isabelle

mpwd

January 14, 2020

Contents

1 Minimal Logic

theory *Minimal-Logic*
  imports *Main*
begin

This theory presents *minimal logic*, 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*: $\vdash \varphi \rightarrow \psi \implies \vdash \varphi \implies \vdash \psi$

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*: $\Gamma :\vdash \varphi \rightarrow \psi \Longrightarrow \Gamma :\vdash \varphi \Longrightarrow \Gamma :\vdash \psi$

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*:
$\vdash \Gamma :\to \psi \to (removeAll\ \varphi\ \Gamma) :\to (\varphi \to \psi)$
proof −

```
  have ∀ ψ. ⊢ Γ :→ ψ → (removeAll φ Γ) :→ (φ → ψ)
  proof(induct Γ)
    case Nil
    then show ?case by (simp, meson Axiom-1)
  next
    case (Cons χ Γ)
    assume inductive-hypothesis: ∀ ψ. ⊢ Γ :→ ψ → removeAll φ Γ :→ (φ → ψ)
    moreover {
      assume φ ≠ χ
      with inductive-hypothesis
      have ∀ ψ. ⊢ (χ # Γ) :→ ψ → removeAll φ (χ # Γ) :→ (φ → ψ)
        by (simp, meson Modus-Ponens hypothetical-syllogism)
    }
    moreover {
      fix ψ
      assume φ-equals-χ: φ = χ
      moreover with inductive-hypothesis
      have ⊢ Γ :→ (χ → ψ) → removeAll φ (χ # Γ) :→ (φ → χ → ψ) by simp
      hence ⊢ Γ :→ (χ → ψ) → removeAll φ (χ # Γ) :→ (φ → ψ)
      by (metis calculation Modus-Ponens implication-absorption list-flip-implication1
              list-flip-implication2 list-implication.simps(2))
      ultimately have ⊢ (χ # Γ) :→ ψ → removeAll φ (χ # Γ) :→ (φ → ψ)
        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:
  set Σ ⊆ set Γ ⟹ ⊢ Σ :→ φ → Γ :→ φ
proof −
  assume set Σ ⊆ set Γ
  moreover have ∀ Σ φ. set Σ ⊆ set Γ ⟶ ⊢ Σ :→ φ → Γ :→ φ
  proof(induct Γ)
    case Nil
    then show ?case
     by (metis list-implication.simps(1) list-implication-Axiom-1 set-empty subset-empty)
  next
    case (Cons ψ Γ)
    assume inductive-hypothesis: ∀Σ φ. set Σ ⊆ set Γ ⟶ ⊢ Σ :→ φ → Γ :→ φ
    {
      fix Σ
      fix φ
      assume Σ-subset-relation: set Σ ⊆ set (ψ # Γ)
      have ⊢ Σ :→ φ → (ψ # Γ) :→ φ
```

proof −
{
assume *set* $\Sigma \subseteq$ *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* $\Sigma \subseteq$ *set* Γ)
hence *set ?*$\Delta \subseteq$ *set* Γ **using** Σ*-subset-relation* **by** *auto*
hence $\vdash$ *?*$\Delta :\rightarrow (\psi \rightarrow \varphi) \rightarrow \Gamma :\rightarrow (\psi \rightarrow \varphi)$ **using** *inductive-hypothesis*
by *auto*
hence $\vdash$ *?*$\Delta :\rightarrow (\psi \rightarrow \varphi) \rightarrow (\psi \# \Gamma) :\rightarrow \varphi$
by (*metis Modus-Ponens*
flip-implication
list-flip-implication2
list-implication.simps(*2*))
moreover have $\vdash \Sigma :\rightarrow \varphi \rightarrow$ *?*$\Delta :\rightarrow (\psi \rightarrow \varphi)$
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 $\Sigma \subseteq$ *set* $\Gamma \Longrightarrow \Sigma :\vdash \varphi \Longrightarrow \Gamma :\vdash \varphi$
unfolding *list-deduction-def*
using *Modus-Ponens list-implication-monotonic*
by *blast*

1.6 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*:
$(\varphi \# \Gamma) :\vdash \psi = ($*removeAll* $\varphi\ \Gamma) :\vdash \varphi \rightarrow \psi$
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 $[\![set\ ?\Sigma \subseteq set\ ?\Gamma;\ ?\Sigma :\vdash\ ?\varphi]\!] \Longrightarrow\ ?\Gamma :\vdash\ ?\varphi$ among the other rules provided.

lemma (**in** *Minimal-Logic*) *list-deduction-reflection*: $\varphi \in set\ \Gamma \Longrightarrow \Gamma :\vdash \varphi$
by (*metis list-deduction-def*
insert-subset
list.simps(15)
list-deduction-monotonic
list-implication.simps(2)
list-implication-Axiom-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 and 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 Cut" (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*:
$(\Phi\ @\ \Gamma) :\vdash \psi \Longrightarrow \forall\ \varphi \in set\ \Phi.\ \Delta :\vdash \varphi \Longrightarrow \Gamma\ @\ \Delta :\vdash \psi$

```
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
```

2 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 (:⊢).

2.1 Definition of Deduction

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

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

2.1.1 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*: $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$
proof −
assume $\Gamma \Vdash \varphi \to \psi$
then obtain Φ **where** *A*: *set* $\Phi \subseteq \Gamma$ **and** *B*: $\Phi :\vdash \varphi \to \psi$
using *set-deduction-def* **by** *blast*
assume $\Gamma \Vdash \varphi$
then obtain Ψ **where** *C*: *set* $\Psi \subseteq \Gamma$ **and** *D*: $\Psi :\vdash \varphi$
using *set-deduction-def* **by** *blast*
from *B D* **have** $\Phi\ @\ \Psi :\vdash \psi$
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* $\lambda\ \varphi.\ \Gamma \Vdash \varphi$ $(\to)$
proof
fix $\varphi\ \psi$
show $\Gamma \Vdash \varphi \to \psi \to \varphi$ **by** (*metis Axiom-1 set-deduction-weaken*)
next
fix $\varphi\ \psi\ \chi$
show $\Gamma \Vdash (\varphi \to \psi \to \chi) \to (\varphi \to \psi) \to \varphi \to \chi$ **by** (*metis Axiom-2 set-deduction-weaken*)
next
fix $\varphi\ \psi$
show $\Gamma \Vdash \varphi \to \psi \Longrightarrow \Gamma \Vdash \varphi \Longrightarrow \Gamma \Vdash \psi$ **using** *set-deduction-modus-ponens* **by** *metis*

qed
end

2.2 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

2.3 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*)

2.4 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*)

2.5 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)

2.6 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 $\varphi\ \Gamma \Vdash \psi \Longrightarrow \Delta \Vdash \varphi \Longrightarrow \Gamma \cup \Delta \Vdash \psi$
proof −
  assume *insert* $\varphi\ \Gamma \Vdash \psi$
  hence $\Gamma \Vdash \varphi \rightarrow \psi$ **using** *set-deduction-theorem* **by** *auto*
  hence $\Gamma \cup \Delta \Vdash \varphi \rightarrow \psi$ **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 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 $\forall\,\varphi \in insert\ \chi\ \Phi.\ \Delta \Vdash \varphi$
  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** $\forall\,\varphi \in (insert\ \chi\ \Phi).\ \Psi_1 @ \Psi_2 :\vdash \varphi$
    by (*metis insert-iff le-sup-iff list-deduction-monotonic order-refl set-append*)
  ultimately show *?case* **by** *blast*
qed

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

theorem (**in** *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* $(\Phi' @ \Gamma')$ = *set* Σ **using** A **by** *auto*
  hence E: $\Phi' @ \Gamma' :\vdash \psi$ **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* $(\Gamma' @ \Delta') \subseteq \Gamma \cup \Delta$ **and** $\Gamma' @ \Delta' :\vdash \psi$ **by** *auto*
  thus *?thesis* **using** *set-deduction-def* **by** *blast*
qed

end

3 Classical Propositional Logic

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

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

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

3.1 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*: $\neg \vdash \bot$

3.2 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 $[\varphi \to \bot, (\varphi \to \psi) \to \varphi] :\vdash \bot$
by (*meson list.set-intros(1) list-deduction-modus-ponens list-deduction-reflection*)
hence $[(\varphi \to \psi) \to \varphi] :\vdash (\varphi \to \bot) \to \bot$
using *list-deduction-theorem* **by** *blast*
hence $[(\varphi \to \psi) \to \varphi] :\vdash \varphi$
using *Double-Negation list-deduction-modus-ponens list-deduction-weaken* **by**
blast
thus *?thesis*
using *list-deduction-def*
by *auto*
qed

lemma (**in** *Classical-Propositional-Logic*) *excluded-middle-elimination*:
$\vdash (\varphi \to \psi) \to ((\varphi \to \bot) \to \psi) \to \psi$

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

3.3 Maximally Consistent Sets

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

lemma (**in** *Minimal-Logic*) *Formula-Consistent-Extension*:
  assumes $\varphi-$*Consistent* Γ
  shows ($\varphi-$*Consistent insert* ψ Γ) $\vee$ ($\varphi-$*Consistent insert* $(\psi \to \varphi)$ Γ)
proof –
  {
    assume $\sim$ $\varphi-$*Consistent insert* ψ Γ
    hence $\Gamma \Vdash \psi \to \varphi$
      using *set-deduction-theorem*
      unfolding *Formula-Consistent-def*
      by *simp*
    hence $\varphi-$*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 ⇒ 'a set ⇒ bool (-−MCS - [100] 100)
where
  [simp]: φ−MCS Γ ≡ (φ−Consistent Γ) ∧ (∀ ψ. ψ ∈ Γ ∨ (ψ → φ) ∈ Γ)

theorem (in Minimal-Logic) Formula-Maximally-Consistent-Extension:
  assumes φ−Consistent Γ
  shows ∃ Ω. (φ−MCS Ω) ∧ Γ ⊆ Ω
proof −
  let ?Γ-Extensions = {Σ. (φ−Consistent Σ) ∧ Γ ⊆ Σ}
  have ∃ Ω ∈ ?Γ-Extensions. ∀Σ ∈ ?Γ-Extensions. Ω ⊆ Σ ⟶ Σ = Ω
  proof (rule subset-Zorn)
    fix 𝒞 :: 'a set set
    assume subset-chain-𝒞: subset.chain ?Γ-Extensions 𝒞
    hence 𝒞: ∀ Σ ∈ 𝒞. Γ ⊆ Σ ∀ Σ ∈ 𝒞. φ−Consistent Σ
      unfolding subset.chain-def by blast+
    show ∃ Ω ∈ ?Γ-Extensions. ∀ Σ ∈ 𝒞. Σ ⊆ Ω
    proof cases
      assume 𝒞 = {} thus ?thesis using assms by blast
    next
      let ?Ω = ⋃ 𝒞
      assume 𝒞 ≠ {}
      hence Γ ⊆ ?Ω by (simp add: 𝒞(1) less-eq-Sup)
      moreover have φ−Consistent ?Ω
      proof −
        {
          assume ~ φ−Consistent ?Ω
          then obtain ω where ω: finite ω ω ⊆ ?Ω ~ φ−Consistent ω
            unfolding Formula-Consistent-def
                      set-deduction-def
            by auto
          from ω(1) ω(2) have ∃ Σ ∈ 𝒞. ω ⊆ Σ
          proof (induct ω rule: finite-induct)
            case empty thus ?case using ‹𝒞 ≠ {}› by blast
          next
            case (insert ψ ω)
            from this obtain Σ₁ Σ₂ where
              Σ₁: ω ⊆ Σ₁ Σ₁ ∈ 𝒞 and
              Σ₂: ψ ∈ Σ₂ Σ₂ ∈ 𝒞
              by auto
            hence Σ₁ ⊆ Σ₂ ∨ Σ₂ ⊆ Σ₁
              using subset-chain-𝒞
              unfolding subset.chain-def
              by blast
```

hence (*insert* ψ ω) $\subseteq \Sigma_1 \vee$ (*insert* ψ ω) $\subseteq \Sigma_2$ **using** Σ_1 Σ_2 **by** *blast*
thus *?case* **using** Σ_1 Σ_2 **by** *blast*
qed
hence $\exists\ \Sigma \in \mathcal{C}.$ ($\varphi-$*Consistent* Σ) $\wedge$ ~ ($\varphi-$*Consistent* Σ)
using $\mathcal{C}(2)$ $\omega(3)$
unfolding *Formula-Consistent-def*
set-deduction-def
by *auto*
hence *False* **by** *auto*
}
thus *?thesis* **by** *blast*
qed
ultimately show *?thesis* **by** *blast*
qed
qed
then obtain Ω **where** Ω: $\Omega \in$ *?*Γ*-Extensions*
$\forall \Sigma \in$ *?*Γ*-Extensions*. $\Omega \subseteq \Sigma \longrightarrow \Sigma = \Omega$ **by** *auto+*
{
fix ψ
have ($\varphi-$*Consistent insert* ψ Ω) $\vee$ ($\varphi-$*Consistent insert* ($\psi \to \varphi$) Ω)
$\Gamma \subseteq$ *insert* ψ Ω
$\Gamma \subseteq$ *insert* ($\psi \to \varphi$) Ω
using $\Omega(1)$ *Formula-Consistent-Extension Formula-Consistent-def* **by** *auto*
hence *insert* ψ $\Omega \in$ *?*Γ*-Extensions* $\vee$ *insert* ($\psi \to \varphi$) $\Omega \in$ *?*Γ*-Extensions* **by** *blast*
hence $\psi \in \Omega \vee (\psi \to \varphi) \in \Omega$ **using** $\Omega(2)$ **by** *blast*
}
thus *?thesis* **using** $\Omega(1)$ **unfolding** *Formula-Maximally-Consistent-Set-def* **by** *blast*
qed

lemma (**in** *Minimal-Logic*) *Formula-Maximally-Consistent-Set-reflection*:
$\varphi-$*MCS* $\Gamma \Longrightarrow \psi \in \Gamma = \Gamma \Vdash \psi$
proof −
assume $\varphi-$*MCS* Γ
{
assume $\Gamma \Vdash \psi$
moreover from ‹$\varphi-$*MCS* Γ› **have** $\psi \in \Gamma \vee (\psi \to \varphi) \in \Gamma$ ~ $\Gamma \Vdash \varphi$
unfolding *Formula-Maximally-Consistent-Set-def Formula-Consistent-def*
by *auto*
ultimately have $\psi \in \Gamma$
using *set-deduction-reflection set-deduction-modus-ponens*
by *metis*
}
thus $\psi \in \Gamma = \Gamma \Vdash \psi$
using *set-deduction-reflection*
by *metis*
qed

definition (**in** *Classical-Propositional-Logic*)
  Consistent :: *'a set* ⇒ *bool* **where**
    [*simp*]: *Consistent* $\Gamma \equiv \bot-$*Consistent* Γ

definition (**in** *Classical-Propositional-Logic*)
  Maximally-Consistent-Set :: *'a set* ⇒ *bool* (*MCS*) **where**
    [*simp*]: *MCS* $\Gamma \equiv \bot-$*MCS* Γ

lemma (**in** *Classical-Propositional-Logic*) *Formula-Maximal-Consistent-Set-negation*:
  $\varphi-$*MCS* $\Gamma \Longrightarrow \varphi \to \bot \in \Gamma$
proof −
  assume $\varphi-$*MCS* Γ
  {
    assume $\varphi \to \bot \notin \Gamma$
    hence $(\varphi \to \bot) \to \varphi \in \Gamma$
      using ‹$\varphi-$*MCS* Γ›
      unfolding *Formula-Maximally-Consistent-Set-def*
      by *blast*
    hence $\Gamma \Vdash (\varphi \to \bot) \to \varphi$
      using *set-deduction-reflection*
      by *simp*
    hence $\Gamma \Vdash \varphi$
      using *Peirces-law*
            set-deduction-modus-ponens
            set-deduction-weaken
        by *metis*
    hence *False*
      using ‹$\varphi-$*MCS* Γ›
      unfolding *Formula-Maximally-Consistent-Set-def*
            Formula-Consistent-def
      by *simp*
  }
  thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *Formula-Maximal-Consistency*:
  $(\exists \varphi.\ \varphi-$*MCS* $\Gamma) =$ *MCS* Γ
proof −
  {
    fix φ
    have $\varphi-$*MCS* $\Gamma \Longrightarrow$ *MCS* Γ
    proof −
      assume $\varphi-$*MCS* Γ
      have *Consistent* Γ
        using ‹$\varphi-$*MCS* Γ›
            Ex-Falso-Quodlibet [**where** $\varphi{=}\varphi$]
            set-deduction-weaken [**where** $\Gamma{=}\Gamma$]
            set-deduction-modus-ponens
        unfolding *Formula-Maximally-Consistent-Set-def*

```
              Consistent-def
              Formula-Consistent-def
        by metis
      moreover {
        fix ψ
        have ψ → ⊥ ∉ Γ ⟹ ψ ∈ Γ
        proof −
          assume ψ → ⊥ ∉ Γ
          hence (ψ → ⊥) → φ ∈ Γ
            using ‹φ−MCS Γ›
            unfolding Formula-Maximally-Consistent-Set-def
            by blast
          hence Γ ⊩ (ψ → ⊥) → φ
            using set-deduction-reflection
            by simp
          also have Γ ⊩ φ → ⊥
            using ‹φ−MCS Γ›
                  Formula-Maximal-Consistent-Set-negation
                  set-deduction-reflection
            by simp
          hence Γ ⊩ (ψ → ⊥) → ⊥
            using calculation
                  hypothetical-syllogism [where φ=ψ → ⊥ and ψ=φ and χ=⊥]
                  set-deduction-weaken [where Γ=Γ]
                  set-deduction-modus-ponens
            by metis
          hence Γ ⊩ ψ
            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
```

theorem (**in** *Minimal-Logic*) *Formula-Maximally-Consistent-Set-implication-elimination*:
  assumes φ−*MCS* Ω
  shows $(\psi \to \chi) \in \Omega \Longrightarrow \psi \in \Omega \longrightarrow \chi \in \Omega$
  using *assms*
      Formula-Maximally-Consistent-Set-reflection
      set-deduction-modus-ponens
  by *blast*

lemma (**in** *Classical-Propositional-Logic*) *Formula-Maximally-Consistent-Set-implication*:
  assumes φ−*MCS* Γ
  shows $\psi \to \chi \in \Gamma = (\psi \in \Gamma \longrightarrow \chi \in \Gamma)$
proof −
  {
    assume *hypothesis*: $\psi \in \Gamma \longrightarrow \chi \in \Gamma$
    {
      assume $\psi \notin \Gamma$
      have $\forall \psi.\ \varphi \to \psi \in \Gamma$
        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 $\forall \chi\ \psi.$ *insert* $\chi\ \Gamma \Vdash \psi \vee \chi \to \varphi \notin \Gamma$
        by (*meson assms*
            Axiom-1
            Formula-Maximally-Consistent-Set-reflection
            set-deduction-modus-ponens
            set-deduction-theorem
            set-deduction-weaken)
      hence $\psi \to \chi \in \Gamma$
        by (*meson* $\langle\psi \notin \Gamma\rangle$
            assms
            Formula-Maximally-Consistent-Set-def
            Formula-Maximally-Consistent-Set-reflection
            set-deduction-theorem)
    }
    moreover {
      assume $\chi \in \Gamma$
      hence $\psi \to \chi \in \Gamma$
        by (*metis assms*
            calculation
            insert-absorb
            Formula-Maximally-Consistent-Set-reflection
            set-deduction-theorem)
    }
    ultimately have $\psi \to \chi \in \Gamma$ **using** *hypothesis* **by** *blast*
  }
  thus *?thesis*

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

end

4 Classical Propositional Calculus Soundness And Completeness

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

4.1 Syntax

datatype *'a Classical-Propositional-Formula* =
Falsum ($\bot$)
| *Proposition 'a* ($\langle$ - $\rangle$ [*45*])
| *Implication 'a Classical-Propositional-Formula*
'a Classical-Propositional-Formula (**infixr** $\rightarrow$ *70*)

4.2 Propositional Calculus

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

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

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

4.3 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+*)

4.4 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** $\models\!\models_{prop}$ *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* ::
$'a\ Classical\text{-}Propositional\text{-}Formula\ set \Rightarrow 'a\ set$ ($\{\!\!\{\ \text{-}\ \}\!\!\}$ [*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** $\Gamma'(1)$ **by** *auto*
hence $\forall\ \varphi.\ \Gamma' :\vdash \varphi \longrightarrow \mathfrak{M} \models_{prop} \varphi$
proof (*induct* Γ')
case *Nil*
then show *?case*
by (*simp add*: *Classical-Propositional-Calculus-Soundness*
list-deduction-def)
next
case (*Cons* ψ Γ')
thus *?case* **using** *list-deduction-theorem* **by** *fastforce*
qed
with $\Gamma'(2)$ **have** $\mathfrak{M} \models_{prop} \varphi$ **by** *blast*
}
thus $\Gamma \models_{prop} \varphi$
using *Strong-Classical-Propositional-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]
```

4.5 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) $\subseteq$# *mset* Γ
by (*simp*,
metis $\heartsuit$
‹*mset* Ψ $\subseteq$# *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 $\heartsuit$ **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 $\neg$ ($\exists$ ψ $\in$ *set* Ψ. ψ $\in$ *set* Σ)
and *mset* Ψ $\subseteq$# *mset* (Σ @ Γ)
shows *mset* Ψ $\subseteq$# *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 $\in$ *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

4.6 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 ∃ Φ'. mset Φ' ⊆# mset Γ ∧ Φ = (map f Φ')
proof −
```

have ∀ Φ. *mset* Φ ⊆# *mset* (*map f* Γ) ⟶ (∃ Φ′. *mset* Φ′ ⊆# *mset* Γ ∧ Φ = (*map f* Φ′))
proof (*induct* Γ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* γ Γ)
{
fix Φ
assume *mset* Φ ⊆# *mset* (*map f* (γ # Γ))
have ∃Φ′. *mset* Φ′ ⊆# *mset* (γ # Γ) ∧ Φ = *map f* Φ′
proof *cases*
assume *f* γ ∈ *set* Φ
hence *f* γ # (*remove1* (*f* γ) Φ) <~~> Φ
by (*simp add*: *perm-remove perm-sym*)
with ‹*mset* Φ ⊆# *mset* (*map f* (γ # Γ))›
have *mset* (*remove1* (*f* γ) Φ) ⊆# *mset* (*map f* Γ)
by (*metis insert-subset-eq-iff*
list.simps(*9*)
mset.simps(*2*)
mset-eq-perm
mset-remove1
remove-hd)
from *this Cons* **obtain** Φ′ **where** Φ′:
mset Φ′ ⊆# *mset* Γ
remove1 (*f* γ) Φ = *map f* Φ′
by *blast*
hence *mset* (γ # Φ′) ⊆# *mset* (γ # Γ)
and *f* γ # (*remove1* (*f* γ) Φ) = *map f* (γ # Φ′)
by *simp+*
hence Φ <~~> *map f* (γ # Φ′)
using ‹*f* γ ∈ *set* Φ› *perm-remove* **by** *force*
from *this* **obtain** Φ″ **where** Φ″:
Φ = *map f* Φ″
Φ″ <~~> γ # Φ′
using *perm-map-perm-list-exists*
by *blast*
hence *mset* Φ″ ⊆# *mset* (γ # Γ)
by (*metis* ‹*mset* (γ # Φ′) ⊆# *mset* (γ # Γ)› *mset-eq-perm*)
thus *?thesis* **using** Φ″ **by** *blast*
next
assume *f* γ ∉ *set* Φ
have *mset* Φ − {#*f* γ#} = *mset* Φ
by (*metis* (*no-types*) ‹*f* γ ∉ *set* Φ› *diff-single-trivial set-mset-mset*)
moreover have *mset* (*map f* (γ # Γ)) = *add-mset* (*f* γ) (*image-mset f* (*mset* Γ))
by *simp*
ultimately have *mset* Φ ⊆# *mset* (*map f* Γ)
by (*metis* (*no-types*) *Diff-eq-empty-iff-mset*

$\langle$*mset* $\Phi \subseteq\#$ *mset* (*map f* $(\gamma \# \Gamma))\rangle$
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

4.7 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

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

4.9 List Duplicates

primrec *duplicates* :: *'a list* ⇒ *'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 $\{y.\ 2 \leq$ *count-list* $(x \# xs)$ $y\}$ = *insert x* $\{y.\ 2 \leq$ *count-list xs y*$\}$

by (*simp*, *blast*)
thus *?thesis* **using** *inductive-hypothesis* ‹$x \in$ *set xs*›
by *simp*
next
assume $x \notin$ *set xs*
hence $\{y.\ 2 \leq$ *count-list* $(x \# xs)\ y\} = \{y.\ 2 \leq$ *count-list xs y*$\}$
by (*simp*, *auto*)
thus *?thesis* **using** *inductive-hypothesis* ‹$x \notin$ *set xs*›
by *simp*
qed
qed

4.10 List Subtraction

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

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

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

lemma *listSubtract-remove1-cons-perm*:
$\Phi \ominus (\varphi \# \Lambda)$ <~~> (*remove1* φ Φ) $\ominus$ Λ
by (*induct* Λ, *simp*, *simp*, *metis perm-remove-perm remove1-commute*)

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

lemma *listSubtract-cons-absorb*:
assumes *count-list* Φ φ $\geq$ *count-list* Λ φ
shows $\varphi \# (\Phi \ominus \Lambda)$ <~~> $(\varphi \# \Phi) \ominus \Lambda$
using *assms*
proof −
have $\forall$ Φ. *count-list* Φ φ $\geq$ *count-list* Λ φ $\longrightarrow$ $\varphi \# (\Phi \ominus \Lambda)$ <~~> $(\varphi \# \Phi) \ominus$
Λ
proof (*induct* Λ)
case *Nil*
thus *?case* **using** *listSubtract-cons* **by** *fastforce*
next
case (*Cons* ψ Λ)

assume *inductive-hypothesis*:
$\forall \Phi.$ *count-list* $\Lambda\ \varphi \leq$ *count-list* $\Phi\ \varphi \longrightarrow \varphi \ \#\ \Phi \ominus \Lambda <\sim\sim> (\varphi\ \#\ \Phi) \ominus$
Λ
{
fix Φ :: *'a list*
assume *count-list* $(\psi\ \#\ \Lambda)\ \varphi \leq$ *count-list* $\Phi\ \varphi$
have *count-list* $\Lambda\ \varphi \leq$ *count-list* (*remove1* $\psi\ \Phi)\ \varphi$
proof (*cases* $\varphi = \psi$)
case *True*
hence *1* + *count-list* $\Lambda\ \varphi \leq$ *count-list* $\Phi\ \varphi$
using ‹*count-list* $(\psi\ \#\ \Lambda)\ \varphi \leq$ *count-list* $\Phi\ \varphi$›
by *auto*
moreover from *this* **have** $\varphi \in$ *set* Φ
using *not-one-le-zero* **by** *fastforce*
hence $\Phi <\sim\sim> \varphi\ \#$ (*remove1* $\psi\ \Phi$)
using *True*
by (*simp add*: *True perm-remove*)
ultimately show *?thesis* **by** (*simp add*: *perm-count-list*)
next
case *False*
hence *count-list* $(\psi\ \#\ \Lambda)\ \varphi =$ *count-list* $\Lambda\ \varphi$
by *simp*
moreover have *count-list* $\Phi\ \varphi =$ *count-list* (*remove1* $\psi\ \Phi)\ \varphi$
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\varphi'\ \Phi$)
show *?case*
proof (*cases* $\varphi' = \varphi$)
case *True*
with ‹$\varphi \neq \psi$›
have *count-list* $(\varphi'\ \#\ \Phi)\ \varphi =$ *1* + *count-list* $\Phi\ \varphi$
count-list (*remove1* $\psi\ (\varphi'\ \#\ \Phi))\ \varphi =$ *1* + *count-list* (*remove1* $\psi\ \Phi$)
φ
by *simp*+
with *Cons* **show** *?thesis* **by** *linarith*
next
case *False*
with *Cons* **show** *?thesis* **by** (*cases* $\varphi' = \psi$, *simp*+)
qed
qed
ultimately show *?thesis*
using ‹*count-list* $(\psi\ \#\ \Lambda)\ \varphi \leq$ *count-list* $\Phi\ \varphi$›
by *auto*
qed
hence $\varphi\ \#$ ((*remove1* $\psi\ \Phi) \ominus \Lambda) <\sim\sim> (\varphi\ \#$ (*remove1* $\psi\ \Phi)) \ominus \Lambda$
using *inductive-hypothesis* **by** *blast*
moreover have $\varphi\ \#$ ((*remove1* $\psi\ \Phi) \ominus \Lambda) <\sim\sim> \varphi\ \#\ (\Phi \ominus (\psi\ \#\ \Lambda))$

by (*induct* Λ, *simp*, *simp add*: *perm-remove-perm remove1-commute*)
ultimately have $\star$: $\varphi \# (\Phi \ominus (\psi \# \Lambda)) <\sim\sim> (\varphi \# (remove1\ \psi\ \Phi)) \ominus \Lambda$
by (*meson perm.trans perm-sym*)
have $\varphi \# (\Phi \ominus (\psi \# \Lambda)) <\sim\sim> (\varphi \# \Phi) \ominus (\psi \# \Lambda)$
proof *cases*
assume $\varphi = \psi$
hence $(\varphi \# \Phi) \ominus (\psi \# \Lambda) <\sim\sim> \Phi \ominus \Lambda$
using *listSubtract-remove1-cons-perm* **by** *fastforce*
moreover have $\varphi \in set\ \Phi$
using ‹$\varphi = \psi$› ‹$count\text{-}list\ (\psi \# \Lambda)\ \varphi \leq count\text{-}list\ \Phi\ \varphi$› *leD* **by** *force*
hence $\Phi \ominus \Lambda <\sim\sim> (\varphi \# (remove1\ \varphi\ \Phi)) \ominus \Lambda$
by (*induct* Λ, *simp add*: *perm-remove*, *simp add*: *perm-remove-perm*)
ultimately show *?thesis*
using $\star$
by (*metis* ‹$\varphi = \psi$› *mset-eq-perm*)
next
assume $\varphi \neq \psi$
hence $(\varphi \# (remove1\ \psi\ \Phi)) \ominus \Lambda <\sim\sim> (\varphi \# \Phi) \ominus (\psi \# \Lambda)$
by (*induct* Λ, *simp*, *simp add*: *perm-remove-perm remove1-commute*)
then show *?thesis* **using** $\star$ **by** *blast*
qed
}
then show *?case* **by** *blast*
qed
with *assms* **show** *?thesis* **by** *blast*
qed

lemma *listSubtract-remove1-perm*:
assumes $\varphi \in set\ \Lambda$
shows $\Phi \ominus \Lambda <\sim\sim> (remove1\ \varphi\ \Phi) \ominus (remove1\ \varphi\ \Lambda)$
proof −
from ‹$\varphi \in set\ \Lambda$›
have $mset\ (\Phi \ominus \Lambda) = mset\ ((remove1\ \varphi\ \Phi) \ominus (remove1\ \varphi\ \Lambda))$
by *simp*
thus *?thesis*
using *mset-eq-perm* **by** *blast*
qed

lemma *listSubtract-cons-remove1-perm*:
assumes $\varphi \in set\ \Lambda$
shows $(\varphi \# \Phi) \ominus \Lambda <\sim\sim> \Phi \ominus (remove1\ \varphi\ \Lambda)$
using *assms listSubtract-remove1-perm* **by** *fastforce*

lemma *listSubtract-removeAll-perm*:
assumes $count\text{-}list\ \Phi\ \varphi \leq count\text{-}list\ \Lambda\ \varphi$
shows $\Phi \ominus \Lambda <\sim\sim> (removeAll\ \varphi\ \Phi) \ominus (removeAll\ \varphi\ \Lambda)$
proof −
have $\forall\ \Lambda.\ count\text{-}list\ \Phi\ \varphi \leq count\text{-}list\ \Lambda\ \varphi \longrightarrow \Phi \ominus \Lambda <\sim\sim> (removeAll\ \varphi\ \Phi) \ominus (removeAll\ \varphi\ \Lambda)$

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) ⊖ (removeAll φ (remove1 ξ Λ)) <~~>
              (removeAll φ Φ) ⊖ (remove1 ξ (removeAll φ Λ))
          by (induct Λ, simp, simp add: filter-remove1 removeAll-filter-not-eq)
        hence (removeAll φ Φ) ⊖ (removeAll φ (remove1 ξ Λ)) <~~>
               (removeAll φ (ξ # Φ)) ⊖ (removeAll φ Λ)
          by (simp add: ‹ξ ∈ set Λ›
                        filter-remove1
                        listSubtract-cons-remove1-perm
                        perm-sym
                        removeAll-filter-not-eq)
        ultimately show ?thesis by blast
      next
        assume ξ ∉ set Λ
        hence (ξ # Φ) ⊖ Λ <~~> ξ # (Φ ⊖ Λ)
          by (simp add: listSubtract-cons-absorb perm-sym)
        hence (ξ # Φ) ⊖ Λ <~~> ξ # ((removeAll φ Φ) ⊖ (removeAll φ Λ))
          using ‹Φ ⊖ Λ <~~> removeAll φ Φ ⊖ removeAll φ Λ› by blast
        hence (ξ # Φ) ⊖ Λ <~~> (ξ # (removeAll φ Φ)) ⊖ (removeAll φ Λ)
          by (simp add: ‹ξ ∉ set Λ› listSubtract-cons)
        thus ?thesis using ‹ξ ≠ φ› by auto
      qed
    qed
  }
  then show ?case by auto
 qed
 with assms show ?thesis by blast
qed

lemma listSubtract-permute:
 assumes Φ <~~> Ψ
 shows Φ ⊖ Λ <~~> Ψ ⊖ Λ
proof −
 from ‹Φ <~~> Ψ› have mset Φ = mset Ψ
  by (simp add: mset-eq-perm)
 hence mset (Φ ⊖ Λ) = mset (Ψ ⊖ Λ)
  by simp
 thus ?thesis
  using mset-eq-perm by blast
qed

lemma append-perm-listSubtract-intro:
 assumes A <~~> B @ C
 shows A ⊖ C <~~> B
proof −
 from ‹A <~~> B @ C› have mset A = mset (B @ C)
  using mset-eq-perm by blast
 hence mset (A ⊖ 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
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow ((\mathit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow (\mathit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
using *Cons.hyps* **by** *blast*
moreover have $(\mathit{remove1}\ \psi\ \Phi) \ominus \Psi$ <~~> $\Phi \ominus (\psi \# \Psi)$
by (*meson listSubtract-remove1-cons-perm perm-sym*)
hence $(\sum \varphi' \leftarrow ((\mathit{remove1}\ \psi\ \Phi) \ominus \Psi).\ f\ \varphi') = (\sum \varphi' \leftarrow (\Phi \ominus (\psi \# \Psi)).\ f\ \varphi')$
using *perm-list-summation* **by** *blast*
ultimately have
$(\sum \psi' \leftarrow \Psi.\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus (\psi \# \Psi)).\ f\ \varphi') = (\sum \varphi' \leftarrow (\mathit{remove1}\ \psi\ \Phi).\ f\ \varphi')$
by *simp*
hence
$(\sum \psi' \leftarrow (\psi \# \Psi).\ f\ \psi') + (\sum \varphi' \leftarrow (\Phi \ominus (\psi \# \Psi)).\ f\ \varphi') =$
$(\sum \varphi' \leftarrow (\psi \# (\mathit{remove1}\ \psi\ \Phi)).\ f\ \varphi')$
by (*simp add*: *add.assoc*)
moreover have $\psi \in$ *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 φ ∈ set Γ
      by (metis ‹mset (Γ ⊖ (φ # Φ)) = mset (remove1 φ Γ ⊖ Φ)›
               ‹mset (φ # Φ) ⊆# mset Γ›
               ‹mset Φ ⊆# mset (remove1 φ Γ)›
               add-diff-cancel-left'
               add-right-cancel
               eq-iff
               impossible-Cons
               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 ∉ set A
 shows A ⊖ B = A ⊖ (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 ⊆# mset B
 shows mset (A ⊖ C) ⊆# mset (B ⊖ 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) ⊖ (map f B)) ⊆# mset (map f (A ⊖ 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 ⊆# mset A
  shows mset ((map f A) ⊖ (map f B)) = mset (map f (A ⊖ B))
  using assms
  by (induct B, simp, simp add: image-mset-Diff)

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

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

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

lemma *triple-list-exists*:
  assumes *mset* (*map snd* Ψ) $\subseteq$# *mset* Σ
    and *mset* Σ $\subseteq$# *mset* (*map snd* Δ)
  shows $\exists$ Ω. *map* (λ (ψ, σ, -). (ψ, σ)) Ω = Ψ $\wedge$
      mset (*map* (λ (-, σ, γ). (γ, σ)) Ω) $\subseteq$# *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) $\subseteq$# *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 *?*Δ_Ω = *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)) $\subseteq$# *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* (λ(*b*, *a*, *c*). (*c*, *a*)) Ω) + {#*case* (*snd* ψ, γ) *of* (*a*, *c*) ⇒ (*c*, *a*)#}

$= mset\ (map\ (\lambda(b,\ a,\ c).\ (c,\ a))\ ((fst\ \psi,\ snd\ \psi,\ \gamma)\ \#\ \Omega))$
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 $mset\ \Delta = mset\ (map\ (\lambda(b,\ a,\ c).\ (c,\ a))\ \Omega) + mm\ (mset\ \Delta)\ (mset\ (map\ (\lambda(b,\ a,\ c).\ (c,\ a))\ \Omega))$
by (*metis* $\Omega(2)$ *subset-mset.le-iff-add*)
then have $mset\ (map\ (\lambda\ (\text{-},\ \sigma,\ \gamma).\ (\gamma,\ \sigma))\ ?\Omega) \subseteq\#\ mset\ \Delta$
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

4.12 List Intersection

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

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

lemma *list-diff-intersect-comp*:
$mset\ \Phi = mset\ (\Phi \ominus \Psi) + mset\ (\Phi \cap \Psi)$
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

5 Classical Propositional Connectives

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

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

5.1 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*

5.2 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*

5.3 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*:
$\vdash (\varphi \leftrightarrow \psi) \to \varphi \to \psi$

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*)

5.4 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*

5.5 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* $\Rightarrow$ $'a$ ($\bigsqcup$)
  where
    $\bigsqcup$ [] = $\bot$
  | $\bigsqcup$ (φ # Φ) = $\varphi \sqcup \bigsqcup$ Φ

lemma (**in** *Classical-Propositional-Logic*) *disjunction-elimination*:
 $\vdash (\varphi \to \chi) \to (\psi \to \chi) \to (\varphi \sqcup \psi) \to \chi$
proof −
  let *?*Γ = [$\varphi \to \chi$, $\psi \to \chi$, $\varphi \sqcup \psi$]
  have *?*Γ :$\vdash$ ($\varphi \to \bot$) $\to \chi$
    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 *?*Γ :$\vdash$ χ
    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*]:

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

5.6 Mutual Exclusion

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

5.7 Subtraction

definition (**in** *Classical-Propositional-Logic*) *subtraction* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\backslash$ *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*

5.8 Common Rules

5.8.1 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 $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\langle\varphi\rangle \leftrightarrow \langle\psi\rangle) \to (\langle\psi\rangle \leftrightarrow \langle\chi\rangle) \to (\langle\varphi\rangle \leftrightarrow \langle\chi\rangle)$

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*

5.8.2 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)

5.8.3 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** $\Gamma=\Gamma$]
set-deduction-modus-ponens [**where** $\Gamma=\Gamma$]
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** $\Gamma=\Gamma$]
list-deduction-modus-ponens [**where** $\Gamma=\Gamma$]
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 Φ ⊆ set Ψ ⟹ ⊢ ⨅ Ψ → ⨅ Φ
proof −
  have ∀ Φ. set Φ ⊆ set Ψ ⟶ ⊢ ⨅ Ψ → ⨅ Φ
  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 Φ ⊆ set (ψ # Ψ)
      have ⊢ ⨅ (ψ # Ψ) → ⨅ Φ
      proof (cases ψ ∈ set Φ)
        assume ψ ∈ set Φ
                have ∀ φ ∈ set Φ. ⊢ ⨅ Φ ↔ (φ ⊓ ⨅ (removeAll φ Φ))
        proof (induct Φ)
          case Nil
          then show ?case by simp
        next
          case (Cons χ Φ)
          {
            fix φ
            assume φ ∈ set (χ # Φ)
            have ⊢ ⨅ (χ # Φ) ↔ (φ ⊓ ⨅ (removeAll φ (χ # Φ)))
            proof cases
              assume φ ∈ set Φ
              hence ⊢ ⨅ Φ ↔ (φ ⊓ ⨅ (removeAll φ Φ))
                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
      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 $\vdash (\psi \sqcap \bigsqcap\ (\textit{removeAll}\ \psi\ \Phi)) \to \bigsqcap\ \Phi$
  using *Modus-Ponens biconditional-right-elimination* ‹$\psi \in$ *set* Φ›
  by *blast*
moreover
from ‹$\psi \in$ *set* Φ› ‹*set* $\Phi \subseteq$ *set* $(\psi\ \#\ \Psi)$› *Cons.hyps*
have $\vdash \bigsqcap\ \Psi \to \bigsqcap\ (\textit{removeAll}\ \psi\ \Phi)$
  by (*simp add*: *subset-insert-iff insert-absorb*)
hence $\vdash \bigsqcap\ (\psi\ \#\ \Psi) \to (\psi \sqcap \bigsqcap\ (\textit{removeAll}\ \psi\ \Phi))$
  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 $\psi \notin$ *set* Φ
hence $\vdash \bigsqcap\ \Psi \to \bigsqcap\ \Phi$
  using *Cons.hyps* ‹*set* $\Phi \subseteq$ *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 *set* $\Phi \subseteq$ *set* $\Psi \Longrightarrow \vdash \bigsqcap\ \Psi \to \bigsqcap\ \Phi$ **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *arbitrary-conjunction-remdups*:
$\vdash (\bigsqcap\ \Phi) \leftrightarrow \bigsqcap\ (\textit{remdups}\ \Phi)$
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** *?*φ=φ
and *?*ψ=$\bigsqcap\ \Phi$
and *?*χ=χ]
show *?case*
unfolding *biconditional-def*
by (*simp*, *metis Modus-Ponens hypothetical-syllogism*)
qed

5.8.4 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* $\Psi \Longrightarrow \vdash \bigsqcup\ \Phi \to \bigsqcup\ \Psi$
proof −
have $\forall\ \Phi.$ *set* $\Phi \subseteq$ *set* $\Psi \longrightarrow \vdash \bigsqcup\ \Phi \to \bigsqcup\ \Psi$
proof (*induct* Ψ)
case *Nil*
then show *?case* **using** *verum-def verum-tautology* **by** *auto*
next
case (*Cons* ψ Ψ)
{
fix Φ
assume *set* $\Phi \subseteq$ *set* $(\psi \# \Psi)$
have $\vdash \bigsqcup\ \Phi \to \bigsqcup\ (\psi \# \Psi)$
proof *cases*
assume $\psi \in$ *set* Φ
have $\forall\ \varphi \in$ *set* $\Phi.\ \vdash \bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup$ (*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 $\vdash \bigsqcup\ (\chi \# \Phi) \leftrightarrow (\varphi \sqcup \bigsqcup$ (*removeAll* φ $(\chi \# \Phi)$)))
proof *cases*
assume $\varphi \in$ *set* Φ
hence $\vdash \bigsqcup\ \Phi \leftrightarrow (\varphi \sqcup \bigsqcup$ (*removeAll* φ Φ))
using *Cons.hyps* ‹$\varphi \in$ *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 ⊢ (ψ ⊔ ⨆ (removeAll ψ Φ)) → ⨆ (ψ # Ψ)
```

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* Φ
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* Φ)
by (*simp add*: *arbitrary-disjunction-monotone biconditional-def*)

5.8.5 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 $\vdash$ ($\bigsqcup$ (ψ # Ψ) $\sqcap$ φ) $\leftrightarrow$ (($\psi \sqcap \varphi$) $\sqcup$ (($\bigsqcup$ Ψ) $\sqcap$ φ))

using *conjunction-distribution* **by** *auto*

moreover

from *Cons* **have** $\vdash$ (($\psi \sqcap \varphi$) $\sqcup$ (($\bigsqcup$ Ψ) $\sqcap$ φ)) $\leftrightarrow$ (($\psi \sqcap \varphi$) $\sqcup$ ($\bigsqcup$ [$\psi \sqcap \varphi$. $\psi \leftarrow$ Ψ]))

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*:

$\vdash$ ($\bigsqcup$ Ψ $\setminus$ φ) $\leftrightarrow$ $\bigsqcup$ [$\psi \setminus \varphi$. $\psi \leftarrow \Psi$]

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*:

$\vdash$ (Φ :$\to$ ($\psi \sqcap \chi$)) $\leftrightarrow$ ((Φ :$\to$ ψ) $\sqcap$ (Φ :$\to$ χ))

proof (*induct* Φ)

case *Nil*

then show *?case*

by (*simp add*: *biconditional-reflection*)

next

case (*Cons* φ Φ)

hence $\vdash$ (φ # Φ) :$\to$ ($\psi \sqcap \chi$) $\leftrightarrow$ ($\varphi \to$ (Φ :$\to$ $\psi \sqcap \Phi$:$\to$ χ))

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

lemma (**in** *Classical-Propositional-Logic*) *list-implication-arbitrary-distribution*:
$\vdash (\Phi :\rightarrow \prod \Psi) \leftrightarrow \prod [\Phi :\rightarrow \psi.\ \psi \leftarrow \Psi]$
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 $\vdash \Phi :\to \bigsqcap (\psi \# \Psi) \leftrightarrow (\Phi :\to \psi \sqcap \Phi :\to \bigsqcap \Psi)$
using *list-implication-distribution*
by *fastforce*
moreover
from *biconditional-conjunction-weaken-rule*
Cons
have $\vdash (\Phi :\to \psi \sqcap \Phi :\to \bigsqcap \Psi) \leftrightarrow \bigsqcap [\Phi :\to \psi.\ \psi \leftarrow (\psi \# \Psi)]$
by *simp*
ultimately show *?case*
by (*metis biconditional-transitivity-rule*)
qed

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

5.8.6 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*

5.9 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 $\vdash \sim (\chi \sqcap \varphi) \rightarrow \sim (\varphi \sqcap \chi)$
      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 $\varphi \in set\ \Phi$

moreover {
  fix φ ψ χ
  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* $\Phi.\ \Gamma \Vdash \sim \varphi$
    $\forall\ \varphi \in$ *set* $\Phi.\ \forall\ \psi \in$ *set* $\Phi.\ (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$
  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* $(\varphi \# \Phi)$. $\Gamma \Vdash \sim \varphi$
and *B*: $\forall \chi \in$ *set* $(\varphi \# \Phi)$. $\forall \psi \in$ *set* $(\varphi \# \Phi)$. $\chi \neq \psi \longrightarrow \Gamma \Vdash \sim (\chi \sqcap \psi)$
hence *C*: $\Gamma \Vdash \coprod \Phi$ **using** *Cons.hyps* **by** *simp*
then show *?case*
proof *cases*
assume $\varphi \in$ *duplicates* $(\varphi \# \Phi)$
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* $(\psi \# \Phi)$
then show $\vdash \sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup (\psi \# \Phi))$
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* $(\psi \# \Phi)$› **by** *auto*
hence $\vdash \sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi)$ **using** *Cons.hyps* **by** *auto*
moreover have $\vdash (\sim \varphi \leftrightarrow \sim (\varphi \sqcap \bigsqcup \Phi)) \rightarrow (\sim \varphi \leftrightarrow \sim (\varphi \sqcap (\psi \sqcup \bigsqcup$
$\Phi)))$
proof $-$
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle)) \rightarrow$
$(\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup \Phi\rangle)))$
by *auto*
hence $\vdash (\!| (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap \langle\bigsqcup \Phi\rangle)) \rightarrow (\sim \langle\varphi\rangle \leftrightarrow \sim (\langle\varphi\rangle \sqcap (\langle\psi\rangle$
$\sqcup \langle\bigsqcup \Phi\rangle))) |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed

ultimately have *?thesis* **using** *Modus-Ponens* **by** *simp*
}
ultimately show *?thesis* **by** *auto*
qed
qed
with ‹$\Gamma \Vdash \sim \varphi$› **have** $\Gamma \Vdash \sim(\varphi \sqcap \bigsqcup \Phi)$
using *biconditional-weaken set-deduction-weaken* **by** *blast*
with ‹$\Gamma \Vdash \coprod \Phi$› **show** *?thesis* **by** *simp*
next
assume $\varphi \notin$ *duplicates* $(\varphi \# \Phi)$
hence $\varphi \notin$ *set* Φ **by** *auto*
with B **have** $\forall \psi \in set\ \Phi.\ \Gamma \Vdash \sim (\varphi \sqcap \psi)$ **by** (*simp*, *metis*)
hence $\Gamma \Vdash \sim (\varphi \sqcap \bigsqcup \Phi)$
by (*simp add*: *disjuction-exclusion-equivalence*)
with ‹$\Gamma \Vdash \coprod \Phi$› **show** *?thesis* **by** *simp*
qed
qed
}
thus *?thesis*
by (*metis exclusive-elimination1 exclusive-elimination2*)
qed

end

6 Archimedean Fields, Floor and Ceiling Functions

theory *Archimedean-Field*
imports *Main*
begin

lemma *cInf-abs-ge*:
fixes S :: *'a*::{*linordered-idom*,*conditionally-complete-linorder*} *set*
assumes $S \neq \{\}$
and *bdd*: $\bigwedge x.\ x \in S \Longrightarrow |x| \leq a$
shows $|Inf\ S| \leq a$
proof −
have *Sup* (*uminus* ‘ S) = − (*Inf* S)
proof (*rule antisym*)
show − (*Inf* S) ≤ *Sup* (*uminus* ‘ S)
apply (*subst minus-le-iff*)
apply (*rule cInf-greatest* [*OF* ‹$S \neq \{\}$›])
apply (*subst minus-le-iff*)
apply (*rule cSup-upper*)
apply *force*
using *bdd*
apply (*force simp*: *abs-le-iff bdd-above-def*)

done
next
show *Sup* (*uminus* ‘ *S*) $\leq$ − *Inf S*
apply (*rule cSup-least*)
using ‹*S* $\neq$ {}›
apply *force*
apply *clarsimp*
apply (*rule cInf-lower*)
apply *assumption*
using *bdd*
apply (*simp add*: *bdd-below-def*)
apply (*rule-tac x* = − *a* **in** *exI*)
apply *force*
done
qed
with *cSup-abs-le* [*of uminus* ‘ *S*] *assms* **show** *?thesis*
by *fastforce*
qed

lemma *cSup-asclose*:
fixes *S* :: ′*a*::{*linordered-idom*,*conditionally-complete-linorder*} *set*
assumes *S*: *S* $\neq$ {}
and *b*: $\forall x \in S$. $|x - l| \leq e$
shows $|Sup\ S - l| \leq e$
proof −
have ∗: $|x - l| \leq e \longleftrightarrow l - e \leq x \wedge x \leq l + e$ **for** *x l e* :: ′*a*
by *arith*
have *bdd-above S*
using *b* **by** (*auto intro*!: *bdd-aboveI*[*of* - *l* + *e*])
with *S b* **show** *?thesis*
unfolding ∗ **by** (*auto intro*!: *cSup-upper2 cSup-least*)
qed

lemma *cInf-asclose*:
fixes *S* :: ′*a*::{*linordered-idom*,*conditionally-complete-linorder*} *set*
assumes *S*: *S* $\neq$ {}
and *b*: $\forall x \in S$. $|x - l| \leq e$
shows $|Inf\ S - l| \leq e$
proof −
have ∗: $|x - l| \leq e \longleftrightarrow l - e \leq x \wedge x \leq l + e$ **for** *x l e* :: ′*a*
by *arith*
have *bdd-below S*
using *b* **by** (*auto intro*!: *bdd-belowI*[*of* - *l* − *e*])
with *S b* **show** *?thesis*
unfolding ∗ **by** (*auto intro*!: *cInf-lower2 cInf-greatest*)
qed

