons.hyps* **have** *distinct* (*map* (λa. *length* (*fst a*)) Σ) **by** *simp*
moreover
obtain δ Δ **where** $\sigma = (\Delta, \delta)$
by *fastforce*
hence $(\Delta, \delta) \in$ *set* ($\mathfrak{V}$ Ψ)
using *Cons.prems mset-subset-eq-insertD*
by *fastforce*
hence $\forall$ $(\Sigma,\sigma) \in$ *set* ($\mathfrak{V}$ Ψ). (*length* Δ = *length* Σ) = $((\Delta, \delta) = (\Sigma, \sigma))$
using *core-optimal-pre-witness-length-iff-eq* [**where** Ψ=Ψ]
by *fastforce*
hence $\forall$ $(\Sigma,\sigma) \in$ *set* Σ. (*length* Δ = *length* Σ) = $((\Delta, \delta) = (\Sigma, \sigma))$
using ‹*mset* $\Sigma \subseteq\#$ *mset* ($\mathfrak{V}$ Ψ)›
by (*metis* (*no-types*, *lifting*) *Un-iff mset-le-perm-append perm-set-eq set-append*)
hence *length* (*fst* σ) $\notin$ *set* (*map* (λa. *length* (*fst a*)) Σ)
using *Cons.prems*(*2*) ‹$\sigma = (\Delta, \delta)$›
by *fastforce*
ultimately show *?case* **by** *simp*
qed
moreover have *length* (*map* (*length* $\circ$ *fst*) Σ) = *length* Σ **by** *simp*
moreover have *map* (*length* $\circ$ *fst*) $\Sigma \neq$ [] **using** *assms* **by** *simp*
ultimately show *?thesis*
using *distinct-pigeon-hole*
by *fastforce*
qed

abbreviation (**in** *Classical-Propositional-Logic*)
core-optimal-witness :: $'a \Rightarrow 'a$ *list* $\Rightarrow$ $('a \times 'a)$ *list* ($\mathfrak{W}$)
where $\mathfrak{W}$ φ $\Xi \equiv$ *map* ($\lambda(\Psi,\psi)$. (Ψ :$\rightarrow$ φ, ψ)) ($\mathfrak{V}$ Ξ)

abbreviation (**in** *Classical-Propositional-Logic*)
disjunction-core-optimal-witness :: $'a \Rightarrow 'a$ *list* $\Rightarrow 'a$ *list* ($\mathfrak{W}_\sqcup$)
where $\mathfrak{W}_\sqcup$ φ $\Psi \equiv$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{W}$ φ Ψ)

abbreviation (**in** *Classical-Propositional-Logic*)
implication-core-optimal-witness :: $'a \Rightarrow 'a$ *list* $\Rightarrow 'a$ *list* ($\mathfrak{W}_\rightarrow$)
where $\mathfrak{W}_\rightarrow$ φ $\Psi \equiv$ *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{W}$ φ Ψ)

lemma (**in** *Classical-Propositional-Logic*) *core-optimal-witness-conjunction-identity*:
$\vdash \bigsqcap$ ($\mathfrak{W}_\sqcup$ φ Ψ) $\leftrightarrow$ ($\varphi \sqcup \bigsqcap$ Ψ)
proof (*induct* Ψ)
case *Nil*
then show *?case*
unfolding *biconditional-def*
disjunction-def
using *Axiom-1*

Modus-Ponens
verum-tautology
by (*simp*, *blast*)
next
case (*Cons* ψ Ψ)
have $\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\prod \Psi \rightarrow \varphi)$
by (*simp add*: *list-curry-uncurry*)
hence $\vdash \prod$ (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{W}$ φ (ψ # Ψ)))
$\leftrightarrow ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap \prod$ (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{W}$ φ Ψ)))
unfolding *biconditional-def*
using *conjunction-monotonic*
disjunction-monotonic
by *simp*
moreover have $\vdash ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap \prod$ (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{W}$ φ Ψ)))
$\leftrightarrow ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap (\varphi \sqcup \prod \Psi))$
using *Cons.hyps biconditional-conjunction-weaken-rule*
by *blast*
moreover
{
fix φ ψ χ
have $\vdash ((\chi \rightarrow \varphi \sqcup \psi) \sqcap (\varphi \sqcup \chi)) \leftrightarrow (\varphi \sqcup (\psi \sqcap \chi))$
proof −
let $?\varphi = ((\langle\chi\rangle \rightarrow \langle\varphi\rangle \sqcup \langle\psi\rangle) \sqcap (\langle\varphi\rangle \sqcup \langle\chi\rangle)) \leftrightarrow (\langle\varphi\rangle \sqcup (\langle\psi\rangle \sqcap \langle\chi\rangle))$
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 $\vdash \prod$ (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{W}$ φ (ψ # Ψ))) $\leftrightarrow (\varphi \sqcup (\psi \sqcap \prod$
$\Psi))$
using *biconditional-transitivity-rule*
by *blast*
then show *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *core-optimal-witness-deduction*:
$\vdash \mathfrak{W}_\sqcup\ \varphi\ \Psi :\rightarrow \varphi \leftrightarrow \Psi :\rightarrow \varphi$
proof −
have $\vdash \mathfrak{W}_\sqcup\ \varphi\ \Psi :\rightarrow \varphi \leftrightarrow (\prod (\mathfrak{W}_\sqcup\ \varphi\ \Psi) \rightarrow \varphi)$
by (*simp add*: *list-curry-uncurry*)
moreover
{
fix α β γ
have $\vdash (\alpha \leftrightarrow \beta) \rightarrow ((\alpha \rightarrow \gamma) \leftrightarrow (\beta \rightarrow \gamma))$
proof −
let $?\varphi = (\langle\alpha\rangle \leftrightarrow \langle\beta\rangle) \rightarrow ((\langle\alpha\rangle \rightarrow \langle\gamma\rangle) \leftrightarrow (\langle\beta\rangle \rightarrow \langle\gamma\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*

qed
}
ultimately have $\vdash \mathfrak{W}_{\sqcup}\ \varphi\ \Psi :\rightarrow \varphi \leftrightarrow ((\varphi \sqcup \bigsqcap\ \Psi) \rightarrow \varphi)$
using *Modus-Ponens*
biconditional-transitivity-rule
core-optimal-witness-conjunction-identity
by *blast*
moreover
{
fix $\alpha\ \beta$
have $\vdash ((\alpha \sqcup \beta) \rightarrow \alpha) \leftrightarrow (\beta \rightarrow \alpha)$
proof −
let $?\varphi = ((\langle\alpha\rangle \sqcup \langle\beta\rangle) \rightarrow \langle\alpha\rangle) \leftrightarrow (\langle\beta\rangle \rightarrow \langle\alpha\rangle)$
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 $\vdash \mathfrak{W}_{\sqcup}\ \varphi\ \Psi :\rightarrow \varphi \leftrightarrow (\bigsqcap\ \Psi \rightarrow \varphi)$
using *biconditional-transitivity-rule* **by** *blast*
thus *?thesis*
using *biconditional-symmetry-rule*
biconditional-transitivity-rule
list-curry-uncurry
by *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *optimal-witness-split-identity*:
$\vdash (\mathfrak{W}_{\sqcup}\ \varphi\ (\psi\ \#\ \Xi)) :\rightarrow \varphi \rightarrow (\mathfrak{W}_{\rightarrow}\ \varphi\ (\psi\ \#\ \Xi)) :\rightarrow \varphi \rightarrow \Xi :\rightarrow \varphi$
proof (*induct* Ξ)
case *Nil*
have $\vdash ((\varphi \sqcup \psi) \rightarrow \varphi) \rightarrow ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi$
proof −
let $?\varphi = ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \rightarrow \langle\varphi\rangle) \rightarrow ((\langle\varphi\rangle \rightarrow \langle\psi\rangle) \rightarrow \langle\varphi\rangle) \rightarrow \langle\varphi\rangle$
have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
let $?A = \mathfrak{W}_{\sqcup}\ \varphi\ \Xi :\rightarrow \varphi$
let $?B = \mathfrak{W}_{\rightarrow}\ \varphi\ \Xi :\rightarrow \varphi$
let $?X = \Xi :\rightarrow \varphi$
from *Cons.hyps* **have** $\vdash ((?X \sqcup \psi) \rightarrow ?A) \rightarrow ((?X \rightarrow \psi) \rightarrow ?B) \rightarrow ?X$ **by** *simp*
moreover
have $\vdash (((?X \sqcup \psi) \rightarrow ?A) \rightarrow ((?X \rightarrow \psi) \rightarrow ?B) \rightarrow ?X)$
$\rightarrow ((\xi \rightarrow ?X \sqcup \psi) \rightarrow (?X \sqcup \xi) \rightarrow ?A) \rightarrow (((\xi \rightarrow ?X) \rightarrow \psi) \rightarrow (?X \rightarrow \xi)$

$\to$ *?B*) $\to \xi \to$ *?X*
proof −
let *?φ* =$(((\langle ?X \rangle \sqcup \langle \psi \rangle) \to \langle ?A \rangle) \to ((\langle ?X \rangle \to \langle \psi \rangle) \to \langle ?B \rangle) \to \langle ?X \rangle) \to$
$((\langle \xi \rangle \to \langle ?X \rangle \sqcup \langle \psi \rangle) \to (\langle ?X \rangle \sqcup \langle \xi \rangle) \to \langle ?A \rangle) \to$
$(((\langle \xi \rangle \to \langle ?X \rangle) \to \langle \psi \rangle) \to (\langle ?X \rangle \to \langle \xi \rangle) \to \langle ?B \rangle) \to$
$\langle \xi \rangle \to$
$\langle ?X \rangle$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash (\!|$ *?φ* $|\!)$ **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-Propositional-Logic*) *disj-conj-impl-duality*:
$\vdash (\varphi \to \chi \sqcap \psi \to \chi) \leftrightarrow ((\varphi \sqcup \psi) \to \chi)$
proof −
let *?φ* = $(\langle \varphi \rangle \to \langle \chi \rangle \sqcap \langle \psi \rangle \to \langle \chi \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcup \langle \psi \rangle) \to \langle \chi \rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?φ* **by** *fastforce*
hence $\vdash (\!|$ *?φ* $|\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

lemma (**in** *Classical-Propositional-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-Propositional-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 $\vdash \bigsqcup\ (\Phi\ @\ \Psi) \leftrightarrow (\bigsqcup\ \Phi \sqcup \bigsqcup\ \Psi) \rightarrow (\varphi \sqcup \bigsqcup\ (\Phi\ @\ \Psi)) \leftrightarrow ((\varphi \sqcup \bigsqcup\ \Phi) \sqcup \bigsqcup\ \Psi)$
  proof −
    let $?\varphi =$
      $(\langle \bigsqcup\ (\Phi\ @\ \Psi)\rangle \leftrightarrow (\langle \bigsqcup\ \Phi\rangle \sqcup \langle \bigsqcup\ \Psi\rangle)) \rightarrow (\langle\varphi\rangle \sqcup \langle \bigsqcup\ (\Phi\ @\ \Psi)\rangle) \leftrightarrow ((\langle\varphi\rangle \sqcup \langle \bigsqcup\ \Phi\rangle) \sqcup \langle \bigsqcup\ \Psi\rangle)$
    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-Propositional-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-Propositional-Logic*) *conj-absorption*:
  assumes $\chi \in$ *set* Φ
  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-Propositional-Logic*) *conj-extract*: $\vdash \bigsqcup$ (*map* (($\sqcap$) φ) Ψ) $\leftrightarrow$ $(\varphi \sqcap \bigsqcup \Psi)$
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) $\leftrightarrow (\varphi \sqcap \bigsqcup \Psi)$
      $\to ((\varphi \sqcap \psi) \sqcup \bigsqcup$ (*map* (($\sqcap$) φ) Ψ)) $\leftrightarrow (\varphi \sqcap (\psi \sqcup \bigsqcup \Psi))$
  proof −
    let *?*φ = $\langle\bigsqcup$ (*map* (($\sqcap$) φ) Ψ)$\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcup \Psi\rangle)$
          $\to ((\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcup \langle\bigsqcup$ (*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 $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
then show *?case* **using** *Cons Modus-Ponens* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *conj-multi-extract*:
$\vdash \bigsqcup\ (map \bigsqcap\ (map\ ((@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcup\ (map \bigsqcap\ \Sigma))$
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
$\vdash\ \ \bigsqcup\ (map \bigsqcap\ (map\ ((@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcup\ (map \bigsqcap\ \Sigma))$
$\rightarrow \bigsqcap\ (\Delta\ @\ \sigma) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcap\ \sigma)$
$\rightarrow (\bigsqcap\ (\Delta\ @\ \sigma) \sqcup \bigsqcup\ (map\ (\bigsqcap \circ (@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap (\bigsqcap\ \sigma \sqcup \bigsqcup\ (map \bigsqcap$
$\Sigma)))$
proof $-$
let $?\varphi =$
$\langle \bigsqcup\ (map \bigsqcap\ (map\ ((@)\ \Delta)\ \Sigma))\rangle \leftrightarrow (\langle \bigsqcap\ \Delta\rangle \sqcap \langle \bigsqcup\ (map \bigsqcap\ \Sigma)\rangle)$
$\rightarrow \langle \bigsqcap\ (\Delta\ @\ \sigma)\rangle \leftrightarrow (\langle \bigsqcap\ \Delta\rangle \sqcap \langle \bigsqcap\ \sigma\rangle)$
$\rightarrow (\langle \bigsqcap\ (\Delta\ @\ \sigma)\rangle \sqcup \langle \bigsqcup\ (map\ (\bigsqcap \circ (@)\ \Delta)\ \Sigma)\rangle) \leftrightarrow (\langle \bigsqcap\ \Delta\rangle \sqcap (\langle \bigsqcap\ \sigma\rangle \sqcup \langle \bigsqcup$
$(map \bigsqcap\ \Sigma)\rangle))$
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
$\vdash (\bigsqcap\ (\Delta\ @\ \sigma) \sqcup \bigsqcup\ (map\ (\bigsqcap \circ (@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap (\bigsqcap\ \sigma \sqcup \bigsqcup\ (map \bigsqcap$
$\Sigma)))$
using *Cons.hyps arbitrary-conj-concat-equiv Modus-Ponens* **by** *blast*
then show *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-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
$\vdash \bigsqcup\ (map\ ?\Pi\Delta\ \Psi) \leftrightarrow (\bigsqcap\ ?\Delta' \sqcap \bigsqcup\ (map\ ?\Pi\ \Psi))$
$\to \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) \leftrightarrow (?\chi' \sqcap \bigsqcup\ (map\ ?\Pi\Delta\ \Psi))$
$\to \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi) \leftrightarrow ((?\chi' \sqcap \bigsqcap\ ?\Delta') \sqcap \bigsqcup\ (map\ ?\Pi\ \Psi))$
proof $-$
let $?\varphi = \langle \bigsqcup\ (map\ ?\Pi\Delta\ \Psi)\rangle \leftrightarrow (\langle \bigsqcap\ ?\Delta'\rangle \sqcap \langle \bigsqcup\ (map\ ?\Pi\ \Psi)\rangle)$
$\to \langle \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi)\rangle \leftrightarrow (\langle ?\chi'\rangle \sqcap \langle \bigsqcup\ (map\ ?\Pi\Delta\ \Psi)\rangle)$
$\to \langle \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap ?\Pi\Delta\ \varphi)\ \Psi)\rangle \leftrightarrow ((\langle ?\chi'\rangle \sqcap \langle \bigsqcap\ ?\Delta'\rangle) \sqcap \langle \bigsqcup$
$(map\ ?\Pi\ \Psi)\rangle)$
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 $\vdash \bigsqcup\ (map\ (\lambda\varphi.\ ?\chi' \sqcap \bigsqcap\ (?\Delta'\ @\ map\ snd\ \varphi))\ \Psi)$
$\leftrightarrow ((?\chi' \sqcap \bigsqcap\ ?\Delta') \sqcap \bigsqcup\ (map\ (\lambda\varphi.\ \bigsqcap\ (map\ snd\ \varphi))\ \Psi))$
using *Modus-Ponens* **by** *blast*
thus *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *extract-inner-concat-remdups*:
$\vdash \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \leftrightarrow$
$(\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
proof $-$
have $\forall\ \Psi.\ \vdash \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi) \leftrightarrow$
$(\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
proof (*induct* Δ)
case *Nil*
then show *?case*
by (*simp*,
meson Modus-Ponens
biconditional-introduction
conjunction-introduction
conjunction-right-elimination
verum-tautology)