6.1 Class of Archimedean fields

Archimedean fields have no infinite elements.

```
class archimedean-field = linordered-field +
  assumes ex-le-of-int: ∃ z. x ≤ of-int z

lemma ex-less-of-int: ∃ z. x < of-int z
  for x :: 'a::archimedean-field
proof −
  from ex-le-of-int obtain z where x ≤ of-int z ..
  then have x < of-int (z + 1) by simp
  then show ?thesis ..
qed

lemma ex-of-int-less: ∃ z. of-int z < x
  for x :: 'a::archimedean-field
proof −
  from ex-less-of-int obtain z where − x < of-int z ..
  then have of-int (− z) < x by simp
  then show ?thesis ..
qed

lemma reals-Archimedean2: ∃ n. x < of-nat n
  for x :: 'a::archimedean-field
proof −
  obtain z where x < of-int z
    using ex-less-of-int ..
  also have ... ≤ of-int (int (nat z))
    by simp
  also have ... = of-nat (nat z)
    by (simp only: of-int-of-nat-eq)
  finally show ?thesis ..
qed

lemma real-arch-simple: ∃ n. x ≤ of-nat n
  for x :: 'a::archimedean-field
proof −
  obtain n where x < of-nat n
    using reals-Archimedean2 ..
  then have x ≤ of-nat n
    by simp
  then show ?thesis ..
qed
```

Archimedean fields have no infinitesimal elements.

```
lemma reals-Archimedean:
  fixes x :: 'a::archimedean-field
  assumes 0 < x
  shows ∃ n. inverse (of-nat (Suc n)) < x
```

proof −
  from ‹$0 < x$› **have** $0 < inverse\ x$
    by (*rule positive-imp-inverse-positive*)
  obtain n **where** $inverse\ x < of\text{-}nat\ n$
    using *reals-Archimedean2* **..**
  then obtain m **where** $inverse\ x < of\text{-}nat\ (Suc\ m)$
    using ‹$0 < inverse\ x$› **by** (*cases n*) (*simp-all del*: *of-nat-Suc*)
  then have $inverse\ (of\text{-}nat\ (Suc\ m)) < inverse\ (inverse\ x)$
    using ‹$0 < inverse\ x$› **by** (*rule less-imp-inverse-less*)
  then have $inverse\ (of\text{-}nat\ (Suc\ m)) < x$
    using ‹$0 < x$› **by** (*simp add*: *nonzero-inverse-inverse-eq*)
  then show *?thesis* **..**
qed

lemma *ex-inverse-of-nat-less*:
  fixes x :: *'a::archimedean-field*
  assumes $0 < x$
  shows $\exists n>0.\ inverse\ (of\text{-}nat\ n) < x$
  using *reals-Archimedean* [*OF* ‹$0 < x$›] **by** *auto*

lemma *ex-less-of-nat-mult*:
  fixes x :: *'a::archimedean-field*
  assumes $0 < x$
  shows $\exists n.\ y < of\text{-}nat\ n * x$
proof −
  obtain n **where** $y\ /\ x < of\text{-}nat\ n$
    using *reals-Archimedean2* **..**
  with ‹$0 < x$› **have** $y < of\text{-}nat\ n * x$
    by (*simp add*: *pos-divide-less-eq*)
  then show *?thesis* **..**
qed

6.2 Existence and uniqueness of floor function

lemma *exists-least-lemma*:
  assumes $\neg\ P\ 0$ **and** $\exists n.\ P\ n$
  shows $\exists n.\ \neg\ P\ n \wedge P\ (Suc\ n)$
proof −
  from ‹$\exists n.\ P\ n$› **have** $P\ (Least\ P)$
    by (*rule LeastI-ex*)
  with ‹$\neg\ P\ 0$› **obtain** n **where** $Least\ P = Suc\ n$
    by (*cases Least P*) *auto*
  then have $n < Least\ P$
    by *simp*
  then have $\neg\ P\ n$
    by (*rule not-less-Least*)
  then have $\neg\ P\ n \wedge P\ (Suc\ n)$
    using ‹$P\ (Least\ P)$› ‹$Least\ P = Suc\ n$› **by** *simp*
  then show *?thesis* **..**

qed

lemma *floor-exists*:
  fixes x :: $'a$::*archimedean-field*
  shows $\exists z.$ *of-int* $z \leq x \wedge x <$ *of-int* $(z + 1)$
proof (*cases* $0 \leq x$)
  case *True*
  then have $\neg\ x <$ *of-nat* 0
    by *simp*
  then have $\exists n.\ \neg\ x <$ *of-nat* $n \wedge x <$ *of-nat* (*Suc* n)
    using *reals-Archimedean2* **by** (*rule exists-least-lemma*)
  then obtain n **where** $\neg\ x <$ *of-nat* $n \wedge x <$ *of-nat* (*Suc* n) **..**
  then have *of-int* (*int* n) $\leq x \wedge x <$ *of-int* (*int* $n + 1$)
    by *simp*
  then show *?thesis* **..**
next
  case *False*
  then have $\neg - x \leq$ *of-nat* 0
    by *simp*
  then have $\exists n.\ \neg - x \leq$ *of-nat* $n \wedge - x \leq$ *of-nat* (*Suc* n)
    using *real-arch-simple* **by** (*rule exists-least-lemma*)
  then obtain n **where** $\neg - x \leq$ *of-nat* $n \wedge - x \leq$ *of-nat* (*Suc* n) **..**
  then have *of-int* ($-$ *int* $n - 1$) $\leq x \wedge x <$ *of-int* ($-$ *int* $n - 1 + 1$)
    by *simp*
  then show *?thesis* **..**
qed

lemma *floor-exists1*: $\exists!z.$ *of-int* $z \leq x \wedge x <$ *of-int* $(z + 1)$
  for x :: $'a$::*archimedean-field*
proof (*rule ex-ex1I*)
  show $\exists z.$ *of-int* $z \leq x \wedge x <$ *of-int* $(z + 1)$
    by (*rule floor-exists*)
next
  fix $y\ z$
  assume *of-int* $y \leq x \wedge x <$ *of-int* $(y + 1)$
    and *of-int* $z \leq x \wedge x <$ *of-int* $(z + 1)$
  with *le-less-trans* [*of of-int* $y\ x$ *of-int* $(z + 1)$]
      le-less-trans [*of of-int* $z\ x$ *of-int* $(y + 1)$] **show** $y = z$
    by (*simp del*: *of-int-add*)
qed

6.3 Floor function

class *floor-ceiling* = *archimedean-field* +
  fixes *floor* :: $'a \Rightarrow$ *int* ($\lfloor$-$\rfloor$)
  assumes *floor-correct*: *of-int* $\lfloor x \rfloor \leq x \wedge x <$ *of-int* $(\lfloor x \rfloor + 1)$

lemma *floor-unique*: *of-int* $z \leq x \Longrightarrow x <$ *of-int* $z + 1 \Longrightarrow \lfloor x \rfloor = z$
  using *floor-correct* [*of* x] *floor-exists1* [*of* x] **by** *auto*

lemma *floor-eq-iff*: $\lfloor x \rfloor = a \longleftrightarrow$ *of-int* $a \leq x \wedge x <$ *of-int* $a + 1$
using *floor-correct floor-unique* **by** *auto*

lemma *of-int-floor-le* [*simp*]: *of-int* $\lfloor x \rfloor \leq x$
  using *floor-correct* **..**

lemma *le-floor-iff*: $z \leq \lfloor x \rfloor \longleftrightarrow$ *of-int* $z \leq x$
proof
  assume $z \leq \lfloor x \rfloor$
  then have (*of-int* z :: $'a$) $\leq$ *of-int* $\lfloor x \rfloor$ **by** *simp*
  also have *of-int* $\lfloor x \rfloor \leq x$ **by** (*rule of-int-floor-le*)
  finally show *of-int* $z \leq x$ **.**
next
  assume *of-int* $z \leq x$
  also have $x <$ *of-int* $(\lfloor x \rfloor + 1)$ **using** *floor-correct* **..**
  finally show $z \leq \lfloor x \rfloor$ **by** (*simp del*: *of-int-add*)
qed

lemma *floor-less-iff*: $\lfloor x \rfloor < z \longleftrightarrow x <$ *of-int* z
  by (*simp add*: *not-le* [*symmetric*] *le-floor-iff*)

lemma *less-floor-iff*: $z < \lfloor x \rfloor \longleftrightarrow$ *of-int* $z + 1 \leq x$
  using *le-floor-iff* [*of* $z + 1$ x] **by** *auto*

lemma *floor-le-iff*: $\lfloor x \rfloor \leq z \longleftrightarrow x <$ *of-int* $z + 1$
  by (*simp add*: *not-less* [*symmetric*] *less-floor-iff*)

lemma *floor-split*[*arith-split*]: P $\lfloor t \rfloor \longleftrightarrow (\forall\, i.$ *of-int* $i \leq t \wedge t <$ *of-int* $i + 1 \longrightarrow$
P $i)$
  by (*metis floor-correct floor-unique less-floor-iff not-le order-refl*)

lemma *floor-mono*:
  assumes $x \leq y$
  shows $\lfloor x \rfloor \leq \lfloor y \rfloor$
proof −
  have *of-int* $\lfloor x \rfloor \leq x$ **by** (*rule of-int-floor-le*)
  also note ‹$x \leq y$›
  finally show *?thesis* **by** (*simp add*: *le-floor-iff*)
qed

lemma *floor-less-cancel*: $\lfloor x \rfloor < \lfloor y \rfloor \Longrightarrow x < y$
  by (*auto simp add*: *not-le* [*symmetric*] *floor-mono*)

lemma *floor-of-int* [*simp*]: $\lfloor$*of-int* $z\rfloor = z$
  by (*rule floor-unique*) *simp-all*

lemma *floor-of-nat* [*simp*]: $\lfloor$*of-nat* $n\rfloor =$ *int* n
  using *floor-of-int* [*of of-nat* n] **by** *simp*

lemma *le-floor-add*: $\lfloor x\rfloor + \lfloor y\rfloor \le \lfloor x + y\rfloor$
by (*simp only*: *le-floor-iff of-int-add add-mono of-int-floor-le*)

Floor with numerals.

lemma *floor-zero* [*simp*]: $\lfloor 0\rfloor = 0$
using *floor-of-int* [*of 0*] **by** *simp*

lemma *floor-one* [*simp*]: $\lfloor 1\rfloor = 1$
using *floor-of-int* [*of 1*] **by** *simp*

lemma *floor-numeral* [*simp*]: $\lfloor \mathit{numeral}\ v\rfloor = \mathit{numeral}\ v$
using *floor-of-int* [*of numeral v*] **by** *simp*

lemma *floor-neg-numeral* [*simp*]: $\lfloor - \mathit{numeral}\ v\rfloor = - \mathit{numeral}\ v$
using *floor-of-int* [*of* $-$ *numeral v*] **by** *simp*

lemma *zero-le-floor* [*simp*]: $0 \le \lfloor x\rfloor \longleftrightarrow 0 \le x$
by (*simp add*: *le-floor-iff*)

lemma *one-le-floor* [*simp*]: $1 \le \lfloor x\rfloor \longleftrightarrow 1 \le x$
by (*simp add*: *le-floor-iff*)

lemma *numeral-le-floor* [*simp*]: $\mathit{numeral}\ v \le \lfloor x\rfloor \longleftrightarrow \mathit{numeral}\ v \le x$
by (*simp add*: *le-floor-iff*)

lemma *neg-numeral-le-floor* [*simp*]: $- \mathit{numeral}\ v \le \lfloor x\rfloor \longleftrightarrow - \mathit{numeral}\ v \le x$
by (*simp add*: *le-floor-iff*)

lemma *zero-less-floor* [*simp*]: $0 < \lfloor x\rfloor \longleftrightarrow 1 \le x$
by (*simp add*: *less-floor-iff*)

lemma *one-less-floor* [*simp*]: $1 < \lfloor x\rfloor \longleftrightarrow 2 \le x$
by (*simp add*: *less-floor-iff*)

lemma *numeral-less-floor* [*simp*]: $\mathit{numeral}\ v < \lfloor x\rfloor \longleftrightarrow \mathit{numeral}\ v + 1 \le x$
by (*simp add*: *less-floor-iff*)

lemma *neg-numeral-less-floor* [*simp*]: $- \mathit{numeral}\ v < \lfloor x\rfloor \longleftrightarrow - \mathit{numeral}\ v + 1 \le x$
by (*simp add*: *less-floor-iff*)

lemma *floor-le-zero* [*simp*]: $\lfloor x\rfloor \le 0 \longleftrightarrow x < 1$
by (*simp add*: *floor-le-iff*)

lemma *floor-le-one* [*simp*]: $\lfloor x\rfloor \le 1 \longleftrightarrow x < 2$
by (*simp add*: *floor-le-iff*)

lemma *floor-le-numeral* [*simp*]: $\lfloor x\rfloor \le \mathit{numeral}\ v \longleftrightarrow x < \mathit{numeral}\ v + 1$

by (*simp add*: *floor-le-iff*)

lemma *floor-le-neg-numeral* [*simp*]: $\lfloor x \rfloor \leq -$ *numeral* $v \longleftrightarrow x < -$ *numeral* v + *1*
by (*simp add*: *floor-le-iff*)

lemma *floor-less-zero* [*simp*]: $\lfloor x \rfloor < 0 \longleftrightarrow x < 0$
by (*simp add*: *floor-less-iff*)

lemma *floor-less-one* [*simp*]: $\lfloor x \rfloor < 1 \longleftrightarrow x < 1$
by (*simp add*: *floor-less-iff*)

lemma *floor-less-numeral* [*simp*]: $\lfloor x \rfloor <$ *numeral* $v \longleftrightarrow x <$ *numeral* v
by (*simp add*: *floor-less-iff*)

lemma *floor-less-neg-numeral* [*simp*]: $\lfloor x \rfloor < -$ *numeral* $v \longleftrightarrow x < -$ *numeral* v
by (*simp add*: *floor-less-iff*)

lemma *le-mult-floor-Ints*:
assumes $0 \leq a$ $a \in$ *Ints*
shows *of-int* $(\lfloor a \rfloor * \lfloor b \rfloor) \leq$ (*of-int*$\lfloor a * b \rfloor$ :: *'a* :: *linordered-idom*)
by (*metis Ints-cases assms floor-less-iff floor-of-int linorder-not-less mult-left-mono of-int-floor-le of-int-less-iff of-int-mult*)

Addition and subtraction of integers.

lemma *floor-add-int*: $\lfloor x \rfloor + z = \lfloor x +$ *of-int* $z \rfloor$
using *floor-correct* [*of x*] **by** (*simp add*: *floor-unique*[*symmetric*])

lemma *int-add-floor*: $z + \lfloor x \rfloor = \lfloor$*of-int* $z + x \rfloor$
using *floor-correct* [*of x*] **by** (*simp add*: *floor-unique*[*symmetric*])

lemma *one-add-floor*: $\lfloor x \rfloor + 1 = \lfloor x + 1 \rfloor$
using *floor-add-int* [*of x 1*] **by** *simp*

lemma *floor-diff-of-int* [*simp*]: $\lfloor x -$ *of-int* $z \rfloor = \lfloor x \rfloor - z$
using *floor-add-int* [*of x* $- z$] **by** (*simp add*: *algebra-simps*)

lemma *floor-uminus-of-int* [*simp*]: $\lfloor -$ (*of-int* z)$\rfloor = - z$
by (*metis floor-diff-of-int* [*of 0*] *diff-0 floor-zero*)

lemma *floor-diff-numeral* [*simp*]: $\lfloor x -$ *numeral* $v \rfloor = \lfloor x \rfloor -$ *numeral* v
using *floor-diff-of-int* [*of x numeral v*] **by** *simp*

lemma *floor-diff-one* [*simp*]: $\lfloor x - 1 \rfloor = \lfloor x \rfloor - 1$
using *floor-diff-of-int* [*of x 1*] **by** *simp*

lemma *le-mult-floor*:
assumes $0 \leq a$ **and** $0 \leq b$
shows $\lfloor a \rfloor * \lfloor b \rfloor \leq \lfloor a * b \rfloor$

proof $-$
have $\textit{of-int}\ \lfloor a \rfloor \leq a$ **and** $\textit{of-int}\ \lfloor b \rfloor \leq b$
by ($\textit{auto intro}$: $\textit{of-int-floor-le}$)
then have $\textit{of-int}\ (\lfloor a \rfloor * \lfloor b \rfloor) \leq a * b$
using $\textit{assms}$ **by** ($\textit{auto intro}$!: $\textit{mult-mono}$)
also have $a * b < \textit{of-int}\ (\lfloor a * b \rfloor + \mathit{1})$
using $\textit{floor-correct}[\textit{of}\ a * b]$ **by** $\textit{auto}$
finally show $\textit{?thesis}$
unfolding $\textit{of-int-less-iff}$ **by** $\textit{simp}$
qed

lemma $\textit{floor-divide-of-int-eq}$: $\lfloor \textit{of-int}\ k\ /\ \textit{of-int}\ l \rfloor = k\ \textit{div}\ l$
for $k\ l :: \textit{int}$
proof ($\textit{cases}\ l = \mathit{0}$)
case $\textit{True}$
then show $\textit{?thesis}$ **by** $\textit{simp}$
next
case $\textit{False}$
have $*$: $\lfloor \textit{of-int}\ (k\ \textit{mod}\ l)\ /\ \textit{of-int}\ l :: {'a} \rfloor = \mathit{0}$
proof ($\textit{cases}\ l > \mathit{0}$)
case $\textit{True}$
then show $\textit{?thesis}$
by ($\textit{auto intro}$: $\textit{floor-unique}$)
next
case $\textit{False}$
obtain r **where** $r = -\ l$
by $\textit{blast}$
then have l: $l = -\ r$
by $\textit{simp}$
with $\langle l \neq \mathit{0} \rangle$ $\textit{False}$ **have** $r > \mathit{0}$
by $\textit{simp}$
with l **show** $\textit{?thesis}$
using $\textit{pos-mod-bound}\ [\textit{of}\ r]$
by ($\textit{auto simp add}$: $\textit{zmod-zminus2-eq-if less-le field-simps intro}$: $\textit{floor-unique}$)
qed
have $(\textit{of-int}\ k :: {'a}) = \textit{of-int}\ (k\ \textit{div}\ l * l + k\ \textit{mod}\ l)$
by $\textit{simp}$
also have $\ldots = (\textit{of-int}\ (k\ \textit{div}\ l) + \textit{of-int}\ (k\ \textit{mod}\ l)\ /\ \textit{of-int}\ l) * \textit{of-int}\ l$
using $\textit{False}$ **by** ($\textit{simp only}$: $\textit{of-int-add}$) ($\textit{simp add}$: $\textit{field-simps}$)
finally have $(\textit{of-int}\ k\ /\ \textit{of-int}\ l :: {'a}) = \ldots\ /\ \textit{of-int}\ l$
by $\textit{simp}$
then have $(\textit{of-int}\ k\ /\ \textit{of-int}\ l :: {'a}) = \textit{of-int}\ (k\ \textit{div}\ l) + \textit{of-int}\ (k\ \textit{mod}\ l)\ /\ \textit{of-int}\ l$
using $\textit{False}$ **by** ($\textit{simp only}$:) ($\textit{simp add}$: $\textit{field-simps}$)
then have $\lfloor \textit{of-int}\ k\ /\ \textit{of-int}\ l :: {'a} \rfloor = \lfloor \textit{of-int}\ (k\ \textit{div}\ l) + \textit{of-int}\ (k\ \textit{mod}\ l)\ /\ \textit{of-int}\ l :: {'a} \rfloor$
by $\textit{simp}$
then have $\lfloor \textit{of-int}\ k\ /\ \textit{of-int}\ l :: {'a} \rfloor = \lfloor \textit{of-int}\ (k\ \textit{mod}\ l)\ /\ \textit{of-int}\ l + \textit{of-int}\ (k\ \textit{div}\ l) :: {'a} \rfloor$

by (*simp add*: *ac-simps*)
then have $\lfloor$*of-int k* / *of-int l* :: *'a*$\rfloor$ = $\lfloor$*of-int* (*k mod l*) / *of-int l* :: *'a*$\rfloor$ + *k div l*
by (*simp add*: *floor-add-int*)
with ∗ **show** *?thesis*
by *simp*
qed

lemma *floor-divide-of-nat-eq*: $\lfloor$*of-nat m* / *of-nat n*$\rfloor$ = *of-nat* (*m div n*)
for *m n* :: *nat*
proof (*cases n* = *0*)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
then have ∗: $\lfloor$*of-nat* (*m mod n*) / *of-nat n* :: *'a*$\rfloor$ = *0*
by (*auto intro*: *floor-unique*)
have (*of-nat m* :: *'a*) = *of-nat* (*m div n* ∗ *n* + *m mod n*)
by *simp*
also have . . . = (*of-nat* (*m div n*) + *of-nat* (*m mod n*) / *of-nat n*) ∗ *of-nat n*
using *False* **by** (*simp only*: *of-nat-add*) (*simp add*: *field-simps*)
finally have (*of-nat m* / *of-nat n* :: *'a*) = . . . / *of-nat n*
by *simp*
then have (*of-nat m* / *of-nat n* :: *'a*) = *of-nat* (*m div n*) + *of-nat* (*m mod n*) / *of-nat n*
using *False* **by** (*simp only*:) *simp*
then have $\lfloor$*of-nat m* / *of-nat n* :: *'a*$\rfloor$ = $\lfloor$*of-nat* (*m div n*) + *of-nat* (*m mod n*) / *of-nat n* :: *'a*$\rfloor$
by *simp*
then have $\lfloor$*of-nat m* / *of-nat n* :: *'a*$\rfloor$ = $\lfloor$*of-nat* (*m mod n*) / *of-nat n* + *of-nat* (*m div n*) :: *'a*$\rfloor$
by (*simp add*: *ac-simps*)
moreover have (*of-nat* (*m div n*) :: *'a*) = *of-int* (*of-nat* (*m div n*))
by *simp*
ultimately have $\lfloor$*of-nat m* / *of-nat n* :: *'a*$\rfloor$ =
$\lfloor$*of-nat* (*m mod n*) / *of-nat n* :: *'a*$\rfloor$ + *of-nat* (*m div n*)
by (*simp only*: *floor-add-int*)
with ∗ **show** *?thesis*
by *simp*
qed

lemma *floor-divide-lower*:
fixes *q* :: *'a::floor-ceiling*
shows *q* > *0* ⟹ *of-int* $\lfloor$*p* / *q*$\rfloor$ ∗ *q* ≤ *p*
using *of-int-floor-le pos-le-divide-eq* **by** *blast*

lemma *floor-divide-upper*:
fixes *q* :: *'a::floor-ceiling*
shows *q* > *0* ⟹ *p* < (*of-int* $\lfloor$*p* / *q*$\rfloor$ + *1*) ∗ *q*
by (*meson floor-eq-iff pos-divide-less-eq*)

6.4 Ceiling function

definition *ceiling* :: *'a::floor-ceiling* ⇒ *int* (⌈-⌉)
where $\lceil x \rceil = - \lfloor - x \rfloor$

lemma *ceiling-correct*: *of-int* $\lceil x \rceil - 1 < x \wedge x \leq$ *of-int* $\lceil x \rceil$
unfolding *ceiling-def* **using** *floor-correct* [*of* $- x$]
by (*simp add*: *le-minus-iff*)

lemma *ceiling-unique*: *of-int* $z - 1 < x \Longrightarrow x \leq$ *of-int* $z \Longrightarrow \lceil x \rceil = z$
unfolding *ceiling-def* **using** *floor-unique* [*of* $- z - x$] **by** *simp*

lemma *ceiling-eq-iff*: $\lceil x \rceil = a \longleftrightarrow$ *of-int* $a - 1 < x \wedge x \leq$ *of-int* a
using *ceiling-correct ceiling-unique* **by** *auto*

lemma *le-of-int-ceiling* [*simp*]: $x \leq$ *of-int* $\lceil x \rceil$
using *ceiling-correct* **..**

lemma *ceiling-le-iff*: $\lceil x \rceil \leq z \longleftrightarrow x \leq$ *of-int* z
unfolding *ceiling-def* **using** *le-floor-iff* [*of* $- z - x$] **by** *auto*

lemma *less-ceiling-iff*: $z < \lceil x \rceil \longleftrightarrow$ *of-int* $z < x$
by (*simp add*: *not-le* [*symmetric*] *ceiling-le-iff*)

lemma *ceiling-less-iff*: $\lceil x \rceil < z \longleftrightarrow x \leq$ *of-int* $z - 1$
using *ceiling-le-iff* [*of* x $z - 1$] **by** *simp*

lemma *le-ceiling-iff*: $z \leq \lceil x \rceil \longleftrightarrow$ *of-int* $z - 1 < x$
by (*simp add*: *not-less* [*symmetric*] *ceiling-less-iff*)

lemma *ceiling-mono*: $x \geq y \Longrightarrow \lceil x \rceil \geq \lceil y \rceil$
unfolding *ceiling-def* **by** (*simp add*: *floor-mono*)

lemma *ceiling-less-cancel*: $\lceil x \rceil < \lceil y \rceil \Longrightarrow x < y$
by (*auto simp add*: *not-le* [*symmetric*] *ceiling-mono*)

lemma *ceiling-of-int* [*simp*]: ⌈*of-int* z⌉ $= z$
by (*rule ceiling-unique*) *simp-all*

lemma *ceiling-of-nat* [*simp*]: ⌈*of-nat* n⌉ $=$ *int* n
using *ceiling-of-int* [*of of-nat* n] **by** *simp*

lemma *ceiling-add-le*: $\lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$
by (*simp only*: *ceiling-le-iff of-int-add add-mono le-of-int-ceiling*)

lemma *mult-ceiling-le*:
assumes $0 \leq a$ **and** $0 \leq b$
shows $\lceil a * b \rceil \leq \lceil a \rceil * \lceil b \rceil$
by (*metis assms ceiling-le-iff ceiling-mono le-of-int-ceiling mult-mono of-int-mult*)

lemma *mult-ceiling-le-Ints*:
assumes $0 \leq a$ $a \in$ *Ints*
shows (*of-int* $\lceil a * b \rceil$ :: $'a$ :: *linordered-idom*) $\leq$ *of-int*($\lceil a \rceil * \lceil b \rceil$)
by (*metis Ints-cases assms ceiling-le-iff ceiling-of-int le-of-int-ceiling mult-left-mono of-int-le-iff of-int-mult*)

lemma *finite-int-segment*:
fixes a :: $'a$::*floor-ceiling*
shows *finite* $\{x \in \mathbb{Z}.\ a \leq x \wedge x \leq b\}$
proof −
have *finite* {*ceiling a..floor b*}
by *simp*
moreover have $\{x \in \mathbb{Z}.\ a \leq x \wedge x \leq b\}$ = *of-int* ' {*ceiling a..floor b*}
by (*auto simp*: *le-floor-iff ceiling-le-iff elim*!: *Ints-cases*)
ultimately show *?thesis*
by *simp*
qed

corollary *finite-abs-int-segment*:
fixes a :: $'a$::*floor-ceiling*
shows *finite* $\{k \in \mathbb{Z}.\ |k| \leq a\}$
using *finite-int-segment* [*of* $-a$ a] **by** (*auto simp add*: *abs-le-iff conj-commute minus-le-iff*)

6.4.1 Ceiling with numerals.

lemma *ceiling-zero* [*simp*]: $\lceil 0 \rceil = 0$
using *ceiling-of-int* [*of 0*] **by** *simp*

lemma *ceiling-one* [*simp*]: $\lceil 1 \rceil = 1$
using *ceiling-of-int* [*of 1*] **by** *simp*

lemma *ceiling-numeral* [*simp*]: $\lceil$*numeral v*$\rceil$ = *numeral v*
using *ceiling-of-int* [*of numeral v*] **by** *simp*

lemma *ceiling-neg-numeral* [*simp*]: $\lceil-$ *numeral v*$\rceil$ = $-$ *numeral v*
using *ceiling-of-int* [*of* $-$ *numeral v*] **by** *simp*

lemma *ceiling-le-zero* [*simp*]: $\lceil x \rceil \leq 0 \longleftrightarrow x \leq 0$
by (*simp add*: *ceiling-le-iff*)

lemma *ceiling-le-one* [*simp*]: $\lceil x \rceil \leq 1 \longleftrightarrow x \leq 1$
by (*simp add*: *ceiling-le-iff*)

lemma *ceiling-le-numeral* [*simp*]: $\lceil x \rceil \leq$ *numeral v* $\longleftrightarrow x \leq$ *numeral v*
by (*simp add*: *ceiling-le-iff*)

lemma *ceiling-le-neg-numeral* [*simp*]: $\lceil x \rceil \leq -$ *numeral v* $\longleftrightarrow x \leq -$ *numeral v*
by (*simp add*: *ceiling-le-iff*)

lemma *ceiling-less-zero* [*simp*]: $\lceil x \rceil < 0 \longleftrightarrow x \leq -1$
by (*simp add*: *ceiling-less-iff*)

lemma *ceiling-less-one* [*simp*]: $\lceil x \rceil < 1 \longleftrightarrow x \leq 0$
by (*simp add*: *ceiling-less-iff*)

lemma *ceiling-less-numeral* [*simp*]: $\lceil x \rceil <$ *numeral* $v \longleftrightarrow x \leq$ *numeral* $v - 1$
by (*simp add*: *ceiling-less-iff*)

lemma *ceiling-less-neg-numeral* [*simp*]: $\lceil x \rceil < -$ *numeral* $v \longleftrightarrow x \leq -$ *numeral* $v - 1$
by (*simp add*: *ceiling-less-iff*)

lemma *zero-le-ceiling* [*simp*]: $0 \leq \lceil x \rceil \longleftrightarrow -1 < x$
by (*simp add*: *le-ceiling-iff*)

lemma *one-le-ceiling* [*simp*]: $1 \leq \lceil x \rceil \longleftrightarrow 0 < x$
by (*simp add*: *le-ceiling-iff*)

lemma *numeral-le-ceiling* [*simp*]: *numeral* $v \leq \lceil x \rceil \longleftrightarrow$ *numeral* $v - 1 < x$
by (*simp add*: *le-ceiling-iff*)

lemma *neg-numeral-le-ceiling* [*simp*]: $-$ *numeral* $v \leq \lceil x \rceil \longleftrightarrow -$ *numeral* $v - 1 < x$
by (*simp add*: *le-ceiling-iff*)

lemma *zero-less-ceiling* [*simp*]: $0 < \lceil x \rceil \longleftrightarrow 0 < x$
by (*simp add*: *less-ceiling-iff*)

lemma *one-less-ceiling* [*simp*]: $1 < \lceil x \rceil \longleftrightarrow 1 < x$
by (*simp add*: *less-ceiling-iff*)

lemma *numeral-less-ceiling* [*simp*]: *numeral* $v < \lceil x \rceil \longleftrightarrow$ *numeral* $v < x$
by (*simp add*: *less-ceiling-iff*)

lemma *neg-numeral-less-ceiling* [*simp*]: $-$ *numeral* $v < \lceil x \rceil \longleftrightarrow -$ *numeral* $v < x$
by (*simp add*: *less-ceiling-iff*)

lemma *ceiling-altdef*: $\lceil x \rceil =$ (*if* $x =$ *of-int* $\lfloor x \rfloor$ *then* $\lfloor x \rfloor$ *else* $\lfloor x \rfloor + 1$)
by (*intro ceiling-unique*; *simp*, *linarith?*)

lemma *floor-le-ceiling* [*simp*]: $\lfloor x \rfloor \leq \lceil x \rceil$
by (*simp add*: *ceiling-altdef*)

6.4.2 Addition and subtraction of integers.

lemma *ceiling-add-of-int* [*simp*]: $\lceil x +$ *of-int* $z \rceil = \lceil x \rceil + z$
using *ceiling-correct* [*of x*] **by** (*simp add*: *ceiling-def*)

lemma *ceiling-add-numeral* [*simp*]: $\lceil x + numeral\ v\rceil = \lceil x\rceil + numeral\ v$
  using *ceiling-add-of-int* [*of x numeral v*] **by** *simp*

lemma *ceiling-add-one* [*simp*]: $\lceil x + 1\rceil = \lceil x\rceil + 1$
  using *ceiling-add-of-int* [*of x 1*] **by** *simp*

lemma *ceiling-diff-of-int* [*simp*]: $\lceil x - of\text{-}int\ z\rceil = \lceil x\rceil - z$
  using *ceiling-add-of-int* [*of x − z*] **by** (*simp add*: *algebra-simps*)

lemma *ceiling-diff-numeral* [*simp*]: $\lceil x - numeral\ v\rceil = \lceil x\rceil - numeral\ v$
  using *ceiling-diff-of-int* [*of x numeral v*] **by** *simp*

lemma *ceiling-diff-one* [*simp*]: $\lceil x - 1\rceil = \lceil x\rceil - 1$
  using *ceiling-diff-of-int* [*of x 1*] **by** *simp*

lemma *ceiling-split*[*arith-split*]: $P\ \lceil t\rceil \longleftrightarrow (\forall i.\ of\text{-}int\ i - 1 < t \wedge t \le of\text{-}int\ i \longrightarrow P\ i)$
  by (*auto simp add*: *ceiling-unique ceiling-correct*)

lemma *ceiling-diff-floor-le-1*: $\lceil x\rceil - \lfloor x\rfloor \le 1$
proof −
  have $of\text{-}int\ \lceil x\rceil - 1 < x$
    using *ceiling-correct*[*of x*] **by** *simp*
  also have $x < of\text{-}int\ \lfloor x\rfloor + 1$
    using *floor-correct*[*of x*] **by** *simp-all*
  finally have $of\text{-}int\ (\lceil x\rceil - \lfloor x\rfloor) < (of\text{-}int\ 2::'a)$
    by *simp*
  then show *?thesis*
    unfolding *of-int-less-iff* **by** *simp*
qed

lemma *nat-approx-posE*:
  fixes *e*:: *'a*::{*archimedean-field*,*floor-ceiling*}
  assumes $0 < e$
  obtains *n* :: *nat* **where** $1\ /\ of\text{-}nat(Suc\ n) < e$
proof
  have $(1::'a)\ /\ of\text{-}nat\ (Suc\ (nat\ \lceil 1/e\rceil)) < 1\ /\ of\text{-}int\ (\lceil 1/e\rceil)$
  proof (*rule divide-strict-left-mono*)
    show $(of\text{-}int\ \lceil 1\ /\ e\rceil::'a) < of\text{-}nat\ (Suc\ (nat\ \lceil 1\ /\ e\rceil))$
      using *assms* **by** (*simp add*: *field-simps*)
    show $(0::'a) < of\text{-}nat\ (Suc\ (nat\ \lceil 1\ /\ e\rceil)) * of\text{-}int\ \lceil 1\ /\ e\rceil$
      using *assms* **by** (*auto simp*: *zero-less-mult-iff pos-add-strict*)
  qed *auto*
  also have $1\ /\ of\text{-}int\ (\lceil 1/e\rceil) \le 1\ /\ (1/e)$
    by (*rule divide-left-mono*) (*auto simp*: ‹$0 < e$› *ceiling-correct*)
  also have $\ldots = e$ **by** *simp*
  finally show $1\ /\ of\text{-}nat\ (Suc\ (nat\ \lceil 1\ /\ e\rceil)) < e$
    by *metis*

qed

lemma *ceiling-divide-upper*:
fixes $q :: 'a::floor\text{-}ceiling$
shows $q > 0 \Longrightarrow p \le of\text{-}int\ (ceiling\ (p\ /\ q)) * q$
by (*meson divide-le-eq le-of-int-ceiling*)

lemma *ceiling-divide-lower*:
fixes $q :: 'a::floor\text{-}ceiling$
shows $q > 0 \Longrightarrow (of\text{-}int\ \lceil p\ /\ q \rceil - 1) * q < p$
by (*meson ceiling-eq-iff pos-less-divide-eq*)

6.5 Negation

lemma *floor-minus*: $\lfloor - x \rfloor = - \lceil x \rceil$
unfolding *ceiling-def* **by** *simp*

lemma *ceiling-minus*: $\lceil - x \rceil = - \lfloor x \rfloor$
unfolding *ceiling-def* **by** *simp*

6.6 Natural numbers

lemma *of-nat-floor*: $r \ge 0 \Longrightarrow of\text{-}nat\ (nat\ \lfloor r \rfloor) \le r$
by *simp*

lemma *of-nat-ceiling*: $of\text{-}nat\ (nat\ \lceil r \rceil) \ge r$
by (*cases* $r \ge 0$) *auto*

6.7 Frac Function

definition $frac :: 'a \Rightarrow 'a::floor\text{-}ceiling$
where $frac\ x \equiv x - of\text{-}int\ \lfloor x \rfloor$

lemma *frac-lt-1*: $frac\ x < 1$
by (*simp add*: *frac-def*) *linarith*

lemma *frac-eq-0-iff* [*simp*]: $frac\ x = 0 \longleftrightarrow x \in \mathbb{Z}$
by (*simp add*: *frac-def*) (*metis Ints-cases Ints-of-int floor-of-int*)

lemma *frac-ge-0* [*simp*]: $frac\ x \ge 0$
unfolding *frac-def* **by** *linarith*

lemma *frac-gt-0-iff* [*simp*]: $frac\ x > 0 \longleftrightarrow x \notin \mathbb{Z}$
by (*metis frac-eq-0-iff frac-ge-0 le-less less-irrefl*)

lemma *frac-of-int* [*simp*]: $frac\ (of\text{-}int\ z) = 0$
by (*simp add*: *frac-def*)

lemma *frac-frac* [*simp*]: $frac\ (frac\ x) = frac\ x$
by (*simp add*: *frac-def*)

lemma *floor-add*: $\lfloor x + y \rfloor = (\mathit{if}\ \mathit{frac}\ x + \mathit{frac}\ y < 1\ \mathit{then}\ \lfloor x \rfloor + \lfloor y \rfloor\ \mathit{else}\ (\lfloor x \rfloor + \lfloor y \rfloor) + 1)$
proof −
  have $x + y < 1 + (\mathit{of\text{-}int}\ \lfloor x \rfloor + \mathit{of\text{-}int}\ \lfloor y \rfloor) \Longrightarrow \lfloor x + y \rfloor = \lfloor x \rfloor + \lfloor y \rfloor$
    by (*metis add.commute floor-unique le-floor-add le-floor-iff of-int-add*)
  moreover
  have $\neg\ x + y < 1 + (\mathit{of\text{-}int}\ \lfloor x \rfloor + \mathit{of\text{-}int}\ \lfloor y \rfloor) \Longrightarrow \lfloor x + y \rfloor = 1 + (\lfloor x \rfloor + \lfloor y \rfloor)$
    apply (*simp add*: *floor-eq-iff*)
    apply (*auto simp add*: *algebra-simps*)
    apply *linarith*
    done
  ultimately show *?thesis* **by** (*auto simp add*: *frac-def algebra-simps*)
qed

lemma *floor-add2*[*simp*]: $x \in \mathbb{Z} \lor y \in \mathbb{Z} \Longrightarrow \lfloor x + y \rfloor = \lfloor x \rfloor + \lfloor y \rfloor$
by (*metis add.commute add.left-neutral frac-lt-1 floor-add frac-eq-0-iff*)

lemma *frac-add*:
  $\mathit{frac}\ (x + y) = (\mathit{if}\ \mathit{frac}\ x + \mathit{frac}\ y < 1\ \mathit{then}\ \mathit{frac}\ x + \mathit{frac}\ y\ \mathit{else}\ (\mathit{frac}\ x + \mathit{frac}\ y) - 1)$
  by (*simp add*: *frac-def floor-add*)

lemma *frac-unique-iff*: $\mathit{frac}\ x = a \longleftrightarrow x - a \in \mathbb{Z} \land 0 \le a \land a < 1$
  for $x :: 'a{::}\mathit{floor\text{-}ceiling}$
  apply (*auto simp*: *Ints-def frac-def algebra-simps floor-unique*)
   apply *linarith+*
  done

lemma *frac-eq*: $\mathit{frac}\ x = x \longleftrightarrow 0 \le x \land x < 1$
  by (*simp add*: *frac-unique-iff*)

lemma *frac-neg*: $\mathit{frac}\ (-\ x) = (\mathit{if}\ x \in \mathbb{Z}\ \mathit{then}\ 0\ \mathit{else}\ 1 - \mathit{frac}\ x)$
  for $x :: 'a{::}\mathit{floor\text{-}ceiling}$
  apply (*auto simp add*: *frac-unique-iff*)
   apply (*simp add*: *frac-def*)
  apply (*meson frac-lt-1 less-iff-diff-less-0 not-le not-less-iff-gr-or-eq*)
  done

lemma *frac-in-Ints-iff* [*simp*]: $\mathit{frac}\ x \in \mathbb{Z} \longleftrightarrow x \in \mathbb{Z}$
proof *safe*
  assume $\mathit{frac}\ x \in \mathbb{Z}$
  hence $\mathit{of\text{-}int}\ \lfloor x \rfloor + \mathit{frac}\ x \in \mathbb{Z}$ **by** *auto*
  also have $\mathit{of\text{-}int}\ \lfloor x \rfloor + \mathit{frac}\ x = x$ **by** (*simp add*: *frac-def*)
  finally show $x \in \mathbb{Z}$ **.**
qed (*auto simp*: *frac-def*)

6.8 Rounding to the nearest integer

definition *round* :: *'a::floor-ceiling* ⇒ *int*
 where *round* $x = \lfloor x + 1/2 \rfloor$

lemma *of-int-round-ge*: *of-int* (*round* x) $\geq x - 1/2$
 and *of-int-round-le*: *of-int* (*round* x) $\leq x + 1/2$
 and *of-int-round-abs-le*: |*of-int* (*round* x) $- x| \leq 1/2$
 and *of-int-round-gt*: *of-int* (*round* x) $> x - 1/2$
proof −
 from *floor-correct*[*of* $x + 1/2$] **have** $x + 1/2 <$ *of-int* (*round* x) $+ 1$
  by (*simp add*: *round-def*)
 from *add-strict-right-mono*[*OF this*, *of* -1] **show** *A*: *of-int* (*round* x) $> x - 1/2$
  by *simp*
 then show *of-int* (*round* x) $\geq x - 1/2$
  by *simp*
 from *floor-correct*[*of* $x + 1/2$] **show** *of-int* (*round* x) $\leq x + 1/2$
  by (*simp add*: *round-def*)
 with *A* **show** |*of-int* (*round* x) $- x| \leq 1/2$
  by *linarith*
qed

lemma *round-of-int* [*simp*]: *round* (*of-int* n) $= n$
 unfolding *round-def* **by** (*subst floor-eq-iff*) *force*

lemma *round-0* [*simp*]: *round* $0 = 0$
 using *round-of-int*[*of* 0] **by** *simp*

lemma *round-1* [*simp*]: *round* $1 = 1$
 using *round-of-int*[*of* 1] **by** *simp*

lemma *round-numeral* [*simp*]: *round* (*numeral* n) = *numeral* n
 using *round-of-int*[*of numeral* n] **by** *simp*

lemma *round-neg-numeral* [*simp*]: *round* (−*numeral* n) = −*numeral* n
 using *round-of-int*[*of* −*numeral* n] **by** *simp*

lemma *round-of-nat* [*simp*]: *round* (*of-nat* n) = *of-nat* n
 using *round-of-int*[*of int* n] **by** *simp*

lemma *round-mono*: $x \leq y \Longrightarrow$ *round* $x \leq$ *round* y
 unfolding *round-def* **by** (*intro floor-mono*) *simp*

lemma *round-unique*: *of-int* $y > x - 1/2 \Longrightarrow$ *of-int* $y \leq x + 1/2 \Longrightarrow$ *round* $x = y$
 unfolding *round-def*
proof (*rule floor-unique*)
 assume $x - 1 / 2 <$ *of-int* y
 from *add-strict-left-mono*[*OF this*, *of* 1] **show** $x + 1 / 2 <$ *of-int* $y + 1$

by *simp*
qed

lemma *round-unique'*: $|x - \textit{of-int } n| < 1/2 \Longrightarrow \textit{round } x = n$
by (*subst* (*asm*) *abs-less-iff*, *rule round-unique*) (*simp-all add*: *field-simps*)

lemma *round-altdef*: $\textit{round } x = (\textit{if frac } x \geq 1/2 \textit{ then } \lceil x \rceil \textit{ else } \lfloor x \rfloor)$
by (*cases frac* $x \geq 1/2$)
(*rule round-unique*, ((*simp add*: *frac-def field-simps ceiling-altdef*; *linarith*)+)[*2*])+

lemma *floor-le-round*: $\lfloor x \rfloor \leq \textit{round } x$
unfolding *round-def* **by** (*intro floor-mono*) *simp*

lemma *ceiling-ge-round*: $\lceil x \rceil \geq \textit{round } x$
unfolding *round-altdef* **by** *simp*

lemma *round-diff-minimal*: $|z - \textit{of-int } (\textit{round } z)| \leq |z - \textit{of-int } m|$
for *z* :: *'a*::*floor-ceiling*
proof (*cases of-int* $m \geq z$)
case *True*
then have $|z - \textit{of-int } (\textit{round } z)| \leq |\textit{of-int } \lceil z \rceil - z|$
unfolding *round-altdef* **by** (*simp add*: *field-simps ceiling-altdef frac-def*) *linarith*
also have $\textit{of-int } \lceil z \rceil - z \geq 0$
by *linarith*
with *True* **have** $|\textit{of-int } \lceil z \rceil - z| \leq |z - \textit{of-int } m|$
by (*simp add*: *ceiling-le-iff*)
finally show *?thesis* **.**
next
case *False*
then have $|z - \textit{of-int } (\textit{round } z)| \leq |\textit{of-int } \lfloor z \rfloor - z|$
unfolding *round-altdef* **by** (*simp add*: *field-simps ceiling-altdef frac-def*) *linarith*
also have $z - \textit{of-int } \lfloor z \rfloor \geq 0$
by *linarith*
with *False* **have** $|\textit{of-int } \lfloor z \rfloor - z| \leq |z - \textit{of-int } m|$
by (*simp add*: *le-floor-iff*)
finally show *?thesis* **.**
qed

end

7 Rational numbers

theory *Rat*
imports *Archimedean-Field*
begin

7.1 Rational numbers as quotient

7.1.1 Construction of the type of rational numbers

definition *ratrel* :: $(int \times int) \Rightarrow (int \times int) \Rightarrow bool$
where $ratrel = (\lambda x\ y.\ snd\ x \neq 0 \wedge snd\ y \neq 0 \wedge fst\ x * snd\ y = fst\ y * snd\ x)$

lemma *ratrel-iff* [*simp*]: $ratrel\ x\ y \longleftrightarrow snd\ x \neq 0 \wedge snd\ y \neq 0 \wedge fst\ x * snd\ y = fst\ y * snd\ x$
by (*simp add*: *ratrel-def*)

lemma *exists-ratrel-refl*: $\exists x.\ ratrel\ x\ x$
by (*auto intro*!: *one-neq-zero*)

lemma *symp-ratrel*: *symp ratrel*
by (*simp add*: *ratrel-def symp-def*)

lemma *transp-ratrel*: *transp ratrel*
proof (*rule transpI*, *unfold split-paired-all*)
fix $a\ b\ a'\ b'\ a''\ b''$ :: *int*
assume $*$: $ratrel\ (a, b)\ (a', b')$
assume $**$: $ratrel\ (a', b')\ (a'', b'')$
have $b' * (a * b'') = b'' * (a * b')$ **by** *simp*
also from $*$ **have** $a * b' = a' * b$ **by** *auto*
also have $b'' * (a' * b) = b * (a' * b'')$ **by** *simp*
also from $**$ **have** $a' * b'' = a'' * b'$ **by** *auto*
also have $b * (a'' * b') = b' * (a'' * b)$ **by** *simp*
finally have $b' * (a * b'') = b' * (a'' * b)$ **.**
moreover from $**$ **have** $b' \neq 0$ **by** *auto*
ultimately have $a * b'' = a'' * b$ **by** *simp*
with $*$ $**$ **show** $ratrel\ (a, b)\ (a'', b'')$ **by** *auto*
qed

lemma *part-equivp-ratrel*: *part-equivp ratrel*
by (*rule part-equivpI* [*OF exists-ratrel-refl symp-ratrel transp-ratrel*])

quotient-type $rat = int \times int$ / *partial*: *ratrel*
morphisms *Rep-Rat Abs-Rat*
by (*rule part-equivp-ratrel*)

lemma *Domainp-cr-rat* [*transfer-domain-rule*]: $Domainp\ pcr\text{-}rat = (\lambda x.\ snd\ x \neq 0)$
by (*simp add*: *rat.domain-eq*)

7.1.2 Representation and basic operations

lift-definition *Fract* :: $int \Rightarrow int \Rightarrow rat$
is $\lambda a\ b.\ if\ b = 0\ then\ (0, 1)\ else\ (a, b)$
by *simp*

lemma *eq-rat*:
$\bigwedge a\ b\ c\ d.\ b \neq 0 \Longrightarrow d \neq 0 \Longrightarrow \mathit{Fract}\ a\ b = \mathit{Fract}\ c\ d \longleftrightarrow a * d = c * b$
$\bigwedge a.\ \mathit{Fract}\ a\ 0 = \mathit{Fract}\ 0\ 1$
$\bigwedge a\ c.\ \mathit{Fract}\ 0\ a = \mathit{Fract}\ 0\ c$
by (*transfer*, *simp*)+

lemma *Rat-cases* [*case-names Fract*, *cases type*: *rat*]:
assumes *that*: $\bigwedge a\ b.\ q = \mathit{Fract}\ a\ b \Longrightarrow b > 0 \Longrightarrow \mathit{coprime}\ a\ b \Longrightarrow C$
shows *C*
proof −
obtain *a b* :: *int* **where** *q*: $q = \mathit{Fract}\ a\ b$ **and** *b*: $b \neq 0$
by *transfer simp*
let $?a = a\ \mathit{div}\ \mathit{gcd}\ a\ b$
let $?b = b\ \mathit{div}\ \mathit{gcd}\ a\ b$
from *b* **have** $?b * \mathit{gcd}\ a\ b = b$
by *simp*
with *b* **have** $?b \neq 0$
by *fastforce*
with *q b* **have** *q2*: $q = \mathit{Fract}\ ?a\ ?b$
by (*simp add*: *eq-rat dvd-div-mult mult.commute* [*of a*])
from *b* **have** *coprime*: *coprime ?a ?b*
by (*auto intro*: *div-gcd-coprime*)
show *C*
proof (*cases* $b > 0$)
case *True*
then have $?b > 0$
by (*simp add*: *nonneg1-imp-zdiv-pos-iff*)
from *q2 this coprime* **show** *C* **by** (*rule that*)
next
case *False*
have $q = \mathit{Fract}\ (-\ ?a)\ (-\ ?b)$
unfolding *q2* **by** *transfer simp*
moreover from *False b* **have** $-\ ?b > 0$
by (*simp add*: *pos-imp-zdiv-neg-iff*)
moreover from *coprime* **have** *coprime* $(-\ ?a)\ (-\ ?b)$
by *simp*
ultimately show *C*
by (*rule that*)
qed
qed

lemma *Rat-induct* [*case-names Fract*, *induct type*: *rat*]:
assumes $\bigwedge a\ b.\ b > 0 \Longrightarrow \mathit{coprime}\ a\ b \Longrightarrow P\ (\mathit{Fract}\ a\ b)$
shows *P q*
using *assms* **by** (*cases q*) *simp*

instantiation *rat* :: *field*
begin

lift-definition *zero-rat* :: *rat* **is** $(0, 1)$
by *simp*

lift-definition *one-rat* :: *rat* **is** $(1, 1)$
by *simp*

lemma *Zero-rat-def*: $0 = Fract\ 0\ 1$
by *transfer simp*

lemma *One-rat-def*: $1 = Fract\ 1\ 1$
by *transfer simp*

lift-definition *plus-rat* :: $rat \Rightarrow rat \Rightarrow rat$
is $\lambda x\ y.\ (fst\ x * snd\ y + fst\ y * snd\ x,\ snd\ x * snd\ y)$
by (*auto simp*: *distrib-right*) (*simp add*: *ac-simps*)

lemma *add-rat* [*simp*]:
assumes $b \neq 0$ **and** $d \neq 0$
shows $Fract\ a\ b + Fract\ c\ d = Fract\ (a * d + c * b)\ (b * d)$
using *assms* **by** *transfer simp*

lift-definition *uminus-rat* :: $rat \Rightarrow rat$ **is** $\lambda x.\ (-\ fst\ x,\ snd\ x)$
by *simp*

lemma *minus-rat* [*simp*]: $-\ Fract\ a\ b = Fract\ (-\ a)\ b$
by *transfer simp*

lemma *minus-rat-cancel* [*simp*]: $Fract\ (-\ a)\ (-\ b) = Fract\ a\ b$
by $(cases\ b = 0)$ (*simp-all add*: *eq-rat*)

definition *diff-rat-def*: $q - r = q + -\ r$ **for** $q\ r$ :: *rat*

lemma *diff-rat* [*simp*]:
$b \neq 0 \Longrightarrow d \neq 0 \Longrightarrow Fract\ a\ b - Fract\ c\ d = Fract\ (a * d - c * b)\ (b * d)$
by (*simp add*: *diff-rat-def*)

lift-definition *times-rat* :: $rat \Rightarrow rat \Rightarrow rat$
is $\lambda x\ y.\ (fst\ x * fst\ y,\ snd\ x * snd\ y)$
by (*simp add*: *ac-simps*)

lemma *mult-rat* [*simp*]: $Fract\ a\ b * Fract\ c\ d = Fract\ (a * c)\ (b * d)$
by *transfer simp*

lemma *mult-rat-cancel*: $c \neq 0 \Longrightarrow Fract\ (c * a)\ (c * b) = Fract\ a\ b$
by *transfer simp*

lift-definition *inverse-rat* :: $rat \Rightarrow rat$
is $\lambda x.\ if\ fst\ x = 0\ then\ (0, 1)\ else\ (snd\ x,\ fst\ x)$
by (*auto simp add*: *mult.commute*)

lemma *inverse-rat* [*simp*]: *inverse* (*Fract a b*) = *Fract b a*
  by *transfer simp*

definition *divide-rat-def*: *q div r* = *q* ∗ *inverse r* **for** *q r* :: *rat*

lemma *divide-rat* [*simp*]: *Fract a b div Fract c d* = *Fract* (*a* ∗ *d*) (*b* ∗ *c*)
  by (*simp add*: *divide-rat-def*)

instance
proof
  fix *q r s* :: *rat*
  show (*q* ∗ *r*) ∗ *s* = *q* ∗ (*r* ∗ *s*)
    by *transfer simp*
  show *q* ∗ *r* = *r* ∗ *q*
    by *transfer simp*
  show *1* ∗ *q* = *q*
    by *transfer simp*
  show (*q* + *r*) + *s* = *q* + (*r* + *s*)
    by *transfer* (*simp add*: *algebra-simps*)
  show *q* + *r* = *r* + *q*
    by *transfer simp*
  show *0* + *q* = *q*
    by *transfer simp*
  show − *q* + *q* = *0*
    by *transfer simp*
  show *q* − *r* = *q* + − *r*
    by (*fact diff-rat-def*)
  show (*q* + *r*) ∗ *s* = *q* ∗ *s* + *r* ∗ *s*
    by *transfer* (*simp add*: *algebra-simps*)
  show (*0*::*rat*) ≠ *1*
    by *transfer simp*
  show *inverse q* ∗ *q* = *1* **if** *q* ≠ *0*
    using *that* **by** *transfer simp*
  show *q div r* = *q* ∗ *inverse r*
    by (*fact divide-rat-def*)
  show *inverse 0* = (*0*::*rat*)
    by *transfer simp*
qed

end

lemma *div-add-self1-no-field* [*simp*]:
  assumes *NO-MATCH* (*x* :: *'b* :: *field*) *b* (*b* :: *'a* :: *euclidean-semiring-cancel*) ≠ *0*
  shows (*b* + *a*) *div b* = *a div b* + *1*
  using *assms*(*2*) **by** (*fact div-add-self1*)

lemma *div-add-self2-no-field* [*simp*]:
  assumes *NO-MATCH* (*x* :: *'b* :: *field*) *b* (*b* :: *'a* :: *euclidean-semiring-cancel*) ≠ *0*
  shows (*a* + *b*) *div b* = *a div b* + *1*
  using *assms*(*2*) **by** (*fact div-add-self2*)

lemma *of-nat-rat*: *of-nat k* = *Fract* (*of-nat k*) *1*
  by (*induct k*) (*simp-all add*: *Zero-rat-def One-rat-def*)

lemma *of-int-rat*: *of-int k* = *Fract k 1*
  by (*cases k rule*: *int-diff-cases*) (*simp add*: *of-nat-rat*)

lemma *Fract-of-nat-eq*: *Fract* (*of-nat k*) *1* = *of-nat k*
  by (*rule of-nat-rat* [*symmetric*])

lemma *Fract-of-int-eq*: *Fract k 1* = *of-int k*
  by (*rule of-int-rat* [*symmetric*])

lemma *rat-number-collapse*:
  Fract 0 k = *0*
  Fract 1 1 = *1*
  Fract (*numeral w*) *1* = *numeral w*
  Fract (− *numeral w*) *1* = − *numeral w*
  Fract (− *1*) *1* = − *1*
  Fract k 0 = *0*
  using *Fract-of-int-eq* [*of numeral w*]
    and *Fract-of-int-eq* [*of* − *numeral w*]
  by (*simp-all add*: *Zero-rat-def One-rat-def eq-rat*)

lemma *rat-number-expand*:
  0 = *Fract 0 1*
  1 = *Fract 1 1*
  numeral k = *Fract* (*numeral k*) *1*
  − *1* = *Fract* (− *1*) *1*
  − *numeral k* = *Fract* (− *numeral k*) *1*
  by (*simp-all add*: *rat-number-collapse*)

lemma *Rat-cases-nonzero* [*case-names Fract 0*]:
  assumes *Fract*: $\bigwedge$*a b*. *q* = *Fract a b* ⟹ *b* > *0* ⟹ *a* ≠ *0* ⟹ *coprime a b* ⟹ *C*
    and *0*: *q* = *0* ⟹ *C*
  shows *C*
proof (*cases q* = *0*)
  case *True*
  then show *C* **using** *0* **by** *auto*
next
  case *False*
  then obtain *a b* **where** ∗: *q* = *Fract a b b* > *0 coprime a b*
    by (*cases q*) *auto*

with *False* **have** $0 \neq$ *Fract a b*
  by *simp*
with ‹$b > 0$› **have** $a \neq 0$
  by (*simp add*: *Zero-rat-def eq-rat*)
with *Fract* $*$ **show** *C* **by** *blast*
qed

7.1.3 Function *normalize*

lemma *Fract-coprime*: *Fract* (*a div gcd a b*) (*b div gcd a b*) = *Fract a b*
proof (*cases* $b = 0$)
  case *True*
  then show *?thesis*
    by (*simp add*: *eq-rat*)
next
  case *False*
  moreover have *b div gcd a b* $*$ *gcd a b* = *b*
    by (*rule dvd-div-mult-self*) *simp*
  ultimately have *b div gcd a b* $*$ *gcd a b* $\neq 0$
    by *simp*
  then have *b div gcd a b* $\neq 0$
    by *fastforce*
  with *False* **show** *?thesis*
    by (*simp add*: *eq-rat dvd-div-mult mult.commute* [*of a*])
qed

definition *normalize* :: *int* $\times$ *int* $\Rightarrow$ *int* $\times$ *int*
  where *normalize p* =
   (*if snd p* > 0 *then* (*let a* = *gcd* (*fst p*) (*snd p*) *in* (*fst p div a*, *snd p div a*))
    else if snd p = *0 then* (*0*, *1*)
    else (*let a* = $-$ *gcd* (*fst p*) (*snd p*) *in* (*fst p div a*, *snd p div a*)))

lemma *normalize-crossproduct*:
  assumes $q \neq 0$ $s \neq 0$
  assumes *normalize* (*p*, *q*) = *normalize* (*r*, *s*)
  shows $p * s = r * q$
proof $-$
  have $*$: $p * s = q * r$
    if *p* $*$ *gcd r s* = *sgn* (*q* $*$ *s*) $*$ *r* $*$ *gcd p q* **and** *q* $*$ *gcd r s* = *sgn* (*q* $*$ *s*) $*$ *s* $*$ *gcd p q*
  proof $-$
    from *that* **have** (*p* $*$ *gcd r s*) $*$ (*sgn* (*q* $*$ *s*) $*$ *s* $*$ *gcd p q*) =
        (*q* $*$ *gcd r s*) $*$ (*sgn* (*q* $*$ *s*) $*$ *r* $*$ *gcd p q*)
      by *simp*
    with *assms* **show** *?thesis*
      by (*auto simp add*: *ac-simps sgn-mult sgn-0-0*)
  qed
  from *assms* **show** *?thesis*
   by (*auto simp*: *normalize-def Let-def dvd-div-div-eq-mult mult.commute sgn-mult*

split: *if-splits intro*: $*$)
qed

lemma *normalize-eq*: *normalize* $(a, b) = (p, q) \Longrightarrow$ *Fract p q* $=$ *Fract a b*
by (*auto simp*: *normalize-def Let-def Fract-coprime dvd-div-neg rat-number-collapse* *split*: *if-split-asm*)

lemma *normalize-denom-pos*: *normalize r* $= (p, q) \Longrightarrow q > 0$
by (*auto simp*: *normalize-def Let-def dvd-div-neg pos-imp-zdiv-neg-iff nonneg1-imp-zdiv-pos-iff* *split*: *if-split-asm*)

lemma *normalize-coprime*: *normalize r* $= (p, q) \Longrightarrow$ *coprime p q*
by (*auto simp*: *normalize-def Let-def dvd-div-neg div-gcd-coprime split*: *if-split-asm*)

lemma *normalize-stable* [*simp*]: $q > 0 \Longrightarrow$ *coprime p q* $\Longrightarrow$ *normalize* $(p, q) = (p, q)$
by (*simp add*: *normalize-def*)

lemma *normalize-denom-zero* [*simp*]: *normalize* $(p, 0) = (0, 1)$
by (*simp add*: *normalize-def*)

lemma *normalize-negative* [*simp*]: $q < 0 \Longrightarrow$ *normalize* $(p, q) =$ *normalize* $(- p, - q)$
by (*simp add*: *normalize-def Let-def dvd-div-neg dvd-neg-div*)

Decompose a fraction into normalized, i.e. coprime numerator and denominator:

definition *quotient-of* :: *rat* $\Rightarrow$ *int* $\times$ *int*
where *quotient-of x* $=$
(*THE pair*. $x =$ *Fract* (*fst pair*) (*snd pair*) $\wedge$ *snd pair* $> 0 \wedge$ *coprime* (*fst pair*) (*snd pair*))

lemma *quotient-of-unique*: $\exists!p.$ $r =$ *Fract* (*fst p*) (*snd p*) $\wedge$ *snd p* $> 0 \wedge$ *coprime* (*fst p*) (*snd p*)
proof (*cases r*)
case (*Fract a b*)
then have $r =$ *Fract* (*fst* (a, b)) (*snd* (a, b)) $\wedge$
snd $(a, b) > 0 \wedge$ *coprime* (*fst* (a, b)) (*snd* (a, b))
by *auto*
then show *?thesis*
proof (*rule ex1I*)
fix *p*
assume *r*: $r =$ *Fract* (*fst p*) (*snd p*) $\wedge$ *snd p* $> 0 \wedge$ *coprime* (*fst p*) (*snd p*)
obtain *c d* **where** *p*: $p = (c, d)$ **by** (*cases p*)
with *r* **have** *Fract'*: $r =$ *Fract c d* $d > 0$ *coprime c d*
by *simp-all*
have $(c, d) = (a, b)$
proof (*cases* $a = 0$)
case *True*

with *Fract Fract′* **show** *?thesis*
by (*simp add*: *eq-rat*)
next
case *False*
with *Fract Fract′* **have** ∗: $c * b = a * d$ **and** $c \neq 0$
by (*auto simp add*: *eq-rat*)
then have $c * b > 0 \longleftrightarrow a * d > 0$
by *auto*
with ‹$b > 0$› ‹$d > 0$› **have** $a > 0 \longleftrightarrow c > 0$
by (*simp add*: *zero-less-mult-iff*)
with ‹$a \neq 0$› ‹$c \neq 0$› **have** *sgn*: *sgn a* = *sgn c*
by (*auto simp add*: *not-less*)
from ‹*coprime a b*› ‹*coprime c d*› **have** $|a| * |d| = |c| * |b| \longleftrightarrow |a| = |c| \wedge |d| = |b|$
by (*simp add*: *coprime-crossproduct-int*)
with ‹$b > 0$› ‹$d > 0$› **have** $|a| * d = |c| * b \longleftrightarrow |a| = |c| \wedge d = b$
by *simp*
then have $a * sgn\ a * d = c * sgn\ c * b \longleftrightarrow a * sgn\ a = c * sgn\ c \wedge d = b$
by (*simp add*: *abs-sgn*)
with *sgn* ∗ **show** *?thesis*
by (*auto simp add*: *sgn-0-0*)
qed
with *p* **show** $p = (a, b)$
by *simp*
qed
qed

lemma *quotient-of-Fract* [*code*]: *quotient-of* (*Fract a b*) = *normalize* (*a*, *b*)
proof −
have *Fract a b* = *Fract* (*fst* (*normalize* (*a*, *b*))) (*snd* (*normalize* (*a*, *b*))) (**is** *?Fract*)
by (*rule sym*) (*auto intro*: *normalize-eq*)
moreover have 0 < *snd* (*normalize* (*a*, *b*)) (**is** *?denom-pos*)
by (*cases normalize* (*a*, *b*)) (*rule normalize-denom-pos*, *simp*)
moreover have *coprime* (*fst* (*normalize* (*a*, *b*))) (*snd* (*normalize* (*a*, *b*))) (**is** *?coprime*)
by (*rule normalize-coprime*) *simp*
ultimately have *?Fract* ∧ *?denom-pos* ∧ *?coprime* **by** *blast*
then have (*THE p*. *Fract a b* = *Fract* (*fst p*) (*snd p*) ∧ 0 < *snd p* ∧
coprime (*fst p*) (*snd p*)) = *normalize* (*a*, *b*)
by (*rule the1-equality* [*OF quotient-of-unique*])
then show *?thesis* **by** (*simp add*: *quotient-of-def*)
qed

lemma *quotient-of-number* [*simp*]:
quotient-of 0 = (0, 1)
quotient-of 1 = (1, 1)
quotient-of (*numeral k*) = (*numeral k*, 1)
quotient-of (− 1) = (− 1, 1)

quotient-of (− *numeral k*) = (− *numeral k*, *1*)
by (*simp-all add*: *rat-number-expand quotient-of-Fract*)

lemma *quotient-of-eq*: *quotient-of* (*Fract a b*) = (*p*, *q*) $\Longrightarrow$ *Fract p q* = *Fract a b*
by (*simp add*: *quotient-of-Fract normalize-eq*)

lemma *quotient-of-denom-pos*: *quotient-of r* = (*p*, *q*) $\Longrightarrow$ *q* > *0*
by (*cases r*) (*simp add*: *quotient-of-Fract normalize-denom-pos*)

lemma *quotient-of-denom-pos'*: *snd* (*quotient-of r*) > *0*
using *quotient-of-denom-pos* [*of r*] **by** (*simp add*: *prod-eq-iff*)

lemma *quotient-of-coprime*: *quotient-of r* = (*p*, *q*) $\Longrightarrow$ *coprime p q*
by (*cases r*) (*simp add*: *quotient-of-Fract normalize-coprime*)

lemma *quotient-of-inject*:
assumes *quotient-of a* = *quotient-of b*
shows *a* = *b*
proof −
obtain *p q r s* **where** *a*: *a* = *Fract p q* **and** *b*: *b* = *Fract r s* **and** *q* > *0* **and** *s* > *0*
by (*cases a*, *cases b*)
with *assms* **show** *?thesis*
by (*simp add*: *eq-rat quotient-of-Fract normalize-crossproduct*)
qed

lemma *quotient-of-inject-eq*: *quotient-of a* = *quotient-of b* $\longleftrightarrow$ *a* = *b*
by (*auto simp add*: *quotient-of-inject*)

7.1.4 Various

lemma *Fract-of-int-quotient*: *Fract k l* = *of-int k* / *of-int l*
by (*simp add*: *Fract-of-int-eq* [*symmetric*])

lemma *Fract-add-one*: *n* ≠ *0* $\Longrightarrow$ *Fract* (*m* + *n*) *n* = *Fract m n* + *1*
by (*simp add*: *rat-number-expand*)

lemma *quotient-of-div*:
assumes *r*: *quotient-of r* = (*n*,*d*)
shows *r* = *of-int n* / *of-int d*
proof −
from *theI'*[*OF quotient-of-unique*[*of r*], *unfolded r*[*unfolded quotient-of-def*]]
have *r* = *Fract n d* **by** *simp*
then show *?thesis* **using** *Fract-of-int-quotient*
by *simp*
qed

7.1.5 The ordered field of rational numbers

lift-definition *positive* :: *rat* ⇒ *bool*

is $\lambda x.\ 0 < \mathit{fst}\ x * \mathit{snd}\ x$
proof *clarsimp*
fix $a\ b\ c\ d :: \mathit{int}$
assume $b \neq 0$ **and** $d \neq 0$ **and** $a * d = c * b$
then have $a * d * b * d = c * b * b * d$
by *simp*
then have $a * b * d^2 = c * d * b^2$
unfolding *power2-eq-square* **by** (*simp add*: *ac-simps*)
then have $0 < a * b * d^2 \longleftrightarrow 0 < c * d * b^2$
by *simp*
then show $0 < a * b \longleftrightarrow 0 < c * d$
using ‹$b \neq 0$› **and** ‹$d \neq 0$›
by (*simp add*: *zero-less-mult-iff*)
qed

lemma *positive-zero*: $\neg$ *positive* 0
by *transfer simp*

lemma *positive-add*: *positive* $x \Longrightarrow$ *positive* $y \Longrightarrow$ *positive* $(x + y)$
apply *transfer*
apply (*simp add*: *zero-less-mult-iff*)
apply (*elim disjE*)
apply (*simp-all add*: *add-pos-pos add-neg-neg mult-pos-neg mult-neg-pos mult-neg-neg*)
done

lemma *positive-mult*: *positive* $x \Longrightarrow$ *positive* $y \Longrightarrow$ *positive* $(x * y)$
apply *transfer*
apply (*drule* (*1*) *mult-pos-pos*)
apply (*simp add*: *ac-simps*)
done

lemma *positive-minus*: $\neg$ *positive* $x \Longrightarrow x \neq 0 \Longrightarrow$ *positive* $(- x)$
by *transfer* (*auto simp*: *neq-iff zero-less-mult-iff mult-less-0-iff*)

instantiation *rat* :: *linordered-field*
begin

definition $x < y \longleftrightarrow$ *positive* $(y - x)$

definition $x \leq y \longleftrightarrow x < y \lor x = y$ **for** $x\ y :: \mathit{rat}$

definition $|a| = (\mathit{if}\ a < 0\ \mathit{then}\ - a\ \mathit{else}\ a)$ **for** $a :: \mathit{rat}$

definition $\mathit{sgn}\ a = (\mathit{if}\ a = 0\ \mathit{then}\ 0\ \mathit{else}\ \mathit{if}\ 0 < a\ \mathit{then}\ 1\ \mathit{else}\ - 1)$ **for** $a :: \mathit{rat}$

instance
proof
fix $a\ b\ c :: \mathit{rat}$
show $|a| = (\mathit{if}\ a < 0\ \mathit{then}\ - a\ \mathit{else}\ a)$

```
    by (rule abs-rat-def)
  show a < b ⟷ a ≤ b ∧ ¬ b ≤ a
    unfolding less-eq-rat-def less-rat-def
    apply auto
     apply (drule (1) positive-add)
     apply (simp-all add: positive-zero)
    done
  show a ≤ a
    unfolding less-eq-rat-def by simp
  show a ≤ b ⟹ b ≤ c ⟹ a ≤ c
    unfolding less-eq-rat-def less-rat-def
    apply auto
    apply (drule (1) positive-add)
    apply (simp add: algebra-simps)
    done
  show a ≤ b ⟹ b ≤ a ⟹ a = b
    unfolding less-eq-rat-def less-rat-def
    apply auto
    apply (drule (1) positive-add)
    apply (simp add: positive-zero)
    done
  show a ≤ b ⟹ c + a ≤ c + b
    unfolding less-eq-rat-def less-rat-def by auto
  show sgn a = (if a = 0 then 0 else if 0 < a then 1 else − 1)
    by (rule sgn-rat-def)
  show a ≤ b ∨ b ≤ a
    unfolding less-eq-rat-def less-rat-def
    by (auto dest!: positive-minus)
  show a < b ⟹ 0 < c ⟹ c ∗ a < c ∗ b
    unfolding less-rat-def
    apply (drule (1) positive-mult)
    apply (simp add: algebra-simps)
    done
qed

end

lemmas (in linordered-field) sign-simps = algebra-simps zero-less-mult-iff mult-less-0-iff
lemmas sign-simps = algebra-simps zero-less-mult-iff mult-less-0-iff


instantiation rat :: distrib-lattice
begin

definition (inf :: rat ⇒ rat ⇒ rat) = min

definition (sup :: rat ⇒ rat ⇒ rat) = max

instance
```

by *standard* (*auto simp add*: *inf-rat-def sup-rat-def max-min-distrib2*)

end

lemma *positive-rat*: *positive* (*Fract a b*) $\longleftrightarrow$ *0* < *a* ∗ *b*
by *transfer simp*

lemma *less-rat* [*simp*]:
$b \neq 0 \Longrightarrow d \neq 0 \Longrightarrow$ *Fract a b* < *Fract c d* $\longleftrightarrow$ (*a* ∗ *d*) ∗ (*b* ∗ *d*) < (*c* ∗ *b*) ∗ (*b* ∗ *d*)
by (*simp add*: *less-rat-def positive-rat algebra-simps*)

lemma *le-rat* [*simp*]:
$b \neq 0 \Longrightarrow d \neq 0 \Longrightarrow$ *Fract a b* ≤ *Fract c d* $\longleftrightarrow$ (*a* ∗ *d*) ∗ (*b* ∗ *d*) ≤ (*c* ∗ *b*) ∗ (*b* ∗ *d*)
by (*simp add*: *le-less eq-rat*)

lemma *abs-rat* [*simp*, *code*]: |*Fract a b*| = *Fract* |*a*| |*b*|
by (*auto simp add*: *abs-rat-def zabs-def Zero-rat-def not-less le-less eq-rat zero-less-mult-iff*)

lemma *sgn-rat* [*simp*, *code*]: *sgn* (*Fract a b*) = *of-int* (*sgn a* ∗ *sgn b*)
unfolding *Fract-of-int-eq*
by (*auto simp*: *zsgn-def sgn-rat-def Zero-rat-def eq-rat*)
(*auto simp*: *rat-number-collapse not-less le-less zero-less-mult-iff*)

lemma *Rat-induct-pos* [*case-names Fract*, *induct type*: *rat*]:
assumes *step*: $\bigwedge a\ b.\ 0 < b \Longrightarrow P\ (Fract\ a\ b)$
shows *P q*
proof (*cases q*)
case (*Fract a b*)
have *step′*: *P* (*Fract a b*) **if** *b*: *b* < *0* **for** *a b* :: *int*
proof −
from *b* **have** *0* < − *b*
by *simp*
then have *P* (*Fract* (− *a*) (− *b*))
by (*rule step*)
then show *P* (*Fract a b*)
by (*simp add*: *order-less-imp-not-eq* [*OF b*])
qed
from *Fract* **show** *P q*
by (*auto simp add*: *linorder-neq-iff step step′*)
qed

lemma *zero-less-Fract-iff*: *0* < *b* $\Longrightarrow$ *0* < *Fract a b* $\longleftrightarrow$ *0* < *a*
by (*simp add*: *Zero-rat-def zero-less-mult-iff*)

lemma *Fract-less-zero-iff*: *0* < *b* $\Longrightarrow$ *Fract a b* < *0* $\longleftrightarrow$ *a* < *0*
by (*simp add*: *Zero-rat-def mult-less-0-iff*)

lemma *zero-le-Fract-iff*: $0 < b \Longrightarrow 0 \le Fract\ a\ b \longleftrightarrow 0 \le a$
  by (*simp add*: *Zero-rat-def zero-le-mult-iff*)

lemma *Fract-le-zero-iff*: $0 < b \Longrightarrow Fract\ a\ b \le 0 \longleftrightarrow a \le 0$
  by (*simp add*: *Zero-rat-def mult-le-0-iff*)

lemma *one-less-Fract-iff*: $0 < b \Longrightarrow 1 < Fract\ a\ b \longleftrightarrow b < a$
  by (*simp add*: *One-rat-def mult-less-cancel-right-disj*)

lemma *Fract-less-one-iff*: $0 < b \Longrightarrow Fract\ a\ b < 1 \longleftrightarrow a < b$
  by (*simp add*: *One-rat-def mult-less-cancel-right-disj*)

lemma *one-le-Fract-iff*: $0 < b \Longrightarrow 1 \le Fract\ a\ b \longleftrightarrow b \le a$
  by (*simp add*: *One-rat-def mult-le-cancel-right*)

lemma *Fract-le-one-iff*: $0 < b \Longrightarrow Fract\ a\ b \le 1 \longleftrightarrow a \le b$
  by (*simp add*: *One-rat-def mult-le-cancel-right*)

7.1.6 Rationals are an Archimedean field

lemma *rat-floor-lemma*: $of\text{-}int\ (a\ div\ b) \le Fract\ a\ b \wedge Fract\ a\ b < of\text{-}int\ (a\ div\ b + 1)$
proof −
  have $Fract\ a\ b = of\text{-}int\ (a\ div\ b) + Fract\ (a\ mod\ b)\ b$
    by (*cases* $b = 0$) (*simp*, *simp add*: *of-int-rat*)
  moreover have $0 \le Fract\ (a\ mod\ b)\ b \wedge Fract\ (a\ mod\ b)\ b < 1$
    unfolding *Fract-of-int-quotient*
    by (*rule linorder-cases* [*of b 0*]) (*simp-all add*: *divide-nonpos-neg*)
  ultimately show *?thesis* **by** *simp*
qed

instance *rat* :: *archimedean-field*
proof
  show $\exists z.\ r \le of\text{-}int\ z$ **for** *r* :: *rat*
  proof (*induct r*)
    case (*Fract a b*)
    have $Fract\ a\ b \le of\text{-}int\ (a\ div\ b + 1)$
      using *rat-floor-lemma* [*of a b*] **by** *simp*
    then show $\exists z.\ Fract\ a\ b \le of\text{-}int\ z$ **..**
  qed
qed

instantiation *rat* :: *floor-ceiling*
begin

definition [*code del*]: $\lfloor x \rfloor = (THE\ z.\ of\text{-}int\ z \le x \wedge x < of\text{-}int\ (z + 1))$ **for** *x* :: *rat*

instance

proof
  show *of-int* $\lfloor x \rfloor \leq x \wedge x <$ *of-int* $(\lfloor x \rfloor + 1)$ **for** x :: *rat*
    unfolding *floor-rat-def* **using** *floor-exists1* **by** (*rule theI'*)
qed

end

lemma *floor-Fract*: $\lfloor$*Fract a b*$\rfloor$ = *a div b*
  by (*simp add*: *Fract-of-int-quotient floor-divide-of-int-eq*)

7.2 Linear arithmetic setup

declaration ‹
  K (*Lin-Arith.add-inj-thms* [@{*thm of-nat-le-iff*} *RS iffD2*, @{*thm of-nat-eq-iff*} *RS iffD2*]
    (∗ *not needed because* $x <$ (*y*::*nat*) *can be rewritten as Suc x* <= *y*: *of-nat-less-iff RS iffD2* ∗)
  #> *Lin-Arith.add-inj-thms* [@{*thm of-int-le-iff*} *RS iffD2*, @{*thm of-int-eq-iff*} *RS iffD2*]
    (∗ *not needed because* $x <$ (*y*::*int*) *can be rewritten as* $x + 1$ <= *y*: *of-int-less-iff RS iffD2* ∗)
  #> *Lin-Arith.add-simps* [@{*thm neg-less-iff-less*},
      @{*thm True-implies-equals*},
      @{*thm distrib-left* [*where a* = *numeral v for v*]},
      @{*thm distrib-left* [*where a* = − *numeral v for v*]},
      @{*thm div-by-1*}, @{*thm div-0*},
      @{*thm times-divide-eq-right*}, @{*thm times-divide-eq-left*},
      @{*thm minus-divide-left*} *RS sym*, @{*thm minus-divide-right*} *RS sym*,
      @{*thm add-divide-distrib*}, @{*thm diff-divide-distrib*},
      @{*thm of-int-minus*}, @{*thm of-int-diff*},
      @{*thm of-int-of-nat-eq*}]
  #> *Lin-Arith.add-simprocs* [*Numeral-Simprocs.field-divide-cancel-numeral-factor*]
  #> *Lin-Arith.add-inj-const* (***const-name*** ‹*of-nat*›, ***typ*** ‹*nat* ⇒ *rat*›)
  #> *Lin-Arith.add-inj-const* (***const-name*** ‹*of-int*›, ***typ*** ‹*int* ⇒ *rat*›))
›

7.3 Embedding from Rationals to other Fields

context *field-char-0*
begin

lift-definition *of-rat* :: *rat* ⇒ *'a*
  is λ*x*. *of-int* (*fst x*) / *of-int* (*snd x*)
  by (*auto simp*: *nonzero-divide-eq-eq nonzero-eq-divide-eq*) (*simp only*: *of-int-mult* [*symmetric*])

end

lemma *of-rat-rat*: $b \neq 0 \Longrightarrow$ *of-rat* (*Fract a b*) = *of-int a* / *of-int b*
  by *transfer simp*

lemma *of-rat-0* [*simp*]: *of-rat 0* = *0*
  by *transfer simp*

lemma *of-rat-1* [*simp*]: *of-rat 1* = *1*
  by *transfer simp*

lemma *of-rat-add*: *of-rat* (*a* + *b*) = *of-rat a* + *of-rat b*
  by *transfer* (*simp add*: *add-frac-eq*)

lemma *of-rat-minus*: *of-rat* (− *a*) = − *of-rat a*
  by *transfer simp*

lemma *of-rat-neg-one* [*simp*]: *of-rat* (− *1*) = − *1*
  by (*simp add*: *of-rat-minus*)

lemma *of-rat-diff*: *of-rat* (*a* − *b*) = *of-rat a* − *of-rat b*
  using *of-rat-add* [*of a* − *b*] **by** (*simp add*: *of-rat-minus*)

lemma *of-rat-mult*: *of-rat* (*a* ∗ *b*) = *of-rat a* ∗ *of-rat b*
  by *transfer* (*simp add*: *divide-inverse nonzero-inverse-mult-distrib ac-simps*)

lemma *of-rat-sum*: *of-rat* ($\sum$ *a*∈*A*. *f a*) = ($\sum$ *a*∈*A*. *of-rat* (*f a*))
  by (*induct rule*: *infinite-finite-induct*) (*auto simp*: *of-rat-add*)

lemma *of-rat-prod*: *of-rat* ($\prod$ *a*∈*A*. *f a*) = ($\prod$ *a*∈*A*. *of-rat* (*f a*))
  by (*induct rule*: *infinite-finite-induct*) (*auto simp*: *of-rat-mult*)

lemma *nonzero-of-rat-inverse*: *a* ≠ *0* ⟹ *of-rat* (*inverse a*) = *inverse* (*of-rat a*)
  by (*rule inverse-unique* [*symmetric*]) (*simp add*: *of-rat-mult* [*symmetric*])

lemma *of-rat-inverse*: (*of-rat* (*inverse a*) :: ′*a*::*field-char-0*) = *inverse* (*of-rat a*)
  by (*cases a* = *0*) (*simp-all add*: *nonzero-of-rat-inverse*)

lemma *nonzero-of-rat-divide*: *b* ≠ *0* ⟹ *of-rat* (*a* / *b*) = *of-rat a* / *of-rat b*
  by (*simp add*: *divide-inverse of-rat-mult nonzero-of-rat-inverse*)

lemma *of-rat-divide*: (*of-rat* (*a* / *b*) :: ′*a*::*field-char-0*) = *of-rat a* / *of-rat b*
  by (*cases b* = *0*) (*simp-all add*: *nonzero-of-rat-divide*)

lemma *of-rat-power*: (*of-rat* (*a* ˆ *n*) :: ′*a*::*field-char-0*) = *of-rat a* ˆ *n*
  by (*induct n*) (*simp-all add*: *of-rat-mult*)

lemma *of-rat-eq-iff* [*simp*]: *of-rat a* = *of-rat b* ⟷ *a* = *b*
  apply *transfer*
  apply (*simp add*: *nonzero-divide-eq-eq nonzero-eq-divide-eq*)
  apply (*simp only*: *of-int-mult* [*symmetric*] *of-int-eq-iff*)
  done

lemma *of-rat-eq-0-iff* [*simp*]: *of-rat a = 0* ⟷ *a = 0*
using *of-rat-eq-iff* [*of - 0*] **by** *simp*

lemma *zero-eq-of-rat-iff* [*simp*]: *0 = of-rat a* ⟷ *0 = a*
by *simp*

lemma *of-rat-eq-1-iff* [*simp*]: *of-rat a = 1* ⟷ *a = 1*
using *of-rat-eq-iff* [*of - 1*] **by** *simp*

lemma *one-eq-of-rat-iff* [*simp*]: *1 = of-rat a* ⟷ *1 = a*
by *simp*

lemma *of-rat-less*: (*of-rat r* :: *'a::linordered-field*) < *of-rat s* ⟷ *r* < *s*
proof (*induct r*, *induct s*)
fix *a b c d* :: *int*
assume *not-zero*: *b* > *0 d* > *0*
then have *b* ∗ *d* > *0* **by** *simp*
have *of-int-divide-less-eq*:
(*of-int a* :: *'a*) / *of-int b* < *of-int c* / *of-int d* ⟷
(*of-int a* :: *'a*) ∗ *of-int d* < *of-int c* ∗ *of-int b*
using *not-zero* **by** (*simp add*: *pos-less-divide-eq pos-divide-less-eq*)
show (*of-rat* (*Fract a b*) :: *'a::linordered-field*) < *of-rat* (*Fract c d*) ⟷
Fract a b < *Fract c d*
using *not-zero* ‹*b* ∗ *d* > *0*›
by (*simp add*: *of-rat-rat of-int-divide-less-eq of-int-mult* [*symmetric*] *del*: *of-int-mult*)
qed

lemma *of-rat-less-eq*: (*of-rat r* :: *'a::linordered-field*) ≤ *of-rat s* ⟷ *r* ≤ *s*
unfolding *le-less* **by** (*auto simp add*: *of-rat-less*)

lemma *of-rat-le-0-iff* [*simp*]: (*of-rat r* :: *'a::linordered-field*) ≤ *0* ⟷ *r* ≤ *0*
using *of-rat-less-eq* [*of r 0*, **where** *'a = 'a*] **by** *simp*

lemma *zero-le-of-rat-iff* [*simp*]: *0* ≤ (*of-rat r* :: *'a::linordered-field*) ⟷ *0* ≤ *r*
using *of-rat-less-eq* [*of 0 r*, **where** *'a = 'a*] **by** *simp*

lemma *of-rat-le-1-iff* [*simp*]: (*of-rat r* :: *'a::linordered-field*) ≤ *1* ⟷ *r* ≤ *1*
using *of-rat-less-eq* [*of r 1*] **by** *simp*

lemma *one-le-of-rat-iff* [*simp*]: *1* ≤ (*of-rat r* :: *'a::linordered-field*) ⟷ *1* ≤ *r*
using *of-rat-less-eq* [*of 1 r*] **by** *simp*

lemma *of-rat-less-0-iff* [*simp*]: (*of-rat r* :: *'a::linordered-field*) < *0* ⟷ *r* < *0*
using *of-rat-less* [*of r 0*, **where** *'a = 'a*] **by** *simp*

lemma *zero-less-of-rat-iff* [*simp*]: *0* < (*of-rat r* :: *'a::linordered-field*) ⟷ *0* < *r*
using *of-rat-less* [*of 0 r*, **where** *'a = 'a*] **by** *simp*

lemma *of-rat-less-1-iff* [*simp*]: (*of-rat r* :: *'a::linordered-field*) < *1* ⟷ *r* < *1*

using *of-rat-less* [*of r 1*] **by** *simp*

lemma *one-less-of-rat-iff* [*simp*]: $1 < (\textit{of-rat } r :: 'a::\textit{linordered-field}) \longleftrightarrow 1 < r$
using *of-rat-less* [*of 1 r*] **by** *simp*

lemma *of-rat-eq-id* [*simp*]: *of-rat* = *id*
proof
show *of-rat a* = *id a* **for** *a*
by (*induct a*) (*simp add*: *of-rat-rat Fract-of-int-eq* [*symmetric*])
qed

Collapse nested embeddings.

lemma *of-rat-of-nat-eq* [*simp*]: *of-rat* (*of-nat n*) = *of-nat n*
by (*induct n*) (*simp-all add*: *of-rat-add*)

lemma *of-rat-of-int-eq* [*simp*]: *of-rat* (*of-int z*) = *of-int z*
by (*cases z rule*: *int-diff-cases*) (*simp add*: *of-rat-diff*)

lemma *of-rat-numeral-eq* [*simp*]: *of-rat* (*numeral w*) = *numeral w*
using *of-rat-of-int-eq* [*of numeral w*] **by** *simp*

lemma *of-rat-neg-numeral-eq* [*simp*]: *of-rat* (− *numeral w*) = − *numeral w*
using *of-rat-of-int-eq* [*of* − *numeral w*] **by** *simp*

lemmas *zero-rat* = *Zero-rat-def*
lemmas *one-rat* = *One-rat-def*

abbreviation *rat-of-nat* :: *nat* ⇒ *rat*
where *rat-of-nat* ≡ *of-nat*

abbreviation *rat-of-int* :: *int* ⇒ *rat*
where *rat-of-int* ≡ *of-int*

7.4 The Set of Rational Numbers

context *field-char-0*
begin

definition *Rats* :: *'a set* ($\mathbb{Q}$)
where $\mathbb{Q}$ = *range of-rat*

end

lemma *Rats-cases* [*cases set*: *Rats*]:
assumes $q \in \mathbb{Q}$
obtains (*of-rat*) *r* **where** *q* = *of-rat r*
proof −
from ‹$q \in \mathbb{Q}$› **have** $q \in$ *range of-rat*
by (*simp only*: *Rats-def*)

then obtain *r* **where** *q* = *of-rat r* **..**
then show *thesis* **..**
qed

lemma *Rats-of-rat* [*simp*]: *of-rat r* $\in \mathbb{Q}$
by (*simp add*: *Rats-def*)

lemma *Rats-of-int* [*simp*]: *of-int z* $\in \mathbb{Q}$
by (*subst of-rat-of-int-eq* [*symmetric*]) (*rule Rats-of-rat*)

lemma *Ints-subset-Rats*: $\mathbb{Z} \subseteq \mathbb{Q}$
using *Ints-cases Rats-of-int* **by** *blast*

lemma *Rats-of-nat* [*simp*]: *of-nat n* $\in \mathbb{Q}$
by (*subst of-rat-of-nat-eq* [*symmetric*]) (*rule Rats-of-rat*)

lemma *Nats-subset-Rats*: $\mathbb{N} \subseteq \mathbb{Q}$
using *Ints-subset-Rats Nats-subset-Ints* **by** *blast*

lemma *Rats-number-of* [*simp*]: *numeral w* $\in \mathbb{Q}$
by (*subst of-rat-numeral-eq* [*symmetric*]) (*rule Rats-of-rat*)

lemma *Rats-0* [*simp*]: *0* $\in \mathbb{Q}$
unfolding *Rats-def* **by** (*rule range-eqI*) (*rule of-rat-0* [*symmetric*])

lemma *Rats-1* [*simp*]: *1* $\in \mathbb{Q}$
unfolding *Rats-def* **by** (*rule range-eqI*) (*rule of-rat-1* [*symmetric*])

lemma *Rats-add* [*simp*]: $a \in \mathbb{Q} \Longrightarrow b \in \mathbb{Q} \Longrightarrow a + b \in \mathbb{Q}$
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-add* [*symmetric*])
done

lemma *Rats-minus-iff* [*simp*]: $- a \in \mathbb{Q} \longleftrightarrow a \in \mathbb{Q}$
by (*metis Rats-cases Rats-of-rat add.inverse-inverse of-rat-minus*)

lemma *Rats-diff* [*simp*]: $a \in \mathbb{Q} \Longrightarrow b \in \mathbb{Q} \Longrightarrow a - b \in \mathbb{Q}$
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-diff* [*symmetric*])
done

lemma *Rats-mult* [*simp*]: $a \in \mathbb{Q} \Longrightarrow b \in \mathbb{Q} \Longrightarrow a * b \in \mathbb{Q}$
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-mult* [*symmetric*])
done

lemma *Rats-inverse* [*simp*]: $a \in \mathbb{Q} \Longrightarrow$ *inverse* $a \in \mathbb{Q}$
for a :: $'a$::*field-char-0*
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-inverse* [*symmetric*])
done

lemma *Rats-divide* [*simp*]: $a \in \mathbb{Q} \Longrightarrow b \in \mathbb{Q} \Longrightarrow a \;/\; b \in \mathbb{Q}$
for a b :: $'a$::*field-char-0*
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-divide* [*symmetric*])
done

lemma *Rats-power* [*simp*]: $a \in \mathbb{Q} \Longrightarrow a \ \hat{}\ n \in \mathbb{Q}$
for a :: $'a$::*field-char-0*
apply (*auto simp add*: *Rats-def*)
apply (*rule range-eqI*)
apply (*rule of-rat-power* [*symmetric*])
done

lemma *Rats-induct* [*case-names of-rat*, *induct set*: *Rats*]: $q \in \mathbb{Q} \Longrightarrow (\bigwedge r.\ P$ (*of-rat* r)) $\Longrightarrow P\ q$
by (*rule Rats-cases*) *auto*

lemma *Rats-infinite*: $\neg$ *finite* $\mathbb{Q}$
by (*auto dest*!: *finite-imageD simp*: *inj-on-def infinite-UNIV-char-0 Rats-def*)

7.5 Implementation of rational numbers as pairs of integers

Formal constructor

definition *Frct* :: *int* $\times$ *int* $\Rightarrow$ *rat*
where [*simp*]: *Frct* p = *Fract* (*fst* p) (*snd* p)

lemma [*code abstype*]: *Frct* (*quotient-of* q) = q
by (*cases* q) (*auto intro*: *quotient-of-eq*)

Numerals

declare *quotient-of-Fract* [*code abstract*]

definition *of-int* :: *int* $\Rightarrow$ *rat*
where [*code-abbrev*]: *of-int* = *Int.of-int*

hide-const (**open**) *of-int*

lemma *quotient-of-int* [*code abstract*]: *quotient-of* (*Rat.of-int* a) = (a, *1*)
by (*simp add*: *of-int-def of-int-rat quotient-of-Fract*)

lemma [*code-unfold*]: *numeral* k = *Rat.of-int* (*numeral* k)

by (*simp add*: *Rat.of-int-def*)

lemma [*code-unfold*]: $-$ *numeral k* = *Rat.of-int* ($-$ *numeral k*)
by (*simp add*: *Rat.of-int-def*)

lemma *Frct-code-post* [*code-post*]:
Frct (*0*, *a*) = *0*
Frct (*a*, *0*) = *0*
Frct (*1*, *1*) = *1*
Frct (*numeral k*, *1*) = *numeral k*
Frct (*1*, *numeral k*) = *1* / *numeral k*
Frct (*numeral k*, *numeral l*) = *numeral k* / *numeral l*
Frct ($-$ *a*, *b*) = $-$ *Frct* (*a*, *b*)
Frct (*a*, $-$ *b*) = $-$ *Frct* (*a*, *b*)
$-$ ($-$ *Frct q*) = *Frct q*
by (*simp-all add*: *Fract-of-int-quotient*)

Operations

lemma *rat-zero-code* [*code abstract*]: *quotient-of 0* = (*0*, *1*)
by (*simp add*: *Zero-rat-def quotient-of-Fract normalize-def*)

lemma *rat-one-code* [*code abstract*]: *quotient-of 1* = (*1*, *1*)
by (*simp add*: *One-rat-def quotient-of-Fract normalize-def*)

lemma *rat-plus-code* [*code abstract*]:
quotient-of (*p* + *q*) = (*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q*
in normalize (*a* ∗ *d* + *b* ∗ *c*, *c* ∗ *d*))
by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract*)

lemma *rat-uminus-code* [*code abstract*]:
quotient-of ($-$ *p*) = (*let* (*a*, *b*) = *quotient-of p in* ($-$ *a*, *b*))
by (*cases p*) (*simp add*: *quotient-of-Fract*)

lemma *rat-minus-code* [*code abstract*]:
quotient-of (*p* $-$ *q*) =
(*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q*
in normalize (*a* ∗ *d* $-$ *b* ∗ *c*, *c* ∗ *d*))
by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract*)

lemma *rat-times-code* [*code abstract*]:
quotient-of (*p* ∗ *q*) =
(*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q*
in normalize (*a* ∗ *b*, *c* ∗ *d*))
by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract*)

lemma *rat-inverse-code* [*code abstract*]:
quotient-of (*inverse p*) =
(*let* (*a*, *b*) = *quotient-of p*
in if a = *0 then* (*0*, *1*) *else* (*sgn a* ∗ *b*, |*a*|))

proof (*cases p*)
  case (*Fract a b*)
  then show *?thesis*
   by (*cases 0::int a rule*: *linorder-cases*) (*simp-all add*: *quotient-of-Fract ac-simps*)
qed

lemma *rat-divide-code* [*code abstract*]:
  quotient-of (*p* / *q*) =
    (*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q*
     in normalize (*a* ∗ *d*, *c* ∗ *b*))
  by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract*)

lemma *rat-abs-code* [*code abstract*]:
  quotient-of |*p*| = (*let* (*a*, *b*) = *quotient-of p in* (|*a*|, *b*))
  by (*cases p*) (*simp add*: *quotient-of-Fract*)

lemma *rat-sgn-code* [*code abstract*]: *quotient-of* (*sgn p*) = (*sgn* (*fst* (*quotient-of p*)), *1*)
proof (*cases p*)
  case (*Fract a b*)
  then show *?thesis*
    by (*cases 0::int a rule*: *linorder-cases*) (*simp-all add*: *quotient-of-Fract*)
qed

lemma *rat-floor-code* [*code*]: ⌊*p*⌋ = (*let* (*a*, *b*) = *quotient-of p in a div b*)
  by (*cases p*) (*simp add*: *quotient-of-Fract floor-Fract*)

instantiation *rat* :: *equal*
begin

definition [*code*]: *HOL.equal a b* ⟷ *quotient-of a* = *quotient-of b*

instance
  by *standard* (*simp add*: *equal-rat-def quotient-of-inject-eq*)

lemma *rat-eq-refl* [*code nbe*]: *HOL.equal* (*r*::*rat*) *r* ⟷ *True*
  by (*rule equal-refl*)

end

lemma *rat-less-eq-code* [*code*]:
  p ≤ *q* ⟷ (*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q in a* ∗ *d* ≤ *c* ∗ *b*)
  by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract mult.commute*)

lemma *rat-less-code* [*code*]:
  p < *q* ⟷ (*let* (*a*, *c*) = *quotient-of p*; (*b*, *d*) = *quotient-of q in a* ∗ *d* < *c* ∗ *b*)
  by (*cases p*, *cases q*) (*simp add*: *quotient-of-Fract mult.commute*)

lemma [*code*]: *of-rat p* = (*let* (*a*, *b*) = *quotient-of p in of-int a* / *of-int b*)

by (*cases p*) (*simp add*: *quotient-of-Fract of-rat-rat*)

Quickcheck

definition (**in** *term-syntax*)
  valterm-fract :: *int* × (*unit* ⇒ *Code-Evaluation.term*) ⇒
    int × (*unit* ⇒ *Code-Evaluation.term*) ⇒
    rat × (*unit* ⇒ *Code-Evaluation.term*)
  where [*code-unfold*]: *valterm-fract k l* = *Code-Evaluation.valtermify Fract* {·} *k* {·} *l*

notation *fcomp* (**infixl** ∘> *60*)
notation *scomp* (**infixl** ∘→ *60*)

instantiation *rat* :: *random*
begin

definition
  Quickcheck-Random.random i =
    Quickcheck-Random.random i ∘→ (λ*num. Random.range i* ∘→ (λ*denom. Pair*
      (*let j* = *int-of-integer* (*integer-of-natural* (*denom* + *1*))
       in valterm-fract num (*j*, λ*u. Code-Evaluation.term-of j*))))

instance ..

end

no-notation *fcomp* (**infixl** ∘> *60*)
no-notation *scomp* (**infixl** ∘→ *60*)

instantiation *rat* :: *exhaustive*
begin

definition
  exhaustive-rat f d =
    Quickcheck-Exhaustive.exhaustive
      (λ*l. Quickcheck-Exhaustive.exhaustive*
        (λ*k. f* (*Fract k* (*int-of-integer* (*integer-of-natural l*) + *1*))) *d*) *d*

instance ..

end

instantiation *rat* :: *full-exhaustive*
begin

definition
  full-exhaustive-rat f d =
    Quickcheck-Exhaustive.full-exhaustive
      (λ(*l*, -). *Quickcheck-Exhaustive.full-exhaustive*

($\lambda k.$ *f*
(*let* j = *int-of-integer* (*integer-of-natural* l) + *1*
in valterm-fract k (j, λ-. *Code-Evaluation.term-of* j))) d) d

instance ..

end

instance *rat* :: *partial-term-of* **..**

lemma [*code*]:
partial-term-of (*ty* :: *rat itself*) (*Quickcheck-Narrowing.Narrowing-variable p tt*) $\equiv$
Code-Evaluation.Free (*STR* ''-'') (*Typerep.Typerep* (*STR* ''Rat.rat'') [])
partial-term-of (*ty* :: *rat itself*) (*Quickcheck-Narrowing.Narrowing-constructor 0* [l, k]) $\equiv$
Code-Evaluation.App
(*Code-Evaluation.Const* (*STR* ''Rat.Frct'')
(*Typerep.Typerep* (*STR* ''fun'')
[*Typerep.Typerep* (*STR* ''Product-Type.prod'')
[*Typerep.Typerep* (*STR* ''Int.int'') [], *Typerep.Typerep* (*STR* ''Int.int'') []],
Typerep.Typerep (*STR* ''Rat.rat'') []]))
(*Code-Evaluation.App*
(*Code-Evaluation.App*
(*Code-Evaluation.Const* (*STR* ''Product-Type.Pair'')
(*Typerep.Typerep* (*STR* ''fun'')
[*Typerep.Typerep* (*STR* ''Int.int'') [],
Typerep.Typerep (*STR* ''fun'')
[*Typerep.Typerep* (*STR* ''Int.int'') [],
Typerep.Typerep (*STR* ''Product-Type.prod'')
[*Typerep.Typerep* (*STR* ''Int.int'') [], *Typerep.Typerep* (*STR* ''Int.int'') []]]]))
(*partial-term-of* (*TYPE*(*int*)) l)) (*partial-term-of* (*TYPE*(*int*)) k))
by (*rule partial-term-of-anything*)+

instantiation *rat* :: *narrowing*
begin

definition
narrowing =
Quickcheck-Narrowing.apply
(*Quickcheck-Narrowing.apply*
(*Quickcheck-Narrowing.cons* (λ*nom denom. Fract nom denom*)) *narrowing*)
narrowing

instance ..

end

7.6 Setup for Nitpick

declaration ‹
  Nitpick-HOL.register-frac-type ***type-name*** ‹*rat*›
    [(***const-name*** ‹*Abs-Rat*›, ***const-name*** ‹*Nitpick.Abs-Frac*›),
     (***const-name*** ‹*zero-rat-inst.zero-rat*›, ***const-name*** ‹*Nitpick.zero-frac*›),
     (***const-name*** ‹*one-rat-inst.one-rat*›, ***const-name*** ‹*Nitpick.one-frac*›),
     (***const-name*** ‹*plus-rat-inst.plus-rat*›, ***const-name*** ‹*Nitpick.plus-frac*›),
     (***const-name*** ‹*times-rat-inst.times-rat*›, ***const-name*** ‹*Nitpick.times-frac*›),
     (***const-name*** ‹*uminus-rat-inst.uminus-rat*›, ***const-name*** ‹*Nitpick.uminus-frac*›),
     (***const-name*** ‹*inverse-rat-inst.inverse-rat*›, ***const-name*** ‹*Nitpick.inverse-frac*›),
     (***const-name*** ‹*ord-rat-inst.less-rat*›, ***const-name*** ‹*Nitpick.less-frac*›),
     (***const-name*** ‹*ord-rat-inst.less-eq-rat*›, ***const-name*** ‹*Nitpick.less-eq-frac*›),
     (***const-name*** ‹*field-char-0-class.of-rat*›, ***const-name*** ‹*Nitpick.of-frac*›)]
›

lemmas [*nitpick-unfold*] =
  inverse-rat-inst.inverse-rat
  one-rat-inst.one-rat ord-rat-inst.less-rat
  ord-rat-inst.less-eq-rat plus-rat-inst.plus-rat times-rat-inst.times-rat
  uminus-rat-inst.uminus-rat zero-rat-inst.zero-rat

7.7 Float syntax

syntax *-Float* :: *float-const* ⇒ *'a*   (-)

parse-translation ‹
  let
    fun mk-frac str =
      let
        val {*mant* = *i*, *exp* = *n*} = *Lexicon.read-float str*;
        val exp = *Syntax.const* ***const-syntax*** ‹*Power.power*›;
        val ten = *Numeral.mk-number-syntax 10*;
        val exp10 = *if n* = *1 then ten else exp* $ *ten* $ *Numeral.mk-number-syntax*
n;
      in Syntax.const ***const-syntax*** ‹*Fields.inverse-divide*› $ *Numeral.mk-number-syntax*
i $ *exp10 end*;

    fun float-tr [(*c as Const* (***syntax-const*** ‹*-constrain*›, -)) $ *t* $ *u*] = *c* $ *float-tr*
[*t*] $ *u*
      | *float-tr* [*t as Const* (*str*, -)] = *mk-frac str*
      | *float-tr ts* = *raise TERM* (*float-tr*, *ts*);
  in [(***syntax-const*** ‹*-Float*›, *K float-tr*)] *end*
›

Test:

lemma $123.456 = -111.111 + 200 + 30 + 4 + 5/10 + 6/100 + (7/1000::rat)$
  by *simp*

7.8 Hiding implementation details

hide-const (**open**) *normalize positive*

lifting-update *rat.lifting*
lifting-forget *rat.lifting*

end

8 Development of the Reals using Cauchy Sequences

theory *Real*
imports *Rat*
begin

This theory contains a formalization of the real numbers as equivalence classes of Cauchy sequences of rationals. See `~~/src/HOL/ex/Dedekind_Real.thy` for an alternative construction using Dedekind cuts.