next
  case $(Cons\ \delta\ \Delta)$
  {
    fix Ψ
    have $\vdash \quad \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta \# \Delta)))\ \Psi)$
        $\leftrightarrow (\bigsqcap\ (map\ snd\ (\delta \# \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
    proof $(cases\ \delta \in set\ \Delta)$
      assume $\delta \in set\ \Delta$
      have
        $\vdash \quad \bigsqcap\ (map\ snd\ \Delta) \leftrightarrow (snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta))$
          $\to \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
            $\leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
          $\to \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
            $\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))$
$\Psi))$
      proof −
        let $?\varphi = \quad \langle\bigsqcap\ (map\ snd\ \Delta)\rangle \leftrightarrow (\langle snd\ \delta\rangle \sqcap \langle\bigsqcap\ (map\ snd\ \Delta)\rangle)$
                $\to \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)\rangle$
                  $\leftrightarrow (\langle\bigsqcap\ (map\ snd\ \Delta)\rangle \sqcap \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))$
$\Psi)\rangle)$
                $\to \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)\rangle$
                $\leftrightarrow ((\langle snd\ \delta\rangle \sqcap \langle\bigsqcap\ (map\ snd\ \Delta)\rangle) \sqcap \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ$
$remdups))\ \Psi)\rangle)$
        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 $\vdash \bigsqcap\ (map\ snd\ \Delta) \leftrightarrow (snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta))$
        by (*simp add*: ‹$\delta \in set\ \Delta$› *conj-absorption*)
      ultimately have
        $\vdash \quad \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
          $\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))$
$\Psi))$
        using *Cons.hyps Modus-Ponens* **by** *blast*
      moreover have $map\ snd \circ remdups \circ (@)\ (\delta \# \Delta) = map\ snd \circ remdups$
$\circ (@)\ \Delta$
        using ‹$\delta \in set\ \Delta$› **by** *fastforce*
      ultimately show *?thesis* **using** *Cons* **by** *simp*
    next
      assume $\delta \notin set\ \Delta$
      hence †:
        $\bigsqcap \circ (map\ snd \circ remdups) = (\lambda\psi.\ \bigsqcap\ (map\ snd\ (remdups\ \psi)))$
          $(\lambda\psi.\ \bigsqcap\ (map\ snd\ (if\ \delta \in set\ \psi\ then\ remdups\ (\Delta\ @\ \psi)\ else\ \delta \# remdups$
$(\Delta\ @\ \psi))))$
          $= \bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta \# \Delta))$
        by *fastforce+*
      show *?thesis*
      proof $(induct\ \Psi)$
        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 $\star$:
$\vdash \bigsqcap$ (*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 $\diamondsuit$: $\vdash$ $\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* δ $\sqcap$ $\bigsqcap$ (*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* δ $\sqcap$ $\bigsqcap$ (*map snd* (*remdups* ψ)))
using *conj-absorption* **by** *blast*
have
$\vdash$ ($\bigsqcap$ (*map snd* (*remdups* ψ)) $\leftrightarrow$ (*snd* δ $\sqcap$ $\bigsqcap$ (*map snd* (*remdups* ψ))))
$\rightarrow$ ($\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* δ $\sqcap$ $\bigsqcap$ (*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))))
$\rightarrow$ ($\bigsqcap$ (*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))$\rangle$)))
$\rightarrow$ (($\langle\bigsqcup$ (*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))$\rangle$)))
$\rightarrow$ (($\langle\bigsqcap$ (*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

$\vdash$ ($\bigsqcap$ (*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)$\rangle$)))

$\rightarrow \quad (\langle \bigsqcap \ (map\ snd\ (remdups\ (\Delta\ @\ \psi)))\rangle$
$\leftrightarrow (\langle \bigsqcap \ (map\ snd\ \Delta)\rangle \sqcap \langle \bigsqcap \ (map\ snd\ (remdups\ \psi))\rangle\rangle))$
$\rightarrow \quad (((\langle snd\ \delta\rangle \sqcap \langle \bigsqcap \ (map\ snd\ (remdups\ (\Delta\ @\ \psi)))\rangle)$
$\sqcup \langle \bigsqcup \ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)\rangle)$
$\leftrightarrow (((\langle snd\ \delta\rangle \sqcap \langle \bigsqcap \ (map\ snd\ \Delta)\rangle)$
$\sqcap (\langle \bigsqcap \ (map\ snd\ (remdups\ \psi))\rangle \sqcup \langle \bigsqcup \ (map\ (\bigsqcap \circ (map\ snd \circ$
$remdups))\ \Psi)\rangle)))$

have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence
$\vdash \quad ((snd\ \delta \sqcap \bigsqcap \ (map\ snd\ (remdups\ (\Delta\ @\ \psi))))$
$\sqcup \bigsqcup \ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi))$
$\leftrightarrow ((snd\ \delta \sqcap \bigsqcap \ (map\ snd\ \Delta))$
$\sqcap (\bigsqcap \ (map\ snd\ (remdups\ \psi)) \sqcup \bigsqcup \ (map\ (\bigsqcap \circ (map\ snd \circ$
$remdups))\ \Psi)))$

using $\star$ $\diamondsuit$ *Modus-Ponens* **by** *blast*
then show *?thesis* **using** $\langle\delta \notin set\ \psi\rangle$ $\langle\delta \notin set\ \Delta\rangle$ **by** (*simp add*: $\dagger$)
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 $\forall$ *B*. *mset A* = *mset B* $\longrightarrow$ *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 × ′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) ⊆# mset (remdups (B @ A))
proof -
 have ∀ B. mset (remdups A) ⊆# 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)) ⊆# 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-Propositional-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 $?\Psi' = map\ (\lambda(\Psi,\psi).\ (\Psi :\to \varphi \to \psi))\ \Psi_0$
{
fix $\Psi :: ('a\ list \times 'a)\ list$
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi)$
let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi)$
have $\vdash (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \to \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \to \varphi)$
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 $\Delta :: 'a\ list$
fix $\delta :: 'a$
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 $\vdash (\bigsqcup\ (map\ \bigsqcap\ ?\Phi\ @\ map\ \bigsqcap\ ?\Psi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Phi) \sqcup \bigsqcup\ (map$
$\bigsqcap\ ?\Psi))) \to$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Delta\ @\ map\ \bigsqcap\ ?\Omega) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Delta) \sqcup \bigsqcup\ (map$
$\bigsqcap\ ?\Omega))) \to$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Phi) \leftrightarrow (\bigsqcap\ [\delta] \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))) \to$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Psi) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))) \to$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Delta) \leftrightarrow (\bigsqcap\ [\delta] \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma))) \to$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Omega) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma))) \to$
$((\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \to \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \to \varphi)) \to$
$((\bigsqcup\ (map\ \bigsqcap\ ?\Phi\ @\ map\ \bigsqcap\ ?\Psi) \to \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Delta\ @\ map$
$\bigsqcap\ ?\Omega) \to \varphi))$
proof –
let $?\varphi =$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Phi\ @\ map\ \bigsqcap\ ?\Psi)\rangle \leftrightarrow (\langle\bigsqcup\ (map\ \bigsqcap\ ?\Phi)\rangle \sqcup \langle\bigsqcup\ (map$
$\bigsqcap\ ?\Psi)\rangle))) \to$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Delta\ @\ map\ \bigsqcap\ ?\Omega)\rangle \leftrightarrow (\langle\bigsqcup\ (map\ \bigsqcap\ ?\Delta)\rangle \sqcup \langle\bigsqcup\ (map$
$\bigsqcap\ ?\Omega)\rangle))) \to$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Phi)\rangle \leftrightarrow (\langle\bigsqcap\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle))) \to$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Psi)\rangle \leftrightarrow (\langle\bigsqcap\ \Delta\rangle \sqcap \langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle))) \to$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Delta)\rangle \leftrightarrow (\langle\bigsqcap\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma)\rangle))) \to$
$((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Omega)\rangle \leftrightarrow (\langle\bigsqcap\ \Delta\rangle \sqcap \langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma)\rangle))) \to$
$(((\langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha)\rangle \to \langle\varphi\rangle) \leftrightarrow (\langle\bigsqcup\ (map\ \bigsqcap\ ?\Sigma)\rangle \to \langle\varphi\rangle)) \to$

$((\langle \bigsqcup (map \bigsqcap ?\Phi @ map \bigsqcap ?\Psi)\rangle \to \langle\varphi\rangle) \leftrightarrow (\langle \bigsqcup (map \bigsqcap ?\Delta @ map \bigsqcap ?\Omega)\rangle \to \langle\varphi\rangle))$

have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!| ?\varphi |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
moreover
have *map snd* $(\mathfrak{V}\ \Delta) = \Delta$ **by** (*induct* Δ, *auto*)
hence $\vdash \bigsqcup (map \bigsqcap ?\Phi @ map \bigsqcap ?\Psi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Phi) \sqcup \bigsqcup (map \bigsqcap ?\Psi))$
$\vdash \bigsqcup (map \bigsqcap ?\Delta @ map \bigsqcap ?\Omega) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Delta) \sqcup \bigsqcup (map \bigsqcap ?\Omega))$
$\vdash \bigsqcup (map \bigsqcap ?\Phi) \leftrightarrow (\bigsqcap [\delta] \sqcap \bigsqcup (map \bigsqcap ?\Sigma_\alpha))$
$\vdash \bigsqcup (map \bigsqcap ?\Psi) \leftrightarrow (\bigsqcap \Delta \sqcap \bigsqcup (map \bigsqcap ?\Sigma_\alpha))$
$\vdash \bigsqcup (map \bigsqcap ?\Delta) \leftrightarrow (\bigsqcap [\delta] \sqcap \bigsqcup (map \bigsqcap ?\Sigma))$
$\vdash \bigsqcup (map \bigsqcap ?\Omega) \leftrightarrow (\bigsqcap \Delta \sqcap \bigsqcup (map \bigsqcap ?\Sigma))$
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
$\vdash ((\bigsqcup (map \bigsqcap ?\Sigma_\alpha) \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma) \to \varphi)) \to$
$(\bigsqcup (map \bigsqcap ?\Phi @ map \bigsqcap ?\Psi) \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Delta @ map \bigsqcap ?\Omega) \to \varphi)$
using *Modus-Ponens* **by** *blast*
moreover have $(\#)\ (\Delta, \delta) = (@)\ [(\Delta, \delta)]$ **by** *fastforce*
ultimately have
$\vdash ((\bigsqcup (map \bigsqcap ?\Sigma_\alpha) \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma) \to \varphi)) \to$
$((\bigsqcup (map \bigsqcap ?\Sigma_\alpha{}') \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma') \to \varphi))$
by *auto*
}
hence
$\vdash ((\bigsqcup (map \bigsqcap ?\Sigma_\alpha{}') \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma') \to \varphi))$
using *Cons Modus-Ponens* **by** *blast*
moreover have $\Delta\delta = (?\Delta, ?\delta)$ **by** *fastforce*
ultimately show *?case* **by** *metis*
qed
}
hence $\vdash (\bigsqcup (map \bigsqcap ?\Sigma_\alpha) \to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma) \to \varphi)$ **by** *blast*
moreover have $\vdash (?\Psi' :\to \varphi) \leftrightarrow (\bigsqcup (map \bigsqcap ?\Sigma_\alpha) \to \varphi)$
proof (*induct* Ψ_0)
case *Nil*
have $\vdash \varphi \leftrightarrow ((\top \sqcup \bot) \to \varphi)$
proof −
let $?\varphi = \langle\varphi\rangle \leftrightarrow ((\top \sqcup \bot) \to \langle\varphi\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*

hence $\vdash$ ⦇ *?φ* ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
thus *?case* **by** *simp*
next
case (*Cons* ψ_0 Ψ_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 Ξ :: *'a list*
have $map\ snd\ (\mathfrak{V}\ \Xi) = \Xi$ **by** (*induct* Ξ, *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
$\vdash\ (?\Psi' :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi)$
$\rightarrow\ (?\Xi :\rightarrow \varphi) \leftrightarrow (\bigsqcap\ ?\Xi \rightarrow \varphi)$
$\rightarrow\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))$
$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \sqcup \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)))$
$\rightarrow\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \leftrightarrow (?\delta \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$
$\rightarrow\ \bigsqcup\ (map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \leftrightarrow (\bigsqcap\ ?\Xi \sqcap \bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha))$
$\rightarrow\ ((?\Xi :\rightarrow \varphi \rightarrow ?\delta) \rightarrow ?\Psi' :\rightarrow \varphi)$
$\leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map\ \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \rightarrow \varphi)$
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 $?\mathcal{M} = length \circ fst$
let $?\mathcal{S} = sort\text{-}key\ (-\ ?\mathcal{M})$
let $?\Sigma' = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ (?\mathcal{S}\ \Psi_0))$
have $mset\ \Psi_0 = mset\ (?\mathcal{S}\ \Psi_0)$ **by** *simp*

have $\forall\ \Phi.\ mset\ \Psi_0 = mset\ \Phi \longrightarrow mset\ (map\ mset\ (?T_\Sigma\ \Psi_0)) = mset\ (map\ mset\ (?T_\Sigma\ \Phi))$
proof (*induct* Ψ_0)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ_0)
obtain Δ δ **where** $\psi = (\Delta,\delta)$ **by** *fastforce*
{
fix Φ
assume $mset\ (\psi\ \#\ \Psi_0) = mset\ \Phi$
hence $mset\ \Psi_0 = mset\ (remove1\ \psi\ \Phi)$
by (*simp add*: *union-single-eq-diff*)
have $\psi \in set\ \Phi$ **using** ‹$mset\ (\psi\ \#\ \Psi_0) = mset\ \Phi$›
using *mset-eq-setD* **by** *fastforce*
hence $mset\ (map\ mset\ (?T_\Sigma\ \Phi)) = mset\ (map\ mset\ (?T_\Sigma\ (\psi\ \#\ (remove1$

ψ Φ))))
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
then show *?case* **proof** (*cases* $\varphi = \psi$)
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 $\varphi = (\Delta',\delta')$
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*
$?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((@) ($\mathfrak{V}$ Δ)) ∘ *image-mset* ((#) φ))) (*mset*
$?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((@) ($\mathfrak{V}$ Δ)) ∘ *image-mset* ((@) ($\mathfrak{V}$ Δ'))))
(*mset* $?\Sigma$)
using ‹$\psi = (\Delta,\delta)$› ‹$\varphi = (\Delta',\delta')$›
by (*simp add*: *multiset.map-comp*)
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
($?\mathfrak{m}$ ∘ (*image-mset* ((#) φ) ∘ *image-mset* ((#) ψ))) (*mset* $?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((@) ($\mathfrak{V}$ Δ')) ∘ *image-mset* ((#) ψ))) (*mset*
$?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((#) φ) ∘ *image-mset* ((@) ($\mathfrak{V}$ Δ)))) (*mset*
$?\Sigma$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((@) ($\mathfrak{V}$ Δ')) ∘ *image-mset* ((@) ($\mathfrak{V}$ Δ))))
(*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* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
($?\mathfrak{m}$ ∘ (*image-mset* ((#) φ))) (*mset* $?\Sigma'$) +
($?\mathfrak{m}$ ∘ (*image-mset* ((@) ($\mathfrak{V}$ Δ')))) (*mset* $?\Sigma'$)

```
          using ‹ψ = (Δ,δ)›
          by (simp add: multiset.map-comp)
        hence mset (map mset (?T_Σ (ψ # remove1 ψ (φ # Φ)))) =
              image-mset ((+) {#φ#}) (mset (map mset ?Σ')) +
              image-mset ((+) (mset (𝔙 Δ'))) (mset (map mset ?Σ'))
          by (simp add: image-mset-cons-homomorphism
                        image-mset-append-homomorphism)
        hence mset (map mset (?T_Σ (ψ # remove1 ψ (φ # Φ)))) =
              image-mset ((+) {#φ#}) (mset (map mset (?T_Σ Φ))) +
              image-mset ((+) (mset (𝔙 Δ'))) (mset (map mset (?T_Σ Φ)))
          using † by auto
        hence mset (map mset (?T_Σ (ψ # remove1 ψ (φ # Φ)))) =
              (?𝔪 ∘ (image-mset ((#) φ))) (mset (?T_Σ Φ)) +
              (?𝔪 ∘ (image-mset ((@) (𝔙 Δ')))) (mset (?T_Σ Φ))
          by (simp add: image-mset-cons-homomorphism
                        image-mset-append-homomorphism)
        thus ?thesis using ‹φ = (Δ',δ')› by (simp add: multiset.map-comp)
      qed
    qed
     hence   image-mset mset (image-mset ((#) ψ) (mset (?T_Σ (remove1 ψ
Φ)))) +
             image-mset mset (image-mset ((@) (𝔙 Δ)) (mset (?T_Σ (remove1
ψ Φ))))
           = image-mset mset (mset (?T_Σ Φ))
      by (simp add: ‹ψ = (Δ,δ)› multiset.map-comp)
    hence
        image-mset ((+) {# ψ #}) (image-mset mset (mset (?T_Σ (remove1 ψ
Φ)))) +
       image-mset ((+) (mset (𝔙 Δ))) (image-mset mset (mset (?T_Σ (remove1
ψ Φ))))
       = image-mset mset (mset (?T_Σ Φ))
   by (simp add: image-mset-cons-homomorphism image-mset-append-homomorphism)
    hence
      image-mset ((+) {# ψ #}) (image-mset mset (mset (?T_Σ Ψ_0))) +
       image-mset ((+) (mset (𝔙 Δ))) (image-mset mset (mset (?T_Σ Ψ_0)))
     = image-mset mset (mset (?T_Σ Φ))
      using Cons ‹mset Ψ_0 = mset (remove1 ψ Φ)›
      by fastforce
    hence
      image-mset mset (image-mset ((#) ψ) (mset (?T_Σ Ψ_0))) +
       image-mset mset (image-mset ((@) (𝔙 Δ)) (mset (?T_Σ Ψ_0)))
     = image-mset mset (mset (?T_Σ Φ))
   by (simp add: image-mset-cons-homomorphism image-mset-append-homomorphism)
    hence mset (map mset (?T_Σ (ψ # Ψ_0))) = mset (map mset (?T_Σ Φ))
      by (simp add: ‹ψ = (Δ,δ)› multiset.map-comp)
  }
  then show ?case by blast
 qed
 hence mset (map mset (?T_Σ Ψ_0)) = mset (map mset (?T_Σ (?𝒮 Ψ_0)))
```

using ‹*mset* Ψ_0 = *mset* (*?S* Ψ_0)› **by** *blast*
hence *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) (*?T*$_\Sigma$ Ψ_0))
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *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 ∀ σ ∈ *set ?Σ*. ∃ σ′ ∈ *set ?Σ′*. *mset* σ = *mset* σ′
by *fastforce*
hence ∀ σ ∈ *set ?Σ*. ∃ σ′ ∈ *set ?Σ′*. *length* σ = *length* σ′
using *mset-eq-length* **by** *blast*
have *mset* (*?S* Ψ_0) ⊆# *mset* (𝔙 Ξ)
by (*simp add*: Ψ_0(*1*))
{
fix *n*
have ∀ Ψ. *mset* Ψ ⊆# *mset* (𝔙 Ξ) ⟶
sorted (*map* (− *?M*) Ψ) ⟶
length Ψ = *n* ⟶
(∀ σ′ ∈ *set* (*map* (*map snd* ∘ *remdups*) (*?T*$_\Sigma$ Ψ)). *length* σ′ + *1*
≥ *n*)
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Ψ :: (′*a list* × ′*a*) *list*
assume *A*: *mset* Ψ ⊆# *mset* (𝔙 Ξ)
and *B*: *sorted* (*map* (− *?M*) Ψ)
and *C*: *length* Ψ = *n* + *1*
obtain Δ δ **where** (Δ, δ) = *hd* Ψ
using *prod.collapse* **by** *blast*
let *?Ψ′* = *tl* Ψ
have *mset ?Ψ′* ⊆# *mset* (𝔙 Ξ) **using** *A*
by (*induct* Ψ, *simp*, *simp*, *meson mset-subset-eq-insertD subset-mset-def*)
moreover
have *sorted* (*map* (− *?M*) (*tl* Ψ))
using *B*
by (*simp add*: *map-tl sorted-tl*)
moreover have *length ?Ψ′* = *n* **using** *C*
by *simp*
ultimately have ⋆: ∀ σ′ ∈ *set* (*map* (*map snd* ∘ *remdups*) (*?T*$_\Sigma$ *?Ψ′*)).
length σ′ + *1* ≥ *n*
using *Suc*
by *blast*
from *C* **have** Ψ = (Δ, δ) # *?Ψ′*
by (*metis* ‹(Δ, δ) = *hd* Ψ›
One-nat-def

```
              add-is-0
              list.exhaust-sel
              list.size(3)
              nat.simps(3))
      have distinct ((Δ, δ) # ?Ψ′)
        using A ‹Ψ = (Δ, δ) # ?Ψ′›
            core-optimal-pre-witness-distinct
            mset-distinct-msub-down
        by fastforce
      hence set ((Δ, δ) # ?Ψ′) ⊆ set (𝔙 Ξ)
        by (metis A ‹Ψ = (Δ, δ) # ?Ψ′›
              Un-iff
              mset-le-perm-append
              perm-set-eq set-append
              subsetI)
      hence ∀ (Δ′, δ′) ∈ set ?Ψ′. (Δ, δ) ≠ (Δ′, δ′)
            ∀ (Δ′, δ′) ∈ set (𝔙 Ξ). ((Δ, δ) ≠ (Δ′, δ′)) ⟶ (length Δ ≠ length
Δ′)
            set ?Ψ′ ⊆ set (𝔙 Ξ)
        using core-optimal-pre-witness-length-iff-eq [where Ψ=Ξ]
            ‹distinct ((Δ, δ) # ?Ψ′)›
        by auto
      hence ∀ (Δ′, δ′) ∈ set ?Ψ′. length Δ ≠ length Δ′
            sorted (map (− ?ℳ) ((Δ, δ) # ?Ψ′))
        using B ‹Ψ = (Δ, δ) # ?Ψ′›
        by (fastforce, auto)
      hence ∀ (Δ′, δ′) ∈ set ?Ψ′. length Δ > 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 (𝔙 Δ) ∪ (set σ')
            σ' ∈ set (?T_Σ Ψ)
            by auto
          have ∀ (Δ', δ') ∈ set (𝔙 Δ). length Δ > length Δ'
            by (metis (mono-tags, lifting)
                      case-prodI2
                      core-optimal-pre-witness-nonelement
                      not-le)
          hence ∀ (Δ', δ') ∈ set (𝔙 Δ). n > length Δ'
            using ‹n > length Δ› by auto
          then show ?thesis using 0 σ' ‹ψ' ∈ 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 $\forall\ \sigma \in$ *set* (*?T*$_\Sigma$ *?*Ψ'). $\forall\ (\Delta', \delta') \in$ *set* σ. *length* $\Delta >$ *length* Δ'
using ‹$\forall\ (\Delta', \delta') \in$ *set ?*Ψ'. *length* $\Delta >$ *length* Δ'›
by *blast*
then show *?thesis* **using** *True* $\star$ ψ(*1*) **by** *fastforce*
next
case *False*
have $\forall\ (\Delta', \delta') \in$ *set ?*Ψ'. *length* $\Delta \geq$ *length* Δ'
using ‹$\forall\ (\Delta', \delta') \in$ *set ?*Ψ'. *length* $\Delta >$ *length* Δ'›
by *auto*
hence $\forall\ (\Delta', \delta') \in$ *set* Ψ. *length* $\Delta \geq$ *length* Δ'
using ‹$\Psi = (\Delta, \delta)$ # *?*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* $\Delta \geq n$
using *C*
by *simp*
have *length* Δ = *length* ($\mathfrak{V}\ \Delta$)
by (*induct* Δ, *simp+*)
hence *length* (*remdups* ($\mathfrak{V}\ \Delta$)) = *length* ($\mathfrak{V}\ \Delta$)
by (*simp add*: *core-optimal-pre-witness-distinct*)
hence *length* (*remdups* ($\mathfrak{V}\ \Delta$)) $\geq n$
using ‹*length* Δ = *length* ($\mathfrak{V}\ \Delta$)› ‹$n \leq$ *length* Δ›
by *linarith*
have *mset* (*remdups* ($\mathfrak{V}\ \Delta$ @ ψ)) = *mset* (*remdups* (ψ @ $\mathfrak{V}\ \Delta$))
by (*simp add*: *mset-remdups*)
hence *length* (*remdups* ($\mathfrak{V}\ \Delta$ @ ψ)) $\geq$ *length* (*remdups* ($\mathfrak{V}\ \Delta$))
by (*metis le-cases length-sub-mset mset-remdups-append-msub size-mset*)
hence *length* (*remdups* ($\mathfrak{V}\ \Delta$ @ ψ)) $\geq n$
using ‹$n \leq$ *length* (*remdups* ($\mathfrak{V}\ \Delta$))› *dual-order.trans* **by** *blast*
thus *?thesis* **using** *False* ψ(*2*)
by *simp*
qed
}
hence $\forall\ \sigma' \in$ *set* (*map* (*map snd* $\circ$ *remdups*) (*?T*$_\Sigma$ Ψ)). *length* $\sigma' \geq n$
by *blast*
}
then show *?case* **by** *fastforce*
qed
}
hence $\forall\ \sigma' \in$ *set ?*Σ'. *length* σ' + *1* $\geq$ *length* (*?*$\mathcal{S}$ Ψ_0)
using ‹*mset* (*?*$\mathcal{S}$ Ψ_0) $\subseteq$# *mset* ($\mathfrak{V}\ \Xi$)›
by *fastforce*
hence $\forall\ \sigma' \in$ *set ?*Σ'. *length* σ' + *1* $\geq$ *length* Ψ_0 **by** *simp*
hence $\forall\ \sigma \in$ *set ?*Σ. *length* σ + *1* $\geq$ *length* Ψ_0
using ‹$\forall\ \sigma \in$ *set ?*Σ. $\exists\ \sigma' \in$ *set ?*Σ'. *length* σ = *length* σ'›