8.1 Preliminary lemmas

Useful in convergence arguments

lemma *inverse-of-nat-le*:
  fixes $n::nat$ **shows** $[\![n \leq m;\ n\neq 0]\!] \Longrightarrow 1\ /\ \textit{of-nat}\ m \leq (1::'a::\textit{linordered-field})\ /\ \textit{of-nat}\ n$
  by (*simp add*: *frac-le*)

lemma *add-diff-add*: $(a + c) - (b + d) = (a - b) + (c - d)$
  for $a\ b\ c\ d :: 'a::\textit{ab-group-add}$
  by *simp*

lemma *minus-diff-minus*: $-\ a - -\ b = -\ (a - b)$
  for $a\ b :: 'a::\textit{ab-group-add}$
  by *simp*

lemma *mult-diff-mult*: $(x * y - a * b) = x * (y - b) + (x - a) * b$
  for $x\ y\ a\ b :: 'a::\textit{ring}$
  by (*simp add*: *algebra-simps*)

lemma *inverse-diff-inverse*:
  fixes $a\ b :: 'a::\textit{division-ring}$
  assumes $a \neq 0$ **and** $b \neq 0$
  shows $\textit{inverse}\ a - \textit{inverse}\ b = -\ (\textit{inverse}\ a * (a - b) * \textit{inverse}\ b)$
  using *assms* **by** (*simp add*: *algebra-simps*)

lemma *obtain-pos-sum*:
  fixes $r :: rat$ **assumes** r: $0 < r$
  obtains $s\ t$ **where** $0 < s$ **and** $0 < t$ **and** $r = s + t$

proof
  from *r* **show** $0 < r/2$ **by** *simp*
  from *r* **show** $0 < r/2$ **by** *simp*
  show $r = r/2 + r/2$ **by** *simp*
qed

8.2 Sequences that converge to zero

definition *vanishes* :: $(nat \Rightarrow rat) \Rightarrow bool$
  where *vanishes* $X \longleftrightarrow (\forall r>0.\ \exists k.\ \forall n\geq k.\ |X\ n| < r)$

lemma *vanishesI*: $(\bigwedge r.\ 0 < r \Longrightarrow \exists k.\ \forall n\geq k.\ |X\ n| < r) \Longrightarrow$ *vanishes* X
  unfolding *vanishes-def* **by** *simp*

lemma *vanishesD*: *vanishes* $X \Longrightarrow 0 < r \Longrightarrow \exists k.\ \forall n\geq k.\ |X\ n| < r$
  unfolding *vanishes-def* **by** *simp*

lemma *vanishes-const* [*simp*]: *vanishes* $(\lambda n.\ c) \longleftrightarrow c = 0$
proof (*cases* $c = 0$)
  case *True*
  then show *?thesis*
    by (*simp add*: *vanishesI*)
next
  case *False*
  then show *?thesis*
    unfolding *vanishes-def*
    using *zero-less-abs-iff* **by** *blast*
qed

lemma *vanishes-minus*: *vanishes* $X \Longrightarrow$ *vanishes* $(\lambda n.\ -\ X\ n)$
  unfolding *vanishes-def* **by** *simp*

lemma *vanishes-add*:
  assumes *X*: *vanishes X*
    and *Y*: *vanishes Y*
  shows *vanishes* $(\lambda n.\ X\ n + Y\ n)$
proof (*rule vanishesI*)
  fix *r* :: *rat*
  assume $0 < r$
  then obtain *s t* **where** *s*: $0 < s$ **and** *t*: $0 < t$ **and** *r*: $r = s + t$
    by (*rule obtain-pos-sum*)
  obtain *i* **where** *i*: $\forall n\geq i.\ |X\ n| < s$
    using *vanishesD* [*OF X s*] **..**
  obtain *j* **where** *j*: $\forall n\geq j.\ |Y\ n| < t$
    using *vanishesD* [*OF Y t*] **..**
  have $\forall n\geq max\ i\ j.\ |X\ n + Y\ n| < r$
  proof *clarsimp*
    fix *n*
    assume *n*: $i \leq n\ j \leq n$

have $|X\ n\ +\ Y\ n| \le |X\ n| + |Y\ n|$
by (*rule abs-triangle-ineq*)
also have $\ldots < s + t$
by (*simp add*: *add-strict-mono i j n*)
finally show $|X\ n\ +\ Y\ n| < r$
by (*simp only*: *r*)
qed
then show $\exists\, k.\ \forall\, n{\ge}k.\ |X\ n\ +\ Y\ n| < r$ **..**
qed

lemma *vanishes-diff*:
assumes *vanishes X vanishes Y*
shows *vanishes* $(\lambda n.\ X\ n\ -\ Y\ n)$
unfolding *diff-conv-add-uminus* **by** (*intro vanishes-add vanishes-minus assms*)

lemma *vanishes-mult-bounded*:
assumes X: $\exists\, a{>}0.\ \forall\, n.\ |X\ n| < a$
assumes Y: *vanishes* $(\lambda n.\ Y\ n)$
shows *vanishes* $(\lambda n.\ X\ n * Y\ n)$
proof (*rule vanishesI*)
fix r :: *rat*
assume r: $0 < r$
obtain a **where** a: $0 < a\ \forall\, n.\ |X\ n| < a$
using X **by** *blast*
obtain b **where** b: $0 < b\ r = a * b$
proof
show $0 < r\ /\ a$ **using** $r\ a$ **by** *simp*
show $r = a * (r\ /\ a)$ **using** a **by** *simp*
qed
obtain k **where** k: $\forall\, n{\ge}k.\ |Y\ n| < b$
using *vanishesD* [*OF Y b*(*1*)] **..**
have $\forall\, n{\ge}k.\ |X\ n * Y\ n| < r$
by (*simp add*: *b*(*2*) *abs-mult mult-strict-mono′ a k*)
then show $\exists\, k.\ \forall\, n{\ge}k.\ |X\ n * Y\ n| < r$ **..**
qed

8.3 Cauchy sequences

definition *cauchy* :: $(nat \Rightarrow rat) \Rightarrow bool$
where *cauchy* $X \longleftrightarrow (\forall\, r{>}0.\ \exists\, k.\ \forall\, m{\ge}k.\ \forall\, n{\ge}k.\ |X\ m\ -\ X\ n| < r)$

lemma *cauchyI*: $(\bigwedge r.\ 0 < r \Longrightarrow \exists\, k.\ \forall\, m{\ge}k.\ \forall\, n{\ge}k.\ |X\ m\ -\ X\ n| < r) \Longrightarrow$ *cauchy* X
unfolding *cauchy-def* **by** *simp*

lemma *cauchyD*: *cauchy* $X \Longrightarrow 0 < r \Longrightarrow \exists\, k.\ \forall\, m{\ge}k.\ \forall\, n{\ge}k.\ |X\ m\ -\ X\ n| < r$
unfolding *cauchy-def* **by** *simp*

lemma *cauchy-const* [*simp*]: *cauchy* $(\lambda n.\ x)$

unfolding *cauchy-def* **by** *simp*

lemma *cauchy-add* [*simp*]:
assumes X: *cauchy* X **and** Y: *cauchy* Y
shows *cauchy* $(\lambda n.\ X\ n + Y\ n)$
proof (*rule cauchyI*)
fix r :: *rat*
assume $0 < r$
then obtain s t **where** s: $0 < s$ **and** t: $0 < t$ **and** r: $r = s + t$
by (*rule obtain-pos-sum*)
obtain i **where** i: $\forall m \geq i.\ \forall n \geq i.\ |X\ m - X\ n| < s$
using *cauchyD* [*OF X s*] **..**
obtain j **where** j: $\forall m \geq j.\ \forall n \geq j.\ |Y\ m - Y\ n| < t$
using *cauchyD* [*OF Y t*] **..**
have $\forall m \geq max\ i\ j.\ \forall n \geq max\ i\ j.\ |(X\ m + Y\ m) - (X\ n + Y\ n)| < r$
proof *clarsimp*
fix m n
assume $*$: $i \leq m$ $j \leq m$ $i \leq n$ $j \leq n$
have $|(X\ m + Y\ m) - (X\ n + Y\ n)| \leq |X\ m - X\ n| + |Y\ m - Y\ n|$
unfolding *add-diff-add* **by** (*rule abs-triangle-ineq*)
also have $\ldots < s + t$
by (*rule add-strict-mono*) (*simp-all add*: i j $*$)
finally show $|(X\ m + Y\ m) - (X\ n + Y\ n)| < r$ **by** (*simp only*: r)
qed
then show $\exists k.\ \forall m \geq k.\ \forall n \geq k.\ |(X\ m + Y\ m) - (X\ n + Y\ n)| < r$ **..**
qed

lemma *cauchy-minus* [*simp*]:
assumes X: *cauchy* X
shows *cauchy* $(\lambda n.\ - X\ n)$
using *assms* **unfolding** *cauchy-def*
unfolding *minus-diff-minus abs-minus-cancel* **.**

lemma *cauchy-diff* [*simp*]:
assumes *cauchy* X *cauchy* Y
shows *cauchy* $(\lambda n.\ X\ n - Y\ n)$
using *assms* **unfolding** *diff-conv-add-uminus* **by** (*simp del*: *add-uminus-conv-diff*)

lemma *cauchy-imp-bounded*:
assumes *cauchy* X
shows $\exists b>0.\ \forall n.\ |X\ n| < b$
proof −
obtain k **where** k: $\forall m \geq k.\ \forall n \geq k.\ |X\ m - X\ n| < 1$
using *cauchyD* [*OF assms zero-less-one*] **..**
show $\exists b>0.\ \forall n.\ |X\ n| < b$
proof (*intro exI conjI allI*)
have $0 \leq |X\ 0|$ **by** *simp*
also have $|X\ 0| \leq Max\ (abs\ `\ X\ `\ \{..k\})$ **by** *simp*
finally have $0 \leq Max\ (abs\ `\ X\ `\ \{..k\})$ **.**

then show $0 < Max\ (abs\ `\ X\ `\ \{..k\}) + 1$ **by** *simp*
next
fix $n :: nat$
show $|X\ n| < Max\ (abs\ `\ X\ `\ \{..k\}) + 1$
proof (*rule linorder-le-cases*)
assume $n \leq k$
then have $|X\ n| \leq Max\ (abs\ `\ X\ `\ \{..k\})$ **by** *simp*
then show $|X\ n| < Max\ (abs\ `\ X\ `\ \{..k\}) + 1$ **by** *simp*
next
assume $k \leq n$
have $|X\ n| = |X\ k + (X\ n - X\ k)|$ **by** *simp*
also have $|X\ k + (X\ n - X\ k)| \leq |X\ k| + |X\ n - X\ k|$
by (*rule abs-triangle-ineq*)
also have $\ldots < Max\ (abs\ `\ X\ `\ \{..k\}) + 1$
by (*rule add-le-less-mono*) (*simp-all add*: k ‹$k \leq n$›)
finally show $|X\ n| < Max\ (abs\ `\ X\ `\ \{..k\}) + 1$ **.**
qed
qed
qed

lemma *cauchy-mult* [*simp*]:
assumes X: *cauchy* X **and** Y: *cauchy* Y
shows *cauchy* $(\lambda n.\ X\ n * Y\ n)$
proof (*rule cauchyI*)
fix $r :: rat$ **assume** $0 < r$
then obtain $u\ v$ **where** u: $0 < u$ **and** v: $0 < v$ **and** $r = u + v$
by (*rule obtain-pos-sum*)
obtain a **where** a: $0 < a\ \forall n.\ |X\ n| < a$
using *cauchy-imp-bounded* [*OF X*] **by** *blast*
obtain b **where** b: $0 < b\ \forall n.\ |Y\ n| < b$
using *cauchy-imp-bounded* [*OF Y*] **by** *blast*
obtain $s\ t$ **where** s: $0 < s$ **and** t: $0 < t$ **and** r: $r = a * t + s * b$
proof
show $0 < v/b$ **using** $v\ b(1)$ **by** *simp*
show $0 < u/a$ **using** $u\ a(1)$ **by** *simp*
show $r = a * (u/a) + (v/b) * b$
using $a(1)\ b(1)$ ‹$r = u + v$› **by** *simp*
qed
obtain i **where** i: $\forall m \geq i.\ \forall n \geq i.\ |X\ m - X\ n| < s$
using *cauchyD* [*OF X s*] **..**
obtain j **where** j: $\forall m \geq j.\ \forall n \geq j.\ |Y\ m - Y\ n| < t$
using *cauchyD* [*OF Y t*] **..**
have $\forall m \geq max\ i\ j.\ \forall n \geq max\ i\ j.\ |X\ m * Y\ m - X\ n * Y\ n| < r$
proof *clarsimp*
fix $m\ n$
assume $*$: $i \leq m\ j \leq m\ i \leq n\ j \leq n$
have $|X\ m * Y\ m - X\ n * Y\ n| = |X\ m * (Y\ m - Y\ n) + (X\ m - X\ n) * Y$
$n|$
unfolding *mult-diff-mult* **..**

also have ... $\leq$ $|X\ m * (Y\ m - Y\ n)| + |(X\ m - X\ n) * Y\ n|$
by (*rule abs-triangle-ineq*)
also have ... $= |X\ m| * |Y\ m - Y\ n| + |X\ m - X\ n| * |Y\ n|$
unfolding *abs-mult* **..**
also have ... $< a * t + s * b$
by (*simp-all add*: *add-strict-mono mult-strict-mono′ a b i j* $*$)
finally show $|X\ m * Y\ m - X\ n * Y\ n| < r$
by (*simp only*: *r*)
qed
then show $\exists\, k.\ \forall\, m{\geq}k.\ \forall\, n{\geq}k.\ |X\ m * Y\ m - X\ n * Y\ n| < r$ **..**
qed

lemma *cauchy-not-vanishes-cases*:
assumes *X*: *cauchy X*
assumes *nz*: $\neg$ *vanishes X*
shows $\exists\, b{>}0.\ \exists\, k.\ (\forall\, n{\geq}k.\ b < -\ X\ n) \vee (\forall\, n{\geq}k.\ b < X\ n)$
proof −
obtain *r* **where** $0 < r$ **and** *r*: $\forall\, k.\ \exists\, n{\geq}k.\ r \leq |X\ n|$
using *nz* **unfolding** *vanishes-def* **by** (*auto simp add*: *not-less*)
obtain *s t* **where** *s*: $0 < s$ **and** *t*: $0 < t$ **and** $r = s + t$
using ‹$0 < r$› **by** (*rule obtain-pos-sum*)
obtain *i* **where** *i*: $\forall\, m{\geq}i.\ \forall\, n{\geq}i.\ |X\ m - X\ n| < s$
using *cauchyD* [*OF X s*] **..**
obtain *k* **where** $i \leq k$ **and** $r \leq |X\ k|$
using *r* **by** *blast*
have *k*: $\forall\, n{\geq}k.\ |X\ n - X\ k| < s$
using *i* ‹$i \leq k$› **by** *auto*
have $X\ k \leq -\ r \vee r \leq X\ k$
using ‹$r \leq |X\ k|$› **by** *auto*
then have $(\forall\, n{\geq}k.\ t < -\ X\ n) \vee (\forall\, n{\geq}k.\ t < X\ n)$
unfolding ‹$r = s + t$› **using** *k* **by** *auto*
then have $\exists\, k.\ (\forall\, n{\geq}k.\ t < -\ X\ n) \vee (\forall\, n{\geq}k.\ t < X\ n)$ **..**
then show $\exists\, t{>}0.\ \exists\, k.\ (\forall\, n{\geq}k.\ t < -\ X\ n) \vee (\forall\, n{\geq}k.\ t < X\ n)$
using *t* **by** *auto*
qed

lemma *cauchy-not-vanishes*:
assumes *X*: *cauchy X*
and *nz*: $\neg$ *vanishes X*
shows $\exists\, b{>}0.\ \exists\, k.\ \forall\, n{\geq}k.\ b < |X\ n|$
using *cauchy-not-vanishes-cases* [*OF assms*]
by (*elim ex-forward conj-forward asm-rl*) *auto*

lemma *cauchy-inverse* [*simp*]:
assumes *X*: *cauchy X*
and *nz*: $\neg$ *vanishes X*
shows *cauchy* ($\lambda n.$ *inverse* (*X n*))
proof (*rule cauchyI*)
fix *r* :: *rat*

assume $0 < r$
obtain b i **where** b: $0 < b$ **and** i: $\forall n \geq i.\ b < |X\ n|$
  using *cauchy-not-vanishes* [*OF X nz*] **by** *blast*
from b i **have** *nz*: $\forall n \geq i.\ X\ n \neq 0$ **by** *auto*
obtain s **where** s: $0 < s$ **and** r: $r = inverse\ b * s * inverse\ b$
proof
  show $0 < b * r * b$ **by** (*simp add*: ‹$0 < r$› b)
  show $r = inverse\ b * (b * r * b) * inverse\ b$
    using b **by** *simp*
qed
obtain j **where** j: $\forall m \geq j.\ \forall n \geq j.\ |X\ m - X\ n| < s$
  using *cauchyD* [*OF X s*] **..**
have $\forall m \geq max\ i\ j.\ \forall n \geq max\ i\ j.\ |inverse\ (X\ m) - inverse\ (X\ n)| < r$
proof *clarsimp*
  fix m n
  assume $*$: $i \leq m$ $j \leq m$ $i \leq n$ $j \leq n$
  have $|inverse\ (X\ m) - inverse\ (X\ n)| = inverse\ |X\ m| * |X\ m - X\ n| * inverse$
$|X\ n|$
    by (*simp add*: *inverse-diff-inverse nz* $*$ *abs-mult*)
  also have $\ldots < inverse\ b * s * inverse\ b$
    by (*simp add*: *mult-strict-mono less-imp-inverse-less* i j b $*$ s)
  finally show $|inverse\ (X\ m) - inverse\ (X\ n)| < r$ **by** (*simp only*: r)
qed
then show $\exists k.\ \forall m \geq k.\ \forall n \geq k.\ |inverse\ (X\ m) - inverse\ (X\ n)| < r$ **..**
qed

lemma *vanishes-diff-inverse*:
  assumes X: *cauchy* X $\neg$ *vanishes* X
    and Y: *cauchy* Y $\neg$ *vanishes* Y
    and XY: *vanishes* $(\lambda n.\ X\ n - Y\ n)$
  shows *vanishes* $(\lambda n.\ inverse\ (X\ n) - inverse\ (Y\ n))$
proof (*rule vanishesI*)
  fix r :: *rat*
  assume r: $0 < r$
  obtain a i **where** a: $0 < a$ **and** i: $\forall n \geq i.\ a < |X\ n|$
    using *cauchy-not-vanishes* [*OF X*] **by** *blast*
  obtain b j **where** b: $0 < b$ **and** j: $\forall n \geq j.\ b < |Y\ n|$
    using *cauchy-not-vanishes* [*OF Y*] **by** *blast*
  obtain s **where** s: $0 < s$ **and** $inverse\ a * s * inverse\ b = r$
  proof
    show $0 < a * r * b$
      using a r b **by** *simp*
    show $inverse\ a * (a * r * b) * inverse\ b = r$
      using a r b **by** *simp*
  qed
  obtain k **where** k: $\forall n \geq k.\ |X\ n - Y\ n| < s$
    using *vanishesD* [*OF XY s*] **..**
  have $\forall n \geq max\ (max\ i\ j)\ k.\ |inverse\ (X\ n) - inverse\ (Y\ n)| < r$
  proof *clarsimp*

fix n
assume n: $i \leq n$ $j \leq n$ $k \leq n$
with i j a b **have** $X\ n \neq 0$ **and** $Y\ n \neq 0$
by *auto*
then have $|inverse\ (X\ n) - inverse\ (Y\ n)| = inverse\ |X\ n| * |X\ n - Y\ n| * inverse\ |Y\ n|$
by (*simp add*: *inverse-diff-inverse abs-mult*)
also have $\ldots < inverse\ a * s * inverse\ b$
by (*intro mult-strict-mono′ less-imp-inverse-less*) (*simp-all add*: *a b i j k n*)
also note ‹$inverse\ a * s * inverse\ b = r$›
finally show $|inverse\ (X\ n) - inverse\ (Y\ n)| < r$ **.**
qed
then show $\exists k.\ \forall n \geq k.\ |inverse\ (X\ n) - inverse\ (Y\ n)| < r$ **..**
qed

8.4 Equivalence relation on Cauchy sequences

definition *realrel* :: $(nat \Rightarrow rat) \Rightarrow (nat \Rightarrow rat) \Rightarrow bool$
where $realrel = (\lambda X\ Y.\ cauchy\ X \land cauchy\ Y \land vanishes\ (\lambda n.\ X\ n - Y\ n))$

lemma *realrelI* [*intro?*]: $cauchy\ X \Longrightarrow cauchy\ Y \Longrightarrow vanishes\ (\lambda n.\ X\ n - Y\ n) \Longrightarrow realrel\ X\ Y$
by (*simp add*: *realrel-def*)

lemma *realrel-refl*: $cauchy\ X \Longrightarrow realrel\ X\ X$
by (*simp add*: *realrel-def*)

lemma *symp-realrel*: *symp realrel*
by (*simp add*: *abs-minus-commute realrel-def symp-def vanishes-def*)

lemma *transp-realrel*: *transp realrel*
unfolding *realrel-def*
by (*rule transpI*) (*force simp add*: *dest*: *vanishes-add*)

lemma *part-equivp-realrel*: *part-equivp realrel*
by (*blast intro*: *part-equivpI symp-realrel transp-realrel realrel-refl cauchy-const*)

8.5 The field of real numbers

quotient-type $real = nat \Rightarrow rat$ / *partial*: *realrel*
morphisms *rep-real Real*
by (*rule part-equivp-realrel*)

lemma *cr-real-eq*: $pcr\text{-}real = (\lambda x\ y.\ cauchy\ x \land Real\ x = y)$
unfolding *real.pcr-cr-eq cr-real-def realrel-def* **by** *auto*

lemma *Real-induct* [*induct type*: *real*]:
assumes $\bigwedge X.\ cauchy\ X \Longrightarrow P\ (Real\ X)$
shows $P\ x$
proof (*induct x*)

case (*1 X*)
then have *cauchy X* **by** (*simp add*: *realrel-def*)
then show *P* (*Real X*) **by** (*rule assms*)
qed

lemma *eq-Real*: *cauchy X* $\Longrightarrow$ *cauchy Y* $\Longrightarrow$ *Real X* = *Real Y* $\longleftrightarrow$ *vanishes* ($\lambda n.$ $X\ n - Y\ n$)
using *real.rel-eq-transfer*
unfolding *real.pcr-cr-eq cr-real-def rel-fun-def realrel-def* **by** *simp*

lemma *Domainp-pcr-real* [*transfer-domain-rule*]: *Domainp pcr-real* = *cauchy*
by (*simp add*: *real.domain-eq realrel-def*)

instantiation *real* :: *field*
begin

lift-definition *zero-real* :: *real* **is** $\lambda n.\ 0$
by (*simp add*: *realrel-refl*)

lift-definition *one-real* :: *real* **is** $\lambda n.\ 1$
by (*simp add*: *realrel-refl*)

lift-definition *plus-real* :: *real* $\Rightarrow$ *real* $\Rightarrow$ *real* **is** $\lambda X\ Y\ n.\ X\ n + Y\ n$
unfolding *realrel-def add-diff-add*
by (*simp only*: *cauchy-add vanishes-add simp-thms*)

lift-definition *uminus-real* :: *real* $\Rightarrow$ *real* **is** $\lambda X\ n.\ -\ X\ n$
unfolding *realrel-def minus-diff-minus*
by (*simp only*: *cauchy-minus vanishes-minus simp-thms*)

lift-definition *times-real* :: *real* $\Rightarrow$ *real* $\Rightarrow$ *real* **is** $\lambda X\ Y\ n.\ X\ n * Y\ n$
proof −
fix *f1 f2 f3 f4*
have $[\![$*cauchy f1*; *cauchy f4*; *vanishes* ($\lambda n.\ f1\ n - f2\ n$); *vanishes* ($\lambda n.\ f3\ n - f4\ n$)$]\!]$
$\Longrightarrow$ *vanishes* ($\lambda n.\ f1\ n * (f3\ n - f4\ n) + f4\ n * (f1\ n - f2\ n)$)
by (*simp add*: *vanishes-add vanishes-mult-bounded cauchy-imp-bounded*)
then show $[\![$*realrel f1 f2*; *realrel f3 f4*$]\!]$ $\Longrightarrow$ *realrel* ($\lambda n.\ f1\ n * f3\ n$) ($\lambda n.\ f2\ n * f4\ n$)
by (*simp add*: *mult.commute realrel-def mult-diff-mult*)
qed

lift-definition *inverse-real* :: *real* $\Rightarrow$ *real*
is $\lambda X.$ *if vanishes X then* ($\lambda n.\ 0$) *else* ($\lambda n.$ *inverse* ($X\ n$))
proof −
fix *X Y*
assume *realrel X Y*
then have *X*: *cauchy X* **and** *Y*: *cauchy Y* **and** *XY*: *vanishes* ($\lambda n.\ X\ n - Y\ n$)
by (*simp-all add*: *realrel-def*)

have *vanishes X* $\longleftrightarrow$ *vanishes Y*
proof
assume *vanishes X*
from *vanishes-diff* [*OF this XY*] **show** *vanishes Y*
by *simp*
next
assume *vanishes Y*
from *vanishes-add* [*OF this XY*] **show** *vanishes X*
by *simp*
qed
then show *?thesis X Y*
by (*simp add*: *vanishes-diff-inverse X Y XY realrel-def*)
qed

definition $x - y = x + - y$ **for** $x\ y$ :: *real*

definition $x\ div\ y = x * inverse\ y$ **for** $x\ y$ :: *real*

lemma *add-Real*: *cauchy X* $\Longrightarrow$ *cauchy Y* $\Longrightarrow$ *Real X* + *Real Y* = *Real* ($\lambda n.$ *X n* + *Y n*)
using *plus-real.transfer* **by** (*simp add*: *cr-real-eq rel-fun-def*)

lemma *minus-Real*: *cauchy X* $\Longrightarrow$ − *Real X* = *Real* ($\lambda n.$ − *X n*)
using *uminus-real.transfer* **by** (*simp add*: *cr-real-eq rel-fun-def*)

lemma *diff-Real*: *cauchy X* $\Longrightarrow$ *cauchy Y* $\Longrightarrow$ *Real X* − *Real Y* = *Real* ($\lambda n.$ *X n* − *Y n*)
by (*simp add*: *minus-Real add-Real minus-real-def*)

lemma *mult-Real*: *cauchy X* $\Longrightarrow$ *cauchy Y* $\Longrightarrow$ *Real X* ∗ *Real Y* = *Real* ($\lambda n.$ *X n* ∗ *Y n*)
using *times-real.transfer* **by** (*simp add*: *cr-real-eq rel-fun-def*)

lemma *inverse-Real*:
cauchy X $\Longrightarrow$ *inverse* (*Real X*) = (*if vanishes X then 0 else Real* ($\lambda n.$ *inverse* (*X n*)))
using *inverse-real.transfer zero-real.transfer*
unfolding *cr-real-eq rel-fun-def* **by** (*simp split*: *if-split-asm*, *metis*)

instance
proof
fix *a b c* :: *real*
show $a + b = b + a$
by *transfer* (*simp add*: *ac-simps realrel-def*)
show $(a + b) + c = a + (b + c)$
by *transfer* (*simp add*: *ac-simps realrel-def*)
show $0 + a = a$
by *transfer* (*simp add*: *realrel-def*)
show $- a + a = 0$

```
    by transfer (simp add: realrel-def)
  show a − b = a + − b
    by (rule minus-real-def)
  show (a ∗ b) ∗ c = a ∗ (b ∗ c)
    by transfer (simp add: ac-simps realrel-def)
  show a ∗ b = b ∗ a
    by transfer (simp add: ac-simps realrel-def)
  show 1 ∗ a = a
    by transfer (simp add: ac-simps realrel-def)
  show (a + b) ∗ c = a ∗ c + b ∗ c
    by transfer (simp add: distrib-right realrel-def)
  show (0::real) ≠ (1::real)
    by transfer (simp add: realrel-def)
  have vanishes (λn. inverse (X n) ∗ X n − 1) if X: cauchy X ¬ vanishes X for
X
  proof (rule vanishesI)
    fix r::rat
    assume 0 < r
    obtain b k where b>0 ∀ n≥k. b < |X n|
      using X cauchy-not-vanishes by blast
    then show ∃ k. ∀ n≥k. |inverse (X n) ∗ X n − 1| < r
      using ‹0 < r› by force
  qed
  then show a ≠ 0 =⇒ inverse a ∗ a = 1
    by transfer (simp add: realrel-def)
  show a div b = a ∗ inverse b
    by (rule divide-real-def)
  show inverse (0::real) = 0
    by transfer (simp add: realrel-def)
qed

end
```

8.6 Positive reals

```
lift-definition positive :: real ⇒ bool
  is λX. ∃ r>0. ∃ k. ∀ n≥k. r < X n
proof −
  have 1: ∃ r>0. ∃ k. ∀ n≥k. r < Y n
    if ∗: realrel X Y and ∗∗: ∃ r>0. ∃ k. ∀ n≥k. r < X n for X Y
  proof −
    from ∗ have XY: vanishes (λn. X n − Y n)
      by (simp-all add: realrel-def)
    from ∗∗ obtain r i where 0 < r and i: ∀ n≥i. r < X n
      by blast
    obtain s t where s: 0 < s and t: 0 < t and r: r = s + t
      using ‹0 < r› by (rule obtain-pos-sum)
    obtain j where j: ∀ n≥j. |X n − Y n| < s
      using vanishesD [OF XY s] ..
```

have $\forall n \geq max\ i\ j.\ t < Y\ n$
proof *clarsimp*
fix *n*
assume n: $i \leq n$ $j \leq n$
have $|X\ n - Y\ n| < s$ **and** $r < X\ n$
using *i j n* **by** *simp-all*
then show $t < Y\ n$ **by** (*simp add*: *r*)
qed
then show *?thesis* **using** *t* **by** *blast*
qed
fix *X Y* **assume** *realrel X Y*
then have *realrel X Y* **and** *realrel Y X*
using *symp-realrel* **by** (*auto simp*: *symp-def*)
then show *?thesis X Y*
by (*safe elim*!: *1*)
qed

lemma *positive-Real*: *cauchy* $X \Longrightarrow$ *positive* (*Real X*) $\longleftrightarrow$ $(\exists r>0.\ \exists k.\ \forall n\geq k.\ r < X\ n)$
using *positive.transfer* **by** (*simp add*: *cr-real-eq rel-fun-def*)

lemma *positive-zero*: $\neg$ *positive 0*
by *transfer auto*

lemma *positive-add*:
assumes *positive x positive y* **shows** *positive* $(x + y)$
proof $-$
have $*$: $[\![\forall n\geq i.\ a < x\ n;\ \forall n\geq j.\ b < y\ n;\ 0 < a;\ 0 < b;\ n \geq max\ i\ j]\!]$
$\Longrightarrow a+b < x\ n + y\ n$ **for** *x y* **and** *a b*::*rat* **and** *i j n*::*nat*
by (*simp add*: *add-strict-mono*)
show *?thesis*
using *assms*
by *transfer* (*blast intro*: $*$ *pos-add-strict*)
qed

lemma *positive-mult*:
assumes *positive x positive y* **shows** *positive* $(x * y)$
proof $-$
have $*$: $[\![\forall n\geq i.\ a < x\ n;\ \forall n\geq j.\ b < y\ n;\ 0 < a;\ 0 < b;\ n \geq max\ i\ j]\!]$
$\Longrightarrow a*b < x\ n * y\ n$ **for** *x y* **and** *a b*::*rat* **and** *i j n*::*nat*
by (*simp add*: *mult-strict-mono'*)
show *?thesis*
using *assms*
by *transfer* (*blast intro*: $*$ *mult-pos-pos*)
qed

lemma *positive-minus*: $\neg$ *positive x* $\Longrightarrow x \neq 0 \Longrightarrow$ *positive* $(- x)$
apply *transfer*
apply (*simp add*: *realrel-def*)

apply (*blast dest*: *cauchy-not-vanishes-cases*)
done

instantiation *real* :: *linordered-field*
begin

definition $x < y \longleftrightarrow$ *positive* $(y - x)$

definition $x \leq y \longleftrightarrow x < y \lor x = y$ **for** $x\ y$ :: *real*

definition $|a| = (if\ a < 0\ then\ - a\ else\ a)$ **for** a :: *real*

definition $sgn\ a = (if\ a = 0\ then\ 0\ else\ if\ 0 < a\ then\ 1\ else\ - 1)$ **for** a :: *real*

instance
proof
  fix $a\ b\ c$ :: *real*
  show $|a| = (if\ a < 0\ then\ - a\ else\ a)$
    by (*rule abs-real-def*)
  show $a < b \longleftrightarrow a \leq b \land \neg\ b \leq a$
      $a \leq b \Longrightarrow b \leq c \Longrightarrow a \leq c \quad a \leq a$
      $a \leq b \Longrightarrow b \leq a \Longrightarrow a = b$
      $a \leq b \Longrightarrow c + a \leq c + b$
    unfolding *less-eq-real-def less-real-def*
    by (*force simp add*: *positive-zero dest*: *positive-add*)+
  show $sgn\ a = (if\ a = 0\ then\ 0\ else\ if\ 0 < a\ then\ 1\ else\ - 1)$
    by (*rule sgn-real-def*)
  show $a \leq b \lor b \leq a$
    by (*auto dest*!: *positive-minus simp*: *less-eq-real-def less-real-def*)
  show $a < b \Longrightarrow 0 < c \Longrightarrow c * a < c * b$
    unfolding *less-real-def*
    by (*force simp add*: *algebra-simps dest*: *positive-mult*)
qed

end

instantiation *real* :: *distrib-lattice*
begin

definition $(inf :: real \Rightarrow real \Rightarrow real) = min$

definition $(sup :: real \Rightarrow real \Rightarrow real) = max$

instance
  by *standard* (*auto simp add*: *inf-real-def sup-real-def max-min-distrib2*)

end

lemma *of-nat-Real*: $of\text{-}nat\ x = Real\ (\lambda n.\ of\text{-}nat\ x)$

by (*induct x*) (*simp-all add*: *zero-real-def one-real-def add-Real*)

lemma *of-int-Real*: *of-int x* = *Real* (λn. *of-int x*)
by (*cases x rule*: *int-diff-cases*) (*simp add*: *of-nat-Real diff-Real*)

lemma *of-rat-Real*: *of-rat x* = *Real* (λn. *x*)
proof (*induct x*)
  case (*Fract a b*)
  then show *?case*
  apply (*simp add*: *Fract-of-int-quotient of-rat-divide*)
  apply (*simp add*: *of-int-Real divide-inverse inverse-Real mult-Real*)
  done
qed

instance *real* :: *archimedean-field*
proof
  show $\exists z$. $x \leq$ *of-int z* **for** *x* :: *real*
  proof (*induct x*)
    case (*1 X*)
    then obtain *b* **where** $0 < b$ **and** *b*: $\bigwedge n$. $|X\ n| < b$
      by (*blast dest*: *cauchy-imp-bounded*)
    then have *Real X* < *of-int* ($\lceil b \rceil + 1$)
      using *1*
      apply (*simp add*: *of-int-Real less-real-def diff-Real positive-Real*)
      apply (*rule-tac x=1* **in** *exI*)
      apply (*simp add*: *algebra-simps*)
      by (*metis abs-ge-self le-less-trans le-of-int-ceiling less-le*)
    then show *?case*
      using *less-eq-real-def* **by** *blast*
  qed
qed

instantiation *real* :: *floor-ceiling*
begin

definition [*code del*]: $\lfloor x{::}real \rfloor$ = (*THE z*. *of-int z* $\leq x \wedge x <$ *of-int* ($z + 1$))

instance
proof
  show *of-int* $\lfloor x \rfloor \leq x \wedge x <$ *of-int* ($\lfloor x \rfloor + 1$) **for** *x* :: *real*
    unfolding *floor-real-def* **using** *floor-exists1* **by** (*rule theI'*)
qed

end

8.7 Completeness

lemma *not-positive-Real*:
  assumes *cauchy X* **shows** $\neg$ *positive* (*Real X*) $\longleftrightarrow$ ($\forall r>0$. $\exists k$. $\forall n \geq k$. $X\ n \leq$

r) (**is** *?lhs* = *?rhs*)
  unfolding *positive-Real* [*OF assms*]
proof (*intro iffI allI notI impI*)
  show $\exists k.\ \forall n \geq k.\ X\ n \leq r$ **if** *r*: $\neg$ ($\exists r>0.\ \exists k.\ \forall n \geq k.\ r < X\ n$) **and** $0 < r$ **for** *r*
  proof −
    obtain *s t* **where** $s > 0$ $t > 0$ $r = s+t$
      using ‹$r > 0$› *obtain-pos-sum* **by** *blast*
    obtain *k* **where** *k*: $\bigwedge m\ n.\ [\![m \geq k;\ n \geq k]\!] \Longrightarrow |X\ m - X\ n| < t$
      using *cauchyD* [*OF assms* ‹$t > 0$›] **by** *blast*
    obtain *n* **where** $n \geq k$ $X\ n \leq s$
      by (*meson r* ‹$0 < s$› *not-less*)
    then have $X\ l \leq r$ **if** $l \geq n$ **for** *l*
      using *k* [*OF* ‹$n \geq k$›, *of l*] *that* ‹$r = s+t$› **by** *linarith*
    then show *?thesis*
      by *blast*
    qed
qed (*meson le-cases not-le*)

lemma *le-Real*:
  assumes *cauchy X cauchy Y*
  shows $Real\ X \leq Real\ Y = (\forall r>0.\ \exists k.\ \forall n \geq k.\ X\ n \leq Y\ n + r)$
  unfolding *not-less* [*symmetric*, **where** *'a=real*] *less-real-def*
  apply (*simp add*: *diff-Real not-positive-Real assms*)
  apply (*simp add*: *diff-le-eq ac-simps*)
  done

lemma *le-RealI*:
  assumes *Y*: *cauchy Y*
  shows $\forall n.\ x \leq of\text{-}rat\ (Y\ n) \Longrightarrow x \leq Real\ Y$
proof (*induct x*)
  fix *X*
  assume *X*: *cauchy X* **and** $\forall n.\ Real\ X \leq of\text{-}rat\ (Y\ n)$
  then have *le*: $\bigwedge m\ r.\ 0 < r \Longrightarrow \exists k.\ \forall n \geq k.\ X\ n \leq Y\ m + r$
    by (*simp add*: *of-rat-Real le-Real*)
  then have $\exists k.\ \forall n \geq k.\ X\ n \leq Y\ n + r$ **if** $0 < r$ **for** *r* :: *rat*
  proof −
    from *that* **obtain** *s t* **where** *s*: $0 < s$ **and** *t*: $0 < t$ **and** *r*: $r = s + t$
      by (*rule obtain-pos-sum*)
    obtain *i* **where** *i*: $\forall m \geq i.\ \forall n \geq i.\ |Y\ m - Y\ n| < s$
      using *cauchyD* [*OF Y s*] **..**
    obtain *j* **where** *j*: $\forall n \geq j.\ X\ n \leq Y\ i + t$
      using *le* [*OF t*] **..**
    have $\forall n \geq max\ i\ j.\ X\ n \leq Y\ n + r$
    proof *clarsimp*
      fix *n*
      assume *n*: $i \leq n$ $j \leq n$
      have $X\ n \leq Y\ i + t$
        using *n j* **by** *simp*

```
      moreover have |Y i − Y n| < s
        using n i by simp
      ultimately show X n ≤ Y n + r
        unfolding r by simp
    qed
    then show ?thesis ..
  qed
  then show Real X ≤ Real Y
    by (simp add: of-rat-Real le-Real X Y)
qed

lemma Real-leI:
  assumes X: cauchy X
  assumes le: ∀ n. of-rat (X n) ≤ y
  shows Real X ≤ y
proof −
  have − y ≤ − Real X
    by (simp add: minus-Real X le-RealI of-rat-minus le)
  then show ?thesis by simp
qed

lemma less-RealD:
  assumes cauchy Y
  shows x < Real Y =⇒ ∃ n. x < of-rat (Y n)
  apply (erule contrapos-pp)
  apply (simp add: not-less)
  apply (erule Real-leI [OF assms])
  done

lemma of-nat-less-two-power [simp]: of-nat n < (2::'a::linordered-idom) ^ n
  apply (induct n)
   apply simp
   apply (metis add-le-less-mono mult-2 of-nat-Suc one-le-numeral one-le-power
power-Suc)
  done

lemma complete-real:
  fixes S :: real set
  assumes ∃ x. x ∈ S and ∃ z. ∀ x∈S. x ≤ z
  shows ∃ y. (∀ x∈S. x ≤ y) ∧ (∀ z. (∀ x∈S. x ≤ z) −→ y ≤ z)
proof −
  obtain x where x: x ∈ S using assms(1) ..
  obtain z where z: ∀ x∈S. x ≤ z using assms(2) ..

  define P where P x ←→ (∀ y∈S. y ≤ of-rat x) for x
  obtain a where a: ¬ P a
  proof
    have of-int ⌊x − 1⌋ ≤ x − 1 by (rule of-int-floor-le)
    also have x − 1 < x by simp
```

```
    finally have of-int ⌊x − 1⌋ < x .
    then have ¬ x ≤ of-int ⌊x − 1⌋ by (simp only: not-le)
    then show ¬ P (of-int ⌊x − 1⌋)
      unfolding P-def of-rat-of-int-eq using x by blast
  qed
  obtain b where b: P b
  proof
    show P (of-int ⌈z⌉)
    unfolding P-def of-rat-of-int-eq
    proof
      fix y assume y ∈ S
      then have y ≤ z using z by simp
      also have z ≤ of-int ⌈z⌉ by (rule le-of-int-ceiling)
      finally show y ≤ of-int ⌈z⌉ .
    qed
  qed

  define avg where avg x y = x/2 + y/2 for x y :: rat
  define bisect where bisect = (λ(x, y). if P (avg x y) then (x, avg x y) else (avg
x y, y))
  define A where A n = fst ((bisect ^^ n) (a, b)) for n
  define B where B n = snd ((bisect ^^ n) (a, b)) for n
  define C where C n = avg (A n) (B n) for n
  have A-0 [simp]: A 0 = a unfolding A-def by simp
  have B-0 [simp]: B 0 = b unfolding B-def by simp
  have A-Suc [simp]: ⋀n. A (Suc n) = (if P (C n) then A n else C n)
    unfolding A-def B-def C-def bisect-def split-def by simp
  have B-Suc [simp]: ⋀n. B (Suc n) = (if P (C n) then C n else B n)
    unfolding A-def B-def C-def bisect-def split-def by simp

  have width: B n − A n = (b − a) / 2^n for n
  proof (induct n)
    case (Suc n)
    then show ?case
      by (simp add: C-def eq-divide-eq avg-def algebra-simps)
  qed simp
  have twos: ∃n. y / 2 ^ n < r if 0 < r for y r :: rat
  proof −
    obtain n where y / r < rat-of-nat n
      using ‹0 < r› reals-Archimedean2 by blast
    then have ∃n. y < r ∗ 2 ^ n
      by (metis divide-less-eq less-trans mult.commute of-nat-less-two-power that)
    then show ?thesis
      by (simp add: divide-simps)
  qed
  have PA: ¬ P (A n) for n
    by (induct n) (simp-all add: a)
  have PB: P (B n) for n
    by (induct n) (simp-all add: b)
```

have *ab*: $a < b$
using *a b* **unfolding** *P-def*
by (*meson leI less-le-trans of-rat-less*)
have *AB*: $A\ n < B\ n$ **for** *n*
by (*induct n*) (*simp-all add*: *ab C-def avg-def*)
have $A\ i \le A\ j \wedge\ B\ j \le B\ i$ **if** $i < j$ **for** *i j*
using *that*
proof (*induction rule*: *less-Suc-induct*)
case (*1 i*)
then show *?case*
apply (*clarsimp simp add*: *C-def avg-def add-divide-distrib* [*symmetric*])
apply (*rule AB* [*THEN less-imp-le*])
done
qed *simp*
then have *A-mono*: $A\ i \le A\ j$ **and** *B-mono*: $B\ j \le B\ i$ **if** $i \le j$ **for** *i j*
by (*metis eq-refl le-neq-implies-less that*)+
have *cauchy-lemma*: *cauchy X* **if** $*$: $\bigwedge n\ i.\ i \ge n \Longrightarrow A\ n \le X\ i \wedge X\ i \le B\ n$ **for**
X
proof (*rule cauchyI*)
fix *r*::*rat*
assume $0 < r$
then obtain *k* **where** *k*: $(b - a)\ /\ 2\ \hat{}\ k < r$
using *twos* **by** *blast*
have $|X\ m - X\ n| < r$ **if** $m \ge k\ n \ge k$ **for** *m n*
proof −
have $|X\ m - X\ n| \le B\ k - A\ k$
by (*simp add*: $*$ *abs-rat-def diff-mono that*)
also have $\ldots < r$
by (*simp add*: *k width*)
finally show *?thesis* **.**
qed
then show $\exists\, k.\ \forall\, m \ge k.\ \forall\, n \ge k.\ |X\ m - X\ n| < r$
by *blast*
qed
have *cauchy A*
by (*rule cauchy-lemma*) (*meson AB A-mono B-mono dual-order.strict-implies-order*
less-le-trans)
have *cauchy B*
by (*rule cauchy-lemma*) (*meson AB A-mono B-mono dual-order.strict-implies-order*
le-less-trans)
have $\forall\, x \in S.\ x \le$ *Real B*
proof
fix *x*
assume $x \in S$
then show $x \le$ *Real B*
using *PB* [*unfolded P-def*] ‹*cauchy B*›
by (*simp add*: *le-RealI*)
qed
moreover have $\forall\, z.\ (\forall\, x \in S.\ x \le z) \longrightarrow$ *Real A* $\le z$

```
    by (meson PA Real-leI P-def ‹cauchy A› le-cases order.trans)
  moreover have vanishes (λn. (b − a) / 2 ^ n)
  proof (rule vanishesI)
    fix r :: rat
    assume 0 < r
    then obtain k where k: |b − a| / 2 ^ k < r
      using twos by blast
    have ∀n≥k. |(b − a) / 2 ^ n| < r
    proof clarify
      fix n
      assume n: k ≤ n
      have |(b − a) / 2 ^ n| = |b − a| / 2 ^ n
        by simp
      also have ... ≤ |b − a| / 2 ^ k
        using n by (simp add: divide-left-mono)
      also note k
      finally show |(b − a) / 2 ^ n| < r .
    qed
    then show ∃k. ∀n≥k. |(b − a) / 2 ^ n| < r ..
  qed
  then have Real B = Real A
    by (simp add: eq-Real ‹cauchy A› ‹cauchy B› width)
  ultimately show ∃y. (∀x∈S. x ≤ y) ∧ (∀z. (∀x∈S. x ≤ z) ⟶ y ≤ z)
    by force
qed

instantiation real :: linear-continuum
begin
```