```
      by fastforce
    thus ?thesis using Ψ₀ by fastforce
  qed
  have III: ∀ σ ∈ set ?Σ. mset σ ⊆# mset Ξ
  proof −
    have remdups (𝔙 Ξ) = 𝔙 Ξ
      by (simp add: core-optimal-pre-witness-distinct distinct-remdups-id)
    from Ψ₀(1) have set Ψ₀ ⊆ set (𝔙 Ξ)
      by (metis (no-types, lifting) ‹remdups (𝔙 Ξ) = 𝔙 Ξ›
                                   mset-remdups-set-sub-iff
                                   mset-remdups-subset-eq
                                   subset-mset.dual-order.trans)
    hence ∀ σ ∈ set (?T_Σ Ψ₀). set σ ⊆ set (𝔙 Ξ)
    proof (induct Ψ₀)
      case Nil
      then show ?case by simp
    next
      case (Cons ψ Ψ₀)
      hence ∀ σ ∈ set (?T_Σ Ψ₀). set σ ⊆ set (𝔙 Ξ) by auto
      obtain Δ δ where ψ = (Δ,δ) by fastforce
      hence (Δ, δ) ∈ set (𝔙 Ξ) using Cons by simp
      {
        fix σ :: ('a list × 'a) list
        assume ⋆: σ ∈ (#) (Δ, δ) ` set (?T_Σ Ψ₀) ∪ (@) (𝔙 Δ) ` set (?T_Σ Ψ₀)
        have set σ ⊆ set (𝔙 Ξ)
        proof (cases σ ∈ (#) (Δ, δ) ` set (?T_Σ Ψ₀))
          case True
          then show ?thesis
            using ‹∀ σ ∈ set (?T_Σ Ψ₀). set σ ⊆ set (𝔙 Ξ)› ‹(Δ, δ) ∈ set (𝔙 Ξ)›
            by fastforce
        next
          case False
          hence σ ∈ (@) (𝔙 Δ) ` set (?T_Σ Ψ₀) using ⋆ by simp
          moreover have set (𝔙 Δ) ⊆ set (𝔙 Ξ)
            using core-optimal-pre-witness-element-inclusion ‹(Δ, δ) ∈ set (𝔙 Ξ)›
            by fastforce
          ultimately show ?thesis
            using ‹∀ σ ∈ set (?T_Σ Ψ₀). set σ ⊆ set (𝔙 Ξ)›
            by force
        qed
      }
      hence ∀σ∈(#) (Δ, δ) ` set (?T_Σ Ψ₀) ∪ (@) (𝔙 Δ) ` set (?T_Σ Ψ₀). set σ
⊆ set (𝔙 Ξ)
        by auto
      thus ?case using ‹ψ = (Δ, δ)› by simp
    qed
    hence ∀ σ ∈ set (?T_Σ Ψ₀). mset (remdups σ) ⊆# mset (remdups (𝔙 Ξ))
      using mset-remdups-set-sub-iff by blast
    hence ∀ σ ∈ set ?Σ. mset σ ⊆# mset (map snd (𝔙 Ξ))
```

using *map-monotonic* ⟨*remdups* $(\mathfrak{V}\ \Xi) = \mathfrak{V}\ \Xi$⟩
by *auto*
moreover have *map snd* $(\mathfrak{V}\ \Xi) = \Xi$ **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:
$\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ \Sigma_0) \rightarrow \varphi)$
$\forall\ \sigma \in set\ \Sigma_0.\ mset\ \sigma \subseteq\#\ mset\ \Xi \wedge length\ \sigma + 1 \geq length\ \Psi$
by *blast*
moreover
have $(\Phi\ @\ \Psi) :\rightarrow \varphi = \Phi :\rightarrow (\Psi :\rightarrow \varphi)$ **by** (*induct* Φ, *simp*+)
hence $\vdash ((\Phi\ @\ \Psi) :\rightarrow \varphi) \leftrightarrow (\bigsqcap\ \Phi \rightarrow (\Psi :\rightarrow \varphi))$
by (*simp add*: *list-curry-uncurry*)
moreover have $\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ \Sigma_0) \rightarrow \varphi)$
$\rightarrow (\Phi\ @\ \Psi) :\rightarrow \varphi \leftrightarrow (\bigsqcap\ \Phi \rightarrow \Psi :\rightarrow \varphi)$
$\rightarrow (\Phi\ @\ \Psi) :\rightarrow \varphi \leftrightarrow ((\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Sigma_0)) \rightarrow \varphi)$
proof −
let $?\varphi = \langle\Psi :\rightarrow \varphi\rangle \leftrightarrow (\langle\bigsqcup\ (map\ \bigsqcap\ \Sigma_0)\rangle \rightarrow \langle\varphi\rangle)$
$\rightarrow \langle(\Phi\ @\ \Psi) :\rightarrow \varphi\rangle \leftrightarrow (\langle\bigsqcap\ \Phi\rangle \rightarrow \langle\Psi :\rightarrow \varphi\rangle)$
$\rightarrow \langle(\Phi\ @\ \Psi) :\rightarrow \varphi\rangle \leftrightarrow ((\langle\bigsqcap\ \Phi\rangle \sqcap \langle\bigsqcup\ (map\ \bigsqcap\ \Sigma_0)\rangle) \rightarrow \langle\varphi\rangle)$
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
let $?\Sigma = map\ ((@)\ \Phi)\ \Sigma_0$
have $\forall\,\varphi\ \psi\ \chi.\ \vdash (\varphi \rightarrow \psi) \rightarrow \chi \rightarrow \psi \vee \neg \vdash \chi \rightarrow \varphi$
by (*meson Modus-Ponens flip-hypothetical-syllogism*)
hence $\vdash ((\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap\ \Sigma_0)) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
using *append-dnf-distribute biconditional-def* **by** *fastforce*
ultimately have $\vdash (\Phi\ @\ \Psi) :\rightarrow \varphi \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
using *Modus-Ponens biconditional-transitivity-rule*
by *blast*
moreover
{
fix σ
assume $\sigma \in set\ ?\Sigma$
from *this* **obtain** σ_0 **where** σ_0: $\sigma = \Phi\ @\ \sigma_0\ \sigma_0 \in set\ \Sigma_0$ **by** (*simp*, *blast*)
hence $mset\ \sigma_0 \subseteq\#\ mset\ \Xi$ **using** $\Sigma_0(2)$ **by** *blast*
hence $mset\ \sigma \subseteq\#\ mset\ (\Phi\ @\ \Xi)$ **using** $\sigma_0(1)$ **by** *simp*
hence $mset\ \sigma \subseteq\#\ mset\ \Gamma$ **using** *assms*(*1*) *assms*(*2*)
by (*simp*, *meson subset-mset.dual-order.trans subset-mset.le-diff-conv2*)
moreover
have $length\ \sigma + 1 \geq length\ (\Phi\ @\ \Psi)$ **using** $\Sigma_0(2)\ \sigma_0$ **by** *simp*
ultimately have $mset\ \sigma \subseteq\#\ mset\ \Gamma\ length\ \sigma + 1 \geq length\ (\Phi\ @\ \Psi)$ **by** *auto*
}
ultimately

  show *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *unproving-core-optimal-witness*:
  assumes $\neg \vdash \varphi$
  shows $0 < (\| \Gamma \|_\varphi)$
    $= (\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
        $map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi\ \wedge$
        $1 + (\|\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \|_\varphi) = \|\ \Gamma\ \|_\varphi)$
proof (*rule iffI*)
  assume $0 < \|\ \Gamma\ \|_\varphi$
  from *this* **obtain** Ξ **where** Ξ: $\Xi \in \mathcal{C}\ \Gamma\ \varphi\ length\ \Xi < length\ \Gamma$
    using ‹$\neg \vdash \varphi$›
      complement-core-size-def
      core-size-intro
      unproving-core-existence
    by *fastforce*
  from *this* **obtain** ψ **where** ψ: $\psi \in set\ (\Gamma \ominus \Xi)$
    by (*metis* ‹$0 < \|\ \Gamma\ \|_\varphi$›
        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 $?\Sigma_B = \mathfrak{W}_\rightarrow\ \varphi\ (\psi\ \#\ \Xi)$
  have $\Diamond$: $mset\ (\psi\ \#\ \Xi) \subseteq\#\ mset\ \Gamma$
      $\psi\ \#\ \Xi :\vdash \varphi$
    using Ξ(*1*) ψ
      unproving-core-def
      list-deduction-theorem
      unproving-core-complement-deduction
      msub-listSubtract-elem-cons-msub [**where** Ξ=Ξ]
    by *blast+*
  moreover have $map\ snd\ ?\Sigma = \psi\ \#\ \Xi$ **by** (*induct* Ξ, *simp+*)
  ultimately have $?\Sigma_A :\vdash \varphi$
        $mset\ (map\ snd\ ?\Sigma) \subseteq\#\ mset\ \Gamma$
    using *core-optimal-witness-deduction*
      list-deduction-def weak-biconditional-weaken
    by (*metis+*)
  moreover
  {
    let $?\Gamma' = ?\Sigma_B\ @\ \Gamma \ominus map\ snd\ ?\Sigma$
    have *A*: $length\ ?\Sigma_B = 1 + length\ \Xi$
      by (*induct* Ξ, *simp+*)
    have *B*: $?\Sigma_B \in \mathcal{C}\ ?\Gamma'\ \varphi$
    proof −
      have $\neg\ ?\Sigma_B :\vdash \varphi$

```
  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 :⊢ φ = Ψ :⊢ φ
      using list-deduction-monotonic by blast
    ultimately have Ψ :⊢ φ by metis
    thus False using Ψ(1) unfolding unproving-core-def by blast
  qed
  moreover have ∃ Ψ. Ψ ∈ 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: ∀Ξ Γ φ. Ξ ∈ 𝒞 Γ φ ⟶ length Ξ = | Γ |φ
  using core-size-intro by blast
then have D: length Ξ = | Γ |φ
  using ‹Ξ ∈ 𝒞 Γ φ› by blast
```

have
$\forall(\Sigma :: 'a\ list)\ \Gamma\ n.\ (\neg\ mset\ \Sigma \subseteq\#\ mset\ \Gamma \vee length\ (\Gamma \ominus \Sigma) \neq n) \vee length\ \Gamma = n + length\ \Sigma$
using *listSubtract-msub-eq* **by** *blast*
then have E: $length\ \Gamma = length\ (\Gamma \ominus map\ snd\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi))) + length\ (\psi\ \#\ \Xi)$
using $\langle map\ snd\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi)) = \psi\ \#\ \Xi\rangle$ $\langle mset\ (\psi\ \#\ \Xi) \subseteq\#\ mset\ \Gamma\rangle$ **by** *presburger*
have $1 + length\ \Xi = |\ \mathfrak{W}_{\rightarrow}\ \varphi\ (\psi\ \#\ \Xi)\ @\ \Gamma \ominus map\ snd\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi))\ |_{\varphi}$
using $C\ B\ A$ **by** *presburger*
hence $1 + (\|\ map\ (uncurry\ (\rightarrow))\ ?\Sigma\ @\ \Gamma \ominus map\ snd\ ?\Sigma\ \|_{\varphi}) = \|\ \Gamma\ \|_{\varphi}$
using $D\ E$ $\langle map\ snd\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi)) = \psi\ \#\ \Xi\rangle$ *complement-core-size-def* **by** *force*
}
ultimately
show $\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
$map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi\ \wedge$
$1 + (\|\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \|_{\varphi}) = \|\ \Gamma\ \|_{\varphi}$
by *metis*
next
assume $\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
$map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi\ \wedge$
$1 + (\|\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \|_{\varphi}) = \|\ \Gamma\ \|_{\varphi}$
thus $0 < \|\ \Gamma\ \|_{\varphi}$
by *auto*
qed

primrec (**in** *Minimal-Logic*) *core-witness* :: $('a \times 'a)\ list \Rightarrow 'a\ list \Rightarrow ('a \times 'a)\ list$ $(\mathfrak{U})$
where
$\mathfrak{U}$ - [] = []
$|\ \mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi) = (case\ find\ (\lambda\ \sigma.\ \xi = snd\ \sigma)\ \Sigma\ of$
$None \Rightarrow \mathfrak{U}\ \Sigma\ \Xi$
$|\ Some\ \sigma \Rightarrow \sigma\ \#\ (\mathfrak{U}\ (remove1\ \sigma\ \Sigma)\ \Xi))$

lemma (**in** *Minimal-Logic*) *core-witness-right-msub*:
$mset\ (map\ snd\ (\mathfrak{U}\ \Sigma\ \Xi)) \subseteq\#\ mset\ \Xi$
proof −
have $\forall\ \Sigma.\ mset\ (map\ snd\ (\mathfrak{U}\ \Sigma\ \Xi)) \subseteq\#\ mset\ \Xi$
proof (*induct* Ξ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
{
fix Σ
have $mset\ (map\ snd\ (\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi))) \subseteq\#\ mset\ (\xi\ \#\ \Xi)$
proof (*cases find* $(\lambda\ \sigma.\ \xi = snd\ \sigma)\ \Sigma$)
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 Minimal-Logic) core-witness-left-msub:
  mset (𝔘 Σ Ξ) ⊆# mset Σ
proof −
  have ∀ Σ. mset (𝔘 Σ Ξ) ⊆# mset Σ
  proof (induct Ξ)
    case Nil
    then show ?case by simp
  next
    case (Cons ξ Ξ)
    {
      fix Σ
      have mset (𝔘 Σ (ξ # Ξ)) ⊆# 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 σ ∈ 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 *Minimal-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-Propositional-Logic*) *witness-list-implication-rule*:
$\vdash$ (*map* (*uncurry* ($\sqcup$)) Σ $:\rightarrow$ φ) $\rightarrow$ $\bigsqcap$ (*map* (λ (χ, ξ). ($\chi \rightarrow \xi$) $\rightarrow \varphi$) Σ) $\rightarrow \varphi$
proof (*induct* Σ)
case *Nil*
then show *?case* **using** *Axiom-1* **by** *simp*
next
case (*Cons* σ Σ)
let $?\chi$ = *fst* σ
let $?\xi$ = *snd* σ
let $?\Sigma_A$ = *map* (*uncurry* ($\sqcup$)) Σ
let $?\Sigma_B$ = *map* (λ (χ, ξ). ($\chi \rightarrow \xi$) $\rightarrow \varphi$) Σ
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 (λ σ. (*fst* σ $\rightarrow$ *snd* σ) $\rightarrow \varphi$) = (λ (χ, ξ). ($\chi \rightarrow \xi$) $\rightarrow \varphi$)
uncurry ($\sqcup$) = (λ σ. *fst* σ $\sqcup$ *snd* σ)
by *fastforce+*

hence $(\lambda\ (\chi, \xi).\ (\chi \to \xi) \to \varphi)\ \sigma = (?\chi \to ?\xi) \to \varphi$
uncurry $(\sqcup)\ \sigma = ?\chi \sqcup ?\xi$
by *metis+*
ultimately show *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *witness-core-size-increase*:
assumes $\neg \vdash \varphi$
and *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ
and *map* (*uncurry* $(\sqcup)$) Σ $:\vdash \varphi$
shows $(\mid \Gamma \mid_\varphi) < (\mid$ *map* (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ *map snd* $\Sigma \mid_\varphi)$
proof −
from ‹$\neg \vdash \varphi$› **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* $(\to)$) ($\mathfrak{U}\ \Sigma\ \Xi$)
have *mset* $\Sigma =$ *mset* ($\mathfrak{U}\ \Sigma\ \Xi$ @ $?\Sigma'$) **by** (*simp add*: *core-witness-left-msub*)
hence *set* (*map* (*uncurry* $(\sqcup)$) Σ) = *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 $((\Phi$ @ $\Psi) :\to \varphi) = (\Phi :\to (\Psi :\to \varphi))$
by (*induct* Φ, *simp+*)
hence $(\Phi$ @ $\Psi) :\vdash \varphi = \Phi :\vdash (\Psi :\to \varphi)$
unfolding *list-deduction-def*
by (*induct* Φ, *simp+*)
}
ultimately have *map* (*uncurry* $(\sqcup)$) ($\mathfrak{U}\ \Sigma\ \Xi$) $:\vdash$ *map* (*uncurry* $(\sqcup)$) $?\Sigma' :\to \varphi$
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* $(\sqcup)$) $?\Sigma' :\to \varphi$
using *list-deduction-monotonic* **by** *blast*
hence $?\Sigma\Xi' :\vdash \prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \to \gamma) \to \varphi)\ ?\Sigma'$) $\to \varphi$
using *list-deduction-modus-ponens*
list-deduction-weaken
witness-list-implication-rule
by *blast*
hence $?\Sigma\Xi'$ \$⊢ [$\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \to \gamma) \to \varphi)\ ?\Sigma'$) $\to \varphi$]
using *segmented-deduction-one-collapse* **by** *metis*
hence
$?\Sigma\Xi'$ @ (*map snd* ($\mathfrak{U}\ \Sigma\ \Xi$)) $\ominus$ (*map snd* ($\mathfrak{U}\ \Sigma\ \Xi$))
\$⊢ [$\prod$ (*map* $(\lambda\ (\chi, \gamma).\ (\chi \to \gamma) \to \varphi)\ ?\Sigma'$) $\to \varphi$]
by *simp*

hence *map snd* ($\mathfrak{U}$ Σ Ξ) $\$\vdash$ [$\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$]
  using *segmented-witness-left-split* [**where** Γ=*map snd* ($\mathfrak{U}$ Σ Ξ)
    and Σ=$\mathfrak{U}$ Σ Ξ]
  by *fastforce*
hence *map snd* ($\mathfrak{U}$ Σ Ξ) $\$\vdash$ [$\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$]
  using *core-witness-right-projection* **by** *auto*
hence *map snd* ($\mathfrak{U}$ Σ Ξ) :$\vdash$ $\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$
  using *segmented-deduction-one-collapse* **by** *blast*
hence $\star$:
  map snd ($\mathfrak{U}$ Σ Ξ) @ $\Xi \ominus$ (*map snd* Σ) :$\vdash$ $\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$)
?Σ') $\to \varphi$
  (**is** *?*Ξ_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* Σ) $\cap$ Ξ) + *mset* ($\Xi \ominus$ (*map snd* Σ))
  by (*metis subset-mset.inf-commute list-intersect-mset-homomorphism union-commute*)
hence *mset* Ξ = *mset* (*map snd* ($\mathfrak{U}$ Σ Ξ)) + *mset* ($\Xi \ominus$ (*map snd* Σ))
  using *core-witness-right-projection* **by** *simp*
hence *mset* Ξ = *mset* *?*Ξ_0
  by *simp*
hence *set* Ξ = *set* *?*Ξ_0
  by (*metis mset-eq-setD*)
have $\neg$ *?*Ξ_0 :$\vdash$ $\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ')
proof (*rule notI*)
  assume *?*Ξ_0 :$\vdash$ $\bigsqcap$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ')
  hence *?*Ξ_0 :$\vdash$ φ
    using $\star$ *list-deduction-modus-ponens* **by** *blast*
  hence Ξ :$\vdash$ φ
    using *list-deduction-monotonic* ‹*set* Ξ = *set* *?*Ξ_0› **by** *blast*
  thus *False*
    using Ξ *unproving-core-def* **by** *blast*
qed
moreover
have *mset* (*map snd* ($\mathfrak{U}$ Σ Ξ)) $\subseteq$# *mset* *?*Ξ_0
  mset (*map* (*uncurry* ($\to$)) ($\mathfrak{U}$ Σ Ξ) @ *?*Ξ_0 $\ominus$ *map snd* ($\mathfrak{U}$ Σ Ξ))
  = *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{U}$ Σ Ξ) @ $\Xi \ominus$ (*map snd* Σ))
  (**is** - = *mset* *?*Ξ_1)
  by *auto*
hence *?*Ξ_1 $\preceq$ *?*Ξ_0
  by (*metis add.commute*
    witness-stronger-theory
    add-diff-cancel-right'
    listSubtract.simps(*1*)
    listSubtract-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
    ¬ ?Ξ₁ :⊢ ⨅ (map (λ (χ, γ). (χ → γ) → φ) ?Σ')
    using stronger-theory-deduction-monotonic by blast
  from this obtain χ γ where
    (χ,γ) ∈ set ?Σ'
    ¬ (χ → γ) # ?Ξ₁ :⊢ φ
    using list-deduction-theorem
    by fastforce
  have mset (χ → γ # ?Ξ₁) ⊆# 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 ?Σ'› 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-listSubtract-mset-equivalence)
      hence γ ∈# mset (map snd Σ ⊖ (map snd Σ ∩ Ξ))
        by (metis core-witness-right-projection listSubtract-mset-homomorphism)
      hence γ ∈# mset (map snd Σ ⊖ Ξ)
        by (metis add-diff-cancel-right'
                listSubtract-mset-homomorphism
                list-diff-intersect-comp)
      moreover from assms(2) have mset (map snd Σ ⊖ Ξ) ⊆# mset (Γ ⊖ Ξ)
         by (simp, metis listSubtract-monotonic listSubtract-mset-homomorphism
mset-map)
      ultimately have γ ∈# mset (Γ ⊖ Ξ)
        by (simp add: mset-subset-eqD)
      hence γ ∈ set (Γ ⊖ Ξ)
        using set-mset-mset by fastforce
      hence γ ∈ set Γ − set Ξ
        using Ξ by simp
      hence γ ∉ set Ξ
        by blast
      hence ∀ Σ. (χ,γ) ∉ set (𝔘 Σ Ξ)
      proof (induct Ξ)
        case Nil
        then show ?case by simp
      next
        case (Cons ξ Ξ)
        {
          fix Σ
```