8.8 Supremum of a set of reals

```
definition Sup X = (LEAST z::real. ∀x∈X. x ≤ z)
definition Inf X = − Sup (uminus ` X) for X :: real set

instance
proof
  show Sup-upper: x ≤ Sup X
    if x ∈ X bdd-above X
    for x :: real and X :: real set
  proof −
    from that obtain s where s: ∀y∈X. y ≤ s ⋀z. ∀y∈X. y ≤ z ⟹ s ≤ z
      using complete-real[of X] unfolding bdd-above-def by blast
    then show ?thesis
      unfolding Sup-real-def by (rule LeastI2-order) (auto simp: that)
  qed
  show Sup-least: Sup X ≤ z
    if X ≠ {} and z: ⋀x. x ∈ X ⟹ x ≤ z
    for z :: real and X :: real set
  proof −
```

from *that* **obtain** s **where** s: $\forall\, y{\in}X.\ y \le s\ \bigwedge z.\ \forall\, y{\in}X.\ y \le z \Longrightarrow s \le z$
using *complete-real* [*of X*] **by** *blast*
then have $Sup\ X = s$
unfolding *Sup-real-def* **by** (*best intro*: *Least-equality*)
also from $s\ z$ **have** $\ldots \le z$
by *blast*
finally show *?thesis* **.**
qed
show $Inf\ X \le x$ **if** $x \in X$ *bdd-below* X
for x :: *real* **and** X :: *real set*
using *Sup-upper* [*of* $-x$ *uminus* ‘ X] **by** (*auto simp*: *Inf-real-def that*)
show $z \le Inf\ X$ **if** $X \neq \{\}\ \bigwedge x.\ x \in X \Longrightarrow z \le x$
for z :: *real* **and** X :: *real set*
using *Sup-least* [*of uminus* ‘ X $-\ z$] **by** (*force simp*: *Inf-real-def that*)
show $\exists\, a\ b{::}real.\ a \neq b$
using *zero-neq-one* **by** *blast*
qed

end

8.9 Hiding implementation details

hide-const (**open**) *vanishes cauchy positive Real*

declare *Real-induct* [*induct del*]
declare *Abs-real-induct* [*induct del*]
declare *Abs-real-cases* [*cases del*]

lifting-update *real.lifting*
lifting-forget *real.lifting*

8.10 More Lemmas

BH: These lemmas should not be necessary; they should be covered by existing simp rules and simplification procedures.

lemma *real-mult-less-iff1* [*simp*]: $0 < z \Longrightarrow x * z < y * z \longleftrightarrow x < y$
for $x\ y\ z$ :: *real*
by *simp*

lemma *real-mult-le-cancel-iff1* [*simp*]: $0 < z \Longrightarrow x * z \le y * z \longleftrightarrow x \le y$
for $x\ y\ z$ :: *real*
by *simp*

lemma *real-mult-le-cancel-iff2* [*simp*]: $0 < z \Longrightarrow z * x \le z * y \longleftrightarrow x \le y$
for $x\ y\ z$ :: *real*
by *simp*

8.11 Embedding numbers into the Reals

abbreviation *real-of-nat* :: *nat* ⇒ *real*
  where *real-of-nat* ≡ *of-nat*

abbreviation *real* :: *nat* ⇒ *real*
  where *real* ≡ *of-nat*

abbreviation *real-of-int* :: *int* ⇒ *real*
  where *real-of-int* ≡ *of-int*

abbreviation *real-of-rat* :: *rat* ⇒ *real*
  where *real-of-rat* ≡ *of-rat*

declare [[*coercion-enabled*]]

declare [[*coercion of-nat* :: *nat* ⇒ *int*]]
declare [[*coercion of-nat* :: *nat* ⇒ *real*]]
declare [[*coercion of-int* :: *int* ⇒ *real*]]

declare [[*coercion-map map*]]
declare [[*coercion-map* $\lambda f\ g\ h\ x.\ g\ (h\ (f\ x))$]]
declare [[*coercion-map* $\lambda f\ g\ (x,y).\ (f\ x,\ g\ y)$]]

declare *of-int-eq-0-iff* [*algebra*, *presburger*]
declare *of-int-eq-1-iff* [*algebra*, *presburger*]
declare *of-int-eq-iff* [*algebra*, *presburger*]
declare *of-int-less-0-iff* [*algebra*, *presburger*]
declare *of-int-less-1-iff* [*algebra*, *presburger*]
declare *of-int-less-iff* [*algebra*, *presburger*]
declare *of-int-le-0-iff* [*algebra*, *presburger*]
declare *of-int-le-1-iff* [*algebra*, *presburger*]
declare *of-int-le-iff* [*algebra*, *presburger*]
declare *of-int-0-less-iff* [*algebra*, *presburger*]
declare *of-int-0-le-iff* [*algebra*, *presburger*]
declare *of-int-1-less-iff* [*algebra*, *presburger*]
declare *of-int-1-le-iff* [*algebra*, *presburger*]

lemma *int-less-real-le*: $n < m \longleftrightarrow$ *real-of-int* $n + 1 \leq$ *real-of-int* m
proof −
  have (*0*::*real*) ≤ *1*
    by (*metis less-eq-real-def zero-less-one*)
  then show *?thesis*
    by (*metis floor-of-int less-floor-iff*)
qed

lemma *int-le-real-less*: $n \leq m \longleftrightarrow$ *real-of-int* $n <$ *real-of-int* $m + 1$
  by (*meson int-less-real-le not-le*)

lemma *real-of-int-div-aux*:
  (*real-of-int* x) / (*real-of-int* d) =
    real-of-int (x *div* d) + (*real-of-int* (x *mod* d)) / (*real-of-int* d)
proof −
  have x = (x *div* d) $*$ d + x *mod* d
    by *auto*
  then have *real-of-int* x = *real-of-int* (x *div* d) $*$ *real-of-int* d + *real-of-int*(x *mod* d)
    by (*metis of-int-add of-int-mult*)
  then have *real-of-int* x / *real-of-int* d = ... / *real-of-int* d
    by *simp*
  then show *?thesis*
    by (*auto simp add*: *add-divide-distrib algebra-simps*)
qed

lemma *real-of-int-div*:
  d *dvd* n $\Longrightarrow$ *real-of-int* (n *div* d) = *real-of-int* n / *real-of-int* d **for** d n :: *int*
  by (*simp add*: *real-of-int-div-aux*)

lemma *real-of-int-div2*: $0 \leq$ *real-of-int* n / *real-of-int* x − *real-of-int* (n *div* x)
proof (*cases* $x = 0$)
  case *False*
  then show *?thesis*
    by (*metis diff-ge-0-iff-ge floor-divide-of-int-eq of-int-floor-le*)
qed *simp*

lemma *real-of-int-div3*: *real-of-int* n / *real-of-int* x − *real-of-int* (n *div* x) ≤ 1
  apply (*simp add*: *algebra-simps*)
  by (*metis add.commute floor-correct floor-divide-of-int-eq less-eq-real-def of-int-1 of-int-add*)

lemma *real-of-int-div4*: *real-of-int* (n *div* x) $\leq$ *real-of-int* n / *real-of-int* x
  using *real-of-int-div2* [*of* n x] **by** *simp*

8.12 Embedding the Naturals into the Reals

lemma *real-of-card*: *real* (*card* A) = *sum* ($\lambda x.$ *1*) A
  by *simp*

lemma *nat-less-real-le*: $n < m \longleftrightarrow$ *real* n + $1 \leq$ *real* m
  by (*metis discrete of-nat-1 of-nat-add of-nat-le-iff*)

lemma *nat-le-real-less*: $n \leq m \longleftrightarrow$ *real* $n <$ *real* m + 1
  for m n :: *nat*
  by (*meson nat-less-real-le not-le*)

lemma *real-of-nat-div-aux*: *real* x / *real* d = *real* (x *div* d) + *real* (x *mod* d) / *real* d

proof –
  have $x = (x\ div\ d) * d + x\ mod\ d$
    by *auto*
  then have $real\ x = real\ (x\ div\ d) * real\ d + real(x\ mod\ d)$
    by (*metis of-nat-add of-nat-mult*)
  then have $real\ x\ /\ real\ d = \ldots\ /\ real\ d$
    by *simp*
  then show *?thesis*
    by (*auto simp add*: *add-divide-distrib algebra-simps*)
qed

lemma *real-of-nat-div*: $d\ dvd\ n \Longrightarrow real(n\ div\ d) = real\ n\ /\ real\ d$
  by (*subst real-of-nat-div-aux*) (*auto simp add*: *dvd-eq-mod-eq-0* [*symmetric*])

lemma *real-of-nat-div2*: $0 \leq real\ n\ /\ real\ x - real\ (n\ div\ x)$ **for** $n\ x :: nat$
  apply (*simp add*: *algebra-simps*)
  by (*metis floor-divide-of-nat-eq of-int-floor-le of-int-of-nat-eq*)

lemma *real-of-nat-div3*: $real\ n\ /\ real\ x - real\ (n\ div\ x) \leq 1$ **for** $n\ x :: nat$
proof (*cases* $x = 0$)
  case *False*
  then show *?thesis*
    by (*metis of-int-of-nat-eq real-of-int-div3 zdiv-int*)
qed *auto*

lemma *real-of-nat-div4*: $real\ (n\ div\ x) \leq real\ n\ /\ real\ x$ **for** $n\ x :: nat$
  using *real-of-nat-div2* [*of n x*] **by** *simp*

8.13 The Archimedean Property of the Reals

lemma *real-arch-inverse*: $0 < e \longleftrightarrow (\exists n::nat.\ n \neq 0 \wedge 0 < inverse\ (real\ n) \wedge inverse\ (real\ n) < e)$
  using *reals-Archimedean*[*of e*] *less-trans*[*of 0 1 / real n e* **for** *n::nat*]
  by (*auto simp add*: *field-simps cong*: *conj-cong simp del*: *of-nat-Suc*)

lemma *reals-Archimedean3*: $0 < x \Longrightarrow \forall y.\ \exists n.\ y < real\ n * x$
  by (*auto intro*: *ex-less-of-nat-mult*)

lemma *real-archimedian-rdiv-eq-0*:
  assumes *x0*: $x \geq 0$
    and *c*: $c \geq 0$
    and *xc*: $\bigwedge m::nat.\ m > 0 \Longrightarrow real\ m * x \leq c$
  shows $x = 0$
  by (*metis reals-Archimedean3 dual-order.order-iff-strict le0 le-less-trans not-le x0 xc*)

8.14 Rationals

lemma *Rats-abs-iff* [*simp*]:
  $|(x::real)| \in \mathbb{Q} \longleftrightarrow x \in \mathbb{Q}$

by(*simp add*: *abs-real-def split*: *if-splits*)

lemma *Rats-eq-int-div-int*: $\mathbb{Q} = \{$*real-of-int i* / *real-of-int j* | *i j*. $j \neq 0\}$ (**is** - = *?S*)
proof
  show $\mathbb{Q} \subseteq$ *?S*
  proof
    fix *x* :: *real*
    assume $x \in \mathbb{Q}$
    then obtain *r* **where** *x* = *of-rat r*
      unfolding *Rats-def* **..**
    have *of-rat r* ∈ *?S*
      by (*cases r*) (*auto simp add*: *of-rat-rat*)
    then show *x* ∈ *?S*
      using ‹*x* = *of-rat r*› **by** *simp*
  qed
next
  show *?S* ⊆ $\mathbb{Q}$
  proof (*auto simp*: *Rats-def*)
    fix *i j* :: *int*
    assume $j \neq 0$
    then have *real-of-int i* / *real-of-int j* = *of-rat* (*Fract i j*)
      by (*simp add*: *of-rat-rat*)
    then show *real-of-int i* / *real-of-int j* ∈ *range of-rat*
      by *blast*
  qed
qed

lemma *Rats-eq-int-div-nat*: $\mathbb{Q} = \{$ *real-of-int i* / *real n* | *i n*. $n \neq 0\}$
proof (*auto simp*: *Rats-eq-int-div-int*)
  fix *i j* :: *int*
  assume $j \neq 0$
  show ∃(*i'*::*int*) (*n*::*nat*). *real-of-int i* / *real-of-int j* = *real-of-int i'* / *real n* ∧ *0* < *n*
  proof (*cases j* > *0*)
    case *True*
    then have *real-of-int i* / *real-of-int j* = *real-of-int i* / *real* (*nat j*) ∧ *0* < *nat j*
      by *simp*
    then show *?thesis* **by** *blast*
  next
    case *False*
    with ‹$j \neq 0$›
    have *real-of-int i* / *real-of-int j* = *real-of-int* (− *i*) / *real* (*nat* (− *j*)) ∧ *0* < *nat* (− *j*)
      by *simp*
    then show *?thesis* **by** *blast*
  qed
next
  fix *i* :: *int* **and** *n* :: *nat*

```
  assume 0 < n
  then have real-of-int i / real n = real-of-int i / real-of-int(int n) ∧ int n ≠ 0
    by simp
  then show ∃ i' j. real-of-int i / real n = real-of-int i' / real-of-int j ∧ j ≠ 0
    by blast
qed

lemma Rats-abs-nat-div-natE:
  assumes x ∈ ℚ
  obtains m n :: nat where n ≠ 0 and |x| = real m / real n and coprime m n
proof −
  from ‹x ∈ ℚ› obtain i :: int and n :: nat where n ≠ 0 and x = real-of-int i
/ real n
    by (auto simp add: Rats-eq-int-div-nat)
  then have |x| = real (nat |i|) / real n by simp
  then obtain m :: nat where x-rat: |x| = real m / real n by blast
  let ?gcd = gcd m n
  from ‹n ≠ 0› have gcd: ?gcd ≠ 0 by simp
  let ?k = m div ?gcd
  let ?l = n div ?gcd
  let ?gcd' = gcd ?k ?l
  have ?gcd dvd m ..
  then have gcd-k: ?gcd ∗ ?k = m
    by (rule dvd-mult-div-cancel)
  have ?gcd dvd n ..
  then have gcd-l: ?gcd ∗ ?l = n
    by (rule dvd-mult-div-cancel)
  from ‹n ≠ 0› and gcd-l have ?gcd ∗ ?l ≠ 0 by simp
  then have ?l ≠ 0 by (blast dest!: mult-not-zero)
  moreover
  have |x| = real ?k / real ?l
  proof −
    from gcd have real ?k / real ?l = real (?gcd ∗ ?k) / real (?gcd ∗ ?l)
      by (simp add: real-of-nat-div)
    also from gcd-k and gcd-l have . . . = real m / real n by simp
    also from x-rat have . . . = |x| ..
    finally show ?thesis ..
  qed
  moreover
  have ?gcd' = 1
  proof −
    have ?gcd ∗ ?gcd' = gcd (?gcd ∗ ?k) (?gcd ∗ ?l)
      by (rule gcd-mult-distrib-nat)
    with gcd-k gcd-l have ?gcd ∗ ?gcd' = ?gcd by simp
    with gcd show ?thesis by auto
  qed
  then have coprime ?k ?l
    by (simp only: coprime-iff-gcd-eq-1)
  ultimately show ?thesis ..
```

qed

8.15 Density of the Rational Reals in the Reals

This density proof is due to Stefan Richter and was ported by TN. The original source is *Real Analysis* by H.L. Royden. It employs the Archimedean property of the reals.

lemma *Rats-dense-in-real*:
  fixes x :: *real*
  assumes $x < y$
  shows $\exists\, r{\in}\mathbb{Q}.\ x < r \wedge r < y$
proof −
  from ‹$x < y$› **have** $0 < y - x$ **by** *simp*
  with *reals-Archimedean* **obtain** q :: *nat* **where** q: *inverse* (*real* q) $< y - x$ **and** $0 < q$
    by *blast*
  define p **where** $p = \lceil y * \mathit{real}\ q \rceil - 1$
  define r **where** $r = \mathit{of\text{-}int}\ p\ /\ \mathit{real}\ q$
  from q **have** $x < y - \mathit{inverse}\ (\mathit{real}\ q)$
    by *simp*
  also from ‹$0 < q$› **have** $y - \mathit{inverse}\ (\mathit{real}\ q) \le r$
    by (*simp add*: *r-def p-def le-divide-eq left-diff-distrib*)
  finally have $x < r$ **.**
  moreover from ‹$0 < q$› **have** $r < y$
    by (*simp add*: *r-def p-def divide-less-eq diff-less-eq less-ceiling-iff* [*symmetric*])
  moreover have $r \in \mathbb{Q}$
    by (*simp add*: *r-def*)
  ultimately show *?thesis* **by** *blast*
qed

lemma *of-rat-dense*:
  fixes $x\ y$ :: *real*
  assumes $x < y$
  shows $\exists\, q$ :: *rat*. $x <$ *of-rat* $q \wedge$ *of-rat* $q < y$
  using *Rats-dense-in-real* [*OF* ‹$x < y$›]
  by (*auto elim*: *Rats-cases*)

8.16 Numerals and Arithmetic

declaration ‹
  K (*Lin-Arith.add-inj-thms* [@{*thm of-nat-le-iff*} *RS iffD2*, @{*thm of-nat-eq-iff*} *RS iffD2*]
    (∗ *not needed because x < (y::nat) can be rewritten as Suc x <= y*: *of-nat-less-iff RS iffD2* ∗)
  #> *Lin-Arith.add-inj-thms* [@{*thm of-int-le-iff*} *RS iffD2*, @{*thm of-nat-eq-iff*} *RS iffD2*]
    (∗ *not needed because x < (y::int) can be rewritten as x + 1 <= y*: *of-int-less-iff RS iffD2* ∗)
  #> *Lin-Arith.add-simps* [@{*thm of-nat-0*}, @{*thm of-nat-Suc*}, @{*thm of-nat-add*},

@{*thm of-nat-mult*}, @{*thm of-int-0*}, @{*thm of-int-1*},
@{*thm of-int-add*}, @{*thm of-int-minus*}, @{*thm of-int-diff*},
@{*thm of-int-mult*}, @{*thm of-int-of-nat-eq*},
@{*thm of-nat-numeral*}, @{*thm of-nat-numeral*}, @{*thm of-int-neg-numeral*}]
#> *Lin-Arith.add-inj-const* (***const-name*** ‹*of-nat*›, ***typ*** ‹*nat* ⇒ *real*›)
#> *Lin-Arith.add-inj-const* (***const-name*** ‹*of-int*›, ***typ*** ‹*int* ⇒ *real*›))
›

8.17 Simprules combining $x + y$ and *0*

lemma *real-add-minus-iff* [*simp*]: $x + - a = 0 \longleftrightarrow x = a$
for $x\ a$ :: *real*
by *arith*

lemma *real-add-less-0-iff*: $x + y < 0 \longleftrightarrow y < - x$
for $x\ y$ :: *real*
by *auto*

lemma *real-0-less-add-iff*: $0 < x + y \longleftrightarrow - x < y$
for $x\ y$ :: *real*
by *auto*

lemma *real-add-le-0-iff*: $x + y \leq 0 \longleftrightarrow y \leq - x$
for $x\ y$ :: *real*
by *auto*

lemma *real-0-le-add-iff*: $0 \leq x + y \longleftrightarrow - x \leq y$
for $x\ y$ :: *real*
by *auto*

8.18 Lemmas about powers

lemma *two-realpow-ge-one*: (*1*::*real*) ≤ 2 ^ n
by *simp*

declare *sum-squares-eq-zero-iff* [*simp*] *sum-power2-eq-zero-iff* [*simp*]

lemma *real-minus-mult-self-le* [*simp*]: $- (u * u) \leq x * x$
for $u\ x$ :: *real*
by (*rule order-trans* [**where** $y = 0$]) *auto*

lemma *realpow-square-minus-le* [*simp*]: $- u^2 \leq x^2$
for $u\ x$ :: *real*
by (*auto simp add*: *power2-eq-square*)

8.19 Density of the Reals

lemma *field-lbound-gt-zero*: $0 < d1 \Longrightarrow 0 < d2 \Longrightarrow \exists e.\ 0 < e \wedge e < d1 \wedge e < d2$

for *d1 d2* :: *'a::linordered-field*
by (*rule exI* [**where** $x = min\ d1\ d2\ /\ 2$]) (*simp add*: *min-def*)

lemma *field-less-half-sum*: $x < y \Longrightarrow x < (x + y) / 2$
for *x y* :: *'a::linordered-field*
by *auto*

lemma *field-sum-of-halves*: $x / 2 + x / 2 = x$
for *x* :: *'a::linordered-field*
by *simp*

8.20 Floor and Ceiling Functions from the Reals to the Integers

lemma *real-of-nat-less-numeral-iff* [*simp*]: *real* $n <$ *numeral* $w \longleftrightarrow n <$ *numeral* w
for *n* :: *nat*
by (*metis of-nat-less-iff of-nat-numeral*)

lemma *numeral-less-real-of-nat-iff* [*simp*]: *numeral* $w <$ *real* $n \longleftrightarrow$ *numeral* $w < n$
for *n* :: *nat*
by (*metis of-nat-less-iff of-nat-numeral*)

lemma *numeral-le-real-of-nat-iff* [*simp*]: *numeral* $n \leq$ *real* $m \longleftrightarrow$ *numeral* $n \leq m$
for *m* :: *nat*
by (*metis not-le real-of-nat-less-numeral-iff*)

lemma *of-int-floor-cancel* [*simp*]: *of-int* $\lfloor x \rfloor = x \longleftrightarrow (\exists\, n{::}int.\ x =$ *of-int* $n)$
by (*metis floor-of-int*)

lemma *floor-eq*: *real-of-int* $n < x \Longrightarrow x <$ *real-of-int* $n + 1 \Longrightarrow \lfloor x \rfloor = n$
by *linarith*

lemma *floor-eq2*: *real-of-int* $n \leq x \Longrightarrow x <$ *real-of-int* $n + 1 \Longrightarrow \lfloor x \rfloor = n$
by (*fact floor-unique*)

lemma *floor-eq3*: *real* $n < x \Longrightarrow x <$ *real* (*Suc* n) $\Longrightarrow$ *nat* $\lfloor x \rfloor = n$
by *linarith*

lemma *floor-eq4*: *real* $n \leq x \Longrightarrow x <$ *real* (*Suc* n) $\Longrightarrow$ *nat* $\lfloor x \rfloor = n$
by *linarith*

lemma *real-of-int-floor-ge-diff-one* [*simp*]: $r - 1 \leq$ *real-of-int* $\lfloor r \rfloor$
by *linarith*

lemma *real-of-int-floor-gt-diff-one* [*simp*]: $r - 1 <$ *real-of-int* $\lfloor r \rfloor$
by *linarith*

lemma *real-of-int-floor-add-one-ge* [*simp*]: $r \le$ *real-of-int* $\lfloor r \rfloor + 1$
  by *linarith*

lemma *real-of-int-floor-add-one-gt* [*simp*]: $r <$ *real-of-int* $\lfloor r \rfloor + 1$
  by *linarith*

```
lemma floor-divide-real-eq-div:
  assumes 0 ≤ b
  shows ⌊a / real-of-int b⌋ = ⌊a⌋ div b
proof (cases b = 0)
  case True
  then show ?thesis by simp
next
  case False
  with assms have b: b > 0 by simp
  have j = i div b
    if real-of-int i ≤ a a < 1 + real-of-int i
      real-of-int j * real-of-int b ≤ a a < real-of-int b + real-of-int j * real-of-int b
    for i j :: int
  proof −
    from that have i < b + j * b
      by (metis le-less-trans of-int-add of-int-less-iff of-int-mult)
    moreover have j * b < 1 + i
    proof −
      have real-of-int (j * b) < real-of-int i + 1
        using ‹a < 1 + real-of-int i› ‹real-of-int j * real-of-int b ≤ a› by force
      then show j * b < 1 + i by linarith
    qed
    ultimately have (j − i div b) * b ≤ i mod b i mod b < ((j − i div b) + 1) * b
      by (auto simp: field-simps)
    then have (j − i div b) * b < 1 * b 0 * b < ((j − i div b) + 1) * b
      using pos-mod-bound [OF b, of i] pos-mod-sign [OF b, of i]
      by linarith+
    then show ?thesis using b unfolding mult-less-cancel-right by auto
  qed
  with b show ?thesis by (auto split: floor-split simp: field-simps)
qed

lemma floor-one-divide-eq-div-numeral [simp]:
  ⌊1 / numeral b::real⌋ = 1 div numeral b
by (metis floor-divide-of-int-eq of-int-1 of-int-numeral)

lemma floor-minus-one-divide-eq-div-numeral [simp]:
  ⌊− (1 / numeral b)::real⌋ = − 1 div numeral b
by (metis (mono-tags, hide-lams) div-minus-right minus-divide-right
    floor-divide-of-int-eq of-int-neg-numeral of-int-1)

lemma floor-divide-eq-div-numeral [simp]:
  ⌊numeral a / numeral b::real⌋ = numeral a div numeral b
```

by (*metis floor-divide-of-int-eq of-int-numeral*)

lemma *floor-minus-divide-eq-div-numeral* [*simp*]:
  $\lfloor-$ (*numeral a* / *numeral b*)::*real*$\rfloor$ = $-$ *numeral a div numeral b*
by (*metis divide-minus-left floor-divide-of-int-eq of-int-neg-numeral of-int-numeral*)

lemma *of-int-ceiling-cancel* [*simp*]: *of-int* $\lceil x\rceil = x \longleftrightarrow (\exists\, n$::*int*. $x =$ *of-int* $n)$
  using *ceiling-of-int* **by** *metis*

lemma *ceiling-eq*: *of-int* $n < x \Longrightarrow x \leq$ *of-int* $n + 1 \Longrightarrow \lceil x\rceil = n + 1$
  by (*simp add*: *ceiling-unique*)

lemma *of-int-ceiling-diff-one-le* [*simp*]: *of-int* $\lceil r\rceil - 1 \leq r$
  by *linarith*

lemma *of-int-ceiling-le-add-one* [*simp*]: *of-int* $\lceil r\rceil \leq r + 1$
  by *linarith*

lemma *ceiling-le*: $x \leq$ *of-int* $a \Longrightarrow \lceil x\rceil \leq a$
  by (*simp add*: *ceiling-le-iff*)

lemma *ceiling-divide-eq-div*: $\lceil$*of-int a* / *of-int b*$\rceil = -$ ($-$ *a div b*)
  by (*metis ceiling-def floor-divide-of-int-eq minus-divide-left of-int-minus*)

lemma *ceiling-divide-eq-div-numeral* [*simp*]:
  $\lceil$*numeral a* / *numeral b* :: *real*$\rceil = -$ ($-$ *numeral a div numeral b*)
  using *ceiling-divide-eq-div*[*of numeral a numeral b*] **by** *simp*

lemma *ceiling-minus-divide-eq-div-numeral* [*simp*]:
  $\lceil-$ (*numeral a* / *numeral b* :: *real*)$\rceil = -$ (*numeral a div numeral b*)
  using *ceiling-divide-eq-div*[*of* $-$ *numeral a numeral b*] **by** *simp*

The following lemmas are remnants of the erstwhile functions natfloor and natceiling.

lemma *nat-floor-neg*: $x \leq 0 \Longrightarrow$ *nat* $\lfloor x\rfloor = 0$
  for x :: *real*
  by *linarith*

lemma *le-nat-floor*: *real* $x \leq a \Longrightarrow x \leq$ *nat* $\lfloor a\rfloor$
  by *linarith*

lemma *le-mult-nat-floor*: *nat* $\lfloor a\rfloor * $ *nat* $\lfloor b\rfloor \leq$ *nat* $\lfloor a * b\rfloor$
  by (*cases* $0 \leq a \wedge 0 \leq b$)
    (*auto simp add*: *nat-mult-distrib*[*symmetric*] *nat-mono le-mult-floor*)

lemma *nat-ceiling-le-eq* [*simp*]: *nat* $\lceil x\rceil \leq a \longleftrightarrow x \leq$ *real a*
  by *linarith*

lemma *real-nat-ceiling-ge*: $x \leq$ *real* (*nat* $\lceil x\rceil$)

by *linarith*

lemma *Rats-no-top-le*: $\exists\, q \in \mathbb{Q}.\ x \le q$
for x :: *real*
by (*auto intro*!: *bexI*[*of* - *of-nat* (*nat* $\lceil x \rceil$)]) *linarith*

lemma *Rats-no-bot-less*: $\exists\, q \in \mathbb{Q}.\ q < x$ **for** x :: *real*
by (*auto intro*!: *bexI*[*of* - *of-int* ($\lfloor x \rfloor - 1$)]) *linarith*

8.21 Exponentiation with floor

lemma *floor-power*:
assumes $x =$ *of-int* $\lfloor x \rfloor$
shows $\lfloor x \ \hat{}\ n \rfloor = \lfloor x \rfloor \ \hat{}\ n$
proof −
have $x \ \hat{}\ n =$ *of-int* ($\lfloor x \rfloor \ \hat{}\ n$)
using *assms* **by** (*induct n arbitrary*: x) *simp-all*
then show *?thesis* **by** (*metis floor-of-int*)
qed

lemma *floor-numeral-power* [*simp*]: $\lfloor$*numeral* $x \ \hat{}\ n\rfloor =$ *numeral* $x \ \hat{}\ n$
by (*metis floor-of-int of-int-numeral of-int-power*)

lemma *ceiling-numeral-power* [*simp*]: $\lceil$*numeral* $x \ \hat{}\ n\rceil =$ *numeral* $x \ \hat{}\ n$
by (*metis ceiling-of-int of-int-numeral of-int-power*)

8.22 Implementation of rational real numbers

Formal constructor

definition *Ratreal* :: *rat* ⇒ *real*
where [*code-abbrev*, *simp*]: *Ratreal* = *real-of-rat*

code-datatype *Ratreal*

Quasi-Numerals

lemma [*code-abbrev*]:
real-of-rat (*numeral k*) = *numeral k*
real-of-rat (− *numeral k*) = − *numeral k*
real-of-rat (*rat-of-int a*) = *real-of-int a*
by *simp-all*

lemma [*code-post*]:
real-of-rat 0 = *0*
real-of-rat 1 = *1*
real-of-rat (− *1*) = − *1*
real-of-rat (*1* / *numeral k*) = *1* / *numeral k*
real-of-rat (*numeral k* / *numeral l*) = *numeral k* / *numeral l*
real-of-rat (− (*1* / *numeral k*)) = − (*1* / *numeral k*)
real-of-rat (− (*numeral k* / *numeral l*)) = − (*numeral k* / *numeral l*)

by (*simp-all add*: *of-rat-divide of-rat-minus*)

Operations

lemma *zero-real-code* [*code*]: *0* = *Ratreal 0*
by *simp*

lemma *one-real-code* [*code*]: *1* = *Ratreal 1*
by *simp*

instantiation *real* :: *equal*
begin

definition *HOL.equal x y* $\longleftrightarrow$ *x* − *y* = *0* **for** *x* :: *real*

instance by *standard* (*simp add*: *equal-real-def*)

lemma *real-equal-code* [*code*]: *HOL.equal* (*Ratreal x*) (*Ratreal y*) $\longleftrightarrow$ *HOL.equal x y*
by (*simp add*: *equal-real-def equal*)

lemma [*code nbe*]: *HOL.equal x x* $\longleftrightarrow$ *True*
for *x* :: *real*
by (*rule equal-refl*)

end

lemma *real-less-eq-code* [*code*]: *Ratreal x* ≤ *Ratreal y* $\longleftrightarrow$ *x* ≤ *y*
by (*simp add*: *of-rat-less-eq*)

lemma *real-less-code* [*code*]: *Ratreal x* < *Ratreal y* $\longleftrightarrow$ *x* < *y*
by (*simp add*: *of-rat-less*)

lemma *real-plus-code* [*code*]: *Ratreal x* + *Ratreal y* = *Ratreal* (*x* + *y*)
by (*simp add*: *of-rat-add*)

lemma *real-times-code* [*code*]: *Ratreal x* ∗ *Ratreal y* = *Ratreal* (*x* ∗ *y*)
by (*simp add*: *of-rat-mult*)

lemma *real-uminus-code* [*code*]: − *Ratreal x* = *Ratreal* (− *x*)
by (*simp add*: *of-rat-minus*)

lemma *real-minus-code* [*code*]: *Ratreal x* − *Ratreal y* = *Ratreal* (*x* − *y*)
by (*simp add*: *of-rat-diff*)

lemma *real-inverse-code* [*code*]: *inverse* (*Ratreal x*) = *Ratreal* (*inverse x*)
by (*simp add*: *of-rat-inverse*)

lemma *real-divide-code* [*code*]: *Ratreal x* / *Ratreal y* = *Ratreal* (*x* / *y*)
by (*simp add*: *of-rat-divide*)

lemma *real-floor-code* [*code*]: $\lfloor Ratreal\ x \rfloor = \lfloor x \rfloor$
by (*metis Ratreal-def floor-le-iff floor-unique le-floor-iff of-int-floor-le of-rat-of-int-eq real-less-eq-code*)

Quickcheck

definition (**in** *term-syntax*)
valterm-ratreal :: *rat* × (*unit* ⇒ *Code-Evaluation.term*) ⇒ *real* × (*unit* ⇒ *Code-Evaluation.term*)
where [*code-unfold*]: *valterm-ratreal k* = *Code-Evaluation.valtermify Ratreal* {·} *k*

notation *fcomp* (**infixl** ∘> *60*)
notation *scomp* (**infixl** ∘→ *60*)

instantiation *real* :: *random*
begin

definition
Quickcheck-Random.random i = *Quickcheck-Random.random i* ∘→ (λ*r*. *Pair* (*valterm-ratreal r*))

instance ..

end

no-notation *fcomp* (**infixl** ∘> *60*)
no-notation *scomp* (**infixl** ∘→ *60*)

instantiation *real* :: *exhaustive*
begin

definition
exhaustive-real f d = *Quickcheck-Exhaustive.exhaustive* (λ*r*. *f* (*Ratreal r*)) *d*

instance ..

end

instantiation *real* :: *full-exhaustive*
begin

definition
full-exhaustive-real f d = *Quickcheck-Exhaustive.full-exhaustive* (λ*r*. *f* (*valterm-ratreal r*)) *d*

instance ..

end

instantiation *real* :: *narrowing*
begin

definition
narrowing-real = *Quickcheck-Narrowing.apply* (*Quickcheck-Narrowing.cons Ratreal*) *narrowing*

instance ..

end

8.23 Setup for Nitpick

declaration ‹
Nitpick-HOL.register-frac-type ***type-name*** ‹*real*›
[(***const-name*** ‹*zero-real-inst.zero-real*›, ***const-name*** ‹*Nitpick.zero-frac*›),
(***const-name*** ‹*one-real-inst.one-real*›, ***const-name*** ‹*Nitpick.one-frac*›),
(***const-name*** ‹*plus-real-inst.plus-real*›, ***const-name*** ‹*Nitpick.plus-frac*›),
(***const-name*** ‹*times-real-inst.times-real*›, ***const-name*** ‹*Nitpick.times-frac*›),
(***const-name*** ‹*uminus-real-inst.uminus-real*›, ***const-name*** ‹*Nitpick.uminus-frac*›),
(***const-name*** ‹*inverse-real-inst.inverse-real*›, ***const-name*** ‹*Nitpick.inverse-frac*›),
(***const-name*** ‹*ord-real-inst.less-real*›, ***const-name*** ‹*Nitpick.less-frac*›),
(***const-name*** ‹*ord-real-inst.less-eq-real*›, ***const-name*** ‹*Nitpick.less-eq-frac*›)]
›

lemmas [*nitpick-unfold*] = *inverse-real-inst.inverse-real one-real-inst.one-real*
ord-real-inst.less-real ord-real-inst.less-eq-real plus-real-inst.plus-real
times-real-inst.times-real uminus-real-inst.uminus-real
zero-real-inst.zero-real

8.24 Setup for SMT

ML-file ‹*Tools/SMT/smt-real.ML*›
ML-file ‹*Tools/SMT/z3-real.ML*›

lemma [*z3-rule*]:
$0 + x = x$
$x + 0 = x$
$0 * x = 0$
$1 * x = x$
$-x = -1 * x$
$x + y = y + x$
for *x y* :: *real*
by *auto*

8.25 Setup for Argo

ML-file ‹*Tools/Argo/argo-real.ML*›

```
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 ⊢ φ → ψ → ⊥
  shows Pr (φ ⊔ ψ) = Pr φ + Pr ψ
  using assms
  by (metis Additivity
            Double-Negation-converse
            Modus-Ponens
            conjunction-def
            negation-def)

lemma (in Logical-Probability) complementation:
  Pr (∼ φ) = 1 − Pr φ
  by (metis Alternate-Additivity
            Unity
            bivalence
            negation-elimination
            add.commute
            add-diff-cancel-left')

lemma (in Logical-Probability) unity-upper-bound:
  Pr φ ≤ 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* ⇒ *real*
  assumes *Thesis*: *Pr* ⊤ = *1*
  assumes *Antithesis*: *Pr* ⊥ = *0*
  assumes *Monotonicity*: ⊢ φ → ψ ⟹ *Pr* φ ≤ *Pr* ψ
  assumes *Sum-Rule*: *Pr* φ + *Pr* ψ = *Pr* (φ ⊓ ψ) + *Pr* (φ ⊔ ψ)

sublocale *Weatherson-Probability* ⊆ *Logical-Probability*
proof
  fix φ
  have ⊢ ⊥ → φ
    by (*simp add*: *Ex-Falso-Quodlibet*)
  thus *0* ≤ *Pr* φ
    using *Antithesis Monotonicity* **by** *fastforce*
next
  fix φ
  assume ⊢ φ
  thus *Pr* φ = *1*
    by (*metis Thesis*
          Monotonicity
          eq-iff
          Axiom-1
          Ex-Falso-Quodlibet
          Modus-Ponens
          verum-def)
next
  fix φ ψ
  assume ⊢ φ → ψ → ⊥
  thus *Pr* ((φ → ⊥) → ψ) = *Pr* φ + *Pr* ψ
    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*:
  ⊢ φ → ψ ⟹ *Pr* φ ≤ *Pr* ψ
proof −
  assume ⊢ φ → ψ
  hence ⊢ ∼ (φ ⊓ ∼ ψ)
    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 $\vdash \varphi \rightarrow (\psi \setminus (\varphi \sqcap \psi)) \rightarrow \bot$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} \langle\varphi\rangle \rightarrow (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle)) \rightarrow \bot$
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 $\vdash (\!|\ \langle\varphi\rangle \rightarrow (\langle\psi\rangle \setminus (\langle\varphi\rangle \sqcap \langle\psi\rangle)) \rightarrow \bot\ |\!)$ **using** *propositional-semantics*
by *blast*
thus *?thesis* **by** *simp*
qed

```
  hence Pr (φ ⊔ ψ) = Pr φ + Pr (ψ \ (φ ⊓ ψ))
    using Alternate-Additivity biconditional-equivalence calculation by auto
  moreover have ⊢ ψ ↔ (ψ \ (φ ⊓ ψ) ⊔ (φ ⊓ ψ))
  proof −
    have ∀ 𝔐. 𝔐 ⊨prop ⟨ψ⟩ ↔ (⟨ψ⟩ \ (⟨φ⟩ ⊓ ⟨ψ⟩) ⊔ (⟨φ⟩ ⊓ ⟨ψ⟩))
      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 ⊢ ⦇ ⟨ψ⟩ ↔ (⟨ψ⟩ \ (⟨φ⟩ ⊓ ⟨ψ⟩) ⊔ (⟨φ⟩ ⊓ ⟨ψ⟩)) ⦈ using propositional-semantics
by blast
    thus ?thesis by simp
  qed
  moreover have ⊢ (ψ \ (φ ⊓ ψ)) → (φ ⊓ ψ) → ⊥
    unfolding subtraction-def negation-def conjunction-def
    using conjunction-def conjunction-right-elimination by auto
  hence Pr ψ = Pr (ψ \ (φ ⊓ ψ)) + Pr (φ ⊓ ψ)
    using Alternate-Additivity biconditional-equivalence calculation by auto
  ultimately show ?thesis
    by simp
qed

sublocale Logical-Probability ⊆ Weatherson-Probability
proof
  show Pr ⊤ = 1
    by (simp add: Unity)
next
  show Pr ⊥ = 0
    by (metis add-cancel-left-right
          Additivity
          Ex-Falso-Quodlibet
          Unity
          bivalence
          conjunction-right-elimination
          negation-def)
next
  fix φ ψ
  assume ⊢ φ → ψ
  thus Pr φ ≤ Pr ψ
    using monotonicity
    by auto
next
  fix φ ψ
  show Pr φ + Pr ψ = Pr (φ ⊓ ψ) + Pr (φ ⊔ ψ)
    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 $\vdash \varphi \to \bigsqcup\ \Psi$
  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 $\vdash \varphi \to \bigsqcup\ \Psi$
  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* =
      $\{Pr.\ class.Logical\text{-}Probability\ (\lambda\ \varphi.\ \vdash \varphi)\ (\to)\ \bot\ Pr\ \}$

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

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 (λ χ. *if* $\chi\in\Omega$ *then* (*1* :: *real*) *else* *0*) $\in$ *Dirac-Measures*
(**is** *?Pr* $\in$ *Dirac-Measures*)
proof −
have *class.Logical-Probability* (λ φ. $\vdash$ φ) ($\rightarrow$) $\bot$ *?Pr*
proof (*standard*, *simp*,
meson assms
Formula-Maximally-Consistent-Set-reflection
Maximally-Consistent-Set-def
set-deduction-weaken)
fix φ ψ
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 $\varphi \notin \Omega \vee \psi \notin \Omega$
using *assms*
Formula-Maximally-Consistent-Set-reflection
Maximally-Consistent-Set-def
conjunction-set-deduction-equivalence
by *meson*

have $\varphi \sqcup \psi \in \Omega$ = ($\varphi \in \Omega \vee \psi \in \Omega$)
by (*metis* ‹$\varphi \sqcap \psi \notin \Omega$›
assms
Formula-Maximally-Consistent-Set-implication
Maximally-Consistent-Set-def
conjunction-def
disjunction-def)
have *?Pr* ($\varphi \sqcup \psi$) = *?Pr* φ + *?Pr* ψ
proof (*cases* $\varphi \sqcup \psi \in \Omega$)
case *True*
hence $\diamondsuit$: *1* = *?Pr* ($\varphi \sqcup \psi$) **by** *simp*
show *?thesis*
proof (*cases* $\varphi \in \Omega$)
case *True*
hence $\psi \notin \Omega$
using ‹$\varphi \notin \Omega \vee \psi \notin \Omega$›
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 ⨆ Ψ ∉ Ω ≡ ∀ ψ ∈ set Ψ. ψ ∉ Ω
proof (induct Ψ)
  case Nil
  then show ?case
    using assms
          Formula-Consistent-def
          Formula-Maximally-Consistent-Set-def
          Maximally-Consistent-Set-def
          set-deduction-reflection
    by (simp, blast)
next
  case (Cons ψ Ψ)
  have ⨆ (ψ # Ψ) ∉ Ω = (ψ ∉ Ω ∧ ⨆ Ψ ∉ Ω)
    by (simp add: disjunction-def,
        meson assms
              Formula-Consistent-def
              Formula-Maximally-Consistent-Set-def
```

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-Measures*. $\delta\ \varphi \le (\sum \psi \leftarrow \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
proof −
{
fix δ :: $'a \Rightarrow$ *real*
assume $\delta \in$ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)$ $(\to)$ $\bot$ δ
unfolding *Dirac-Measures-def*
by *auto*
assume $\vdash \varphi \to \bigsqcup\ \Psi$
hence $\delta\ \varphi \le (\sum \psi \leftarrow \Psi.\ \delta\ \psi)$
using *implication-list-summation-inequality*
by *auto*
}
moreover {
assume $\neg \vdash \varphi \to \bigsqcup\ \Psi$
from *this* **obtain** Ω **where** Ω: *MCS* Ω $\varphi \in \Omega$ $\bigsqcup\ \Psi \notin \Omega$
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$\forall\ \psi \in$ *set* $\Psi.\ \psi \notin \Omega$
using *arbitrary-disjunction-exclusion-MCS* **by** *blast*
let $?\delta = \lambda\ \chi.$ *if* $\chi \in \Omega$ *then* (*1* :: *real*) *else 0*
from ‹$\forall\ \psi \in$ *set* $\Psi.\ \psi \notin \Omega$› **have** $(\sum \psi \leftarrow \Psi.\ ?\delta\ \psi) = 0$
by (*induct* Ψ, *simp*, *simp*)
hence $\neg\ ?\delta\ \varphi \le (\sum \psi \leftarrow \Psi.\ ?\delta\ \psi)$
by (*simp add*: $\Omega(2)$)
hence
$\exists\ \delta \in$ *Dirac-Measures*. $\neg\ (\delta\ \varphi \le (\sum \psi \leftarrow \Psi.\ \delta\ \psi))$

using $\Omega(1)$ *MCS-Dirac-Measure* **by** *auto*
}
ultimately show *?thesis* **by** *blast*
qed

theorem (**in** *Classical-Propositional-Logic*) *List-Summation-Completeness*:
$(\forall\ Pr \in \textit{Logical-Probabilities}.\ Pr\ \varphi \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
(**is** *?lhs = ?rhs*)
proof
assume *?lhs*
hence $\forall\ \delta \in \textit{Dirac-Measures}.\ \delta\ \varphi \leq (\sum \psi \leftarrow \Psi.\ \delta\ \psi)$
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 $Pr :: {'a} \Rightarrow \textit{real}$
assume $Pr \in$ *Logical-Probabilities*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\to)\ \bot\ Pr$
unfolding *Logical-Probabilities-def*
by *auto*
show $Pr\ \varphi \leq (\sum \psi \leftarrow \Psi.\ Pr\ \psi)$
using ‹*?rhs*› *implication-list-summation-inequality*
by *simp*
qed
qed

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Set-Summation-Completeness*:
$(\forall\ \delta \in \textit{Dirac-Measures}.\ \delta\ \varphi \leq (\sum \psi \in \textit{set}\ \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
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*:
$(\forall\ \delta \in \textit{Logical-Probabilities}.\ \delta\ \varphi \leq (\sum \psi \in \textit{set}\ \Psi.\ \delta\ \psi)) = \vdash \varphi \to \bigsqcup\ \Psi$
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 Pr (⨆ Φ) = (∑φ←Φ. Pr φ)
  using assms
proof (induct Φ)
  case Nil
  then show ?case
    by (simp add: Antithesis)
next
  case (Cons φ Φ)
  assume ⊢ ∐ (φ # Φ)
  hence ⊢ ∼ (φ ⊓ ⨆ Φ) ⊢ ∐ Φ by simp+
  hence Pr (⨆(φ # Φ)) = Pr φ + Pr (⨆ Φ)
        Pr (⨆ Φ) = (∑φ←Φ. Pr φ) using Cons.hyps Additivity by auto
  hence Pr (⨆(φ # Φ)) = Pr φ + (∑φ←Φ. Pr φ) by auto
  thus ?case by simp
qed

lemma sum-list-monotone:
  fixes f :: 'a ⇒ real
  assumes ∀ x. f x ≥ 0
     and  set Φ ⊆ set Ψ
     and  distinct Φ
   shows (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
  using assms
proof −
  assume ∀ x. f x ≥ 0
  have ∀Φ. set Φ ⊆ set Ψ ⟶ distinct Φ ⟶ (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
  proof (induct Ψ)
    case Nil
    then show ?case by simp
  next
    case (Cons ψ Ψ)
    {
      fix Φ
      assume set Φ ⊆ set (ψ # Ψ)
         and distinct Φ
      have (∑φ←Φ. f φ) ≤ (∑ψ'←(ψ # Ψ). f ψ')
      proof −
        {
          assume ψ ∉ set Φ
          with ‹set Φ ⊆ set (ψ # Ψ)› have set Φ ⊆ set Ψ by auto
          hence (∑φ←Φ. f φ) ≤ (∑ψ←Ψ. f ψ)
            using Cons.hyps ‹distinct Φ› by auto
          moreover have f ψ ≥ 0 using ‹∀ x. f x ≥ 0› by metis
          ultimately have ?thesis by simp
        }
        moreover
        {
          assume ψ ∈ set Φ
          from ‹ψ ∈ set Φ› have set Φ = insert ψ (set (removeAll ψ Φ))
```

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

lemma count-remove-all-sum-list:
 fixes f :: 'a ⇒ real
 shows real (count-list xs x) * f x + (∑x'←(removeAll x xs). f x') = (∑x←xs.
f x)
 by (induct xs, simp, simp,
    metis (no-types, hide-lams)
          semiring-normalization-rules(3)
          add.commute
          add.left-commute)

lemma (in Classical-Propositional-Logic) Dirac-Exclusive-Implication-Completeness:
 (∀ δ ∈ Dirac-Measures. (∑φ←Φ. δ φ) ≤ δ ψ) = (⊢ ∐ Φ ∧ ⊢ ⨆ Φ → ψ)
proof −
 {
  fix δ
  assume δ ∈ Dirac-Measures
  from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ δ
    unfolding Dirac-Measures-def
    by simp
  assume ⊢ ∐ Φ ⊢ ⨆ Φ → ψ
  hence (∑φ←Φ. δ φ) ≤ δ ψ
    using exclusive-sum-list-identity monotonicity by fastforce
 }
 moreover
 {
  assume ¬ ⊢ ∐ Φ
```

hence $(\exists\ \varphi \in \mathit{set}\ \Phi.\ \exists\ \psi \in \mathit{set}\ \Phi.\ \varphi \neq \psi \wedge \neg \vdash \sim (\varphi \sqcap \psi)) \vee (\exists\ \varphi \in \mathit{duplicates}$
$\Phi.\ \neg \vdash \sim \varphi)$
  using *exclusive-equivalence set-deduction-base-theory* **by** *blast*
 hence $\neg\ (\forall\ \ \delta \in \mathit{Dirac\text{-}Measures}.\ (\sum \varphi \leftarrow \Phi.\ \delta\ \varphi) \leq \delta\ \psi)$
 proof (*elim disjE*)
  assume $\exists\ \varphi \in \mathit{set}\ \Phi.\ \exists\ \chi \in \mathit{set}\ \Phi.\ \varphi \neq \chi \wedge \neg \vdash \sim (\varphi \sqcap \chi)$
  from *this* **obtain** φ **and** χ
   where $\varphi\chi$*-properties*: $\varphi \in \mathit{set}\ \Phi\ \chi \in \mathit{set}\ \Phi\ \varphi \neq \chi\ \neg \vdash \sim (\varphi \sqcap \chi)$
   by *blast*
  from *this* **obtain** Ω **where** Ω: $\mathit{MCS}\ \Omega \sim (\varphi \sqcap \chi) \notin \Omega$
   by (*meson insert-subset*
     Formula-Consistent-def
     Formula-Maximal-Consistency
     Formula-Maximally-Consistent-Extension
     Formula-Maximally-Consistent-Set-def
     set-deduction-base-theory
     set-deduction-reflection
     set-deduction-theorem)
  let $?\delta = \lambda\ \chi.\ \mathit{if}\ \ \chi{\in}\Omega\ \mathit{then}\ (1\ ::\ \mathit{real})\ \mathit{else}\ 0$
  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** $(\sum \varphi \leftarrow [\varphi,\ \chi].\ ?\delta\ \varphi) = 2$
     $\mathit{set}\ [\varphi,\ \chi] \subseteq \mathit{set}\ \Phi$
     $\mathit{distinct}\ [\varphi,\ \chi]$
     $\forall \varphi.\ ?\delta\ \varphi \geq 0$
   by *simp*+
  hence $(\sum \varphi \leftarrow \Phi.\ ?\delta\ \varphi) \geq 2$ **using** *sum-list-monotone* **by** *metis*
  hence $\neg\ (\sum \varphi \leftarrow \Phi.\ ?\delta\ \varphi) \leq\ ?\delta\ (\psi)$ **by** *auto*
  thus *?thesis*
   using $\Omega(1)$ *MCS-Dirac-Measure*
   by *auto*
 next
  assume $\exists\ \varphi \in \mathit{duplicates}\ \Phi.\ \neg \vdash \sim \varphi$
  from *this* **obtain** φ **where** φ: $\varphi \in \mathit{duplicates}\ \Phi\ \neg \vdash \sim \varphi$
   using *exclusive-equivalence* [**where** $\Gamma=\{\}$] *set-deduction-base-theory*
   by *blast*
  from φ **obtain** Ω **where** Ω: $\mathit{MCS}\ \Omega \sim \varphi \notin \Omega$
   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 = λ χ. if χ∈Ω then (1 :: real) else 0
    from φ have count-list Φ φ ≥ 2 using duplicates-alt-def [where xs=Φ]
      by blast
    hence real (count-list Φ φ) ∗ ?δ φ ≥ 2 using ‹φ ∈ Ω› by simp
    moreover
    {
      fix Ψ
      have (∑φ←Ψ. ?δ φ) ≥ 0 by (induct Ψ, simp, simp)
    }
    moreover have (0::real) ≤ (∑a←removeAll φ Φ. if a ∈ Ω then 1 else 0)
      using ‹⋀Ψ. 0 ≤ (∑φ←Ψ. if φ ∈ Ω then 1 else 0)› by presburger
    ultimately have real (count-list Φ φ) ∗ ?δ φ + (∑ φ ← (removeAll φ Φ).
?δ φ) ≥ 2
      using ‹2 ≤ real (count-list Φ φ) ∗ (if φ ∈ Ω then 1 else 0)› by linarith
    hence (∑φ←Φ. ?δ φ) ≥ 2 by (metis count-remove-all-sum-list)
    hence ¬ (∑φ←Φ. ?δ φ) ≤ ?δ (ψ) by auto
    thus ?thesis
      using Ω(1) MCS-Dirac-Measure
      by auto
  qed
}
moreover
{
  assume ¬ ⊢ ⨆ Φ → ψ
  from this obtain Ω φ where Ω: MCS Ω
                       and ψ: ψ ∉ Ω
                       and φ: φ ∈ set Φ φ ∈ Ω
    by (meson insert-subset
              Formula-Consistent-def
              Formula-Maximal-Consistency
              Formula-Maximally-Consistent-Extension
              Formula-Maximally-Consistent-Set-def
              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 ⇒ real
    assume Pr ∈ Logical-Probabilities
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Logical-Probabilities-def
      by simp
    show (∑φ←Φ. Pr φ) ≤ Pr ψ
      using ‹?rhs›
            exclusive-sum-list-identity
            monotonicity
      by fastforce
  qed
qed


lemma (in Classical-Propositional-Logic) Dirac-Inequality-Completeness:
  (∀ δ ∈ Dirac-Measures. δ φ ≤ δ ψ) = ⊢ φ → ψ
proof −
  have ⊢ ∐ [φ]
    by (simp add: conjunction-right-elimination negation-def)
  hence (⊢ ∐ [φ] ∧ ⊢ ⨆ [φ] → ψ) = ⊢ φ → ψ
    by (metis Arbitrary-Disjunction.simps(1)
              Arbitrary-Disjunction.simps(2)
              disjunction-def implication-equivalence
              negation-def
```

weak-biconditional-weaken)
thus *?thesis*
using *Dirac-Exclusive-Implication-Completeness* [**where** Φ=[φ]]
by *auto*
qed

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

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

lemma (**in** *Classical-Propositional-Logic*) *Dirac-Exclusive-Set-Summation-Completeness*:
($\forall$ $\delta \in$ *Dirac-Measures*. δ ($\bigsqcup$ Φ) = ($\sum \varphi \in$ *set* Φ. δ φ)) = $\vdash \coprod$ (*remdups* Φ)
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* $\in$ *Logical-Probabilities*. *Pr* ($\bigsqcup$ Φ) = ($\sum \varphi \in$ *set* Φ. *Pr* φ)) = $\vdash \coprod$ (*remdups*
Φ)

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

end
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 $\vdash (\!|\ ((\langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle \leftrightarrow (\langle\varphi\rangle \rightarrow \langle\bigsqcup\ \Phi\rangle)) \rightarrow$
      $(\sim \langle\psi\rangle \rightarrow \langle \boldsymbol{\sim}\ \Phi :\rightarrow \sim \varphi\rangle) \leftrightarrow (\langle\varphi\rangle \rightarrow (\langle\psi\rangle \sqcup \langle\bigsqcup\ \Phi\rangle))\ |\!)$
      using *propositional-semantics* **by** *blast*
    thus *?thesis*
      by *simp*
  qed
  with *Cons* **show** *?case*
    by (*metis 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* Ψ Λ)
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** $\vdash \bigsqcup\ (map\ \bigsqcap\ (map\ (\lambda\ \Psi.\ \Phi\ @\ \Psi)\ \Lambda)) \leftrightarrow (\bigsqcap\ \Phi \sqcap \bigsqcup\ (map\ \bigsqcap$
$\Lambda))$
(**is** $\vdash \bigsqcup\ (map\ \bigsqcap\ ?A) \leftrightarrow (?B \sqcap\ ?C)$)
by *meson*
moreover have $\vdash \bigsqcup\ (map\ \bigsqcap\ ?A) \leftrightarrow (?B \sqcap\ ?C)$
$\rightarrow (\bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ ?A) \leftrightarrow (\varphi \sqcap \bigsqcup\ (map\ \bigsqcap\ ?A)))$
$\rightarrow \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ ?A) \leftrightarrow ((\varphi \sqcap\ ?B) \sqcap\ ?C)$
proof −
let $?\varphi = \langle \bigsqcup\ (map\ \bigsqcap\ ?A)\rangle \leftrightarrow (\langle ?B\rangle \sqcap \langle ?C\rangle)$
$\rightarrow (\langle \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ ?A)\rangle \leftrightarrow (\langle \varphi\rangle \sqcap \langle \bigsqcup\ (map\ \bigsqcap\ ?A)\rangle))$
$\rightarrow \langle \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ ?A)\rangle \leftrightarrow ((\langle \varphi\rangle \sqcap \langle ?B\rangle) \sqcap \langle ?C\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}\ ?\varphi$ **by** *simp*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis*
by *simp*
qed
ultimately have $\vdash \bigsqcup\ (map\ (\bigsqcap \circ (\lambda\ \varphi s.\ \varphi\ \#\ \varphi s))\ ?A) \leftrightarrow ((\varphi \sqcap\ ?B) \sqcap\ ?C)$
using *Modus-Ponens conj-dnf-distribute*
by *blast*
moreover
have $\bigsqcap \circ (@)\ (\varphi\ \#\ \Phi) = \bigsqcap \circ (\#)\ \varphi \circ (@)\ \Phi$ **by** *auto*
hence
$\vdash \bigsqcup\ (map\ (\bigsqcap \circ (@)\ (\varphi\ \#\ \Phi))\ \Lambda) \leftrightarrow (\bigsqcap\ (\varphi\ \#\ \Phi) \sqcap\ ?C)$
$= \vdash \bigsqcup\ (map\ (\bigsqcap \circ (\#)\ \varphi)\ ?A) \leftrightarrow ((\varphi \sqcap\ ?B) \sqcap\ ?C)$
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 \$⊢ (φ # Φ) = ($\exists$ Ψ. *mset* (*map snd* Ψ) ⊆# *mset* Γ $\wedge$
map (*uncurry* ($\sqcup$)) Ψ :⊢ φ $\wedge$
map (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ (*map snd* Ψ) \$⊢ Φ)

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

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 $\Sigma \preceq \Gamma$
proof −
let *?*$\Delta\Sigma$ = *map* (λ *x*. (*x*,*x*)) Σ
have *map snd ?*$\Delta\Sigma$ = Σ
by (*induct* Σ, *simp*, *simp*)
moreover have *map fst ?*$\Delta\Sigma$ = Σ
by (*induct* Σ, *simp*, *simp*)
hence *mset* (*map fst ?*$\Delta\Sigma$) ⊆# *mset* Γ
using *assms* **by** *simp*
moreover have $\forall$ (γ,σ) $\in$ *set ?*$\Delta\Sigma$. ⊢ $\gamma \rightarrow \sigma$
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*]: $\Gamma \preceq \Gamma$
using *msub-stronger-theory-intro* **by** *auto*

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

lemma (**in** *Minimal-Logic*) *stronger-theory-empty-list-intro* [*simp*]:
assumes $\Gamma \preceq$ []
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 ?Φ_0 = remove1 (γ, δ) Φ
      let ?Σ_0 = map snd ?Φ_0
      from γ Φ(2) have mset (map fst ?Φ_0) ⊆# mset (remove1 γ Γ)
        by (metis ex-mset
                listSubtract-monotonic
                listSubtract-mset-homomorphism
                mset-remove1
                remove1-pairs-list-projections-fst)
      moreover have mset ?Φ_0 ⊆# mset Φ by simp
      with Φ(3) have ∀ (γ,δ) ∈ set ?Φ_0. ⊢ γ → δ by fastforce
      ultimately have ?Σ_0 ≼ remove1 γ Γ
        unfolding stronger-theory-relation-def by blast
```

moreover have $\Delta <\sim\sim> (\textit{remove1}\ \delta\ \Sigma)$ **using** $\langle\Sigma <\sim\sim> (\delta \# \Delta)\rangle$
by (*metis perm-remove-perm perm-sym remove-hd*)
moreover from $\gamma\ \Phi(\mathit{1})$ **have** *mset* $?\Sigma_0$ = *mset* (*remove1* $\delta\ \Sigma$)
using *remove1-pairs-list-projections-snd*
by *fastforce*
hence $?\Sigma_0 <\sim\sim>$ *remove1* $\delta\ \Sigma$
using *mset-eq-perm* **by** *blast*
ultimately have $\Delta \preceq$ *remove1* $\gamma\ \Gamma$ **using** *Cons*
by (*meson perm.trans perm-sym*)
from *this* **obtain** Ψ_0 **where** Ψ_0:
map snd $\Psi_0 = \Delta$
mset (*map fst* Ψ_0) $\subseteq\#$ *mset* (*remove1* $\gamma\ \Gamma$)
$\forall\ (\gamma,\delta) \in \mathit{set}\ \Psi_0.\ \vdash \gamma \to \delta$
using *stronger-theory-relation-def* **by** *fastforce*
let $?\Psi = (\gamma,\ \delta)\ \#\ \Psi_0$
have *map snd* $?\Psi = (\delta\ \#\ \Delta)$
by (*simp add*: $\Psi_0(\mathit{1})$)
moreover have *mset* (*map fst* $?\Psi$) $\subseteq\#$ *mset* (γ # (*remove1* $\gamma\ \Gamma$))
using $\Psi_0(\mathit{2})$ **by** *auto*
moreover from $\gamma\ \Phi(\mathit{3})\ \Psi_0(\mathit{3})$ **have** $\forall\ (\gamma,\sigma) \in \mathit{set}\ ?\Psi.\ \vdash \gamma \to \sigma$ **by** *auto*
ultimately have $(\delta\ \#\ \Delta) \preceq (\gamma\ \#\ (\mathit{remove1}\ \gamma\ \Gamma))$
unfolding *stronger-theory-relation-def* **by** *metis*
moreover from $\gamma\ \Phi(\mathit{2})$ **have** $\gamma \in\#$ *mset* Γ
using *mset-subset-eqD* **by** *fastforce*
hence $(\gamma\ \#\ (\mathit{remove1}\ \gamma\ \Gamma)) <\sim\sim> \Gamma$
by (*simp add*: *perm-remove perm-sym*)
ultimately have $(\delta\ \#\ \Delta) \preceq \Gamma$
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* $\sigma\ \Sigma$)
{
fix $\Delta\ \Gamma$
assume $(\sigma\ \#\ \Sigma) \preceq \Delta\ \Delta \preceq \Gamma$
from *this* **obtain** Φ **where** Φ:
map snd $\Phi = \sigma\ \#\ \Sigma$

mset (*map fst* Φ) ⊆# *mset* Δ
∀ (δ,σ) ∈ *set* Φ. ⊢ δ → σ
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let *?δ* = *fst* (*hd* Φ)
from Φ(*1*) **have** Φ ≠ [] **by** (*induct* Φ, *simp+*)
hence *?δ* ∈# *mset* (*map fst* Φ) **by** (*induct* Φ, *simp+*)
with Φ(*2*) **have** *?δ* ∈# *mset* Δ **by** (*meson mset-subset-eqD*)
with ‹Φ ≠ []› Φ(*2*) **have** *mset* (*map fst* (*remove1* (*hd* Φ) Φ)) ⊆# *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 ‹Φ ≠ []› **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** ∀ (δ,σ) ∈ *set* (*tl* Φ). ⊢ δ → σ
by (*simp add*: ‹Φ ≠ []› *list.set-sel*(*2*))
ultimately have Σ ⪯ *remove1 ?δ* Δ
using *stronger-theory-relation-def* **by** *auto*
from ‹*?δ* ∈# *mset* Δ› **have** *?δ* # (*remove1 ?δ* Δ) <~~> Δ
by (*simp add*: *perm-remove perm-sym*)
with ‹Δ ⪯ Γ› **have** (*?δ* # (*remove1 ?δ* Δ)) ⪯ Γ
using *stronger-theory-left-permutation perm-sym* **by** *blast*
from *this* **obtain** Ψ **where** Ψ:
map snd Ψ = (*?δ* # (*remove1 ?δ* Δ))
mset (*map fst* Ψ) ⊆# *mset* Γ
∀ (γ,δ) ∈ *set* Ψ. ⊢ γ → δ
using *stronger-theory-relation-def* **by** (*simp*, *metis*)
let *?γ* = *fst* (*hd* Ψ)
from Ψ(*1*) **have** Ψ ≠ [] **by** (*induct* Ψ, *simp+*)
hence *?γ* ∈# *mset* (*map fst* Ψ) **by** (*induct* Ψ, *simp+*)
with Ψ(*2*) **have** *?γ* ∈# *mset* Γ **by** (*meson mset-subset-eqD*)
with ‹Ψ ≠ []› Ψ(*2*) **have** *mset* (*map fst* (*remove1* (*hd* Ψ) Ψ)) ⊆# *mset* (*remove1 ?γ* Γ)
by (*simp*,
metis diff-single-eq-union
hd-in-set
image-mset-add-mset
insert-subset-eq-iff
set-mset-mset)
moreover from ‹Ψ ≠ []› **have** *remove1* (*hd* Ψ) Ψ = *tl* Ψ **by** (*induct* Ψ, *simp+*)
moreover from Ψ(*1*) **have** *map snd* (*tl* Ψ) = (*remove1 ?δ* Δ)
by (*simp add*: *map-tl*)
moreover from Ψ(*3*) **have** ∀ (γ,δ) ∈ *set* (*tl* Ψ). ⊢ γ → δ

```
      by (simp add: ‹Ψ ≠ []› list.set-sel(2))
    ultimately have remove1 ?δ Δ ≼ remove1 ?γ Γ
      using stronger-theory-relation-def by auto
    with ‹Σ ≼ remove1 ?δ Δ› Cons.hyps have Σ ≼ remove1 ?γ Γ
      by blast
    from this obtain Ω0 where Ω0:
      map snd Ω0 = Σ
      mset (map fst Ω0) ⊆# mset (remove1 ?γ Γ)
      ∀ (γ,σ) ∈ set Ω0. ⊢ γ → σ
      using stronger-theory-relation-def by (simp, metis)
    let ?Ω = (?γ, σ) # Ω0
    from Ω0(1) have map snd ?Ω = σ # Σ by simp
    moreover from Ω0(2) have mset (map fst ?Ω) ⊆# mset (?γ # (remove1
?γ Γ))
      by simp
     moreover from Φ(1) Ψ(1) have σ = snd (hd Φ) ?δ = snd (hd Ψ) by
fastforce+
    with Φ(3) Ψ(3) ‹Φ ≠ []› ‹Ψ ≠ []› hd-in-set have ⊢ ?δ → σ ⊢ ?γ → ?δ
      by fastforce+
    hence ⊢ ?γ → σ using Modus-Ponens hypothetical-syllogism by blast
    with Ω0(3) have ∀ (γ,σ) ∈ set ?Ω. ⊢ γ → σ
      by auto
    ultimately have (σ # Σ) ≼ (?γ # (remove1 ?γ Γ))
      unfolding stronger-theory-relation-def
      by metis
    moreover from ‹?γ ∈# mset Γ› have (?γ # (remove1 ?γ Γ)) <~~> Γ
      by (simp add: perm-remove perm-sym)
    ultimately have (σ # Σ) ≼ Γ
      using stronger-theory-right-permutation
      by blast
  }
  then show ?case by blast
 qed
 thus ?thesis using assms by blast
qed

lemma (in 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 $\gamma \in\# \ mset\ \Gamma$ **using** $\Phi(2)$
by (*meson mset-subset-eqD*)
moreover
let $?\Phi_0 = remove1\ (\gamma, \sigma)\ \Phi$
let $?\Sigma_0 = map\ snd\ ?\Phi_0$
from $\gamma\ \Phi(2)$ **have** $mset\ (map\ fst\ ?\Phi_0) \subseteq\# \ mset\ (remove1\ \gamma\ \Gamma)$
by (*metis ex-mset*
listSubtract-monotonic
listSubtract-mset-homomorphism
remove1-pairs-list-projections-fst
mset-remove1)
moreover have $mset\ ?\Phi_0 \subseteq\# \ mset\ \Phi$ **by** *simp*
with $\Phi(3)$ **have** $\forall\ (\gamma,\sigma) \in set\ ?\Phi_0.\ \vdash \gamma \to \sigma$ **by** *fastforce*
ultimately have $?\Sigma_0 \preceq remove1\ \gamma\ \Gamma$
unfolding *stronger-theory-relation-def* **by** *blast*
moreover from $\gamma\ \Phi(1)$ **have** $mset\ ?\Sigma_0 = mset\ (remove1\ \sigma\ \Sigma)$
using *remove1-pairs-list-projections-snd*
by *fastforce*
hence $?\Sigma_0$ <~~> $remove1\ \sigma\ \Sigma$
using *mset-eq-perm* **by** *blast*
ultimately have $remove1\ \sigma\ \Sigma \preceq remove1\ \gamma\ \Gamma$
using *stronger-theory-left-permutation* **by** *auto*
moreover from $\gamma\ \Phi(3)$ **have** $\vdash \gamma \to \sigma$ **by** (*simp*, *fast*)
moreover from $\gamma\ \Phi(2)$ **have** $\gamma \in\# \ mset\ \Gamma$
using *mset-subset-eqD* **by** *fastforce*
ultimately show $\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \land (remove1\ \sigma\ \Sigma) \preceq (remove1\ \gamma\ \Gamma)$ **by** *auto*
next
assume $\exists\ \gamma \in set\ \Gamma.\ \vdash \gamma \to \sigma \land (remove1\ \sigma\ \Sigma) \preceq (remove1\ \gamma\ \Gamma)$
from *this* **obtain** $\Phi\ \gamma$ **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** $\sigma\ \#\ (remove1\ \sigma\ \Sigma)$ <~~> Σ
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:
    $map\ snd\ \Phi = \Sigma$
    $mset\ (map\ fst\ \Phi) \subseteq\# mset\ \Gamma$
    $\forall (\gamma, \sigma) \in set\ \Phi.\ \vdash \gamma \to \sigma$
    unfolding *stronger-theory-relation-def*
    by *auto*
  hence $mset\ (map\ fst\ \Phi) \subseteq\# mset\ (\gamma \# \Gamma)$
    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* Φ) = *mset* $\Sigma \wedge$
        mset (*map fst* Φ) $\subseteq\#$ *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* Φ) = *mset* $\Sigma \wedge$
        mset (*map fst* Φ) $\subseteq\#$ *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* Φ) = *mset* $\Sigma \wedge$
          mset (*map fst* Φ) $\subseteq\#$ *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* Φ) = *mset* $\Sigma \wedge$
          mset (*map fst* Φ) $\subseteq\#$ *mset* $(\gamma \# \Gamma) \wedge$

```
          (∀(γ, σ)∈set Φ. ⊢ γ → σ)
    unfolding stronger-theory-relation-def
    by metis
next
  assume ∃Φ. mset (map snd Φ) = mset Σ ∧
             mset (map fst Φ) ⊆# mset (γ # Γ) ∧
             (∀(γ, σ)∈set Φ. ⊢ γ → σ)
  from this obtain Φ where Φ:
    mset (map snd Φ) = mset Σ
    mset (map fst Φ) ⊆# mset (γ # Γ)
    ∀(γ, σ)∈set Φ. ⊢ γ → σ
    by metis
  show Σ ≼ (γ # Γ)
  proof (cases ∃ σ. (γ, σ) ∈ set Φ)
    assume ∃ σ. (γ, σ) ∈ set Φ
    from this obtain σ where σ: (γ, σ) ∈ set Φ by auto
    let ?Φ = remove1 (γ, σ) Φ
    from σ have mset (map snd ?Φ) = mset (remove1 σ Σ)
      using Φ(1) remove1-pairs-list-projections-snd by force+
    moreover
    from σ have mset (map fst ?Φ) = mset (remove1 γ (map fst Φ))
      using Φ(1) remove1-pairs-list-projections-fst by force+
    with Φ(2) have mset (map fst ?Φ) ⊆# mset Γ
      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 :⊢ $\sigma \to \varphi$ **by** *blast*
moreover
have $\sigma \in$ *set* (σ # Σ) **by** *auto*
with ‹(σ # Σ) $\preceq$ Γ› **obtain** γ **where** γ: $\gamma \in$ *set* $\Gamma \vdash \gamma \to \sigma$
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 $\forall$ Σ Γ. *mset* Σ ⊆# *mset* $\Gamma \longrightarrow \Sigma$ \$⊢ $\Phi \longrightarrow \Gamma$ \$⊢ 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* ($\sqcup$)) Ψ :⊢ φ
map (*uncurry* ($\to$)) Ψ @ $\Sigma \ominus$ (*map snd* Ψ) \$⊢ Φ
using *segmented-deduction.simps*(*2*) **by** *blast*
let *?*Ψ = *map snd* Ψ
let *?*Ψ' = *map* (*uncurry* ($\to$)) Ψ
let *?*Σ' = *?*Ψ' @ ($\Sigma \ominus$ *?*Ψ)
let *?*Γ' = *?*Ψ' @ ($\Gamma \ominus$ *?*Ψ)
from Ψ **have** *mset ?*Ψ ⊆# *mset* Γ
using ‹*mset* Σ ⊆# *mset* Γ› *subset-mset.order.trans* **by** *blast*
moreover have *mset* ($\Sigma \ominus$ *?*Ψ) ⊆# *mset* ($\Gamma \ominus$ *?*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) $\subseteq\#$ *mset* Γ
shows *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq \Gamma$
proof −
have $\forall$ Γ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma \longrightarrow$ *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq \Gamma$
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
{
fix Γ
assume *mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ
hence *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*remove1* (*snd* σ) Γ)
by (*simp add*: *insert-subset-eq-iff*)
with *Cons* **have** *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq$ *remove1* (*snd* σ) Γ **by** *blast*
moreover have *uncurry* ($\sqcup$) = (λ σ. *fst* σ $\sqcup$ *snd* σ) **by** *fastforce*
hence *uncurry* ($\sqcup$) σ = *fst* σ $\sqcup$ *snd* σ **by** *simp*
moreover have $\vdash$ *snd* $\sigma \rightarrow$ (*fst* σ $\sqcup$ *snd* σ)
unfolding *disjunction-def*
by (*simp add*: *Axiom-1*)
ultimately have *map* (*uncurry* ($\sqcup$)) (σ # Σ) $\preceq$ (*snd* σ # (*remove1* (*snd* σ)
Γ))
by (*simp add*: *stronger-theory-left-right-cons*)
moreover have *mset* (*snd* σ # (*remove1* (*snd* σ) Γ)) = *mset* Γ
using ‹*mset* (*map snd* (σ # Σ)) $\subseteq\#$ *mset* Γ›
by (*simp*, *meson insert-DiffM mset-subset-eq-insertD*)
ultimately have *map* (*uncurry* ($\sqcup$)) (σ # Σ) $\preceq \Gamma$
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 \$$\vdash$ [φ] = Γ :$\vdash$ φ
proof (*rule iffI*)
assume Γ \$$\vdash$ [φ]
from *this* **obtain** Σ **where**
Σ: *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ
map (*uncurry* ($\sqcup$)) Σ :$\vdash$ φ
by *auto*
hence *map* (*uncurry* ($\sqcup$)) $\Sigma \preceq \Gamma$
using *witness-weaker-theory* **by** *blast*
thus Γ :$\vdash$ φ **using** Σ(*2*)
using *stronger-theory-deduction-monotonic* **by** *blast*
next
assume Γ :$\vdash$ φ

```
  let ?Σ = map (λ γ. (⊥, γ)) Γ
  have Γ ⪯ 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 $\Sigma(1)$ **have** *mset* (*map snd* $?\Sigma$) = *mset* (*remove1* φ Φ)
  using *remove1-pairs-list-projections-snd* **by** *fastforce*
 moreover from φ **have** *mset* (*map fst* $?\Sigma$) = *mset* (*remove1* δ (*map fst*
Σ))
  using *remove1-pairs-list-projections-fst* **by** *fastforce*
hence *mset* (*map fst* $?\Sigma$) $\subseteq\#$ *mset* Δ
  using $\Sigma(2)$ *mset.simps*(*1*) *subset-eq-diff-conv* **by** *force*
moreover from $\Sigma(3)$ **have** $\forall$ $(\delta,\varphi) \in$ *set* $?\Sigma$. $\vdash \delta \to \varphi$ **by** *auto*
ultimately have *remove1* φ $\Phi \preceq \Delta$
  unfolding *stronger-theory-relation-alt-def* **by** *blast*
hence (*remove1* φ Φ @ Ψ) $\preceq$ (Δ @ Γ) **using** *Cons* **by** *auto*
from *this* **obtain** Ω **where** Ω:
  map snd Ω = (*remove1* φ Φ) @ Ψ
  mset (*map fst* Ω) $\subseteq\#$ *mset* (Δ @ Γ)
  $\forall$ $(\alpha,\beta) \in$ *set* Ω. $\vdash \alpha \to \beta$
  unfolding *stronger-theory-relation-def*
  by *blast*
let $?\Omega = (\delta, \varphi)$ # Ω
have *map snd* $?\Omega = \varphi$ # *remove1* φ Φ @ Ψ
  using $\Omega(1)$ **by** *simp*
moreover have *mset* (*map fst* $?\Omega$) $\subseteq\#$ *mset* ((δ # Δ) @ Γ)
  using $\Omega(2)$ **by** *simp*
moreover have $\vdash \delta \to \varphi$
  using $\Sigma(3)$ φ **by** *blast*
hence $\forall$ $(\alpha,\beta) \in$ *set* $?\Omega$. $\vdash \alpha \to \beta$ **using** $\Omega(3)$ **by** *auto*
ultimately have (φ # *remove1* φ Φ @ Ψ) $\preceq$ ((δ # Δ) @ Γ)
  by (*metis stronger-theory-relation-def*)
moreover have $\varphi \in$ *set* Φ
  using $\Sigma(1)$ 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 $\nexists\varphi$. $(\delta, \varphi) \in$ *set* Σ
hence $\delta \notin$ *set* (*map fst* Σ)
    mset Δ + *add-mset* δ (*mset* []) = *mset* (δ # Δ)
  by *auto*
hence *mset* (*map fst* Σ) $\subseteq\#$ *mset* Δ
  by (*metis* (*no-types*) ‹*mset* (*map fst* Σ) $\subseteq\#$ *mset* (δ # Δ)›
    diff-single-trivial
    mset.simps(*1*)
    set-mset-mset
    subset-eq-diff-conv)
with $\Sigma(1)$ $\Sigma(3)$ **have** $\Phi \preceq \Delta$
  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* ($\sqcup$)) $\Psi \preceq$ *map* (*uncurry* ($\sqcup$)) *?*Θ
proof $-$
let *?*Φ = *map* (λ (ψ, σ, γ). ($\psi \sqcup \gamma$, $\psi \sqcup \sigma$)) Ω
have *map snd ?*Φ = *map* (*uncurry* ($\sqcup$)) Ψ
using Ω(*1*) **by** *fastforce*
moreover have *map fst ?*Φ = *map* (*uncurry* ($\sqcup$)) *?*Θ
by *fastforce*
hence *mset* (*map fst ?*Φ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) *?*Θ)
by (*metis subset-mset.dual-order.refl*)
moreover
have *mset* (*map* (λ(ψ, σ, -). (ψ, σ)) Ω) $\subseteq$# *mset* Ψ
using Ω(*1*) **by** *simp*
hence $\forall$ (φ,χ) $\in$ *set ?*Φ. $\vdash \varphi \to \chi$ **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). ($\psi \sqcup \gamma$, $\psi \sqcup \sigma$)) (ω # Ω)
let *?*Φ' = *map* (λ (ψ, σ, γ). ($\psi \sqcup \gamma$, $\psi \sqcup \sigma$)) Ω
have *mset* (*map* (λ(ψ, σ, -). (ψ, σ)) Ω) $\subseteq$# *mset* Ψ
mset (*map* (λ(-, σ, γ). (γ, σ)) Ω) $\subseteq$# *mset* Δ
using *Cons.prems*(*1*) *Cons.prems*(*2*) *subset-mset.dual-order.trans* **by**
fastforce+
with *Cons* **have** $\forall$ (φ,χ) $\in$ *set ?*Φ'. $\vdash \varphi \to \chi$ **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 $\vdash$ *?*γ $\to$ *?*σ
using *Cons.prems*(*2*) *mset-subset-eqD* Δ(*3*)
by *fastforce*
hence $\vdash$ (*?*ψ $\sqcup$ *?*γ) $\to$ (*?*ψ $\sqcup$ *?*σ)
unfolding *disjunction-def*
using *Modus-Ponens hypothetical-syllogism*
by *blast*
moreover have
(λ(ψ, σ, γ). ($\psi \sqcup \gamma$, $\psi \sqcup \sigma$)) =
(λ ω. (((λ (ψ, -, -). ψ) ω) $\sqcup$ ((λ (-, -, γ). γ) ω),
((λ (ψ, -, -). ψ) ω) $\sqcup$ ((λ (-, σ, -). σ) ω)))
by *auto*
hence (λ(ψ, σ, γ). ($\psi \sqcup \gamma$, $\psi \sqcup \sigma$)) ω = ((*?*ψ $\sqcup$ *?*γ), (*?*ψ $\sqcup$ *?*σ)) **by** *metis*
ultimately show *?case* **by** *simp*
qed
ultimately show *?thesis*

unfolding *stronger-theory-relation-def*
by *blast*
qed
with $\Psi(2)$ **have** *map* (*uncurry* ($\sqcup$)) *?*Θ :$\vdash$ φ
by (*metis stronger-theory-deduction-monotonic*)
moreover have
(*map* (*uncurry* ($\rightarrow$)) Ψ @ Σ $\ominus$ (*map snd* Ψ)) $\preceq$
(*map* (*uncurry* ($\rightarrow$)) *?*Θ @ Γ $\ominus$ (*map snd ?*Θ))
proof −
have *map* (*uncurry* ($\rightarrow$)) Ψ $\preceq$ *map* (*uncurry* ($\rightarrow$)) *?*Θ
proof −
let *?*Φ = *map* (λ (ψ, σ, γ). ($\psi \rightarrow \gamma$, $\psi \rightarrow \sigma$)) Ω
have *map snd ?*Φ = *map* (*uncurry* ($\rightarrow$)) Ψ
using $\Omega(1)$ **by** *fastforce*
moreover have *map fst ?*Φ = *map* (*uncurry* ($\rightarrow$)) *?*Θ
by *fastforce*
hence *mset* (*map fst ?*Φ) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) *?*Θ)
by (*metis subset-mset.dual-order.refl*)
moreover
have *mset* (*map* (λ(ψ, σ, -). (ψ, σ)) Ω) $\subseteq$# *mset* Ψ
using $\Omega(1)$ **by** *simp*
hence $\forall$ (φ,χ) $\in$ *set ?*Φ. $\vdash \varphi \rightarrow \chi$ **using** $\Omega(2)$
proof (*induct* Ω)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ω Ω)
let *?*Φ = *map* (λ (ψ, σ, γ). ($\psi \rightarrow \gamma$, $\psi \rightarrow \sigma$)) (ω # Ω)
let *?*Φ' = *map* (λ (ψ, σ, γ). ($\psi \rightarrow \gamma$, $\psi \rightarrow \sigma$)) Ω
have *mset* (*map* (λ(ψ, σ, -). (ψ, σ)) Ω) $\subseteq$# *mset* Ψ
mset (*map* (λ(-, σ, γ). (γ, σ)) Ω) $\subseteq$# *mset* Δ
using *Cons.prems*(*1*) *Cons.prems*(*2*) *subset-mset.dual-order.trans* **by**
fastforce+
with *Cons* **have** $\forall$ (φ,χ) $\in$ *set ?*Φ'. $\vdash \varphi \rightarrow \chi$ **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 $\vdash$ *?*γ $\rightarrow$ *?*σ
using *Cons.prems*(*2*) *mset-subset-eqD* $\Delta(3)$
by *fastforce*
hence $\vdash$ (*?*ψ $\rightarrow$ *?*γ) $\rightarrow$ (*?*ψ $\rightarrow$ *?*σ)
using *Modus-Ponens hypothetical-syllogism*
by *blast*
moreover have
(λ(ψ, σ, γ). ($\psi \rightarrow \gamma$, $\psi \rightarrow \sigma$)) =

$(\lambda\ \omega.\ (((\lambda\ (\psi,\ \text{-},\ \text{-}).\ \psi)\ \omega) \rightarrow ((\lambda\ (\text{-},\ \text{-},\ \gamma).\ \gamma)\ \omega),$
$((\lambda\ (\psi,\ \text{-},\ \text{-}).\ \psi)\ \omega) \rightarrow ((\lambda\ (\text{-},\ \sigma,\ \text{-}).\ \sigma)\ \omega)))$
by *auto*
hence $(\lambda(\psi,\ \sigma,\ \gamma).\ (\psi \rightarrow \gamma,\ \psi \rightarrow \sigma))\ \omega = ((?\psi \rightarrow ?\gamma),\ (?\psi \rightarrow ?\sigma))$ **by** *metis*
ultimately show *?case* **by** *simp*
qed
ultimately show *?thesis*
unfolding *stronger-theory-relation-def*
by *blast*
qed
moreover
have $(\Sigma \ominus (map\ snd\ \Psi)) \preceq (\Gamma \ominus (map\ snd\ ?\Theta))$
proof −
let $?\Delta = \Delta \ominus (map\ (\lambda\ (\text{-},\ \sigma,\ \gamma).\ (\gamma,\ \sigma))\ \Omega)$
have $mset\ (map\ fst\ ?\Delta) \subseteq\#\ mset\ (\Gamma \ominus (map\ snd\ ?\Theta))$
using $\Delta(2)$
by (*metis* $\Omega(2)$
‹$map\ snd\ (map\ (\lambda(\psi,\ \text{-},\ \gamma).\ (\psi,\ \gamma))\ \Omega) =$
$map\ fst\ (map\ (\lambda(\text{-},\ \sigma,\ \gamma).\ (\gamma,\ \sigma))\ \Omega)$›
listSubtract-monotonic
map-listSubtract-mset-equivalence)
moreover
from $\Omega(2)$ **have** $mset\ ?\Delta \subseteq\#\ mset\ \Delta$ **by** *simp*
hence $\forall\ (\gamma,\sigma) \in set\ ?\Delta.\ \vdash \gamma \rightarrow \sigma$
using $\Delta(3)$
by (*metis mset-subset-eqD set-mset-mset*)
moreover
have $map\ snd\ (map\ (\lambda(\text{-},\ \sigma,\ \gamma).\ (\gamma,\ \sigma))\ \Omega) = map\ snd\ \Psi$
using $\Omega(1)$
by (*induct* Ω, *simp*, *fastforce*)
hence $mset\ (map\ snd\ ?\Delta) = mset\ (\Sigma \ominus (map\ snd\ \Psi))$
by (*metis* $\Delta(1)$ $\Omega(2)$ *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 $map\ (uncurry\ (\rightarrow))\ ?\Theta\ @\ \Gamma \ominus (map\ snd\ ?\Theta)\ \$\vdash\ \Phi$
using $\Psi(3)$ *Cons* **by** *blast*
ultimately have $\Gamma\ \$\vdash\ (\varphi\ \#\ \Phi)$
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 $uncurry\ (\sqcup) = (\lambda\ \psi.\ (fst\ \psi) \sqcup (snd\ \psi))$
by $fastforce$
with γ **have** $uncurry\ (\sqcup)\ \psi = ?\psi \sqcup \sim \gamma$
by $simp$
with $tautology$ **have** $\vdash \sim(\gamma \setminus ?\psi) \rightarrow uncurry\ (\sqcup)\ \psi$
by $simp$
ultimately have $map\ (uncurry\ (\sqcup))\ (\psi \# \Psi) \preceq$
$\boldsymbol{\sim}\ (map\ (uncurry\ (\setminus))\ ((zip\ ((hd\ \Delta) \# (tl\ \Delta))\ (map\ fst\ (\psi \#$
$\Psi)))))$
using $stronger\text{-}theory\text{-}left\text{-}right\text{-}cons\ \gamma(2)$
by $simp$
hence $map\ (uncurry\ (\sqcup))\ (\psi \# \Psi) \preceq$
$\boldsymbol{\sim}\ (map\ (uncurry\ (\setminus))\ (zip\ \Delta\ (map\ fst\ (\psi \# \Psi))))$
using $\langle map\ snd\ (\psi \# \Psi) = \boldsymbol{\sim}\ \Delta \rangle$ **by** $force$
}
thus $?case$ **by** $blast$
qed
with $\Psi(2)\ \Delta(2)$ **have** $\boldsymbol{\sim}\ (map\ (uncurry\ (\setminus))\ ?\Psi) :\vdash \varphi$
using $stronger\text{-}theory\text{-}deduction\text{-}monotonic$ **by** $blast$
moreover
have $(map\ (uncurry\ (\rightarrow))\ \Psi\ @\ \boldsymbol{\sim}\ \Gamma \ominus map\ snd\ \Psi) \preceq$
$\boldsymbol{\sim}\ (map\ (uncurry\ (\sqcap))\ ?\Psi\ @\ \Gamma \ominus (map\ fst\ ?\Psi))$
proof $-$
from $\Delta(1)$ **have** $mset\ (\boldsymbol{\sim}\ \Gamma \ominus \boldsymbol{\sim}\ \Delta) = mset\ (\boldsymbol{\sim}\ (\Gamma \ominus \Delta))$
by $(simp\ add\colon image\text{-}mset\text{-}Diff)$
hence $mset\ (\boldsymbol{\sim}\ \Gamma \ominus map\ snd\ \Psi) = mset\ (\boldsymbol{\sim}\ (\Gamma \ominus map\ fst\ ?\Psi))$
using $\Psi(1)\ \Delta(2)\ \langle map\ fst\ ?\Psi = \Delta \rangle$ **by** $simp$
hence $(\boldsymbol{\sim}\ \Gamma \ominus map\ snd\ \Psi) \preceq \boldsymbol{\sim}\ (\Gamma \ominus map\ fst\ ?\Psi)$
by $(simp\ add\colon msub\text{-}stronger\text{-}theory\text{-}intro)$
moreover have $\forall\ \Delta.\ map\ snd\ \Psi = \boldsymbol{\sim}\ \Delta \longrightarrow$
$map\ (uncurry\ (\rightarrow))\ \Psi \preceq \boldsymbol{\sim}\ (map\ (uncurry\ (\sqcap))\ (zip\ \Delta\ (map$
$fst\ \Psi)))$
proof $(induct\ \Psi)$
case Nil
then show $?case$ **by** $simp$
next
case $(Cons\ \psi\ \Psi)$
let $?\psi = fst\ \psi$
{
fix Δ
assume $map\ snd\ (\psi \# \Psi) = \boldsymbol{\sim}\ \Delta$
from $this$ **obtain** γ **where** $\gamma\colon \sim \gamma = snd\ \psi\ \gamma = hd\ \Delta$ **by** $auto$
from $\langle map\ snd\ (\psi \# \Psi) = \boldsymbol{\sim}\ \Delta \rangle$ **have** $map\ snd\ \Psi = \boldsymbol{\sim}\ (tl\ \Delta)$ **by** $auto$
with $Cons.hyps$ **have**
$map\ (uncurry\ (\rightarrow))\ \Psi \preceq \boldsymbol{\sim}\ (map\ (uncurry\ (\sqcap))\ (zip\ (tl\ \Delta)\ (map\ fst\ \Psi)))$
by $simp$
moreover
{

fix ψ γ
have $\vdash \sim(\gamma \sqcap \psi) \to (\psi \to \sim \gamma)$
unfolding *disjunction-def*
conjunction-def
negation-def
by (*meson Modus-Ponens*
flip-implication
hypothetical-syllogism)
} **note** *tautology* = *this*
have (*uncurry* ($\to$)) = (λ ψ. (*fst* ψ) $\to$ (*snd* ψ))
by *fastforce*
with γ **have** *uncurry* ($\to$) ψ = *?*ψ $\to \sim \gamma$
by *simp*
with *tautology* **have** $\vdash \sim(\gamma \sqcap ?\psi) \to$ (*uncurry* ($\to$)) ψ
by *simp*
ultimately have *map* (*uncurry* ($\to$)) (ψ # Ψ) $\preceq$
$\boldsymbol{\sim}$ (*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* ($\to$)) (ψ # Ψ) $\preceq$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) (*zip* Δ (*map fst* (ψ # Ψ))))
using ‹*map snd* (ψ # Ψ) = $\boldsymbol{\sim}$ Δ› **by** *force*
}
then show *?case* **by** *blast*
qed
ultimately have (*map* (*uncurry* ($\to$)) Ψ @ $\boldsymbol{\sim}$ $\Gamma \ominus$ *map snd* Ψ) $\preceq$
($\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) *?*Ψ) @ $\boldsymbol{\sim}$ ($\Gamma \ominus$ (*map fst ?*Ψ)))
using *stronger-theory-combine* Δ(*2*)
by *metis*
thus *?thesis* **by** *simp*
qed
hence $\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) *?*Ψ @ $\Gamma \ominus$ (*map fst ?*Ψ)) \$$\vdash \Phi$
using Ψ(*3*) *segmented-stronger-theory-left-monotonic*
by *blast*
ultimately show $\exists\,\Psi$. *mset* (*map fst* Ψ) $\subseteq$# *mset* Γ $\wedge$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\backslash$)) Ψ) :$\vdash \varphi \wedge$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) Ψ @ $\Gamma \ominus$ (*map fst* Ψ)) \$$\vdash \Phi$
by *metis*
next
assume $\exists\,\Psi$. *mset* (*map fst* Ψ) $\subseteq$# *mset* Γ $\wedge$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\backslash$)) Ψ) :$\vdash \varphi \wedge$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) Ψ @ $\Gamma \ominus$ *map fst* Ψ) \$$\vdash \Phi$
from *this* **obtain** Ψ **where** Ψ:
mset (*map fst* Ψ) $\subseteq$# *mset* Γ
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\backslash$)) Ψ) :$\vdash \varphi$
$\boldsymbol{\sim}$ (*map* (*uncurry* ($\sqcap$)) Ψ @ $\Gamma \ominus$ *map fst* Ψ) \$$\vdash \Phi$
by *auto*
let *?*Ψ = *zip* (*map snd* Ψ) ($\boldsymbol{\sim}$ (*map fst* Ψ))

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

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

lemma (in Logical-Probability) segmented-deduction-summation-introduction:
  assumes ∼ Γ $⊢ ∼ Φ
  shows (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
proof −
  have ∀ Γ. ∼ Γ $⊢ ∼ Φ ⟶ (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
  proof (induct Φ)
    case Nil
    then show ?case
      by (simp, metis (full-types) ex-map-conv 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:
  mset (*map fst* Ψ) $\subseteq\#$ *mset* Γ
  $\sim$ (*map* (*uncurry* ($\backslash$)) Ψ) $:\vdash \sim \varphi$
  $\sim$ (*map* (*uncurry* ($\sqcap$)) Ψ @ $\Gamma \ominus$ (*map fst* Ψ)) $\$\vdash \sim \Phi$
  using *negated-segmented-deduction* **by** *blast*
let $?\Gamma = \Gamma \ominus$ (*map fst* Ψ)
let $?\Psi_1$ = *map* (*uncurry* ($\backslash$)) Ψ
let $?\Psi_2$ = *map* (*uncurry* ($\sqcap$)) Ψ
have $(\sum \varphi' \leftarrow \Phi.\ Pr\ \varphi') \leq (\sum \varphi \leftarrow (?\Psi_2\ @\ ?\Gamma).\ Pr\ \varphi)$
  using *Cons* Ψ(*3*) **by** *blast*
moreover
have $Pr\ \varphi \leq (\sum \varphi \leftarrow ?\Psi_1.\ Pr\ \varphi)$
  using Ψ(*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* ($\backslash$)) Ψ
  let $?\Psi_2$ = *map* (*uncurry* ($\sqcap$)) Ψ
  let $?\psi_1$ = *uncurry* ($\backslash$) ψ
  let $?\psi_2$ = *uncurry* ($\sqcap$) ψ
  assume $(\sum \varphi' \leftarrow (?\Psi_1\ @\ ?\Psi_2).\ Pr\ \varphi') = (\sum \gamma \leftarrow (map\ fst\ \Psi).\ Pr\ \gamma)$
  moreover
  {
    let $?\gamma$ = *fst* ψ
    let $?\psi$ = *snd* ψ
    have *uncurry* ($\backslash$) = ($\lambda\ \psi$. (*fst* ψ) $\backslash$ (*snd* ψ))
      uncurry ($\sqcap$) = ($\lambda\ \psi$. (*fst* ψ) $\sqcap$ (*snd* ψ))
     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* ($\rightarrow$)) ($\mathfrak{A}$ Ψ (δ # Δ))) =
mset (*map* *snd* ($\mathfrak{B}$ Ψ (δ # Δ)))
proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
case *True*
then show *?thesis* **using** *Cons* **by** *simp*
next
case *False*
from *this* **obtain** ψ **where**
find ($\lambda\psi$. *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
uncurry ($\rightarrow$) ψ = *snd* δ
using *find-Some-predicate*
by *fastforce*
then show *?thesis* **using** *Cons* **by** *simp*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

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

lemma (**in** *Minimal-Logic*) *firstComponent-msub*:
mset ($\mathfrak{A}$ Ψ Δ) $\subseteq$# *mset* Ψ
proof −
have $\forall$ Ψ. *mset* ($\mathfrak{A}$ Ψ Δ) $\subseteq$# *mset* Ψ
proof(*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* ($\mathfrak{A}$ Ψ (δ # Δ)) $\subseteq$# *mset* Ψ
proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
case *True*
then show *?thesis* **using** *Cons* **by** *simp*
next
case *False*
from *this* **obtain** ψ **where**
ψ: *find* ($\lambda\psi$. *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
ψ $\in$ *set* Ψ
using *find-Some-set-membership*
by *fastforce*
have *mset* ($\mathfrak{A}$ (*remove1* ψ Ψ) Δ) $\subseteq$# *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 ($\mathfrak{B}$ Ψ Δ) $\subseteq\#$ *mset* Δ
proof −
have $\forall\,\Psi$. *mset* ($\mathfrak{B}$ Ψ Δ) $\subseteq\#$ *mset* Δ
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* ($\mathfrak{B}$ Ψ (δ # Δ)) $\subseteq\#$ *mset* (δ # Δ)
using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) 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* ($\mathfrak{B}$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
proof −
have $\forall\,\Psi$. *mset* (*map snd* ($\mathfrak{B}$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* ($\mathfrak{B}$ Ψ (δ # Δ))) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
proof (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ = *None*)
case *True*
then show *?thesis*