have $(\chi, \gamma) \notin$ *set* $(\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi))$
proof (*cases find* $(\lambda\sigma.\ \xi =$ *snd* $\sigma)\ \Sigma)$
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 ‹$(\chi,\gamma) \in$ *set* $?\Sigma'$› **have** $(\chi,\gamma) \in$ *set* Σ
by (*meson listSubtract-set-trivial-upper-bound subsetCE*)
ultimately show *?thesis* **by** *fastforce*
qed
with ‹$(\chi, \gamma) \in$ *set* $?\Sigma'$› **have** *mset* $((\chi,\gamma)\ \#\ \mathfrak{U}\ \Sigma\ \Xi) \subseteq\#$ *mset* Σ
by (*meson core-witness-left-msub msub-listSubtract-elem-cons-msub*)
hence *mset* $(\chi \to \gamma\ \#\ ?B) \subseteq\#$ *mset* (*map* (*uncurry* $(\to)$) Σ)
by (*metis* (*no-types*, *lifting*) ‹$(\chi, \gamma) \in$ *set* $?\Sigma'$›
core-witness-left-msub
map-listSubtract-mset-equivalence
map-monotonic
mset-eq-setD msub-listSubtract-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* $\Sigma)) \subseteq\#$ *mset* $(\Gamma \ominus$ (*map snd* $\Sigma))$
using *listSubtract-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* $\Xi =$ *mset* $?\Xi_0$› *mset-eq-length* **by** *fastforce*
hence *length* $((\chi \to \gamma)\ \#\ ?\Xi_1) =$ *length* $\Xi + 1$
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 Ω ≥ (| Γ |φ) + 1
    using unproving-core-def
    by (metis (no-types, lifting) ‹¬ χ → γ # ?Ξ1 :⊢ φ› mem-Collect-eq)
  thus ?thesis
    using Ω core-size-intro by auto
qed

lemma (in Classical-Propositional-Logic) unproving-core-stratified-deduction-lower-bound:
  assumes ¬ ⊢ φ
    shows (Γ #⊢ n φ) = (n ≤ ‖ Γ ‖φ)
proof −
  have ∀ Γ. (Γ #⊢ n φ) = (n ≤ ‖ Γ ‖φ)
  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) ⊆# mset Γ
        map (uncurry (⊔)) Σ :⊢ φ
        map (uncurry (→)) Σ @ Γ ⊖ (map snd Σ) #⊢ n φ
        by fastforce
      let ?Γ′ = map (uncurry (→)) Σ @ Γ ⊖ (map snd Σ)
      have length Γ = length ?Γ′
        using Σ(1) listSubtract-msub-eq by fastforce
      hence (‖ Γ ‖φ) > (‖ ?Γ′ ‖φ)
        by (metis Σ(1) Σ(2) ‹¬ ⊢ φ›
                  witness-core-size-increase
                  length-core-decomposition
                  add-less-cancel-right
                  nat-add-left-cancel-less)
      with Σ(3) Suc.hyps have Suc n ≤ ‖ Γ ‖φ
        by auto
    }
    moreover
    {
      fix Γ
      assume Suc n ≤ ‖ Γ ‖φ
      from this obtain Σ where Σ:
        mset (map snd Σ) ⊆# mset Γ
        map (uncurry (⊔)) Σ :⊢ φ
        1 + (‖ map (uncurry (→)) Σ @ Γ ⊖ map snd Σ ‖φ) = ‖ Γ ‖φ
        (is 1 + (‖ ?Γ′ ‖φ) = ‖ Γ ‖φ)
        by (metis Suc-le-D assms unproving-core-optimal-witness zero-less-Suc)
      have n ≤ ‖ ?Γ′ ‖φ
```

using $\Sigma(3)$ ‹*Suc* $n \leq \| \Gamma \|_\varphi$› **by** *linarith*
hence *?*Γ' $\#\vdash$ 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-Propositional-Logic*) *stratified-deduction-tautology-equiv*:
($\forall$ n. Γ $\#\vdash$ n φ) = $\vdash$ φ
proof (*cases* $\vdash$ φ)
case *True*
then show *?thesis*
by (*simp add*: *stratified-deduction-tautology-weaken*)
next
case *False*
have $\neg$ Γ $\#\vdash$ (*1* + *length* Γ) φ
proof (*rule notI*)
assume Γ $\#\vdash$ (*1* + *length* Γ) φ
hence *1* + *length* $\Gamma \leq \| \Gamma \|_\varphi$
using ‹$\neg$ $\vdash$ φ› *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 ‹$\neg$ $\vdash$ φ› **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *unproving-core-max-stratified-deduction*:
Γ $\#\vdash$ n φ = ($\forall$ $\Phi \in \mathcal{C}$ Γ φ. $n \leq$ *length* ($\Gamma \ominus \Phi$))
proof (*cases* $\vdash$ φ)
case *True*
from ‹$\vdash$ φ› **have** Γ $\#\vdash$ n φ
using *stratified-deduction-tautology-weaken*
by *blast*
moreover from ‹$\vdash$ φ› **have** $\mathcal{C}$ Γ φ = {}
using *unproving-core-existence* **by** *auto*
hence $\forall$ $\Phi \in \mathcal{C}$ Γ φ. $n \leq$ *length* ($\Gamma \ominus \Phi$) **by** *blast*
ultimately show *?thesis* **by** *meson*
next
case *False*
from ‹$\neg$ $\vdash$ φ› **have** (Γ $\#\vdash$ n φ) = ($n \leq \| \Gamma \|_\varphi$)
by (*simp add*: *unproving-core-stratified-deduction-lower-bound*)
moreover have ($n \leq \| \Gamma \|_\varphi$) = ($\forall$ $\Phi \in \mathcal{C}$ Γ φ. $n \leq$ *length* ($\Gamma \ominus \Phi$))
proof (*rule iffI*)
assume $n \leq \| \Gamma \|_\varphi$
{

fix Φ
assume $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
hence $n \leq$ *length* $(\Gamma \ominus \Phi)$
using $\langle n \leq \|\ \Gamma\ \|_\varphi \rangle$ *complement-core-size-intro* **by** *auto*
}
thus $\forall\, \Phi \in \mathcal{C}\ \Gamma\ \varphi.\ n \leq$ *length* $(\Gamma \ominus \Phi)$ **by** *blast*
next
assume $\forall\, \Phi \in \mathcal{C}\ \Gamma\ \varphi.\ n \leq$ *length* $(\Gamma \ominus \Phi)$
with $\langle \neg \vdash \varphi \rangle$ **obtain** Φ **where**
$\Phi \in \mathcal{C}\ \Gamma\ \varphi$
$n \leq$ *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** *Logical-Probability*) *list-probability-upper-bound*:
$(\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma) \leq$ *real* (*length* Γ)
proof (*induct* Γ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* γ Γ)
moreover have $Pr\ \gamma \leq 1$ **using** *unity-upper-bound* **by** *blast*
ultimately have $Pr\ \gamma + (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma) \leq 1 +$ *real* (*length* Γ) **by** *linarith*
then show *?case* **by** *simp*
qed

theorem (**in** *Classical-Propositional-Logic*) *binary-limited-stratified-deduction-completeness*:
$(\forall\ Pr \in$ *Dirac-Measures*. *real* $n * Pr\ \varphi \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)) = \boldsymbol{\sim}\ \Gamma\ \#\!\vdash\ n\ (\sim \varphi)$
proof −
{
fix $Pr :: {}'a \Rightarrow$ *real*
assume $Pr \in$ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *Dirac-Measures-def*
by *auto*
assume $\boldsymbol{\sim}\ \Gamma\ \#\!\vdash\ n\ (\sim \varphi)$
moreover have *replicate* $n\ (\sim \varphi) = \boldsymbol{\sim}$ (*replicate* $n\ \varphi$)
by (*induct* n, *auto*)
ultimately have $\boldsymbol{\sim}\ \Gamma\ \$\!\vdash\ \boldsymbol{\sim}$ (*replicate* $n\ \varphi$)
using *stratified-segmented-deduction-replicate* **by** *metis*
hence $(\sum \varphi \leftarrow ($*replicate* $n\ \varphi).\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
using *segmented-deduction-summation-introduction*
by *blast*
moreover have $(\sum \varphi \leftarrow ($*replicate* $n\ \varphi).\ Pr\ \varphi) =$ *real* $n * Pr\ \varphi$

by (*induct n*, *simp*, *simp add*: *semiring-normalization-rules*(*3*))
ultimately have *real n* $*$ *Pr* $\varphi \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
by *simp*
}
moreover
{
assume $\neg \sim \Gamma \ \#\vdash\ n\ (\sim \varphi)$
have $\exists\ Pr \in$ *Dirac-Measures*. *real n* $*$ *Pr* $\varphi > (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
proof −
have $\exists \Phi.\ \Phi \in \mathcal{C}\ (\sim \Gamma)\ (\sim \varphi)$
using $\langle \neg \sim \Gamma \ \#\vdash\ n\ (\sim \varphi)\rangle$
unproving-core-existence
stratified-deduction-tautology-weaken
by *blast*
from *this* **obtain** Φ **where** Φ: $(\sim \Phi) \in \mathcal{C}\ (\sim \Gamma)\ (\sim \varphi)$ *mset* $\Phi \subseteq\#$ *mset* Γ
by (*metis* (*mono-tags*, *lifting*)
unproving-core-def
mem-Collect-eq
mset-sub-map-list-exists)
hence $\neg \vdash \varphi \rightarrow \bigsqcup\ \Phi$
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
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** *Logical-Probability* $(\lambda\ \varphi. \vdash \varphi)\ (\rightarrow)\ \bot$ *?Pr*
unfolding *Dirac-Measures-def*
by *auto*
have $\forall\ \varphi \in$ *set* Φ. *?Pr* $\varphi = 0$
using Φ(*1*) Ω(*1*) Ω(*3*) *arbitrary-disjunction-exclusion-MCS* **by** *auto*
with Φ(*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 $(Cons\ \varphi\ \Phi)$
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*:
$\forall (f :: 'a \Rightarrow real)\ (\Gamma :: 'a\ list)\ \Phi.$
$\neg$ *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* $(map\ f\ (\varphi\ \#\ \Phi))\ \#\ map\ f\ (\Gamma \ominus (\varphi\ \#\ \Phi)))$
$= ($*sum-list* $(map\ f\ \Gamma)::real)$
using *A Cons.prems(1)* **by** *blast*
have *E*: *mset* $\Phi \subseteq\#$ *mset* Γ
using *Cons.prems(1) subset-mset.dual-order.trans* **by** *force*
then have *F*: $\forall f.\ (0::real) =$ *sum-list* $(map\ f\ \Phi)$
$\vee$ *sum-list* $(map\ f\ \Gamma) \neq$ *sum-list* $(map\ f\ (\Gamma \ominus \Phi))$
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*: $\forall\Gamma\ r::real.\ r = (\sum\gamma\leftarrow\Gamma.\ ?Pr\ \gamma)$
$\vee\ \omega \in$ *set* Φ
$\vee\ r \neq (\sum\gamma\leftarrow(\varphi\ \#\ \Gamma).\ ?Pr\ \gamma)$
using *Cons.prems(2)* **by** *auto*
have $(1::real) \neq 0$ **by** *linarith*
moreover
{ assume $\omega \notin$ *set* Φ
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 $(\sum\gamma\leftarrow\Gamma.\ ?Pr\ \gamma) \leq$ *real* (*length* $(\Gamma \ominus \Phi))$
using *list-probability-upper-bound*
by *auto*
moreover
have *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-listSubtract-length-equiv)
hence *real* (*length* ($\sim$ Γ ⊖ $\sim$ Φ)) < *real n*
by *simp*
with Ω(*2*) **have** *real* (*length* ($\sim$ Γ ⊖ $\sim$ Φ)) < *real n* ∗ *?Pr φ*
by *simp*
moreover
have ($\sim$ (Γ ⊖ Φ)) <~~> ($\sim$ Γ ⊖ $\sim$ Φ)
by (*metis* Φ(*2*) *map-listSubtract-mset-equivalence mset-eq-perm*)
with *perm-length* **have** *length* (Γ ⊖ Φ) = *length* ($\sim$ Γ ⊖ $\sim$ Φ)
by *fastforce*
hence *real* (*length* (Γ ⊖ Φ)) = *real* (*length* ($\sim$ Γ ⊖ $\sim$ Φ))
by *simp*
ultimately show *?thesis*
by *force*
qed
}
ultimately show *?thesis* **by** *fastforce*
qed

lemma (**in** *Classical-Propositional-Logic*) *binary-segmented-deduction-completeness*:
(∀ *Pr* ∈ *Dirac-Measures*. (∑φ←Φ. *Pr* φ) ≤ (∑γ←Γ. *Pr* γ)) = $\sim$ Γ $⊢ $\sim$ Φ
proof −
{
fix *Pr* :: ′*a* ⇒ *real*
assume *Pr* ∈ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* (λ φ. ⊢ φ) (→) ⊥ *Pr*
unfolding *Dirac-Measures-def*
by *auto*
assume $\sim$ Γ $⊢ $\sim$ Φ
hence (∑φ←Φ. *Pr* φ) ≤ (∑γ←Γ. *Pr* γ)
using *segmented-deduction-summation-introduction*
by *blast*
}
moreover
{
assume ¬ $\sim$ Γ $⊢ $\sim$ Φ
have ∃ *Pr* ∈ *Dirac-Measures*. (∑φ←Φ. *Pr* φ) > (∑γ←Γ. *Pr* γ)
proof −
from ‹¬ $\sim$ Γ $⊢ $\sim$ Φ› **have** ¬ $\sim$ ($\sim$ Φ) @ $\sim$ Γ #⊢ (*length* ($\sim$ Φ)) ⊥
using *segmented-stratified-falsum-equiv* **by** *blast*
moreover
have $\sim$ ($\sim$ Φ) @ $\sim$ Γ #⊢ (*length* ($\sim$ Φ)) ⊥ = $\sim$ ($\sim$ Φ) @ $\sim$ Γ #⊢ (*length* Φ) ⊥
by (*induct* Φ, *auto*)
moreover have ⊢ ∼ ⊤ → ⊥
by (*simp add*: *negation-def*)
ultimately have ¬ $\sim$ ($\sim$ Φ @ Γ) #⊢ (*length* Φ) (∼ ⊤)
using *stratified-deduction-implication* **by** *fastforce*
from *this* **obtain** *Pr* **where** *Pr*:

$Pr \in$ *Dirac-Measures*
real (*length* Φ) $*$ Pr $\top$ $>$ ($\sum\gamma\leftarrow$ ($\sim$ Φ @ Γ). Pr γ)
using *binary-limited-stratified-deduction-completeness*
by *fastforce*
from *this* **interpret** *Logical-Probability* (λ φ. $\vdash$ φ) ($\to$) $\perp$ *Pr*
unfolding *Dirac-Measures-def*
by *auto*
from *Pr(2)* **have** *real* (*length* Φ) $>$ ($\sum\gamma\leftarrow$ $\sim$ Φ. Pr γ) + ($\sum\gamma\leftarrow$ Γ. Pr γ)
by (*simp add*: *Unity*)
moreover have ($\sum\gamma\leftarrow$ $\sim$ Φ. Pr γ) = *real* (*length* Φ) $-$ ($\sum\gamma\leftarrow$ Φ. Pr γ)
using *complementation*
by (*induct* Φ, *auto*)
ultimately show *?thesis*
using *Pr(1)* **by** *auto*
qed
}
ultimately show *?thesis* **by** *fastforce*
qed

theorem (**in** *Classical-Propositional-Logic*) *segmented-deduction-completeness*:
($\forall$ $Pr \in$ *Logical-Probabilities*. ($\sum\varphi\leftarrow\Phi$. Pr φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. Pr γ)) = $\sim$ Γ \$$\vdash$ $\sim$ Φ
proof $-$
{
fix Pr :: $'a \Rightarrow$ *real*
assume $Pr \in$ *Logical-Probabilities*
from *this* **interpret** *Logical-Probability* (λ φ. $\vdash$ φ) ($\to$) $\perp$ *Pr*
unfolding *Logical-Probabilities-def*
by *auto*
assume $\sim$ Γ \$$\vdash$ $\sim$ Φ
hence ($\sum\varphi\leftarrow\Phi$. Pr φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. Pr γ)
using *segmented-deduction-summation-introduction*
by *blast*
}
thus *?thesis*
using *Dirac-Measures-subset binary-segmented-deduction-completeness*
by *fastforce*
qed

theorem (**in** *Classical-Propositional-Logic*) *weakly-additive-completeness-collapse*:
($\forall$ $Pr \in$ *Logical-Probabilities*. ($\sum\varphi\leftarrow\Phi$. Pr φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. Pr γ))
= ($\forall$ $Pr \in$ *Dirac-Measures*. ($\sum\varphi\leftarrow\Phi$. Pr φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. Pr γ))
by (*simp add*: *binary-segmented-deduction-completeness*
segmented-deduction-completeness)

lemma (**in** *Classical-Propositional-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-Propositional-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-Propositional-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-Propositional-Logic*) *stratified-deduction-completeness*:
$(\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)) = (\sim \Gamma\ @\ \Phi)\ \#\vdash$
$(\textit{length}\ \Phi)\ \bot$
proof −
have $(\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
$= \sim (\sim \Phi)\ @ \sim \Gamma\ \#\vdash\ (\textit{length}\ (\sim \Phi))\ \bot$
using *binary-segmented-deduction-completeness segmented-stratified-falsum-equiv*
by *blast*
also have ... $= \sim (\sim \Phi)\ @ \sim \Gamma\ \#\vdash\ (\textit{length}\ \Phi)\ \bot$ **by** (*induct* Φ, *auto*)
also have ... $= \sim \Gamma\ @ \sim (\sim \Phi)\ \#\vdash\ (\textit{length}\ \Phi)\ \bot$
by (*simp add*: *segmented-left-commute stratified-segmented-deduction-replicate*)
also have ... $= \sim \Gamma\ @\ \Phi\ \#\vdash\ (\textit{length}\ \Phi)\ \bot$
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-Propositional-Logic*) *complement-core-completeness*:
$(\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)) = (\textit{length}\ \Phi \leq \|$
$\sim \Gamma\ @\ \Phi\ \|_\bot)$
proof (*cases* $\vdash \bot$)
case *True*
hence $\mathcal{C}\ (\sim \Gamma\ @\ \Phi)\ \bot = \{\}$
using *unproving-core-existence* **by** *auto*
hence *length* $(\sim \Gamma\ @\ \Phi) = \| \sim \Gamma\ @\ \Phi\ \|_\bot$
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-Propositional-Logic*) *binary-core-partial-completeness*:
 $(\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)) = ((|\ \sim\Gamma\ @\ \Phi\ |_\perp) \leq \textit{length}\ \Gamma)$
proof −
 {
  fix $Pr :: 'a \Rightarrow \textit{real}$
  obtain $\varrho :: 'a\ \textit{list} \Rightarrow 'a\ \textit{list} \Rightarrow 'a \Rightarrow \textit{real}$ **where**
   $(\forall\Phi\ \Gamma.\ \varrho\ \Phi\ \Gamma \in \textit{Dirac-Measures} \wedge \neg\ (\sum\varphi\leftarrow\Phi.\ (\varrho\ \Phi\ \Gamma)\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ (\varrho\ \Phi\ \Gamma)\ \gamma)$
    $\vee\ \textit{length}\ \Phi \leq \|\ \sim\Gamma\ @\ \Phi\ \|_\perp)$
   $\wedge\ (\forall\Phi\ \Gamma.\ \textit{length}\ \Phi \leq (\|\ \sim\Gamma\ @\ \Phi\ \|_\perp)$
    $\longrightarrow (\forall Pr \in \textit{Dirac-Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)))$
  using *complement-core-completeness* **by** *moura*
 moreover have $\forall\Gamma\ \varphi\ n.\ \textit{length}\ \Gamma - n \leq (\|\ \Gamma\ \|_\varphi) \vee (|\ \Gamma\ |_\varphi) - n \neq \mathit{0}$
  by (*metis add-diff-cancel-right'*
    cancel-ab-semigroup-add-class.diff-right-commute
    diff-is-0-eq length-core-decomposition)
 moreover have $\forall\ \Gamma\ \Phi\ n.\ \textit{length}\ (\Gamma\ @\ \Phi) - n \leq \textit{length}\ \Gamma \vee \textit{length}\ \Phi - n \neq \mathit{0}$
  by *force*
 ultimately have
   $(Pr \in \textit{Dirac-Measures} \longrightarrow (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
   $\wedge\ (|\ \sim\Gamma\ @\ \Phi\ |_\perp) \leq \textit{length}\ (\sim\Gamma)$
  $\vee$ $\neg\ (|\ \sim\Gamma\ @\ \Phi\ |_\perp) \leq \textit{length}\ (\sim\Gamma)$
   $\wedge\ (\exists Pr.\ Pr \in \textit{Dirac-Measures} \wedge \neg\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
  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-Propositional-Logic*) *nat-binary-probability*:
 $\forall\ Pr \in \textit{Dirac-Measures}.\ \exists n :: \textit{nat}.\ \textit{real}\ n = (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi)$
proof (*induct* Φ)
 case *Nil*
 then show *?case* **by** *simp*
next
 case (*Cons* φ Φ)
 {
  fix $Pr :: 'a \Rightarrow \textit{real}$

assume *Pr* $\in$ *Dirac-Measures*
from *Cons this* **obtain** *n* **where** *real n* = ($\sum\varphi'\leftarrow\Phi$. *Pr* φ') **by** *fastforce*
hence $\star$: ($\sum\varphi'\leftarrow\Phi$. *Pr* φ') = *real n* **by** *simp*
have $\exists$ *n*. *real n* = ($\sum\varphi'\leftarrow(\varphi$ # $\Phi)$. *Pr* φ')
proof (*cases Pr* φ = *1*)
  case *True*
  then show *?thesis*
    by (*simp add*: $\star$, *metis of-nat-Suc*)
next
  case *False*
  hence *Pr* φ = *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-Propositional-Logic*) *dirac-ceiling*:
$\forall$ *Pr* $\in$ *Dirac-Measures*.
  (($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)) = (($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ))
proof −
{
  fix *Pr*
  assume *Pr* $\in$ *Dirac-Measures*
  have (($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)) = (($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ))
  proof (*rule iffI*)
    assume *assm*: ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
    show ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
    proof (*rule ccontr*)
      assume $\neg$ ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
      moreover
      obtain *x* :: *int*
        and *y* :: *int*
        and *z* :: *int*
        where *xyz*: *x* = ($\sum\varphi\leftarrow\Phi$. *Pr* φ)
              y = $\lceil c\rceil$
              z = ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
        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* φ) + *c* > ($\sum\gamma\leftarrow\Gamma$. *Pr* γ) **using** *xyz* **by** *linarith*
      thus *False* **using** *assm* **by** *simp*
    qed
  next
    assume ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + $\lceil c\rceil$ $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)
    thus ($\sum\varphi\leftarrow\Phi$. *Pr* φ) + *c* $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ)

by *linarith*
qed
}
thus *?thesis* **by** *blast*
qed

lemma (**in** *Logical-Probability*) *probability-replicate-verum*:
fixes n :: *nat*
shows $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + n = (\sum\varphi\leftarrow(replicate\ n\ \top)\ @\ \Phi.\ Pr\ \varphi)$
using *Unity*
by (*induct n*, *auto*)

lemma (**in** *Classical-Propositional-Logic*) *dirac-collapse*:
$(\forall\ Pr \in Logical\text{-}Probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in Dirac\text{-}Measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lceil c\rceil \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
assume $\forall\ Pr \in Logical\text{-}Probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
hence $\forall\ Pr \in Dirac\text{-}Measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *Dirac-Measures-subset* **by** *fastforce*
thus $\forall\ Pr \in Dirac\text{-}Measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lceil c\rceil \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *dirac-ceiling* **by** *blast*
next
assume *assm*: $\forall\ Pr \in Dirac\text{-}Measures.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lceil c\rceil \leq (\sum\gamma\leftarrow\Gamma.\ Pr$
$\gamma)$
show $\forall\ Pr \in Logical\text{-}Probabilities.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
proof (*cases* $c \geq 0$)
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 $Pr \in Dirac\text{-}Measures$
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\to)\ \bot\ Pr$
unfolding *Dirac-Measures-def*
by *auto*
have $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lceil c\rceil \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *assm* ‹$Pr \in Dirac\text{-}Measures$› **by** *blast*
hence $(\sum\varphi\leftarrow(replicate\ n\ \top)\ @\ \Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using ‹$real\ n = \lceil c\rceil$›
probability-replicate-verum [**where** Φ=Φ **and** n=n]
by *metis*
}
hence $\forall\ Pr \in Dirac\text{-}Measures.\ (\sum\varphi\leftarrow(replicate\ n\ \top)\ @\ \Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.$

$Pr\ \gamma$)
    by *blast*
  hence †: $\forall\ Pr \in$ *Logical-Probabilities*.
      $(\sum\varphi\leftarrow(\textit{replicate}\ n\ \top)\ @\ \Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
    using *weakly-additive-completeness-collapse* **by** *blast*
  {
    fix *Pr*
    assume $Pr \in$ *Logical-Probabilities*
    from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
      unfolding *Logical-Probabilities-def*
      by *auto*
    have $(\sum\varphi\leftarrow(\textit{replicate}\ n\ \top)\ @\ \Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
      using † ‹$Pr \in$ *Logical-Probabilities*› **by** *blast*
    hence $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
      using ‹$\textit{real}\ n = \lceil c\rceil$›
        probability-replicate-verum [**where** $\Phi{=}\Phi$ **and** $n{=}n$]
      by *linarith*
  }
  then show *?thesis* **by** *blast*
next
  case *False*
  hence $\lceil c\rceil \leq 0$ **by** *auto*
  from *this* **obtain** $n :: \textit{nat}$ **where** $\textit{real}\ n = -\ \lceil c\rceil$ **by** (*metis neg-0-le-iff-le of-nat-nat*)
  {
    fix *Pr*
    assume $Pr \in$ *Dirac-Measures*
    from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
      unfolding *Dirac-Measures-def*
      by *auto*
    have $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lceil c\rceil \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
      using *assm* ‹$Pr \in$ *Dirac-Measures*› **by** *blast*
    hence $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(\textit{replicate}\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
      using ‹$\textit{real}\ n = -\ \lceil c\rceil$›
        probability-replicate-verum [**where** $\Phi{=}\Gamma$ **and** $n{=}n$]
      by *linarith*
  }
  hence $\forall\ Pr \in$ *Dirac-Measures*. $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(\textit{replicate}\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
    by *blast*
  hence ‡: $\forall\ Pr \in$ *Logical-Probabilities*.
      $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(\textit{replicate}\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
    using *weakly-additive-completeness-collapse* **by** *blast*
  {
    fix *Pr*
    assume $Pr \in$ *Logical-Probabilities*
    from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
      unfolding *Logical-Probabilities-def*
      by *auto*

have $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
  using ‡ ‹$Pr \in$ *Logical-Probabilities*› **by** *blast*
hence $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
  using ‹$real\ n = -\lceil c\rceil$›
    probability-replicate-verum [**where** $\Phi=\Gamma$ **and** $n=n$]
  by *linarith*
}
then show *?thesis* **by** *blast*
qed
qed

lemma (**in** *Classical-Propositional-Logic*) *dirac-strict-floor*:
$\forall\ Pr \in$ *Dirac-Measures*.
$((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)) = ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
fix $Pr :: 'a \Rightarrow real$
let $?Pr' = (\lambda\ \varphi.\ \lfloor\ Pr\ \varphi\ \rfloor) :: 'a \Rightarrow int$
assume $Pr \in$ *Dirac-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 ‹$\forall\ \varphi.\ Pr\ \varphi = ?Pr'\ \varphi$› **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-Propositional-Logic*) *strict-dirac-collapse*:
$(\forall\ Pr \in$ *Logical-Probabilities*. $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in$ *Dirac-Measures*. $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
assume $\forall\ Pr \in$ *Logical-Probabilities*. $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
hence $\forall\ Pr \in$ *Dirac-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-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-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 $\star$: $\forall\ Pr \in Logical\text{-}Probabilities.\ ((\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
using *dirac-collapse* $[of\ \Phi\ \lfloor c \rfloor + 1\ \Gamma]$
by *auto*
show $\forall\ Pr \in Logical\text{-}Probabilities.\ ((\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + c < (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma))$
proof
fix $Pr :: 'a \Rightarrow real$
assume $Pr \in Logical\text{-}Probabilities$
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-Propositional-Logic*) *unproving-core-verum-extract*:
assumes $\neg \vdash \varphi$
shows $(|\ replicate\ n\ \top\ @\ \Phi\ |_{\varphi}) = n + (|\ \Phi\ |_{\varphi})$
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 $\top \in set\ \Sigma$
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-1
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 ‹$\Sigma \in \mathcal{C}$ ($\top$ # Φ) φ› *unproving-core-def* **by** *blast*
hence *mset* (*remove1* $\top$ Σ) $\subseteq$# *mset* Φ
using *subset-eq-diff-conv* **by** *fastforce*
ultimately have (| Φ |$_\varphi$) $\geq$ *length* (*remove1* $\top$ Σ)
by (*metis* (*no-types*, *lifting*)
core-size-intro
list-deduction-weaken
unproving-core-def
unproving-core-existence
mem-Collect-eq)
hence (| Φ |$_\varphi$) + *1* $\geq$ *length* Σ
by (*simp add*: ‹$\top \in$ *set* Σ› *length-remove1*)
moreover have (| Φ |$_\varphi$) < *length* Σ
proof (*rule ccontr*)
assume $\neg$ (| Φ |$_\varphi$) < *length* Σ
hence (| Φ |$_\varphi$) $\geq$ *length* Σ **by** *linarith*
from *this* **obtain** Δ **where** $\Delta \in \mathcal{C}$ Φ φ *length* $\Delta \geq$ *length* Σ
using *assms core-size-intro unproving-core-existence* **by** *fastforce*
hence $\neg$ ($\top$ # Δ) :$\vdash$ φ
using *list-deduction-modus-ponens*
list-deduction-theorem
list-deduction-weaken
unproving-core-def
verum-tautology
by *blast*
moreover have *mset* ($\top$ # Δ) $\subseteq$# *mset* ($\top$ # Φ)
using ‹$\Delta \in \mathcal{C}$ Φ φ› *unproving-core-def* **by** *auto*
ultimately have *length* $\Sigma \geq$ *length* ($\top$ # Δ)
using ‹$\Sigma \in \mathcal{C}$ ($\top$ # Φ) φ› *unproving-core-def* **by** *blast*
hence *length* $\Delta \geq$ *length* ($\top$ # Δ)
using ‹*length* $\Sigma \leq$ *length* Δ› *dual-order.trans* **by** *blast*
thus *False* **by** *simp*
qed
ultimately have (| $\top$ # Φ |$_\varphi$) = (*1* + | Φ |$_\varphi$)
by (*metis Suc-eq-plus1 Suc-le-eq* ‹$\Sigma \in \mathcal{C}$ ($\top$ # Φ) φ› *add.commute le-antisym core-size-intro*)
}
thus *?case* **using** *Suc* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *unproving-core-neg-verum-elim*:
(| *replicate n* ($\sim$ $\top$) @ Φ |$_\varphi$) = (| Φ |$_\varphi$)
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{

fix Φ
have $(\| (\sim \top) \# \Phi \|_\varphi) = (\| \Phi \|_\varphi)$
proof $(\textit{cases} \vdash \varphi)$
  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
          The-Principle-of-Pseudo-Scotus
          list-deduction-theorem
          list-deduction-weaken
          negation-def
          verum-def)
  hence $\sim \top \notin \textit{set}\ \Sigma$
    by (*meson* $\langle \Sigma \in \mathcal{C}\ (\sim \top \# \Phi)\ \varphi \rangle$
          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* $((\sim \top) \# \Phi)$
    using $\langle \Sigma \in \mathcal{C}\ (\sim \top \# \Phi)\ \varphi \rangle$ *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 $\neg\ (\Sigma :\vdash \varphi)$
    using $\langle \Sigma \in \mathcal{C}\ (\sim \top \# \Phi)\ \varphi \rangle$ *unproving-core-def* **by** *blast*
  ultimately have $(\| \Phi \|_\varphi) \geq$ *length* Σ
    by (*metis* (*no-types*, *lifting*)
          core-size-intro
          list-deduction-weaken
          unproving-core-def
          unproving-core-existence
          mem-Collect-eq)
  hence $(\| \Phi \|_\varphi) \geq (\| (\sim \top) \# \Phi \|_\varphi)$
    using $\langle \Sigma \in \mathcal{C}\ (\sim \top \# \Phi)\ \varphi \rangle$ *core-size-intro* **by** *auto*
  moreover
  have $(\| \Phi \|_\varphi) \leq (\| (\sim \top) \# \Phi \|_\varphi)$
  proof $-$
    obtain Δ **where** $\Delta \in \mathcal{C}\ \Phi\ \varphi$
      using *False unproving-core-existence* **by** *blast*
    hence

$\neg\ \Delta :\vdash \varphi$
mset $\Delta \subseteq\#$ *mset* $((\sim \top)\ \#\ \Phi)$
unfolding *unproving-core-def*
by (*simp*,
metis (*mono-tags*, *lifting*)
Diff-eq-empty-iff-mset
listSubtract.simps(*2*)
listSubtract-mset-homomorphism
unproving-core-def
mem-Collect-eq
mset-zero-iff
remove1.simps(*1*))
hence *length* $\Delta \leq$ *length* Σ
using ‹$\Sigma \in \mathcal{C}\ (\sim \top\ \#\ \Phi)\ \varphi$› *unproving-core-def* **by** *blast*
thus *?thesis*
using ‹$\Delta \in \mathcal{C}\ \Phi\ \varphi$› ‹$\Sigma \in \mathcal{C}\ (\sim \top\ \#\ \Phi)\ \varphi$› *core-size-intro* **by** *auto*
qed
ultimately show *?thesis*
using *le-antisym* **by** *blast*
qed
}
thus *?case* **using** *Suc* **by** *simp*
qed

lemma (**in** *Consistent-Classical-Logic*) *binary-inequality-elim*:
assumes $\forall\ Pr \in$ *Dirac-Measures*. $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + (c :: real) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
shows $((|\ \boldsymbol{\sim}\ \Gamma\ @\ \Phi\ |_\perp) + (c :: real) \leq$ *length* $\Gamma)$
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** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \perp\ Pr$
unfolding *Dirac-Measures-def*
by *auto*
have $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + n \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
by (*metis assms* ‹$Pr \in$ *Dirac-Measures*› ‹*real* $n = \lceil c \rceil$› *dirac-ceiling*)
hence $(\sum\varphi\leftarrow(replicate\ n\ \top)\ @\ \Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *probability-replicate-verum* [**where** Φ=Φ **and** n=n]
by *metis*
}
hence $(|\ \boldsymbol{\sim}\ \Gamma\ @$ *replicate* $n\ \top\ @\ \Phi\ |_\perp) \leq$ *length* Γ
using *binary-core-partial-completeness* **by** *blast*
moreover have *mset* $(\boldsymbol{\sim}\ \Gamma\ @$ *replicate* $n\ \top\ @\ \Phi) =$ *mset* (*replicate* $n\ \top\ @\ \boldsymbol{\sim}\ \Gamma\ @\ \Phi)$
by *simp*