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

$\vee$ *search* Ψ ($\lambda p.$ *uncurry* ($\rightarrow$) p = *snd* δ) $\in$ *set* Ψ
$\wedge$ *uncurry* ($\rightarrow$) (*search* Ψ ($\lambda p.$ *uncurry* ($\rightarrow$) p = *snd* δ)) = *snd* δ)
by *blast*
hence *snd* $\delta \in$ *set* (*map* (*uncurry* ($\rightarrow$)) Ψ)
by (*metis* (*no-types*) *False image-eqI image-set*)
moreover
have *A*: *add-mset* (*uncurry* ($\rightarrow$) ψ) (*image-mset snd* (*mset* Δ))
= *image-mset snd* (*add-mset* δ (*mset* Δ))
by (*simp add*: ‹*snd* δ = *uncurry* ($\rightarrow$) ψ›)
have *B*: {#*snd* δ#} $\subseteq$# *image-mset* (*uncurry* ($\rightarrow$)) (*mset* Ψ)
using ‹*snd* $\delta \in$ *set* (*map* (*uncurry* ($\rightarrow$)) Ψ)› **by** *force*
have *image-mset* (*uncurry* ($\rightarrow$)) (*mset* Ψ) − {#*snd* δ#}
= *image-mset* (*uncurry* ($\rightarrow$)) (*mset* (*remove1* ψ Ψ))
by (*simp add*: ‹$\psi \in$ *set* Ψ› ‹*snd* δ = *uncurry* ($\rightarrow$) ψ› *image-mset-Diff*)
then have *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ (*remove1* ψ Ψ) Δ))
$\subseteq$# *mset* (*remove1* (*snd* ψ) $\Gamma \ominus$ *map snd* (*remove1* ψ Ψ))
by (*metis* (*no-types*)
A B $\diamondsuit$ *Cons.hyps*
calculation(*1*)
calculation(*4*)
insert-subset-eq-iff
mset.simps(*2*)
mset-map
subset-mset.diff-add-assoc2
union-code)
ultimately show *?thesis* **by** *fastforce*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *auto*
qed

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

lemma (**in** *Classical-Propositional-Logic*) *mergeWitness-right-empty* [*simp*]:
$\mathfrak{J}$ [] Δ = Δ
by (*induct* Δ, *simp*+)

lemma (**in** *Classical-Propositional-Logic*) *secondComponent-mergeWitness-snd-projection*:
mset (*map snd* Ψ @ *map snd* ($\Delta \ominus$ ($\mathfrak{B}$ Ψ Δ))) = *mset* (*map snd* ($\mathfrak{J}$ Ψ Δ))
proof −

have $\forall\ \Psi.\ mset\ (map\ snd\ \Psi\ @\ map\ snd\ (\Delta \ominus (\mathfrak{B}\ \Psi\ \Delta))) = mset\ (map\ snd\ (\mathfrak{J}\ \Psi\ \Delta))$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have $mset\ (map\ snd\ \Psi\ @\ map\ snd\ ((\delta \# \Delta) \ominus \mathfrak{B}\ \Psi\ (\delta \# \Delta))) = mset\ (map\ snd\ (\mathfrak{J}\ \Psi\ (\delta \# \Delta)))$
proof (*cases find* $(\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None$)
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: $find\ (\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
by *auto*
moreover have $\psi \in set\ \Psi$
by (*meson* ψ *find-Some-set-membership*)
moreover
let $?\Psi' = remove1\ \psi\ \Psi$
from *Cons* **have**
$mset\ (map\ snd\ ?\Psi'\ @\ map\ snd\ (\Delta \ominus \mathfrak{B}\ ?\Psi'\ \Delta)) = mset\ (map\ snd\ (\mathfrak{J}\ ?\Psi'\ \Delta))$
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 (→)) Δ @ map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ Δ)) ≼
    map (uncurry (→)) (𝔍 Ψ Δ)
proof −
  have ∀ Ψ. (map (uncurry (→)) Δ @
             map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ Δ)) ≼
             map (uncurry (→)) (𝔍 Ψ Δ)
  proof (induct Δ)
    case Nil
    then show ?case
      by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have ⊢ (uncurry (→)) δ → (uncurry (→)) δ
        using Axiom-1 Modus-Ponens implication-absorption by blast
      have
        (map (uncurry (→)) (δ # Δ) @
          map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ))) ≼
          map (uncurry (→)) (𝔍 Ψ (δ # Δ))
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        thus ?thesis
          using Cons
                ‹⊢ (uncurry (→)) δ → (uncurry (→)) δ›
          by (simp, metis stronger-theory-left-right-cons)
      next
        case False
         from this obtain ψ where ψ: find (λψ. uncurry (→) ψ = snd δ) Ψ =
Some ψ
          by auto
        from ψ have snd δ = uncurry (→) ψ
          using find-Some-predicate by fastforce
        from ψ ‹snd δ = uncurry (→) ψ› have
          mset (map (uncurry (→)) (δ # Δ) @
                   map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ))) =
           mset (map (uncurry (→)) (δ # Δ) @
                   map (uncurry (→)) (remove1 ψ Ψ) ⊖
                   map snd (𝔅 (remove1 ψ Ψ) Δ))
          by (simp add: find-Some-set-membership image-mset-Diff)
        hence
          (map (uncurry (→)) (δ # Δ) @
              map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ))) ≼
           (map (uncurry (→)) (δ # Δ) @
             map (uncurry (→)) (remove1 ψ Ψ) ⊖ map snd (𝔅 (remove1 ψ Ψ) Δ))
          by (simp add: msub-stronger-theory-intro)
        with Cons ‹⊢ (uncurry (→)) δ → (uncurry (→)) δ› have
```

```
        (map (uncurry (→)) (δ # Δ) @
          map (uncurry (→)) Ψ ⊖ map snd (𝔅 Ψ (δ # Δ)))
          ≼ ((uncurry (→)) δ # map (uncurry (→)) (ℑ (remove1 ψ Ψ) Δ))
        using stronger-theory-left-right-cons
              stronger-theory-transitive
        by fastforce
      moreover
      let ?α = fst δ
      let ?β = fst ψ
      let ?γ = snd ψ
      have uncurry (→) = (λ δ. fst δ → snd δ) by fastforce
      with ψ have (uncurry (→)) δ = ?α → ?β → ?γ
        using find-Some-predicate by fastforce
      hence ⊢ ((?α ⊓ ?β) → ?γ) → (uncurry (→)) δ
        using biconditional-def curry-uncurry by auto
      with ψ have
        ((uncurry (→)) δ # map (uncurry (→)) (ℑ (remove1 ψ Ψ) Δ)) ≼
         map (uncurry (→)) (ℑ Ψ (δ # Δ))
        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) ⊆# mset Γ
     and mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
Ψ))
   shows mset (map snd (ℑ Ψ Δ)) ⊆# mset Γ
proof −
 have ∀Ψ Γ. mset (map snd Ψ) ⊆# mset Γ ⟶
            mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
Ψ)) ⟶
            mset (map snd (ℑ Ψ Δ)) ⊆# mset Γ
 proof (induct Δ)
  case Nil
  then show ?case by simp
 next
  case (Cons δ Δ)
  {
    fix Ψ :: ('a × 'a) list
    fix Γ :: 'a list
    assume ♢: mset (map snd Ψ) ⊆# mset Γ
              mset (map snd (δ # Δ)) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖
```

$(map\ snd\ \Psi))$

have *mset* $(map\ snd\ (\mathfrak{J}\ \Psi\ (\delta\ \#\ \Delta)))\subseteq\#$ *mset* Γ
proof $(cases\ find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None)$
case *True*
hence $snd\ \delta \notin set\ (map\ (uncurry\ (\rightarrow))\ \Psi)$
proof $(induct\ \Psi)$
case *Nil*
then show *?case* **by** *simp*
next
case $(Cons\ \psi\ \Psi)$
hence $uncurry\ (\rightarrow)\ \psi \neq snd\ \delta$ **by** *fastforce*
with *Cons* **show** *?case* **by** *fastforce*
qed
with $\Diamond(2)$ **have** $snd\ \delta \in\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$
using *mset-subset-eq-insertD* **by** *fastforce*
with $\Diamond(1)$ **have** $mset\ (map\ snd\ \Psi) \subseteq\#\ mset\ (remove1\ (snd\ \delta)\ \Gamma)$
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 $add\text{-}mset\ (snd\ \delta)\ (mset\ (\Gamma \ominus map\ snd\ \Psi) - \{\#snd\ \delta\#\}) = mset\ (\Gamma \ominus map\ snd\ \Psi)$
by (*meson* ‹$snd\ \delta \in\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$› *insert-DiffM*)
then have $image\text{-}mset\ snd\ (mset\ \Delta) - (mset\ \Gamma - add\text{-}mset\ (snd\ \delta)\ (image\text{-}mset\ snd\ (mset\ \Psi)))$
$\subseteq\#\ \{\#x \rightarrow y.\ (x,\ y) \in\#\ mset\ \Psi\#\}$
using $\Diamond(2)$ **by** (*simp*, *metis add-mset-diff-bothsides*
listSubtract-mset-homomorphism
mset-map subset-eq-diff-conv)
hence $mset\ (map\ snd\ \Delta)$
$\subseteq\#\ mset\ (map\ (uncurry\ (\rightarrow))\ \Psi\ @\ (remove1\ (snd\ \delta)\ \Gamma) \ominus (map\ snd\ \Psi))$
using *subset-eq-diff-conv* **by** (*simp*, *blast*)
ultimately have $mset\ (map\ snd\ (\mathfrak{J}\ \Psi\ \Delta)) \subseteq\#\ mset\ (remove1\ (snd\ \delta)\ \Gamma)$
using *Cons* **by** *blast*
hence $mset\ (map\ snd\ (\delta\ \#\ (\mathfrak{J}\ \Psi\ \Delta))) \subseteq\#\ mset\ \Gamma$
by (*simp*, *metis* ‹$snd\ \delta \in\#\ mset\ (\Gamma \ominus map\ snd\ \Psi)$›
cancel-ab-semigroup-add-class.diff-right-commute
diff-single-trivial
insert-subset-eq-iff
listSubtract-mset-homomorphism
multi-drop-mem-not-eq)
with ‹$find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None$›
show *?thesis*
by *simp*
next
case *False*

from *this* **obtain** ψ **where** ψ:
  find ($\lambda\psi$. *uncurry* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
  by *fastforce*
let $?\chi$ = *fst* ψ
let $?\gamma$ = *snd* ψ
have *uncurry* ($\rightarrow$) = (λ ψ. *fst* ψ $\rightarrow$ *snd* ψ)
  by *fastforce*
moreover
from *this* **have** *uncurry* ($\rightarrow$) ψ = $?\chi$ $\rightarrow$ $?\gamma$ **by** *fastforce*
with ψ **have** *A*: ($?\chi$, $?\gamma$) $\in$ *set* Ψ
    and *B*: *snd* δ = $?\chi$ $\rightarrow$ $?\gamma$
  using *find-Some-predicate*
  by (*simp add*: *find-Some-set-membership*, *fastforce*)
let $?\Psi'$ = *remove1* ($?\chi$, $?\gamma$) Ψ
from *B* $\diamondsuit$(*2*) **have**
  mset (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ Γ $\ominus$ *map snd* Ψ) − {# $?\chi$ $\rightarrow$ $?\gamma$ #}
  by (*simp add*: *insert-subset-eq-iff*)
moreover
have *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ)
    = *add-mset* (*case* (*fst* ψ, *snd* ψ) *of* (x, xa) $\Rightarrow$ x $\rightarrow$ xa)
        (*image-mset* (*uncurry* ($\rightarrow$)) (*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) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) $?\Psi'$ @ Γ $\ominus$ *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* $\diamondsuit$
have *mset* (Γ $\ominus$ *map snd* Ψ) = *mset*((*remove1* $?\gamma$ Γ) $\ominus$ (*remove1* $?\gamma$ (*map snd* Ψ)))
  by (*metis image-eqI*
        listSubtract-remove1-perm
        mset-eq-perm
        prod.sel(*2*)
        set-map)
with *A* **have** *mset* (Γ $\ominus$ *map snd* Ψ) = *mset*((*remove1* $?\gamma$ Γ) $\ominus$ (*map snd* $?\Psi'$))

    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* (ℑ *?*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* (ℑ *?*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* δ ⊓ *?*χ, *?*γ) # (ℑ *?*Ψ′ Δ))) ⊆# *mset* Γ
    by *simp*
  moreover from ψ **have**
    ℑ Ψ (δ # Δ) = (*fst* δ ⊓ *?*χ, *?*γ) # (ℑ *?*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* (⊔)) (ℑ Ψ Δ)
proof −
  have ∀ Ψ. *map* (*uncurry* (⊔)) Δ ⪯ *map* (*uncurry* (⊔)) (ℑ 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 (⊔)) (𝔍 Ψ (δ # Δ))
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        hence 𝔍 Ψ (δ # Δ) = δ # 𝔍 Ψ Δ
          by simp
        moreover have ⊢ (uncurry (⊔)) δ → (uncurry (⊔)) δ
          by (metis Axiom-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 (→) = (λ ψ. fst ψ → snd ψ)
             uncurry (⊔) = (λ δ. fst δ ⊔ snd δ)
          by fastforce+
        hence uncurry (⊔) δ = ?μ ⊔ (?χ → ?γ)
          using ψ find-Some-predicate
          by fastforce
        moreover
        {
          fix μ χ γ
          have ⊢ ((μ ⊓ χ) ⊔ γ) → (μ ⊔ (χ → γ))
          proof −
            have ∀𝔐. 𝔐 ⊨_prop (((⟨μ⟩ ⊓ ⟨χ⟩) ⊔ ⟨γ⟩) → (⟨μ⟩ ⊔ (⟨χ⟩ → ⟨γ⟩)))
              by fastforce
            hence ⊢ ⦇ (((⟨μ⟩ ⊓ ⟨χ⟩) ⊔ ⟨γ⟩) → (⟨μ⟩ ⊔ (⟨χ⟩ → ⟨γ⟩))) ⦈
              using propositional-semantics by blast
            thus ?thesis
              by simp
         qed
        }
        ultimately show ?thesis
          using Cons ψ stronger-theory-left-right-cons
          by simp
      qed
    }
    thus ?case by blast
  qed
  thus ?thesis by blast
qed

lemma (in Classical-Propositional-Logic) left-mergeWitness-stronger-theory:
  map (uncurry (⊔)) Ψ ≼ map (uncurry (⊔)) (𝔍 Ψ Δ)
```

proof −
have $\forall\ \Psi.\ map\ (uncurry\ (\sqcup))\ \Psi \preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Psi\ \Delta)$
proof $(induct\ \Delta)$
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 $(cases\ find\ (\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None)$
case *True*
then show *?thesis*
using *Cons stronger-theory-right-cons*
by *auto*
next
case *False*
from *this* **obtain** ψ **where** ψ:
$find\ (\lambda\psi.\ uncurry\ (\rightarrow)\ \psi = snd\ \delta)\ \Psi = Some\ \psi$
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* Ψ
by (*simp add*: *find-Some-set-membership*)
hence *mset* (*map* (*uncurry* ($\sqcup$)) (ψ # (*remove1* ψ Ψ))) =
mset (*map* (*uncurry* ($\sqcup$)) Ψ)
by (*metis insert-DiffM*
mset.simps(*2*)
mset-map
mset-remove1
set-mset-mset)
hence *map* (*uncurry* ($\sqcup$)) $\Psi \preceq$ *map* (*uncurry* ($\sqcup$)) (ψ # (*remove1* ψ Ψ))
by (*simp add*: *msub-stronger-theory-intro*)
ultimately show *?thesis*
using *stronger-theory-transitive* **by** *blast*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *mergeWitness-segmented-deduction-intro*:
assumes *mset* (*map snd* Δ) $\subseteq$# *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
and *map* (*uncurry* ($\rightarrow$)) Δ @ (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ *map snd* Ψ) $\ominus$ *map snd* Δ \$$\vdash$ Φ
(**is** *?*Γ_0 \$$\vdash$ Φ)
shows *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{J}$ 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* ($\rightarrow$)) Δ
let *?B* = *map* (*uncurry* ($\rightarrow$)) 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* (Γ $\ominus$ *map snd* ($\mathfrak{J}$ Ψ Δ))
using *secondComponent-mergeWitness-snd-projection*
by *simp*
with *calculation* **have** (*?A* @ (*?B* $\ominus$ *?C*) @ (*?D* $\ominus$ *?E*)) $\preceq$ *?*Γ
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 *?*Γ_0 $\preceq$ *?*Γ
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*:
Γ \$$\vdash$ (φ # Φ) = Γ \$$\vdash$ (ψ $\sqcup$ φ # ψ $\to$ φ # Φ)
proof (*rule iffI*)
assume Γ \$$\vdash$ (φ # Φ)
from *this* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq$# *mset* Γ
map (*uncurry* ($\sqcup$)) Ψ :$\vdash$ φ
(*map* (*uncurry* ($\to$)) Ψ @ Γ $\ominus$ (*map snd* Ψ)) \$$\vdash$ Φ
by *auto*
let *?*Ψ_1 = *zip* (*map* (λ (χ,γ). ψ $\sqcup$ χ) Ψ) (*map snd* Ψ)
let *?*Γ_1 = *map* (*uncurry* ($\to$)) *?*Ψ_1 @ Γ $\ominus$ (*map snd* *?*Ψ_1)
let *?*Ψ_2 = *zip* (*map* (λ (χ,γ). ψ $\to$ χ) Ψ) (*map* (*uncurry* ($\to$)) *?*Ψ_1)

let $?\Gamma_2 = map\ (uncurry\ (\rightarrow))\ ?\Psi_2\ @\ ?\Gamma_1 \ominus (map\ snd\ ?\Psi_2)$
have $map\ (uncurry\ (\rightarrow))\ \Psi \preceq map\ (uncurry\ (\rightarrow))\ ?\Psi_2$
proof $(induct\ \Psi)$
case Nil
then show $?case$ **by** $simp$
next
case $(Cons\ \delta\ \Psi)$
let $?\chi = fst\ \delta$
let $?\gamma = snd\ \delta$
let $?\Psi_1 = zip\ (map\ (\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi)\ (map\ snd\ \Psi)$
let $?\Psi_2 = zip\ (map\ (\lambda\ (\chi,\gamma).\ \psi \rightarrow \chi)\ \Psi)\ (map\ (uncurry\ (\rightarrow))\ ?\Psi_1)$
let $?T_1 = \lambda\ \Psi.\ map\ (uncurry\ (\rightarrow))\ (zip\ (map\ (\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi)\ (map\ snd\ \Psi))$
let $?T_2 = \lambda\ \Psi.\ map\ (uncurry\ (\rightarrow))\ (zip\ (map\ (\lambda\ (\chi,\gamma).\ \psi \rightarrow \chi)\ \Psi)\ (?T_1\ \Psi))$
{
fix $\delta :: 'a \times 'a$
have $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi) = (\lambda\ \delta.\ \psi \sqcup (fst\ \delta))$
$(\lambda\ (\chi,\gamma).\ \psi \rightarrow \chi) = (\lambda\ \delta.\ \psi \rightarrow (fst\ \delta))$
by $fastforce+$
note $functional\text{-}identities = this$
have $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \delta = \psi \sqcup (fst\ \delta)$
$(\lambda\ (\chi,\gamma).\ \psi \rightarrow \chi)\ \delta = \psi \rightarrow (fst\ \delta)$
by $(simp\ add\colon functional\text{-}identities)+$
}
hence $?T_2\ (\delta\ \#\ \Psi) = ((\psi \rightarrow ?\chi) \rightarrow (\psi \sqcup ?\chi) \rightarrow ?\gamma)\ \#\ (map\ (uncurry\ (\rightarrow))\ ?\Psi_2)$
by $simp$
moreover have $map\ (uncurry\ (\rightarrow))\ (\delta\ \#\ \Psi) = (?\chi \rightarrow ?\gamma)\ \#\ map\ (uncurry\ (\rightarrow))\ \Psi$
by $(simp\ add\colon case\text{-}prod\text{-}beta)$
moreover
{
fix $\chi\ \psi\ \gamma$
have $\vdash ((\psi \rightarrow \chi) \rightarrow (\psi \sqcup \chi) \rightarrow \gamma) \leftrightarrow (\chi \rightarrow \gamma)$
proof −
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\psi\rangle \rightarrow \langle\chi\rangle) \rightarrow (\langle\psi\rangle \sqcup \langle\chi\rangle) \rightarrow \langle\gamma\rangle) \leftrightarrow (\langle\chi\rangle \rightarrow \langle\gamma\rangle))$
by $fastforce$
hence $\vdash (\!|\ (((\langle\psi\rangle \rightarrow \langle\chi\rangle) \rightarrow (\langle\psi\rangle \sqcup \langle\chi\rangle) \rightarrow \langle\gamma\rangle) \leftrightarrow (\langle\chi\rangle \rightarrow \langle\gamma\rangle))\ |\!)$
using $propositional\text{-}semantics$ **by** $blast$
thus $?thesis$ **by** $simp$
qed
}
hence $identity\colon \vdash ((\psi \rightarrow ?\chi) \rightarrow (\psi \sqcup ?\chi) \rightarrow ?\gamma) \rightarrow (?\chi \rightarrow ?\gamma)$
using $biconditional\text{-}def$ **by** $auto$
assume $map\ (uncurry\ (\rightarrow))\ \Psi \preceq map\ (uncurry\ (\rightarrow))\ ?\Psi_2$
with $identity$ **have** $((?\chi \rightarrow ?\gamma)\ \#\ map\ (uncurry\ (\rightarrow))\ \Psi) \preceq$
$(((\psi \rightarrow ?\chi) \rightarrow (\psi \sqcup ?\chi) \rightarrow ?\gamma)\ \#\ (map\ (uncurry\ (\rightarrow))\ ?\Psi_2))$
using $stronger\text{-}theory\text{-}left\text{-}right\text{-}cons$ **by** $blast$
ultimately show $?case$ **by** $simp$

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

moreover
{
fix $\psi\ \chi\ \gamma$
have $\vdash ((\psi \to \chi) \sqcup (\psi \sqcup \chi) \to \gamma) \to (\psi \to (\chi \sqcup \gamma))$
proof –
have $\forall\ \mathfrak{M}.\ \mathfrak{M} \models_{prop} (((\langle\psi\rangle \to \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \to (\langle\psi\rangle \to (\langle\chi\rangle$
$\sqcup \langle\gamma\rangle)))$
by *fastforce*
hence $\vdash (\!|\ (((\langle\psi\rangle \to \langle\chi\rangle) \sqcup (\langle\psi\rangle \sqcup \langle\chi\rangle) \to \langle\gamma\rangle) \to (\langle\psi\rangle \to (\langle\chi\rangle \sqcup \langle\gamma\rangle)))\ |\!)$
using *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
ultimately have $\vdash\ ?\alpha \to\ ?\beta$ **by** *simp*
thus *?case*
using *Cons*
stronger-theory-left-right-cons
by *simp*
qed
moreover have $\forall\ \varphi.\ (map\ (uncurry\ (\sqcup))\ \Psi) :\vdash \varphi \longrightarrow\ ?\Sigma :\vdash \psi \to \varphi$
proof (*induct* Ψ)
case *Nil*
then show *?case*
using *Axiom-1 Modus-Ponens*
by *fastforce*
next
case (*Cons* $\delta\ \Psi$)
let $?\delta' = (\lambda\ (\chi, \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta$
let $?\Sigma = map\ (\lambda\ (\chi, \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \Psi$
let $?\Sigma' = map\ (\lambda\ (\chi, \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ (\delta\ \#\ \Psi)$
{
fix φ
assume $map\ (uncurry\ (\sqcup))\ (\delta\ \#\ \Psi) :\vdash \varphi$
hence $map\ (uncurry\ (\sqcup))\ \Psi :\vdash (uncurry\ (\sqcup))\ \delta \to \varphi$
using *list-deduction-theorem*
by *simp*
hence $?\Sigma :\vdash \psi \to (uncurry\ (\sqcup))\ \delta \to \varphi$
using *Cons*
by *blast*
moreover
{
fix $\alpha\ \beta\ \gamma$
have $\vdash (\alpha \to \beta \to \gamma) \to ((\alpha \to \beta) \to \alpha \to \gamma)$
using *Axiom-2* **by** *auto*
}
ultimately have $?\Sigma :\vdash (\psi \to (uncurry\ (\sqcup))\ \delta) \to \psi \to \varphi$
using *list-deduction-weaken* [**where** $?\Gamma = ?\Sigma$]
list-deduction-modus-ponens [**where** $?\Gamma = ?\Sigma$]
by *metis*

moreover
have $(\lambda\ \delta.\ \psi \to (uncurry\ (\sqcup))\ \delta) = (\lambda\ \delta.\ (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta)$
by *fastforce*
ultimately have $?\Sigma :\vdash (\lambda\ (\chi,\ \gamma).\ (\psi \to (\chi \sqcup \gamma)))\ \delta \to \psi \to \varphi$
by *metis*
hence $?\Sigma' :\vdash \psi \to \varphi$
using *list-deduction-theorem*
by *simp*
}
then show *?case* **by** *simp*
qed
with $\Psi(2)$ **have** $?\Sigma :\vdash \psi \to \varphi$
by *blast*
ultimately show *?thesis*
using *stronger-theory-deduction-monotonic* **by** *auto*
qed
moreover have *mset* (*map snd* $?\Psi_2$) $\subseteq\#$ *mset* $?\Gamma_1$ **by** *simp*
ultimately have $?\Gamma_1 \$\vdash (\psi \to \varphi \# \Phi)$ **using** *segmented-deduction.simps*(*2*) **by** *blast*
moreover have $\vdash$ (*map* (*uncurry* $(\sqcup)$) $\Psi :\to \varphi) \to$ (*map* (*uncurry* $(\sqcup)$) $?\Psi_1$) $:\to (\psi \sqcup \varphi)$
proof (*induct* Ψ)
case *Nil*
then show *?case*
unfolding *disjunction-def*
using *Axiom-1 Modus-Ponens*
by *fastforce*
next
case (*Cons* ν Ψ)
let $?\Delta$ = *map* (*uncurry* $(\sqcup)$) Ψ
let $?\Delta'$ = *map* (*uncurry* $(\sqcup)$) $(\nu \# \Psi)$
let $?\Sigma$ = *map* (*uncurry* $(\sqcup)$) (*zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ \Psi$) (*map snd* Ψ))
let $?\Sigma'$ = *map* (*uncurry* $(\sqcup)$) (*zip* (*map* $(\lambda\ (\chi,\gamma).\ \psi \sqcup \chi)\ (\nu \# \Psi)$) (*map snd* $(\nu \# \Psi)$))
have $\vdash (?\Delta' :\to\ \varphi) \to$ (*uncurry* $(\sqcup)$) $\nu \to ?\Delta :\to \varphi$
by (*simp*, *metis Axiom-1 Axiom-2 Modus-Ponens*)
with *Cons* **have** $\vdash (?\Delta' :\to\ \varphi) \to$ (*uncurry* $(\sqcup)$) $\nu \to ?\Sigma :\to (\psi \sqcup \varphi)$
using *hypothetical-syllogism Modus-Ponens*
by *blast*
hence $(?\Delta' :\to\ \varphi)$ # ((*uncurry* $(\sqcup)$) ν) # $?\Sigma :\vdash \psi \sqcup \varphi$
by (*simp add*: *list-deduction-def*)
moreover have *set* $((?\Delta' :\to\ \varphi)$ # ((*uncurry* $(\sqcup)$) ν) # $?\Sigma)$ =
set (((*uncurry* $(\sqcup)$) ν) # $(?\Delta' :\to\ \varphi)$ # $?\Sigma$)
by *fastforce*
ultimately have ((*uncurry* $(\sqcup)$) ν) # $(?\Delta' :\to\ \varphi)$ # $?\Sigma :\vdash \psi \sqcup \varphi$
using *list-deduction-monotonic* **by** *blast*
hence $(?\Delta' :\to\ \varphi)$ # $?\Sigma :\vdash$ ((*uncurry* $(\sqcup)$) $\nu) \to (\psi \sqcup \varphi)$
using *list-deduction-theorem*
by *simp*

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

```
      by simp
    ultimately have (?Δ' :→  φ) # ?Σ' :⊢ ψ ⊔ φ by simp
    then show ?case by (simp add: list-deduction-def)
  qed
  with Ψ(2) have map (uncurry (⊔)) ?Ψ₁ :⊢ (ψ ⊔ φ)
    unfolding list-deduction-def
    using Modus-Ponens
    by blast
  moreover have mset (map snd ?Ψ₁) ⊆# mset Γ using Ψ(1) by simp
  ultimately show Γ $⊢ (ψ ⊔ φ # ψ → φ # Φ)
    using segmented-deduction.simps(2) by blast
next
  assume Γ $⊢ (ψ ⊔ φ # ψ → φ # Φ)
  from this obtain Ψ where Ψ:
    mset (map snd Ψ) ⊆# mset Γ
    map (uncurry (⊔)) Ψ :⊢ ψ ⊔ φ
    map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ) $⊢ (ψ → φ # Φ)
    using segmented-deduction.simps(2) by blast
  let ?Γ' = map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)
  from Ψ obtain Δ where Δ:
    mset (map snd Δ) ⊆# mset ?Γ'
    map (uncurry (⊔)) Δ :⊢ ψ → φ
    (map (uncurry (→)) Δ @ ?Γ' ⊖ (map snd Δ)) $⊢ Φ
    using segmented-deduction.simps(2) by blast
  let ?Ω = 𝔍 Ψ Δ
  have mset (map snd ?Ω) ⊆# mset Γ
    using Δ(1) Ψ(1) mergeWitness-msub-intro
    by blast
  moreover have map (uncurry (⊔)) ?Ω :⊢ φ
  proof -
    have map (uncurry (⊔)) ?Ω :⊢ ψ ⊔ φ
         map (uncurry (⊔)) ?Ω :⊢ ψ → φ
      using Ψ(2) Δ(2)
            stronger-theory-deduction-monotonic
            right-mergeWitness-stronger-theory
            left-mergeWitness-stronger-theory
      by blast+
    moreover
    have ⊢ (ψ ⊔ φ) → (ψ → φ) → φ
      unfolding disjunction-def
      using Modus-Ponens excluded-middle-elimination flip-implication
      by blast
    ultimately show ?thesis
      using list-deduction-weaken list-deduction-modus-ponens
      by blast
  qed
  moreover have map (uncurry (→)) ?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 :: ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* $\Rightarrow$ ($'a \times 'a$) *list* ($\mathfrak{X}$)
where
$\mathfrak{X}\ \Psi\ [] = []$
$|\ \mathfrak{X}\ \Psi\ (\delta\ \#\ \Delta) =$
(*case find* ($\lambda\ \psi$. (*uncurry* ($\rightarrow$)) ψ = *snd* δ) Ψ *of*
None $\Rightarrow \delta\ \#\ \mathfrak{X}\ \Psi\ \Delta$
| *Some* $\psi \Rightarrow$ (*fst* $\psi \rightarrow$ *fst* δ, *snd* ψ) # ($\mathfrak{X}$ (*remove1* ψ Ψ) Δ))

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

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

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

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

lemma (**in** *Minimal-Logic*) *YWitness-map-snd-decomposition*:
  mset (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* (($\Psi \ominus$ ($\mathfrak{A}$ Ψ Δ)) @ ($\mathfrak{Y}_{\bullet}$ Ψ Δ)))
proof −
 have $\forall$ Ψ. *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ)) = *mset* (*map snd* (($\Psi \ominus$ ($\mathfrak{A}$ Ψ Δ)) @ ($\mathfrak{Y}_{\bullet}$ Ψ Δ)))
 proof (*induct* Δ)
  case *Nil*
  then show *?case* **by** *simp*
 next
  case (*Cons* δ Δ)
  {
   fix Ψ
   have *mset* (*map snd* ($\mathfrak{Y}$ Ψ (δ # Δ))) = *mset* (*map snd* ($\Psi \ominus \mathfrak{A}$ Ψ (δ # Δ) @ $\mathfrak{Y}_{\bullet}$ Ψ (δ # Δ)))
    using *Cons*
    by (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) 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) $\subseteq\#$ *mset* Γ
and *mset* (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
shows *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
proof –
have *mset* (*map snd* ($\Delta \ominus$ ($\mathfrak{B}$ Ψ Δ))) $\subseteq\#$ *mset* ($\Gamma \ominus$ (*map snd* Ψ))
using *assms secondComponent-diff-msub* **by** *blast*
moreover have *mset* (*map snd* ($\mathfrak{A}$ Ψ Δ)) $\subseteq\#$ *mset* (*map snd* Ψ)
using *firstComponent-msub*
by (*simp add*: *image-mset-subseteq-mono*)
moreover have *mset* ((*map snd* Ψ) @ ($\Gamma \ominus$ *map snd* Ψ)) = *mset* Γ
using *assms*(*1*)
by *simp*
moreover have *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* ($\Delta \ominus \mathfrak{B}$ Ψ Δ))
= *mset* (*map snd* ($\mathfrak{X}$ Ψ Δ))
using *XWitness-map-snd-decomposition* **by** *force*
ultimately
show *?thesis*
by (*metis* (*no-types*) *mset-append mset-map subset-mset.add-mono*)
qed

lemma (**in** *Minimal-Logic*) *YComponent-msub*:
mset (*map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}$ Ψ Δ))
proof –
have $\forall$ Ψ. *mset* (*map snd* ($\mathfrak{Y}_\bullet$ Ψ Δ)) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}$ Ψ Δ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* (*map snd* ($\mathfrak{Y}_\bullet$ Ψ (δ # Δ))) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{X}$ Ψ (δ # Δ)))
using *Cons*
by (*cases find* (λ ψ. (*uncurry* ($\rightarrow$)) 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) $\subseteq\#$ *mset* Γ
and *mset* (*map snd* Δ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\rightarrow)$) Ψ @ $\Gamma \ominus$ (*map snd* Ψ))
shows *mset* (*map snd* ($\mathfrak{Y}$ Ψ Δ)) $\subseteq\#$
mset (*map* (*uncurry* $(\rightarrow)$) ($\mathfrak{X}$ Ψ Δ) @ $\Gamma \ominus$ *map snd* ($\mathfrak{X}$ Ψ Δ))
proof −
have *A*: *image-mset snd* (*mset* Ψ) $\subseteq\#$ *mset* Γ **using** *assms* **by** *simp*
have *B*: *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)) $\subseteq\#$ *mset* Γ
using *A XWitness-map-snd-decomposition assms(2) XWitness-msub* **by** *auto*
have *mset* Γ − *image-mset snd* (*mset* Ψ) = *mset* ($\Gamma \ominus$ *map snd* Ψ)
by *simp*
then have *C*: *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ Ψ Δ)) + *image-mset snd* (*mset* Ψ) $\subseteq\#$ *mset* Γ
using *A* **by** (*metis* (*full-types*) *assms(2) secondComponent-diff-msub subset-mset.le-diff-conv2*)
have *image-mset snd* (*mset* ($\Psi \ominus \mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) = *image-mset snd* (*mset* Ψ)
by (*metis* (*no-types*) *image-mset-union*
listSubtract-mset-homomorphism
firstComponent-msub
subset-mset.diff-add)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}$ Ψ Δ))
+ (*image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)))
= *mset* (*map snd* ($\Delta \ominus \mathfrak{B}$ Ψ Δ)) + *image-mset snd* (*mset* Ψ)
by (*simp add*: *union-commute*)
then have *image-mset snd* (*mset* Ψ − *mset* ($\mathfrak{A}$ Ψ Δ))
$\subseteq\#$ *mset* Γ − (*image-mset snd* (*mset* ($\mathfrak{A}$ Ψ Δ)) + *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}$ Ψ Δ)))
by (*metis* (*no-types*) *B C subset-mset.le-diff-conv2*)
hence *mset* (*map snd* ($\Psi \ominus \mathfrak{A}$ Ψ Δ)) $\subseteq\#$ *mset* ($\Gamma \ominus$ *map snd* ($\mathfrak{X}$ Ψ Δ))
using *assms XWitness-map-snd-decomposition*
by *simp*
thus *?thesis*
using *YComponent-msub*
YWitness-map-snd-decomposition
by (*simp add*: *mset-subset-eq-mono-add union-commute*)
qed

lemma (**in** *Classical-Propositional-Logic*) *XWitness-right-stronger-theory*:
map (*uncurry* $(\sqcup)$) $\Delta \preceq$ *map* (*uncurry* $(\sqcup)$) ($\mathfrak{X}$ Ψ Δ)
proof −
have $\forall$ Ψ. *map* (*uncurry* $(\sqcup)$) $\Delta \preceq$ *map* (*uncurry* $(\sqcup)$) ($\mathfrak{X}$ 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 (⊔)) (𝔛 Ψ (δ # Δ))
    proof (cases find (λ ψ. (uncurry (→)) 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 (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
           ψ ∈ set Ψ
           (fst ψ → 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 (λψ. uncurry (→) ψ = snd δ) Ψ = Some ψ
             ψ ∈ set Ψ
             (uncurry (⊔)) ψ = fst ψ ⊔ snd ψ
          using find-Some-set-membership
          by fastforce
        let ?φ = fst ψ ⊔ (fst ψ → fst δ) → snd ψ
        let ?Ψ' = remove1 ψ Ψ
        have map (uncurry (⊔)) ?Ψ' ≼ map (uncurry (⊔)) (𝔜 ?Ψ' Δ)
          using Cons by simp
        moreover
        {
          fix α β γ
          have ⊢ (α ⊔ (α → γ) → β) → (α ⊔ β)
          proof −
            let ?φ = (⟨α⟩ ⊔ (⟨α⟩ → ⟨γ⟩) → ⟨β⟩) → (⟨α⟩ ⊔ ⟨β⟩)
            have ∀𝔐. 𝔐 ⊨prop ?φ by fastforce
            hence ⊢ ⦇ ?φ ⦈ using propositional-semantics by blast
            thus ?thesis by simp
          qed
        }
        hence ⊢ ?φ → (uncurry (⊔)) ψ using ψ(3) by auto
        ultimately
         have map (uncurry (⊔)) (ψ # ?Ψ') ≼ (?φ # map (uncurry (⊔)) (𝔜 ?Ψ'
Δ))
          by (simp add: stronger-theory-left-right-cons)
        moreover
       from ψ have mset (map (uncurry (⊔)) (ψ # ?Ψ')) = mset (map (uncurry
(⊔)) Ψ)
          by (metis mset-eq-perm mset-map perm-remove)
```

ultimately show *?thesis*
using *stronger-theory-relation-alt-def* $\psi(1)$ **by** *auto*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **by** *blast*
qed