```
  ultimately have (| replicate n ⊤ @ ∼ Γ @ Φ |⊥) ≤ length Γ
    unfolding core-size-def unproving-core-def
    by metis
  hence (| ∼ Γ @ Φ |⊥) + ⌈c⌉ ≤ length Γ
    using ‹real n = ⌈c⌉› consistency unproving-core-verum-extract
    by auto
  then show ?thesis by linarith
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 Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Dirac-Measures-def
      by auto
    have (∑φ←Φ. Pr φ) + ⌈c⌉ ≤ (∑γ←Γ. Pr γ)
      using assms ‹Pr ∈ Dirac-Measures› dirac-ceiling
      by blast
    hence (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ) + n
      using ‹real n = − ⌈c⌉› by linarith
    hence (∑φ←Φ. Pr φ) ≤ (∑γ←(replicate n ⊤) @ Γ. Pr γ)
      using probability-replicate-verum [where Φ=Γ and n=n]
      by metis
  }
  hence (| ∼ (replicate n ⊤ @ Γ) @ Φ |⊥) ≤ length (replicate n ⊤ @ Γ)
     using binary-core-partial-completeness [where Φ=Φ and Γ=replicate n ⊤ @
Γ]
    by metis
  hence (| ∼ Γ @ Φ |⊥) ≤ n + length Γ
    by (simp add: unproving-core-neg-verum-elim)
  then show ?thesis using ‹real n = − ⌈c⌉› by linarith
qed

lemma (in Classical-Propositional-Logic) binary-inequality-intro:
  assumes (| ∼ Γ @ Φ |⊥) + (c :: real) ≤ length Γ
  shows ∀ Pr ∈ Dirac-Measures. (∑φ←Φ. Pr φ) + (c :: real) ≤ (∑γ←Γ. Pr γ)
proof (cases ⊢ ⊥)
  assume ⊢ ⊥
  {
    fix Pr
    assume Pr ∈ Dirac-Measures
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Dirac-Measures-def
      by auto
    have False
      using ‹⊢ ⊥› consistency by blast
```

}
then show *?thesis* by *blast*
next
assume $\neg \vdash \bot$
then show *?thesis*
proof (*cases* $c \geq 0$)
assume $c \geq 0$
from *this* obtain n :: *nat* where *real* $n = \lceil c \rceil$
by (*metis ceiling-mono ceiling-zero of-nat-nat*)
hence $n + (| \sim \Gamma \ @ \ \Phi \ |_{\bot}) \leq$ *length* Γ
using *assms* by *linarith*
hence (| *replicate* n $\top$ @ $\sim \Gamma$ @ Φ $|_{\bot}) \leq$ *length* Γ
by (*simp add*: ‹$\neg \vdash \bot$› *unproving-core-verum-extract*)
moreover have *mset* (*replicate* n $\top$ @ $\sim \Gamma$ @ Φ) = *mset* ($\sim \Gamma$ @ *replicate* n $\top$ @ Φ)
by *simp*
ultimately have (| $\sim \Gamma$ @ *replicate* n $\top$ @ Φ $|_{\bot}) \leq$ *length* Γ
unfolding *core-size-def unproving-core-def*
by *metis*
hence $\forall$ *Pr* $\in$ *Dirac-Measures*. ($\sum \varphi \leftarrow$(*replicate* n $\top$) @ Φ. *Pr* φ) $\leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)
using *binary-core-partial-completeness* by *blast*
{
fix *Pr*
assume *Pr* $\in$ *Dirac-Measures*
from *this* interpret *Logical-Probability* (λ φ. $\vdash \varphi$) ($\rightarrow$) $\bot$ *Pr*
unfolding *Dirac-Measures-def*
by *auto*
have ($\sum \varphi \leftarrow$(*replicate* n $\top$) @ Φ. *Pr* φ) $\leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)
using ‹*Pr* $\in$ *Dirac-Measures*›
‹$\forall$ *Pr* $\in$ *Dirac-Measures*. ($\sum \varphi \leftarrow$(*replicate* n $\top$) @ Φ. *Pr* φ) $\leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)›
by *blast*
hence ($\sum \varphi \leftarrow$ Φ. *Pr* φ) + $n \leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)
by (*simp add*: *probability-replicate-verum*)
hence ($\sum \varphi \leftarrow$ Φ. *Pr* φ) + $c \leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)
using ‹*real* n = *real-of-int* $\lceil c \rceil$› 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 :: *nat* where *real* $n = - \lceil c \rceil$
by (*metis neg-0-le-iff-le of-nat-nat*)
hence (| $\sim \Gamma$ @ Φ $|_{\bot}) \leq n +$ *length* Γ
using *assms* by *linarith*
hence (| $\sim$ (*replicate* n $\top$ @ Γ) @ Φ $|_{\bot}) \leq$ *length* (*replicate* n $\top$ @ Γ)
by (*simp add*: *unproving-core-neg-verum-elim*)
hence $\forall$ *Pr* $\in$ *Dirac-Measures*.

$(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
using *binary-core-partial-completeness* **by** *blast*
{
fix *Pr*
assume $Pr \in$ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *Dirac-Measures-def*
by *auto*
have $(\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow(replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
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-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*

end
theory *Dutch-Book*
imports *../../Logic/Classical/Classical-Propositional-Connectives*
Logical-Probability-Completeness
~~/src/HOL/Real
begin

record *'p bet-offer* =
bet :: *'p*
amount :: *real*

record *'p book* =
buys :: (*'p bet-offer*) *list*
sells :: (*'p bet-offer*) *list*

definition *payoff* :: $('p \Rightarrow bool) \Rightarrow 'p\ book \Rightarrow real\ (\pi)$ **where**
$[simp]$: $\pi\ s\ b \equiv (\sum\ i \leftarrow sells\ b.\ (if\ s\ (bet\ i)\ then\ 1\ else\ 0) - amount\ i)$
$+ (\sum\ i \leftarrow buys\ b.\ amount\ i - (if\ s\ (bet\ i)\ then\ 1\ else\ 0))$

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*:
  π *s book* = $(\sum i \leftarrow \textit{sells book}.\ \textit{settle-bet}\ s\ (\textit{bet}\ i) - \textit{amount}\ i)$
  $+ (\sum i \leftarrow \textit{buys book}.\ \textit{amount}\ i - \textit{settle-bet}\ s\ (\textit{bet}\ i))$
  unfolding *settle-bet-def*
  by *simp*

definition *settle* :: $('p \Rightarrow \textit{bool}) \Rightarrow 'p\ \textit{bet-offer list} \Rightarrow \textit{real}$ **where**
  settle s bets $\equiv \sum b \leftarrow \textit{bets}.\ \textit{settle-bet}\ s\ (\textit{bet}\ b)$

definition *total-amount* :: $('p\ \textit{bet-offer})\ \textit{list} \Rightarrow \textit{real}$ **where**
  total-amount offers $\equiv \sum i \leftarrow \textit{offers}.\ \textit{amount}\ i$

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-Propositional-Logic*) *possibility* :: $('a \Rightarrow \textit{bool}) \Rightarrow \textit{bool}$ **where**
  [*simp*]: *possibility p* $\equiv \neg (p\ \bot)$
  $\wedge (\forall \varphi.\ \vdash \varphi \longrightarrow p\ \varphi)$
  $\wedge (\forall \varphi\ \psi\ .\ p\ (\varphi \rightarrow \psi) \longrightarrow p\ \varphi \longrightarrow p\ \psi)$
  $\wedge (\forall \varphi\ .\ p\ \varphi \vee p\ (\varphi \rightarrow \bot))$

definition (**in** *Classical-Propositional-Logic*) *possibilities* :: $('a \Rightarrow \textit{bool})\ \textit{set}$ **where**
  [*simp*]: *possibilities* = {*p*. *possibility p*}

lemma (**in** *Classical-Propositional-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-Propositional-Logic*) *possibilities-logical-closure*:
  assumes *possibility p*
    and $\{x.\ p\ x\} \Vdash \varphi$
  shows $p\ \varphi$
proof −
  {
    fix Γ
    assume *set* $\Gamma \subseteq$ *Collect p*
    hence $\forall\ \varphi.\ \Gamma :\vdash \varphi \longrightarrow p\ \varphi$
    proof (*induct* Γ)
      case *Nil*
      have $\forall \varphi.\ \vdash \varphi \longrightarrow p\ \varphi$
        using ‹*possibility p*› **by** *auto*
      then show *?case*
        using *list-deduction-base-theory* **by** *blast*
    next
      case (*Cons* γ Γ)
      hence $p\ \gamma$
        by *simp*
      have $\forall\ \varphi.\ \Gamma :\vdash \gamma \to \varphi \longrightarrow p\ (\gamma \to \varphi)$
        using *Cons.hyps Cons.prems* **by** *auto*
      then show *?case*
        by (*meson* ‹$p\ \gamma$› ‹*possibility p*› *list-deduction-theorem possibility-def*)
    qed
  }
  thus *?thesis*
    using ‹*Collect p* $\Vdash \varphi$› *set-deduction-def* **by** *auto*
qed

lemma (**in** *Classical-Propositional-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
      Maximally-Consistent-Set-def
      possibilities-logical-closure
      possibility-def
      mem-Collect-eq)

lemma (**in** *Classical-Propositional-Logic*) *MCSs-are-possibilities*:
  assumes *MCS s*
  shows *possibility* $(\lambda\ x.\ x \in s)$
proof −
  have $\bot \notin s$
    using ‹*MCS s*›
      Formula-Consistent-def

Formula-Maximally-Consistent-Set-def
Maximally-Consistent-Set-def
set-deduction-reflection
by *blast*
moreover have $\forall\ \varphi.\ \vdash \varphi \longrightarrow \varphi \in s$
using ‹*MCS s*›
Formula-Maximally-Consistent-Set-reflection
Maximally-Consistent-Set-def
set-deduction-weaken
by *blast*
moreover have $\forall\ \varphi\ \psi.\ (\varphi \rightarrow \psi) \in s \longrightarrow \varphi \in s \longrightarrow \psi \in s$
using ‹*MCS s*›
Formula-Maximal-Consistency
Formula-Maximally-Consistent-Set-implication
by *blast*
moreover have $\forall\ \varphi.\ \varphi \in s \vee (\varphi \rightarrow \bot) \in s$
using *assms*
Formula-Maximally-Consistent-Set-implication
Maximally-Consistent-Set-def
by *blast*
ultimately show *?thesis* **by** *simp*
qed

definition (**in** *Classical-Propositional-Logic*) *negate-bets* ($-^{\sim}$) **where**
$bets^{\sim} = [b\ (\!|\ bet := \sim (bet\ b)\ |\!).\ b \leftarrow bets]$

lemma (**in** *Classical-Propositional-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-offer*) *list*
fix *s* :: ′*a* ⇒ *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** † = *this*
{

```
    fix n :: nat
    have    (∃ p ∈ possibilities. settle p (buys'~ @ sells') ≤ n)
          ⟶ (∃ q ∈ possibilities. settle q (buys'~ @ sells') < 0) (is - ⟶
?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 q_1 where
          q_1 ∈ possibilities
          settle q_1 (buys'~ @ sells') < Suc n
          by (metis ◇ antisym-conv not-less)
        from this obtain q_2 where
          q_2 ∈ possibilities
          settle q_2 (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: $(\exists\, p \in possibilities.\ \pi\ p\ bets = x) \wedge (\forall\, q \in possibilities.\ x \leq \pi\ q\ bets)$
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-Propositional-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*+)

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
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* $\Psi \leq$ *length* $?\Phi$ **by** *linarith*
}
ultimately have $?\Phi \in \mathcal{C}$ *?props* $\bot$
unfolding *unproving-core-def*
by *blast*
hence *MaxSAT ?props* = *length* $?\Phi$
using *core-size-intro* **by** *presburger*
hence *MaxSAT ?props* = *settle p* (*sells*$'^{\sim}$ @ *buys*$'$)
unfolding *settle-alt-def*
by *simp*
thus *MaxSAT ?props* + $k \leq$ *total-amount buys*$'$ + *length sells*$'$ − *total-amount sells*$'$
using *possibility-payoff-dual* [*of p buys*$'$ *sells*$'$]
‹$k \leq \pi_{min}$ *?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* + $k \leq$ *?c*
from *this* **obtain** Φ **where** $\Phi \in \mathcal{C}$ *?props* $\bot$ **and** *length* $\Phi + k \leq$ *?c*
using *consistency core-size-intro unproving-core-existence* **by** *fastforce*
hence $\neg\, \Phi$:⊢ $\bot$
using *unproving-core-def* **by** *blast*
from *this* **obtain** Ω_Φ **where** *MCS* Ω_Φ **and** *set* $\Phi \subseteq \Omega_\Phi$
by (*meson Formula-Consistent-def*
Formula-Maximal-Consistency
Formula-Maximally-Consistent-Extension
list-deduction-monotonic
set-deduction-def)
let *?p* = $\lambda\varphi\ .\ \varphi \in \Omega_\Phi$
have *possibility ?p*
using ‹*MCS* Ω_Φ› *MCSs-are-possibilities* **by** *blast*
have *mset* $\Phi \subseteq\#$ *mset ?props*
using ‹$\Phi \in \mathcal{C}$ *?props* $\bot$› *unproving-core-def* **by** *blast*
have *mset* $\Phi \subseteq\#$ *mset* [$b \leftarrow$ *?props. ?p b*]
by (*metis* ‹*mset* $\Phi \subseteq\#$ *mset ?props*›
‹*set* $\Phi \subseteq \Omega_\Phi$›
filter-True
mset-filter
multiset-filter-mono
subset-code(*1*))
have *mset* Φ = *mset* [$b \leftarrow$ *?props. ?p b*]
proof (*rule ccontr*)
assume *mset* $\Phi \neq$ *mset* [$b \leftarrow$ *?props. ?p b*]
hence *length* $\Phi <$ *length* [$b \leftarrow$ *?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
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* Φ + $k \leq$ *?c*⟩ **by** *linarith*
hence $k \leq \pi$ *?p ?bets*
using ⟨*possibility ?p*⟩
possibility-payoff-dual [*of ?p buys*$'$ *sells*$'$]
⟨*length* Φ + $k \leq$ *?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*)
⟨$\Phi \in \mathcal{C}$ *?props* $\bot$⟩
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*$'$
$\leq$ $-$ *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* $\leq$ π *q* *?bets*
using *possibility-payoff-dual* [*of ?p buys*$'$ *sells*$'$]
possibility-payoff-dual [*of q buys*$'$ *sells*$'$]
‹*possibility ?p*›
‹*q* $\in$ *possibilities*›
unfolding *possibilities-def*
by (*metis mem-Collect-eq*)
qed
have π_{min} *?bets* = π *?p* *?bets*
unfolding *minimum-payoff-def*
proof
show ($\exists$ *p*$\in$*possibilities*. π *p* *?bets* = π *?p* *?bets*) $\wedge$ ($\forall$ *q*$\in$*possibilities*. π *?p* *?bets* $\leq$ π *q* *?bets*)
using ‹$\forall$ *q* $\in$ *possibilities*. π *?p* *?bets* $\leq$ π *q* *?bets*›
‹*possibility ?p*›
unfolding *possibilities-def*
by *blast*
next
fix *n*
assume $\star$: ($\exists$ *p*$\in$*possibilities*. π *p* *?bets* = *n*) $\wedge$ ($\forall$ *q*$\in$*possibilities*. *n* $\leq$ π *q* *?bets*)
from *this* **obtain** *p* **where** π *p* *?bets* = *n* **and** *possibility p*
using *possibilities-def* **by** *blast*
hence π *p* *?bets* $\leq$ π *?p* *?bets*
using $\star$ ‹*possibility ?p*›
unfolding *possibilities-def*
by *blast*
moreover have π *?p* *?bets* $\leq$ π *p* *?bets*
using ‹$\forall$ *q* $\in$ *possibilities*. π *?p* *?bets* $\leq$ 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* $\leq$ π_{min} *?bets*
using ‹*k* $\leq$ π *?p* *?bets*›
by *auto*
qed

lemma (**in** *Consistent-Classical-Logic*) *nonstrict-dutch-book*:
$(k \le \pi_{min}\ (\!|\ buys = buys',\ sells = sells'\ |\!))$
$= (\forall\ Pr \in Logical\text{-}Probabilities.$
$(\sum b{\leftarrow}buys'.\ Pr\ (bet\ b)) + total\text{-}amount\ sells' + k$
$\le (\sum s{\leftarrow}sells'.\ Pr\ (bet\ s)) + total\text{-}amount\ buys')$
(**is** *?lhs* = -)
proof −
let *?tot-ss* = *total-amount sells'* **and** *?tot-bs* = *total-amount buys'*
have $[bet\ b\ .\ b \leftarrow sells'^{\sim}\ @\ buys'] = \sim [bet\ s.\ s \leftarrow sells']\ @\ [bet\ b.\ b \leftarrow buys']$
(**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*.
$(\sum \varphi{\leftarrow}?buy\text{-}\varphi s.\ Pr\ \varphi) + (?tot\text{-}ss - ?tot\text{-}bs + k) \le (\sum \gamma{\leftarrow}?sell\text{-}\varphi s.\ Pr\ \gamma))$
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 $((\sum \varphi{\leftarrow}?buy\text{-}\varphi s.\ Pr\ \varphi) + (?tot\text{-}ss - ?tot\text{-}bs + k) \le (\sum \gamma{\leftarrow}?sell\text{-}\varphi s.\ Pr\ \gamma))$
$= ((\sum b{\leftarrow}buys'.\ Pr\ (bet\ b)) + ?tot\text{-}ss + k \le (\sum s{\leftarrow}sells'.\ Pr\ (bet\ s)) + ?tot\text{-}bs)$
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*:
$(k < \pi_{min}\ (\!|\ buys = buys',\ sells = sells'\ |\!))$
$= (\forall\ Pr \in Logical\text{-}Probabilities.$
$(\sum b{\leftarrow}buys'.\ Pr\ (bet\ b)) + total\text{-}amount\ sells' + k$
$< (\sum s{\leftarrow}sells'.\ Pr\ (bet\ s)) + total\text{-}amount\ buys')$
(**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 ∈ Logical-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 ∈ Logical-Probabilities. (∑ b←buys'. Pr (bet b)) + ?c < (∑ s←sells'.
Pr (bet s))
    using ‹?rhs› by fastforce
  hence ∀ Pr ∈ Logical-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*:
  $(0 < \pi_{min}\ (\!|\ buys = buys',\ sells = sells'\ |\!))$
  $= (\forall\ Pr \in Logical\text{-}Probabilities.$
    $(\sum b \leftarrow buys'.\ Pr\ (bet\ b)) + total\text{-}amount\ sells'$
    $< (\sum s \leftarrow sells'.\ Pr\ (bet\ s)) + total\text{-}amount\ buys')$
 by (*simp add*: *strict-dutch-book*)

end

References

[1] D. A. Turner. Another algorithm for bracket abstraction. *Journal of Symbolic Logic*, 44(2):267–270, June 1979.