lemma (**in** *Minimal-Logic*) *XWitness-secondComponent-diff-decomposition*:
mset $(\mathfrak{X}\ \Psi\ \Delta) =$ *mset* $(\mathfrak{X}_\bullet\ \Psi\ \Delta\ @\ \Delta \ominus \mathfrak{B}\ \Psi\ \Delta)$
proof −
have $\forall\ \Psi.$ *mset* $(\mathfrak{X}\ \Psi\ \Delta) =$ *mset* $(\mathfrak{X}_\bullet\ \Psi\ \Delta\ @\ \Delta \ominus \mathfrak{B}\ \Psi\ \Delta)$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* $(\mathfrak{X}\ \Psi\ (\delta\ \#\ \Delta)) =$
mset $(\mathfrak{X}_\bullet\ \Psi\ (\delta\ \#\ \Delta)\ @\ (\delta\ \#\ \Delta) \ominus \mathfrak{B}\ \Psi\ (\delta\ \#\ \Delta))$
using *Cons*
by (*cases find* $(\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None$,
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 $(\mathfrak{Y}\ \Psi\ \Delta) =$ *mset* $(\Psi \ominus \mathfrak{A}\ \Psi\ \Delta\ @\ \mathfrak{Y}_\bullet\ \Psi\ \Delta)$
proof −
have $\forall\ \Psi.$ *mset* $(\mathfrak{Y}\ \Psi\ \Delta) =$ *mset* $(\Psi \ominus \mathfrak{A}\ \Psi\ \Delta\ @\ \mathfrak{Y}_\bullet\ \Psi\ \Delta)$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Ψ
have *mset* $(\mathfrak{Y}\ \Psi\ (\delta\ \#\ \Delta)) =$
mset $(\Psi \ominus \mathfrak{A}\ \Psi\ (\delta\ \#\ \Delta)\ @\ \mathfrak{Y}_\bullet\ \Psi\ (\delta\ \#\ \Delta))$
using *Cons*
by (*cases find* $(\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None$, *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 (→)) (δ # Δ)
          ≼ map (uncurry (→)) (𝔜 Ψ (δ # Δ) ⊖ ?𝔣 Ψ (δ # Δ) @ ?𝔤 Ψ (δ # Δ))
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        moreover
        from Cons have
          map (uncurry (→)) (δ # Δ) ≼ map (uncurry (→)) (δ # 𝔜 Ψ Δ ⊖ ?𝔣 Ψ
Δ @ ?𝔤 Ψ Δ)
          by (simp add: stronger-theory-left-right-cons trivial-implication)
        moreover
        have mset (map (uncurry (→)) (δ # 𝔜 Ψ Δ ⊖ ?𝔣 Ψ Δ @ ?𝔤 Ψ Δ))
           = mset (map (uncurry (→)) (𝔜 Ψ Δ ⊖ ?𝔣 Ψ Δ @ ((δ # Δ) ⊖ 𝔅 Ψ Δ)))
          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
          ∀Ψ Φ. Ψ ≼ Φ
              = (∃Σ. map snd Σ = Ψ
                    ∧ mset (map fst Σ) ⊆# mset Φ
                    ∧ (∀ξ. ξ ∉ set Σ ∨ ⊢ (uncurry (→) ξ)))
            by (simp add: Ball-def-raw stronger-theory-relation-def)
        moreover have
          ((uncurry (→) δ) # map (uncurry (→)) Δ)
           ≼ ((uncurry (→) δ) # map (uncurry (→)) (𝔜 Ψ Δ ⊖ (?𝔣 Ψ Δ))
              @ map (uncurry (→)) (?𝔤 Ψ Δ))
```

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* ($\rightarrow$) ψ = *snd* δ) Ψ = *Some* ψ
uncurry ($\rightarrow$) ψ = *snd* δ
using *find-Some-predicate*
by *fastforce*
let $?\alpha$ = *fst* ψ
let $?\beta$ = *fst* δ
let $?\gamma$ = *snd* ψ
have (λ δ. *fst* $\delta \rightarrow$ *snd* δ) = *uncurry* ($\rightarrow$) **by** *fastforce*
hence $?\beta \rightarrow ?\alpha \rightarrow ?\gamma$ = *uncurry* ($\rightarrow$) δ **using** $\psi(2)$ **by** *metis*
moreover
let $?A = \mathfrak{Y}$ (*remove1* ψ Ψ) Δ
let $?B = \mathfrak{A}$ (*remove1* ψ Ψ) Δ
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* ($\rightarrow$))
((($?\alpha$, ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$) # $?A$) $\ominus$ *remove1* ψ ($\Psi \ominus ?B$)
@ (*remove1* δ ((δ # Δ) $\ominus$ $?C$))))
= *mset* (($?\alpha \rightarrow$ ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$) # *map* (*uncurry* ($\rightarrow$)) ($?D$ @ ($\Delta \ominus$
$?C$)))
by (*simp*, *metis* (*no-types*, *hide-lams*)
add-mset-add-single
image-mset-add-mset
prod.simps(*2*)
subset-mset.diff-add-assoc2)
moreover
have $\vdash$ ($?\alpha \rightarrow$ ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$) $\rightarrow ?\beta \rightarrow ?\alpha \rightarrow ?\gamma$
proof –
let $?\Gamma$ = [($?\alpha \rightarrow$ ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$), $?\beta$, $?\alpha$]
have $?\Gamma$:$\vdash$ $?\alpha \rightarrow$ ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$
$?\Gamma$:$\vdash$ $?\alpha$
by (*simp add*: *list-deduction-reflection*)+
hence $?\Gamma$:$\vdash$ ($?\alpha \rightarrow ?\beta$) $\rightarrow ?\gamma$
using *list-deduction-modus-ponens* **by** *blast*
moreover have $?\Gamma$:$\vdash$ $?\beta$
by (*simp add*: *list-deduction-reflection*)
hence $?\Gamma$:$\vdash$ $?\alpha \rightarrow ?\beta$
using *Axiom-1 list-deduction-modus-ponens list-deduction-weaken* **by**
blast
ultimately have $?\Gamma$:$\vdash$ $?\gamma$
using *list-deduction-modus-ponens* **by** *blast*
thus *?thesis*

unfolding *list-deduction-def* **by** *simp*
qed
hence $(?\beta \to ?\alpha \to ?\gamma$ # *map* (*uncurry* $(\to)$) $\Delta) \preceq$
$(?\alpha \to (?\alpha \to ?\beta) \to ?\gamma$ # *map* (*uncurry* $(\to)$) $(?D$ @ $(\Delta \ominus ?C)))$
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* $(\to)$) $(\mathfrak{X}_\bullet\ \Psi\ \Delta)$ = *map snd* $(\mathfrak{Y}_\bullet\ \Psi\ \Delta)$
proof –
have $\forall\ \Psi$. *map* (*uncurry* $(\to)$) $(\mathfrak{X}_\bullet\ \Psi\ \Delta)$ = *map snd* $(\mathfrak{Y}_\bullet\ \Psi\ \Delta)$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\delta\ \Delta$)
{
fix Ψ
have *map* (*uncurry* $(\to)$) $(\mathfrak{X}_\bullet\ \Psi\ (\delta$ # $\Delta))$ = *map snd* $(\mathfrak{Y}_\bullet\ \Psi\ (\delta$ # $\Delta))$
using *Cons*
by (*cases find* $(\lambda\ \psi.$ (*uncurry* $(\to)$) ψ = *snd* $\delta)\ \Psi$ = *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) $\subseteq$# *mset* Γ
and *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 \Phi$
(**is** $?\Gamma_0$ \$$\vdash \Phi$)
shows *map* (*uncurry* $(\to)$) $(\mathfrak{Y}\ \Psi\ \Delta)$ @
(*map* (*uncurry* $(\to)$) $(\mathfrak{X}\ \Psi\ \Delta)$ @ $\Gamma \ominus$ *map snd* $(\mathfrak{X}\ \Psi\ \Delta)) \ominus$
map snd $(\mathfrak{Y}\ \Psi\ \Delta)$ \$$\vdash \Phi$
(**is** $?\Gamma$ \$$\vdash \Phi$)
proof –
let $?A$ = *map* (*uncurry* $(\to)$) $(\mathfrak{Y}\ \Psi\ \Delta)$
let $?B$ = *map* (*uncurry* $(\to)$) $(\mathfrak{X}\ \Psi\ \Delta)$
let $?C = \Psi \ominus \mathfrak{A}\ \Psi\ \Delta$

let *?D* = *map* (*uncurry* (→)) *?C*
let *?E* = $\Delta \ominus \mathfrak{B}\ \Psi\ \Delta$
let *?F* = *map* (*uncurry* (→)) *?E*
let *?G* = *map snd* ($\mathfrak{B}\ \Psi\ \Delta$)
let *?H* = *map* (*uncurry* (→)) ($\mathfrak{X}_\bullet\ \Psi\ \Delta$)
let *?I* = $\mathfrak{A}\ \Psi\ \Delta$
let *?J* = *map snd* ($\mathfrak{X}\ \Psi\ \Delta$)
let *?K* = *map snd* ($\mathfrak{Y}\ \Psi\ \Delta$)
have *mset* (*map* (*uncurry* (→)) ($\mathfrak{Y}\ \Psi\ \Delta \ominus$ *?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 $?\Gamma_0$ ≼ ((*?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* ⊆# *mset* (*map* (*uncurry* (→)) Ψ)
    mset ($\mathfrak{B}\ \Psi\ \Delta$) ⊆# *mset* Δ
    mset (*map snd ?E*) ⊆# *mset* (Γ ⊖ *map snd* Ψ)
    mset (*map* (*uncurry* (→)) Ψ ⊖ *?G*) = *mset ?D*
    mset ?D ⊆# *mset ?A*
    mset (*map snd ?I*) ⊆# *mset* (*map snd* Ψ)
    mset (*map snd ?I*) ⊆# *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* ($\mathfrak{B}\ \Psi\ \Delta$) + *mset* ($\mathfrak{B}\ \Psi\ \Delta$) = *mset* Δ
  by *simp*
hence ♡: {#*x* → *y*. (*x*, *y*) ∈# *mset* Ψ#} + (*mset* Γ − *image-mset snd* (*mset*
Ψ))
                − *image-mset snd* (*mset* Δ)
      = {#*x* → *y*. (*x*, *y*) ∈# *mset* Ψ#} + (*mset* Γ − *image-mset snd* (*mset*
Ψ))
                − *image-mset snd* (*mset* Δ − *mset* ($\mathfrak{B}\ \Psi\ \Delta$))
                − *image-mset snd* (*mset* ($\mathfrak{B}\ \Psi\ \Delta$))
      image-mset snd (*mset* Ψ − *mset* ($\mathfrak{A}\ \Psi\ \Delta$)) + *image-mset snd* (*mset* ($\mathfrak{A}$

Ψ Δ))
$= \mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ \Psi)$
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 $\mathit{mset}\ \Gamma - \mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ \Psi)$
$- \mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ \Delta - \mathit{mset}\ (\mathfrak{B}\ \Psi\ \Delta))$
$= \mathit{mset}\ \Gamma - (\mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ \Psi - \mathit{mset}\ (\mathfrak{A}\ \Psi\ \Delta))$
$+ \mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ (\mathfrak{X}\ \Psi\ \Delta)))$
using ♠ **by** (*simp*, *metis* (*full-types*) *diff-diff-add-mset*)
hence $\mathit{mset}\ ((\mathit{map}\ (\mathit{uncurry}\ (\rightarrow))\ \Psi\ @\ \Gamma \ominus \mathit{map}\ \mathit{snd}\ \Psi) \ominus \mathit{map}\ \mathit{snd}\ \Delta)$
$= \mathit{mset}\ (?D\ @\ (\Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
using ♡ ♠ **by** (*simp*, *metis* (*no-types*) *add.commute subset-mset.add-diff-assoc*)
ultimately have $?\Gamma_0 \preceq ((?A \ominus ?D\ @\ ?F)\ @\ ?D\ @\ (\Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
unfolding *stronger-theory-relation-alt-def*
by *simp*
moreover
have $\mathit{mset}\ ?F = \mathit{mset}\ (?B \ominus ?H)$
$\mathit{mset}\ ?D \subseteq\#\ \mathit{mset}\ ?A$
$\mathit{mset}\ (\mathit{map}\ \mathit{snd}\ (\Psi \ominus ?I)) \subseteq\#\ \mathit{mset}\ (\Gamma \ominus ?J)$
by (*simp add*: *XWitness-secondComponent-diff-decomposition*,
simp add: *YWitness-firstComponent-diff-decomposition*,
simp, *metis* (*no-types*, *lifting*)
♡(*2*) ♠(*8*) *add.assoc assms*(*1*) *assms*(*2*) *image-mset-union*
XWitness-msub mergeWitness-msub-intro
secondComponent-mergeWitness-snd-projection
mset-map
subset-mset.le-diff-conv2
union-code)
hence $\mathit{mset}\ ((?A \ominus ?D\ @\ ?F)\ @\ ?D\ @\ (\Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
$= \mathit{mset}\ (?A\ @\ (?B \ominus ?H\ @\ \Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
$\mathit{mset}\ ?H \subseteq\#\ \mathit{mset}\ ?B$
$\{\#x \rightarrow y.\ (x,\ y) \in\#\ \mathit{mset}\ (\mathfrak{X}_\bullet\ \Psi\ \Delta)\#\} = \mathit{mset}\ (\mathit{map}\ \mathit{snd}\ (\mathfrak{Y}_\bullet\ \Psi\ \Delta))$
by (*simp add*: *subset-mset.diff-add-assoc*,
simp add: *XWitness-secondComponent-diff-decomposition*,
metis XComponent-YComponent-connection mset-map uncurry-def)
hence $\mathit{mset}\ ((?A \ominus ?D\ @\ ?F)\ @\ ?D\ @\ (\Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
$= \mathit{mset}\ (?A\ @\ (?B\ @\ \Gamma \ominus ?J) \ominus (?H\ @\ \mathit{map}\ \mathit{snd}\ ?C))$
$\{\#x \rightarrow y.\ (x,\ y) \in\#\ \mathit{mset}\ (\mathfrak{X}_\bullet\ \Psi\ \Delta)\#\} + \mathit{image\text{-}mset}\ \mathit{snd}\ (\mathit{mset}\ \Psi - \mathit{mset}$
$(\mathfrak{A}\ \Psi\ \Delta))$
$= \mathit{mset}\ (\mathit{map}\ \mathit{snd}\ (\mathfrak{Y}\ \Psi\ \Delta))$
using *YWitness-map-snd-decomposition*
by (*simp add*: *subset-mset.diff-add-assoc*, *force*)
hence $\mathit{mset}\ ((?A \ominus ?D\ @\ ?F)\ @\ ?D\ @\ (\Gamma \ominus ?J) \ominus \mathit{map}\ \mathit{snd}\ ?C)$
$= \mathit{mset}\ (?A\ @\ (?B\ @\ \Gamma \ominus ?J) \ominus ?K)$
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 $\Gamma \ \$\vdash (\varphi \ \# \ \psi \ \# \ \Phi)$
shows $\Gamma \ \$\vdash (\psi \ \# \ \varphi \ \# \ \Phi)$
proof −
from *assms* **obtain** Ψ **where** Ψ:
mset (*map snd* Ψ) $\subseteq\#$ *mset* Γ
map (*uncurry* ($\sqcup$)) Ψ :$\vdash \varphi$
map (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ) $\$\vdash (\psi \ \# \ \Phi)$
by *fastforce*
let $?\Gamma_0$ = *map* (*uncurry* ($\to$)) Ψ @ $\Gamma \ominus$ (*map snd* Ψ)
from Ψ(*3*) **obtain** Δ **where** Δ:
mset (*map snd* Δ) $\subseteq\#$ *mset* $?\Gamma_0$
map (*uncurry* ($\sqcup$)) Δ :$\vdash \psi$
(*map* (*uncurry* ($\to$)) Δ @ $?\Gamma_0 \ominus$ (*map snd* Δ)) $\$\vdash \Phi$
using *segmented-deduction.simps*(*2*) **by** *blast*
let $?\Psi' = \mathfrak{X} \ \Psi \ \Delta$
let $?\Gamma_1$ = *map* (*uncurry* ($\to$)) $?\Psi'$ @ $\Gamma \ominus$ (*map snd* $?\Psi'$)
let $?\Delta' = \mathfrak{Y} \ \Psi \ \Delta$
have (*map* (*uncurry* ($\to$)) $?\Delta'$ @ $?\Gamma_1 \ominus$ (*map snd* $?\Delta'$)) $\$\vdash \Phi$
map (*uncurry* ($\sqcup$)) $\Psi \preceq$ *map* (*uncurry* ($\sqcup$)) $?\Delta'$
using Ψ(*1*) Δ(*1*) Δ(*3*)
XWitness-YWitness-segmented-deduction-intro
YWitness-left-stronger-theory
by *auto*
hence $?\Gamma_1 \ \$\vdash (\varphi \ \# \ \Phi)$
using Ψ(*1*) Ψ(*2*) Δ(*1*)
YWitness-msub segmented-deduction.simps(*2*)
stronger-theory-deduction-monotonic
by *blast*
thus *?thesis*
using Ψ(*1*) Δ(*1*) Δ(*2*)
XWitness-msub
XWitness-right-stronger-theory
segmented-deduction.simps(*2*)
stronger-theory-deduction-monotonic
by *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *segmented-cons-remove1*:
assumes $\varphi \in$ *set* Φ
shows $\Gamma \ \$\vdash \Phi = \Gamma \ \$\vdash (\varphi \ \#$ (*remove1* $\varphi \ \Phi$))
proof −
from ‹$\varphi \in$ *set* Φ›
have $\forall \ \Gamma. \ \Gamma \ \$\vdash \Phi = \Gamma \ \$\vdash (\varphi \ \#$ (*remove1* $\varphi \ \Phi$))

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

by *auto*
moreover from ‹Γ $\vdash (\varphi \# \Phi)$› **have** Γ $\vdash \Phi$
using *segmented-deduction.simps(2)*
segmented-stronger-theory-left-monotonic
witness-stronger-theory
by *blast*
ultimately show *?thesis* **using** *Cons* **by** *blast*
qed
}
then show *?case* **by** *blast*
qed
thus *?thesis* **using** *assms* **by** *blast*
qed

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

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

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

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

$\sqcup$ *snd* σ # *map* ($\lambda(x, y)$. $x \sqcup y$) ($\mathfrak{E}$ (*remove1* σ Σ) Δ)) $\succeq$ *map* ($\lambda(x,$ $y)$. $x \sqcup y$) Σ
        by (*metis* (*no-types*, *hide-lams*) *A B stronger-theory-transitive uncurry-def*)
        thus *?thesis* **using** *A B* σ **by** *simp*
      qed
    }
    then show *?case* **by** *auto*
  qed
  thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *submergeWitness-msub*:
  mset (*map snd* ($\mathfrak{E}$ Σ Δ)) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ))
proof −
  have $\forall$ Σ. *mset* (*map snd* ($\mathfrak{E}$ Σ Δ)) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ))
  proof (*induct* Δ)
    case *Nil*
    {
      fix Σ
      have *mset* (*map snd* ($\mathfrak{E}$ Σ [])) $\subseteq$#
            mset (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ []))
        by (*induct* Σ, *simp+*)
    }
    then show *?case* **by** *blast*
  next
    case (*Cons* δ Δ)
    {
      fix Σ
      have *mset* (*map snd* ($\mathfrak{E}$ Σ (δ # Δ))) $\subseteq$#
            mset (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ (δ # Δ)))
        using *Cons*
        by (*cases find* (λ σ. (*uncurry* ($\to$)) 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* ($\sqcup$)) Δ
 $\preceq$ (*map* (*uncurry* ($\to$)) ($\mathfrak{E}$ Σ Δ) @ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{J}$ Σ Δ) $\ominus$ *map snd* ($\mathfrak{E}$ Σ Δ))
proof −

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

$-$ (*add-mset* (*case* δ *of* $(x, xa) \Rightarrow x \sqcup xa$) $\{\#x \sqcup y.\ (x, y) \in\#$ *mset* $(\mathfrak{J}\ \Sigma\ \Delta)\#\}$
$-$ *image-mset snd* (*mset* $(\mathfrak{C}\ \Sigma\ \Delta)$))
$\subseteq\#$ $\{\#x \to y.\ (x, y) \in\#$ *mset* $(\mathfrak{C}\ \Sigma\ \Delta)\#\}$
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))
$\subseteq\#$ $\{\#x \to y.\ (x, y) \in\#$ *mset* $(\mathfrak{C}\ \Sigma\ \Delta)\#\}$
$+$ (*add-mset* (*case* δ *of* $(x, xa) \Rightarrow x \sqcup xa$) $\{\#x \sqcup y.\ (x, y) \in\#$ *mset* $(\mathfrak{J}\ \Sigma\ \Delta)\#\}$
$-$ *image-mset snd* (*mset* $(\mathfrak{C}\ \Sigma\ \Delta)$))
using *subset-eq-diff-conv* **by** *blast*
hence
mset (*map fst* $?\Phi'$) $\subseteq\#$
mset (*map* (*uncurry* $(\to)$) $(\mathfrak{C}\ \Sigma\ (\delta\ \#\ \Delta))$
@ *map* (*uncurry* $(\sqcup)$) $(\mathfrak{J}\ \Sigma\ (\delta\ \#\ \Delta))$
$\ominus$ *map snd* $(\mathfrak{C}\ \Sigma\ (\delta\ \#\ \Delta))$)
using *True* $\Phi(2)$
by *simp*
moreover have $\forall (\gamma, \sigma) \in$ *set* $?\Phi'.\ \vdash \gamma \to \sigma$
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* $(\to)$ σ = *snd* δ) Σ = *Some* σ
uncurry $(\to)$ σ = *snd* δ
using *find-Some-predicate*
by *fastforce*
moreover from *Cons* **have**
map (*uncurry* $(\sqcup)$) $\Delta \preceq$
(*map* (*uncurry* $(\to)$) ($\mathfrak{C}$ (*remove1* σ Σ) Δ) @
remove1 ((*fst* $\delta \sqcap$ *fst* σ) $\sqcup$ *snd* σ)
(((*fst* $\delta \sqcap$ *fst* σ) $\sqcup$ *snd* σ # *map* (*uncurry* $(\sqcup)$) ($\mathfrak{J}$ (*remove1* σ Σ) Δ))
$\ominus$ *map snd* ($\mathfrak{C}$ (*remove1* σ Σ) Δ)))
unfolding *stronger-theory-relation-alt-def*
by *simp*
moreover
{
fix α β γ
have $\vdash (\alpha \to ((\gamma \sqcap \alpha) \sqcup \beta)) \to (\gamma \sqcup (\alpha \to \beta))$
proof $-$
let $?\varphi = (\langle\alpha\rangle \to ((\langle\gamma\rangle \sqcap \langle\alpha\rangle) \sqcup \langle\beta\rangle)) \to (\langle\gamma\rangle \sqcup (\langle\alpha\rangle \to \langle\beta\rangle))$
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
$\mathfrak{P}\ \Sigma\ [] = \Sigma$
$|\ \mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta) =$

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

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

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

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

lemma (**in** *Classical-Propositional-Logic*) *recoverWitnessA-mset-equiv*:
assumes *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
shows *mset* (*map snd* (𝔓 Σ Δ @ $\mathfrak{P}^C$ Σ Δ)) = *mset* (*map snd* Δ)
proof −
have ∀ Σ. *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
⟶ *mset* (*map snd* (𝔓 Σ Δ @ $\mathfrak{P}^C$ Σ Δ)) = *mset* (*map snd* Δ)
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons δ* Δ)
{
fix Σ :: (*'a* × *'a*) *list*
assume ⋆: *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) (*δ* # Δ))
have *mset* (*map snd* (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* ($\sqcup$) δ $\notin$ *set* (*map snd* Σ)
      proof (*induct* Σ)
        case *Nil*
        then show *?case* **by** *simp*
      next
        case (*Cons* σ Σ)
        then show *?case*
          by (*cases* (*uncurry* ($\sqcup$)) δ = *snd* σ, *fastforce*+)
      qed
       moreover have *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ) +
{#*uncurry* ($\sqcup$) δ#}
        using $\star$ **by** *fastforce*
      ultimately have *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
        by (*metis diff-single-trivial*
                in-multiset-in-set
                subset-eq-diff-conv)
      then show *?thesis* **using** *Cons True* **by** *simp*
    next
      case *False*
      from *this* **obtain** σ **where**
        σ: *find* ($\lambda\sigma$. *snd* σ = *uncurry* ($\sqcup$) δ) Σ = *Some* σ
          snd σ = *uncurry* ($\sqcup$) δ
          $\sigma \in$ *set* Σ
        using *find-Some-predicate*
              find-Some-set-membership
        by *fastforce*
      have *A*: *mset* (*map snd* Σ)
          $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ) + *add-mset* (*uncurry* ($\sqcup$) δ) (*mset* [])
        using $\star$ **by** *auto*
      have (*fst* σ, *uncurry* ($\sqcup$) δ) $\in$# *mset* Σ
        by (*metis* (*no-types*) σ(*2*) σ(*3*) *prod.collapse set-mset-mset*)
      then have *B*: *mset* (*map snd* (*remove1* (*fst* σ, *uncurry* ($\sqcup$) δ) Σ))
                = *mset* (*map snd* Σ) − {#*uncurry* ($\sqcup$) δ#}
        by (*meson remove1-pairs-list-projections-snd*)
      have (*fst* σ, *uncurry* ($\sqcup$) δ) = σ
        by (*metis* σ(*2*) *prod.collapse*)
      then have *mset* (*map snd* Σ) − *add-mset* (*uncurry* ($\sqcup$) δ) (*mset* [])
              = *mset* (*map snd* (*remove1* σ Σ))
        using *B* **by** *simp*
      hence *mset* (*map snd* (*remove1* σ Σ)) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
        using *A* **by** (*metis* (*no-types*) *subset-eq-diff-conv*)
      with σ(*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* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
shows (*map* (*uncurry* (→)) Σ @ *map* (*uncurry* (⊔)) Δ ⊖ *map snd* Σ)
⪯ *map* (*uncurry* (⊔)) (𝔔 Σ Δ)
proof −
have ∀ Σ. *mset* (*map snd* Σ) ⊆# *mset* (*map* (*uncurry* (⊔)) Δ)
⟶ (*map* (*uncurry* (→)) Σ @ *map* (*uncurry* (⊔)) Δ ⊖ *map snd* Σ)
⪯ *map* (*uncurry* (⊔)) (𝔔 Σ Δ)
proof(*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Σ :: (′*a* × ′*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 $(uncurry\ (\sqcup)\ \delta\ \#\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ map\ (uncurry\ (\sqcup))\ \Delta \ominus map\ snd\ \Sigma)$
$\preceq map\ (uncurry\ (\sqcup))\ (\mathfrak{Q}\ \Sigma\ (\delta\ \#\ \Delta))$
using *True*
by (*simp add*: *stronger-theory-left-right-cons trivial-implication*)
ultimately show *?thesis*
unfolding *stronger-theory-relation-alt-def*
by *simp*
next
case *False*
let $?\Gamma = map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ (map\ (uncurry\ (\sqcup))\ (\delta\ \#\ \Delta)) \ominus map\ snd\ \Sigma$
from *False* **obtain** σ **where**
σ: $find\ (\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
$snd\ \sigma = uncurry\ (\sqcup)\ \delta$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
let $?\Gamma_0 = map\ (uncurry\ (\rightarrow))\ (remove1\ \sigma\ \Sigma)$
$@\ (map\ (uncurry\ (\sqcup))\ \Delta) \ominus map\ snd\ (remove1\ \sigma\ \Sigma)$
let $?\alpha = fst\ \sigma$
let $?\beta = fst\ \delta$
let $?\gamma = snd\ \delta$
have $uncurry\ (\sqcup) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$
$uncurry\ (\rightarrow) = (\lambda\ \sigma.\ fst\ \sigma \rightarrow snd\ \sigma)$
by *fastforce+*
hence $uncurry\ (\rightarrow)\ \sigma = ?\alpha \rightarrow (?\beta \sqcup ?\gamma)$
using $\sigma(2)$
by *simp*
from $\sigma(3)$ **have** $mset\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)) = mset\ \Sigma$ **by** *simp*
hence ♠: $mset\ (map\ snd\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))) = mset\ (map\ snd\ \Sigma)$
$mset\ (map\ (uncurry\ (\rightarrow))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))) = mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma)$
by (*metis mset-map*)+
hence $mset\ ?\Gamma = mset\ (map\ (uncurry\ (\rightarrow))\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
$@\ (uncurry\ (\sqcup)\ \delta\ \#\ map\ (uncurry\ (\sqcup))\ \Delta)$
$\ominus map\ snd\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)))$
by *simp*
hence $?\Gamma \preceq (?\alpha \rightarrow (?\beta \sqcup ?\gamma)\ \#\ ?\Gamma_0)$
using $\sigma(2)$ ‹$uncurry\ (\rightarrow)\ \sigma = ?\alpha \rightarrow (?\beta \sqcup ?\gamma)$›
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* σ Σ) Δ) **by** *simp*
{
fix α β γ
have $\vdash (\beta \sqcup (\alpha \sqcup \beta) \to \gamma) \to (\alpha \to (\beta \sqcup \gamma))$
proof −
let $?\varphi = (\langle\beta\rangle \sqcup (\langle\alpha\rangle \sqcup \langle\beta\rangle) \to \langle\gamma\rangle) \to (\langle\alpha\rangle \to (\langle\beta\rangle \sqcup \langle\gamma\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
}
hence $\vdash (?\beta \sqcup (?\alpha \sqcup ?\beta) \to ?\gamma) \to (?\alpha \to (?\beta \sqcup ?\gamma))$
by *simp*
hence $(?\alpha \to (?\beta \sqcup ?\gamma)$ # $?\Gamma_0) \preceq$ *map* (*uncurry* ($\sqcup$)) ($\mathfrak{Q}$ Σ (δ # Δ))
using σ(*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* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
shows *mset* (*map snd* ($\mathfrak{Q}$ Σ Δ))
= *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{P}$ Σ Δ) @ *map snd* Δ $\ominus$ *map snd* ($\mathfrak{P}$ Σ Δ))
proof −
have $\forall$ Σ. *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
$\longrightarrow$ *mset* (*map snd* ($\mathfrak{Q}$ Σ Δ)) = *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{P}$ Σ Δ) @ *map snd* ($\mathfrak{P}^C$ Σ Δ))
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Σ :: ('*a* $\times$ '*a*) *list*
assume ⋆: *mset* (*map snd* Σ) $\subseteq$# *mset* (*map* (*uncurry* ($\sqcup$)) (δ # Δ))
have *mset* (*map snd* ($\mathfrak{Q}$ Σ (δ # Δ)))
= *mset* (*map* (*uncurry* ($\to$)) ($\mathfrak{P}$ Σ (δ # Δ)) @ *map snd* ($\mathfrak{P}^C$ Σ (δ # Δ)))

proof (*cases find* ($\lambda\ \sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta$) Σ = *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* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ) +
$\{\#uncurry\ (\sqcup)\ \delta\#\}$
using $\star$ **by** *force*
ultimately have *mset* (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ)
by (*metis diff-single-trivial in-multiset-in-set subset-eq-diff-conv*)
then show *?thesis* **using** *True Cons* **by** *simp*
next
case *False*
from *this* **obtain** σ **where**
σ: *find* ($\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta$) Σ = *Some* σ
snd σ = *uncurry* $(\sqcup)\ \delta$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
hence (*fst* σ, *uncurry* $(\sqcup)\ \delta$) $\in\#$ *mset* Σ
by (*metis* (*full-types*) *prod.collapse set-mset-mset*)
then have *mset* (*map snd* (*remove1* (*fst* σ, *uncurry* $(\sqcup)\ \delta$) Σ))
= *mset* (*map snd* Σ) $-$ $\{\#uncurry\ (\sqcup)\ \delta\#\}$
by (*meson remove1-pairs-list-projections-snd*)
moreover have
mset (*map snd* Σ)
$\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ) + *add-mset* (*uncurry* $(\sqcup)\ \delta$) (*mset* [])
using $\star$ **by** *force*
ultimately have *mset* (*map snd* (*remove1* σ Σ))
$\subseteq\#$ *mset* (*map* (*uncurry* $(\sqcup)$) Δ)
by (*metis* (*no-types*) $\sigma(2)$ *mset.simps*(*1*) *prod.collapse subset-eq-diff-conv*)
with $\sigma(1)$ *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*:

$map\ (uncurry\ (\to))\ \Delta \preceq map\ (uncurry\ (\to))\ (\mathfrak{Q}\ \Sigma\ \Delta)$
proof −
have $\forall\ \Sigma.\ map\ (uncurry\ (\to))\ \Delta \preceq map\ (uncurry\ (\to))\ (\mathfrak{Q}\ \Sigma\ \Delta)$
proof (*induct* Δ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* δ Δ)
{
fix Σ
have $map\ (uncurry\ (\to))\ (\delta \# \Delta) \preceq map\ (uncurry\ (\to))\ (\mathfrak{Q}\ \Sigma\ (\delta \# \Delta))$
proof (*cases find* $(\lambda\ \sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = None$)
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:
$find\ (\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
by *fastforce*
let $?\alpha = fst\ \delta$
let $?\beta = snd\ \delta$
let $?\gamma = fst\ \sigma$
have $uncurry\ (\to) = (\lambda\delta.\ fst\ \delta \to snd\ \delta)$ **by** *fastforce*
hence $uncurry\ (\to)\ \delta = ?\alpha \to ?\beta$ **by** *auto*
moreover have $\vdash (?\alpha \to (?\gamma \sqcup ?\alpha) \to ?\beta) \to ?\alpha \to ?\beta$
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 $mset\ (map\ snd\ \Delta) \subseteq\# mset\ \Gamma$
and $mset\ (map\ snd\ \Sigma) \subseteq\# mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
shows $mset\ (\Gamma \ominus map\ snd\ \Delta)$
$= mset\ ((map\ (uncurry\ (\to))\ (\mathfrak{P}\ \Sigma\ \Delta)\ @\ \Gamma \ominus map\ snd\ (\mathfrak{P}\ \Sigma\ \Delta)) \ominus map\ snd\ (\mathfrak{Q}\ \Sigma\ \Delta))$
proof −
have $mset\ (\Gamma \ominus map\ snd\ \Delta) = mset\ (\Gamma \ominus map\ snd\ (\mathfrak{P}^C\ \Sigma\ \Delta) \ominus map\ snd\ (\mathfrak{P}\ \Sigma\ \Delta))$

using *assms*(*2*) *recoverWitnessA-mset-equiv*
by (*simp add*: *union-commute*)
moreover have $\forall\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
$\longrightarrow mset\ (\Gamma \ominus map\ snd\ (\mathfrak{P}^C\ \Sigma\ \Delta))$
$= (mset\ ((map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ \Delta)\ @\ \Gamma) \ominus map\ snd\ (\mathfrak{Q}\ \Sigma\ \Delta)))$
using *assms*(*1*)
proof (*induct* Δ)
case *Nil*
then show *?case* by *simp*
next
case (*Cons* δ Δ)
from *Cons.prems* have $snd\ \delta \in set\ \Gamma$
using *mset-subset-eqD* by *fastforce*
from *Cons.prems* have $\heartsuit$: $mset\ (map\ snd\ \Delta) \subseteq\#\ mset\ \Gamma$
using *subset-mset.dual-order.trans*
by *fastforce*
{
fix Σ :: $('a \times 'a)\ list$
assume $\star$: $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ (\delta\ \#\ \Delta))$
have $mset\ (\Gamma \ominus map\ snd\ (\mathfrak{P}^C\ \Sigma\ (\delta\ \#\ \Delta)))$
$= mset\ ((map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ (\delta\ \#\ \Delta))\ @\ \Gamma) \ominus map\ snd\ (\mathfrak{Q}\ \Sigma\ (\delta\ \#\ \Delta)))$
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 $\star$ by *auto*
ultimately have $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ (map\ (uncurry\ (\sqcup))\ \Delta)$
by (*metis* (*full-types*) *diff-single-trivial in-multiset-in-set subset-eq-diff-conv*)
with *Cons.hyps* $\heartsuit$ have $mset\ (\Gamma \ominus map\ snd\ (\mathfrak{P}^C\ \Sigma\ \Delta))$
$= mset\ ((map\ (uncurry\ (\rightarrow))\ (\mathfrak{P}\ \Sigma\ \Delta)\ @\ \Gamma) \ominus map\ snd\ (\mathfrak{Q}\ \Sigma\ \Delta))$
by *simp*
thus *?thesis* using *True* ‹$snd\ \delta \in set\ \Gamma$› by *simp*
next
case *False*
from *this* obtain σ where σ:
find $(\lambda\sigma.\ snd\ \sigma = uncurry\ (\sqcup)\ \delta)\ \Sigma = Some\ \sigma$
$snd\ \sigma = uncurry\ (\sqcup)\ \delta$

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

theorem (**in** *Classical-Propositional-Logic*) *segmented-deduction-generalized-witness*:
$\Gamma\ \$\vdash\ (\Phi\ @\ \Psi) = (\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
$map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ \Phi\ \wedge$
$(map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus (map\ snd\ \Sigma))\ \$\vdash\ \Psi)$
proof –
have $\forall\ \Gamma\ \Psi.\ \Gamma\ \$\vdash\ (\Phi\ @\ \Psi) = (\exists\ \Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma\ \wedge$
$map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ \Phi\ \wedge$
$(map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus (map\ snd\ \Sigma))\ \$\vdash\ \Psi)$

proof (*induct* Φ)
  case *Nil*
  {
    fix Γ Ψ
    have Γ $\vdash ([] @ \Psi) = (\exists\,\Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma \wedge$
      $map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ []\ \wedge$
      $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \$\vdash\ \Psi)$
    proof (*rule iffI*)
      assume $\Gamma\ \$\vdash\ ([]\ @\ \Psi)$
      moreover
      have $\Gamma\ \$\vdash\ ([]\ @\ \Psi) = (mset\ (map\ snd\ []) \subseteq\#\ mset\ \Gamma \wedge$
        $map\ (uncurry\ (\sqcup))\ []\ \$\vdash\ []\ \wedge$
        $map\ (uncurry\ (\rightarrow))\ []\ @\ \Gamma \ominus (map\ snd\ [])\ \$\vdash\ \Psi)$
        by *simp*
      ultimately show $\exists\,\Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma \wedge$
        $map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ []\ \wedge$
        $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \$\vdash\ \Psi$
        by *metis*
    next
      assume $\exists\,\Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma \wedge$
        $map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ []\ \wedge$
        $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \$\vdash\ \Psi$
      from *this* **obtain** Σ **where**
        Σ: $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma$
          $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \$\vdash\ ([]\ @\ \Psi)$
        by *fastforce*
      hence $(map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma) \preceq \Gamma$
        using *witness-stronger-theory* **by** *auto*
      with $\Sigma(2)$ **show** $\Gamma\ \$\vdash\ ([]\ @\ \Psi)$
        using *segmented-stronger-theory-left-monotonic* **by** *blast*
    qed
  }
  then show *?case* **by** *blast*
next
  case (*Cons* φ Φ)
  {
    fix Γ Ψ
    have $\Gamma\ \$\vdash\ ((\varphi\ \#\ \Phi)\ @\ \Psi) = (\exists\,\Sigma.\ mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma \wedge$
      $map\ (uncurry\ (\sqcup))\ \Sigma\ \$\vdash\ (\varphi\ \#\ \Phi)\ \wedge$
      $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ \$\vdash\ \Psi)$
    proof (*rule iffI*)
      assume $\Gamma\ \$\vdash\ ((\varphi\ \#\ \Phi)\ @\ \Psi)$
      from *this* **obtain** Σ **where**
        Σ: $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma$
          $map\ (uncurry\ (\sqcup))\ \Sigma\ :\vdash\ \varphi$
          $map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus (map\ snd\ \Sigma)\ \$\vdash\ (\Phi\ @\ \Psi)$
          (**is** $?\Gamma_0\ \$\vdash\ (\Phi\ @\ \Psi)$)
        by *auto*
      from *this(3)* **obtain** Δ **where**

Δ: *mset* (*map snd* Δ) $\subseteq\#$ *mset* $?\Gamma_0$
    map (*uncurry* ($\sqcup$)) Δ $\$\vdash$ Φ
    map (*uncurry* ($\rightarrow$)) Δ @ $?\Gamma_0 \ominus$ (*map snd* Δ) $\$\vdash$ Ψ
  using *Cons*
  by *auto*
let $?\Sigma' = \mathfrak{J}\ \Sigma\ \Delta$
have *map* (*uncurry* ($\sqcup$)) $?\Sigma'$ $\$\vdash$ (φ # Φ)
  using Δ(*1*) Δ(*2*) Σ(*2*) *mergeWitness-cons-segmented-deduction* **by** *blast*
moreover have *mset* (*map snd* $?\Sigma'$) $\subseteq\#$ *mset* Γ
  using Δ(*1*) Σ(*1*) *mergeWitness-msub-intro* **by** *blast*
moreover have *map* (*uncurry* ($\rightarrow$)) $?\Sigma'$ @ $\Gamma \ominus$ *map snd* $?\Sigma'$ $\$\vdash$ Ψ
  using Δ(*1*) Δ(*3*) *mergeWitness-segmented-deduction-intro* **by** *blast*
ultimately show
  $\exists\,\Sigma$. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
      map (*uncurry* ($\sqcup$)) Σ $\$\vdash$ (φ # Φ) $\wedge$
      map (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ $\$\vdash$ Ψ
  by *fast*
next
  assume $\exists\,\Sigma$. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
        map (*uncurry* ($\sqcup$)) Σ $\$\vdash$ (φ # Φ) $\wedge$
        map (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ $\$\vdash$ Ψ
  from *this* **obtain** Δ **where** Δ:
    mset (*map snd* Δ) $\subseteq\#$ *mset* Γ
    map (*uncurry* ($\sqcup$)) Δ $\$\vdash$ (φ # Φ)
    map (*uncurry* ($\rightarrow$)) Δ @ $\Gamma \ominus$ *map snd* Δ $\$\vdash$ Ψ
    by *auto*
  from *this* **obtain** Σ **where** Σ:
    mset (*map snd* Σ) $\subseteq\#$ *mset* (*map* (*uncurry* ($\sqcup$)) Δ)
    map (*uncurry* ($\sqcup$)) Σ $:\vdash$ φ
    map (*uncurry* ($\rightarrow$)) Σ @ (*map* (*uncurry* ($\sqcup$)) Δ) $\ominus$ *map snd* Σ $\$\vdash$ Φ
    by *auto*
  let $?\Omega = \mathfrak{P}\ \Sigma\ \Delta$
  let $?\Xi = \mathfrak{Q}\ \Sigma\ \Delta$
  let $?\Gamma_0$ = *map* (*uncurry* ($\rightarrow$)) $?\Omega$ @ $\Gamma \ominus$ *map snd* $?\Omega$
  let $?\Gamma_1$ = *map* (*uncurry* ($\rightarrow$)) $?\Xi$ @ $?\Gamma_0 \ominus$ *map snd* $?\Xi$
  have *mset* ($\Gamma \ominus$ *map snd* Δ) = *mset* ($?\Gamma_0 \ominus$ *map snd* $?\Xi$)
    using Δ(*1*) Σ(*1*) *recoverWitnesses-mset-equiv* **by** *blast*
  hence ($\Gamma \ominus$ *map snd* Δ) $\preceq$ ($?\Gamma_0 \ominus$ *map snd* $?\Xi$)
    by (*simp add*: *msub-stronger-theory-intro*)
  hence $?\Gamma_1$ $\$\vdash$ Ψ
    using 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 (⊔)) ?Ξ $⊢ Φ
        using Σ(1) recoverWitnessB-stronger-theory
              Σ(3) segmented-stronger-theory-left-monotonic by blast
      ultimately have ?Γ₀ $⊢ (Φ @ Ψ)
        using Cons by fast
      moreover
      have mset (map snd ?Ω) ⊆# mset (map snd Δ)
        using Σ(1) recoverWitnessA-mset-equiv
        by (simp, metis mset-subset-eq-add-left)
      hence mset (map snd ?Ω) ⊆# mset Γ using Δ(1) by simp
      moreover
      have map (uncurry (⊔)) ?Ω :⊢ 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
```

$\ominus$ *map snd* Σ)
using *segmented-deduction-generalized-witness*
by *fastforce*
have *map* (*uncurry* ($\rightarrow$)) Ψ @ $\Gamma \ominus$ *map snd* Ψ $\$\vdash$ Δ
using Σ(*3*) Ψ(*3*) *Cons*
by *auto*
moreover
have *map* (*uncurry* ($\sqcup$)) Ψ $:\vdash \delta$
using Ψ(*2*) Σ(*2*) *segmented-list-deduction-antitonic*
by *blast*
ultimately have Γ $\$\vdash$ (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*:
$\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ Φ = φ # Γ $\$\vdash$ Φ
proof (*rule iffI*)
assume φ # Γ $\$\vdash$ Φ
have $\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ ($\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ)
using *segmented-stronger-theory-intro*
stronger-theory-reflexive
by *blast*
hence $\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ (φ # Γ)
using *segmented-formula-right-split* **by** *blast*
with ‹φ # Γ $\$\vdash$ Φ› **show** $\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ Φ
using *segmented-transitive* **by** *blast*
next
assume $\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ Φ
have φ # Γ $\$\vdash$ (φ # Γ)
using *segmented-stronger-theory-intro*
stronger-theory-reflexive
by *blast*
hence φ # Γ $\$\vdash$ ($\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ)
using *segmented-formula-right-split* **by** *blast*
with ‹$\psi \sqcup \varphi$ # $\psi \rightarrow \varphi$ # Γ $\$\vdash$ Φ› **show** φ # Γ $\$\vdash$ Φ
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)) $\$\vdash$
Φ = Γ $\$\vdash$ Φ
proof –
have $\forall$ Γ. *mset* (*map snd* Σ) $\subseteq$# *mset* Γ $\longrightarrow$

(map ($uncurry$ (⊔)) Σ @ map ($uncurry$ (→)) Σ @ Γ ⊖ (map snd Σ)) $⊢ Φ =
Γ $⊢ Φ
  proof ($induct$ Σ)
    case Nil
    then show $?case$ **by** $simp$
  next
    case ($Cons$ σ Σ)
    {
      fix Γ
      let $?χ$ = fst σ
      let $?γ$ = snd σ
      let $?Γ_0$ = map ($uncurry$ (⊔)) Σ @ map ($uncurry$ (→)) Σ @ Γ ⊖ map snd (σ
Σ)
      let $?Γ'$ = map ($uncurry$ (⊔)) (σ # Σ) @ map ($uncurry$ (→)) (σ # Σ) @ Γ
⊖ map snd (σ # Σ)
      assume $mset$ (map snd (σ # Σ)) ⊆# $mset$ Γ
      hence A: $add\text{-}mset$ (snd σ) ($image\text{-}mset$ snd ($mset$ Σ)) ⊆# $mset$ Γ **by** $simp$
      hence B: $image\text{-}mset$ snd ($mset$ Σ) + ($mset$ Γ − $image\text{-}mset$ snd ($mset$ Σ))
          = $add\text{-}mset$ (snd σ) ($image\text{-}mset$ snd ($mset$ Σ))
            + ($mset$ Γ − $add\text{-}mset$ (snd σ) ($image\text{-}mset$ snd ($mset$ Σ)))
        by ($metis$ ($no\text{-}types$) $mset\text{-}subset\text{-}eq\text{-}insertD$ $subset\text{-}mset.add\text{-}diff\text{-}inverse$
$subset\text{-}mset\text{-}def$)
        have {#x → y. (x, y) ∈# $mset$ Σ#} + $mset$ Γ − $add\text{-}mset$ (snd σ)
($image\text{-}mset$ snd ($mset$ Σ))
          = {#x → y. (x, y) ∈# $mset$ Σ#} + ($mset$ Γ − $add\text{-}mset$ (snd σ)
($image\text{-}mset$ snd ($mset$ Σ)))
        using A $subset\text{-}mset.diff\text{-}add\text{-}assoc$ **by** $blast$
      hence {#x → y. (x, y) ∈# $mset$ Σ#} + ($mset$ Γ − $image\text{-}mset$ snd ($mset$
Σ))
          = $add\text{-}mset$ (snd σ) ({#x → y. (x, y) ∈# $mset$ Σ#}
            + $mset$ Γ − $add\text{-}mset$ (snd σ) ($image\text{-}mset$ 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)
        = $mset$ ($?γ$ # $?Γ_0$)
        using ‹$mset$ (map snd (σ # Σ)) ⊆# $mset$ Γ›
            $subset\text{-}mset.dual\text{-}order.trans$
        by ($fastforce$+)
      hence Γ $⊢ Φ = ($?χ$ ⊔ $?γ$ # $?χ$ → $?γ$ # $?Γ_0$) $⊢ 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 $Cons.hyps$ $segmented\text{-}msub\text{-}left\text{-}monotonic$ **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 $\Sigma(2)$
by *blast*
next
case *False*
from *this* **obtain** σ χ **where**
σ: $\sigma = (\chi, \delta)$
$\sigma \in set\ \Sigma$
using *find-Some-predicate*
find-Some-set-membership
by *fastforce*
let $?\Sigma' = remove1\ \sigma\ \Sigma$
let $?\Sigma_A = map\ (uncurry\ (\sqcup))\ ?\Sigma'$
let $?\Sigma_B = map\ (uncurry\ (\rightarrow))\ ?\Sigma'$
have $mset\ \Sigma = mset\ (?\Sigma' @ [(\chi, \delta)])$
$mset\ \Sigma = mset\ ((\chi, \delta) \# ?\Sigma')$
using σ **by** *simp+*
hence $mset\ (map\ (uncurry\ (\sqcup))\ \Sigma) = mset\ (map\ (uncurry\ (\sqcup))\ (?\Sigma' @ [(\chi, \delta)]))$
$mset\ (map\ snd\ \Sigma) = mset\ (map\ snd\ ((\chi, \delta) \# ?\Sigma'))$
$mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma) = mset\ (map\ (uncurry\ (\rightarrow))\ ((\chi, \delta) \# ?\Sigma'))$
by (*metis mset-map*)+
hence $mset\ (map\ (uncurry\ (\sqcup))\ \Sigma) = mset\ (?\Sigma_A @ [\chi \sqcup \delta])$
$mset\ (map\ (uncurry\ (\rightarrow))\ \Sigma @ ((\delta \# \Delta) @ \Gamma) \ominus map\ snd\ \Sigma)$
$= mset\ (\chi \rightarrow \delta \# ?\Sigma_B @ (\Delta @ \Gamma) \ominus map\ snd\ ?\Sigma')$
by *simp+*
hence
$?\Sigma_A @ [\chi \sqcup \delta]$ $\$\vdash$ $(\Delta @ \Phi)$
$\chi \rightarrow \delta \# (?\Sigma_B @ (\Delta @ \Gamma) \ominus map\ snd\ ?\Sigma')$ $\$\vdash$ $[\delta]$
using $\Sigma(2)$ $\Sigma(3)$
by (*metis segmented-msub-left-monotonic subset-mset.dual-order.refl*, *simp*)
moreover
have $\vdash ((\chi \rightarrow \delta) \rightarrow \delta) \rightarrow (\chi \sqcup \delta)$
unfolding *disjunction-def*
using *Modus-Ponens*
The-Principle-of-Pseudo-Scotus
flip-hypothetical-syllogism
by *blast*
ultimately have $(?\Sigma_A @ ?\Sigma_B @ (\Delta @ \Gamma) \ominus map\ snd\ ?\Sigma')$ $\$\vdash$ $(\Delta @ \Phi)$
using *segmented-deduction-one-collapse*
list-deduction-theorem
list-deduction-modus-ponens
list-deduction-weaken
forward-direction

segmented-transitive
by *meson*
moreover
have δ = *snd* σ
snd $\sigma \in$ *set* (*map snd* Σ)
by (*simp add*: σ(*1*), *simp add*: σ(*2*))
with Σ(*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
σ(*1*) σ(*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 $\vdash \gamma \leftrightarrow \varphi$
shows (γ $\#$ Γ) $\$\vdash$ (φ $\#$ Φ) = Γ $\$\vdash$ Φ
proof −
from *assms* **have** (γ $\#$ Φ) $\preceq$ (φ $\#$ Φ) (φ $\#$ Φ) $\preceq$ (γ $\#$ Φ)
unfolding *biconditional-def*
by (*simp add*: *stronger-theory-left-right-cons*)+
hence (γ $\#$ Φ) $\$\vdash$ (φ $\#$ Φ)
(φ $\#$ Φ) $\$\vdash$ (γ $\#$ Φ)
using *segmented-stronger-theory-intro* **by** *blast*+
moreover
have Γ $\$\vdash$ Φ = (γ $\#$ Γ) $\$\vdash$ (γ $\#$ Φ)
by (*metis append-Cons append-Nil segmented-cancel*)+
ultimately
have Γ $\$\vdash$ $\Phi \Longrightarrow \gamma$ $\#$ Γ $\$\vdash$ (φ $\#$ Φ)
γ $\#$ Γ $\$\vdash$ (φ $\#$ Φ) $\Longrightarrow \Gamma$ $\$\vdash$ Φ
using *segmented-transitive* **by** *blast*+
thus *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *right-segmented-sub*:
assumes $\vdash \varphi \leftrightarrow \psi$
shows Γ $\$\vdash$ (φ $\#$ Φ) = Γ $\$\vdash$ (ψ $\#$ Φ)
proof −
have Γ $\$\vdash$ (φ $\#$ Φ) = (ψ $\#$ Γ) $\$\vdash$ (ψ $\#$ φ $\#$ Φ)

using *segmented-cancel* [**where** Δ=[ψ] **and** Γ=Γ **and** Φ=φ # Φ] **by** *simp*
also have ... = (ψ # Γ) $⊢ ($\varphi$ # ψ # Φ)
using *segmented-cons-cons-right-permute* **by** *blast*
also have ... = Γ $⊢ ($\psi$ # Φ)
using *assms biconditional-symmetry-rule segmented-biconditional-cancel* **by** *blast*
finally show *?thesis* **.**
qed

lemma (**in** *Classical-Propositional-Logic*) *left-segmented-sub*:
assumes $\vdash \gamma \leftrightarrow \chi$
shows (γ # Γ) $⊢ Φ = (χ # Γ) $⊢ Φ
proof −
have (γ # Γ) $⊢ Φ = (χ # γ # Γ) $⊢ ($\chi$ # Φ)
using *segmented-cancel* [**where** Δ=[χ] **and** Γ=(γ # Γ) **and** Φ=Φ] **by** *simp*
also have ... = (γ # χ # Γ) $⊢ ($\chi$ # 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*:
Γ $⊢ ($\alpha$ # β # Φ) = Γ $⊢ ($\alpha \sqcup \beta$ # $\alpha \sqcap \beta$ # Φ)
proof −
have *A*: *mset* ($\alpha \sqcup \beta$ # $\beta \to \alpha$ # β # Φ) = *mset* ($\beta \to \alpha$ # β # $\alpha \sqcup \beta$ # Φ) **by** *simp*
have *B*: $\vdash (\beta \to \alpha) \leftrightarrow (\beta \to (\alpha \sqcap \beta))$
proof −
let *?*φ = $((\langle\beta\rangle \to \langle\alpha\rangle) \leftrightarrow (\langle\beta\rangle \to (\langle\alpha\rangle \sqcap \langle\beta\rangle)))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ (| *?*φ |) **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
have *C*: $\vdash \beta \leftrightarrow (\beta \sqcup (\alpha \sqcap \beta))$
proof −
let *?*φ = $\langle\beta\rangle \leftrightarrow (\langle\beta\rangle \sqcup (\langle\alpha\rangle \sqcap \langle\beta\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ (| *?*φ |) **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
have Γ $⊢ ($\alpha$ # β # Φ) = Γ $⊢ ($\beta \sqcup \alpha$ # $\beta \to \alpha$ # β # Φ)
using *segmented-formula-right-split* **by** *blast*
also have ... = Γ $⊢ ($\alpha \sqcup \beta$ # $\beta \to \alpha$ # β # Φ)
using *disjunction-commutativity right-segmented-sub* **by** *blast*
also have ... = Γ $⊢ ($\beta \to \alpha$ # β # $\alpha \sqcup \beta$ # Φ)
by (*metis A segmented-msub-weaken subset-mset.dual-order.refl*)
also have ... = Γ $⊢ ($\beta \to (\alpha \sqcap \beta)$ # β # $\alpha \sqcup \beta$ # Φ)

using *B right-segmented-sub* **by** *blast*
also have ... = Γ \$$\vdash$ $(\beta \# \beta \to (\alpha \sqcap \beta) \# \alpha \sqcup \beta \# \Phi)$
using *segmented-cons-cons-right-permute* **by** *blast*
also have ... = Γ \$$\vdash$ $(\beta \sqcup (\alpha \sqcap \beta) \# \beta \to (\alpha \sqcap \beta) \# \alpha \sqcup \beta \# \Phi)$
using *C right-segmented-sub* **by** *blast*
also have ... = Γ \$$\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$ Φ
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** $\Delta=[\alpha \sqcup \beta, \alpha \sqcap \beta]$ **and** $\Gamma=(\alpha \# \beta \# \Gamma)$ **and** $\Phi=\Phi$] **by** *simp*
also have ... = $(\alpha \sqcup \beta \# \alpha \sqcap \beta \# \alpha \# \beta \# \Gamma)$ \$$\vdash$ $(\alpha \# \beta \# \Phi)$
using *right-segmented-sum-rule* **by** *blast*
also have ... = $(\alpha \# \beta \# \alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma)$ \$$\vdash$ $(\alpha \# \beta \# \Phi)$
by (*metis* $\star$ *segmented-msub-left-monotonic subset-mset.dual-order.refl*)
also have ... = $(\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma)$ \$$\vdash$ Φ
using *segmented-cancel* [**where** $\Delta=[\alpha, \beta]$ **and** $\Gamma=(\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma)$ **and** $\Phi=\Phi$] **by** *simp*
finally show *?thesis* **.**
qed

lemma (**in** *Classical-Propositional-Logic*) *segmented-exchange*:
$(\gamma \# \Gamma)$ \$$\vdash$ $(\varphi \# \Phi) = (\varphi \to \gamma \# \Gamma)$ \$$\vdash$ $(\gamma \to \varphi \# \Phi)$
proof −
have $(\gamma \# \Gamma)$ \$$\vdash$ $(\varphi \# \Phi)$
$= (\varphi \sqcup \gamma \# \varphi \to \gamma \# \Gamma)$ \$$\vdash$ $(\gamma \sqcup \varphi \# \gamma \to \varphi \# \Phi)$
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 \$$\vdash$ $(\varphi \# \Phi) = (\sim \varphi \# \Gamma)$ \$$\vdash$ $(\bot \# \Phi)$
proof −
have Γ \$$\vdash$ $(\varphi \# \Phi) = (\bot \# \Gamma)$ \$$\vdash$ $(\bot \# \varphi \# \Phi)$
by (*metis append-Cons append-Nil segmented-cancel*)

```
  also have ... = (⊥ # Γ) $⊢ (φ # ⊥ # Φ)
    using segmented-cons-cons-right-permute by blast
  also have ... = (∼ φ # Γ) $⊢ (⊥ → φ # ⊥ # Φ)
    unfolding negation-def
    using segmented-exchange
    by blast
  also have ... = (∼ φ # Γ) $⊢ (⊥ # 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 = (∃ Ψ. mset (map snd Ψ) ⊆# mset Γ ∧
                          map (uncurry (⊔)) Ψ :⊢ φ ∧
                          map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ) #⊢ n φ)

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

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

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

lemma (**in** *Classical-Propositional-Logic*) *stratified-deduction-implication*:
  assumes $\vdash \varphi \to \psi$
    and Γ #⊢ n φ
  shows Γ #⊢ n ψ
proof −
  have *replicate* n $\psi \preceq$ *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 \$⊢ Φ = (∼ Φ @ Γ) #⊢ (*length* Φ) $\bot$
proof −
  have $\forall$ Γ Ψ. Γ \$⊢ (Φ @ Ψ) = (∼ Φ @ Γ) \$⊢ (*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 \$⊢ ((φ # Φ) @ Ψ) = ($\sim \varphi$ # Γ) \$⊢ ($\bot$ # Φ @ Ψ)
        using *segmented-negation-swap* **by** *auto*
      moreover have *mset* (Φ @ ($\bot$ # Ψ)) = *mset* ($\bot$ # Φ @ Ψ)

by *simp*
ultimately have Γ \$⊢ $((\varphi \# \Phi) @ \Psi) = (\sim \varphi \# \Gamma)$ \$⊢ $(\Phi @ (\bot \# \Psi))$
by (*metis segmented-msub-weaken subset-mset.order-refl*)
hence Γ \$⊢ $((\varphi \# \Phi) @ \Psi) = (\boldsymbol{\sim} \Phi @ (\sim \varphi \# \Gamma))$ \$⊢ (*replicate* (*length* Φ) $\bot @ (\bot \# \Psi))$
using *Cons*
by *blast*
moreover have *mset* $(\boldsymbol{\sim} \Phi @ (\sim \varphi \# \Gamma)) =$ *mset* $(\boldsymbol{\sim} (\varphi \# \Phi) @ \Gamma)$
mset (*replicate* (*length* Φ) $\bot @ (\bot \# \Psi))$
$=$ *mset* (*replicate* (*length* $(\varphi \# \Phi)$) $\bot @ \Psi)$
by *simp+*
ultimately have
Γ \$⊢ $((\varphi \# \Phi) @ \Psi) = \boldsymbol{\sim} (\varphi \# \Phi) @ \Gamma$ \$⊢ (*replicate* (*length* $(\varphi \# \Phi)$) $\bot @ \Psi)$
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* ($\mathcal{C}$)
where
$\mathcal{C}\ \Gamma\ \varphi = \{\Phi.$ *mset* $\Phi \subseteq\#$ *mset* Γ
$\wedge \neg \Phi :\vdash \varphi$
$\wedge (\forall\ \Psi.$ *mset* $\Psi \subseteq\#$ *mset* $\Gamma \longrightarrow \neg \Psi :\vdash \varphi \longrightarrow$ *length* $\Psi \leq$ *length* $\Phi)\}$

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

lemma (**in** *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* Γ − *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*:

```
  Γ :⊢ φ = (∀ Φ ∈ 𝒞 Γ φ. 1 ≤ length (Γ ⊖ Φ))
proof cases
  assume ⊢ φ
  hence Γ :⊢ φ 𝒞 Γ φ = {}
    unfolding unproving-core-def
    by (simp add: list-deduction-weaken)+
  then show ?thesis by blast
next
  assume ¬ ⊢ φ
  from this obtain Ω where Ω: Ω ∈ 𝒞 Γ φ
    using unproving-core-existence by blast
  from this have mset Ω ⊆# mset Γ
    unfolding unproving-core-def by blast
  hence ◇: length (Γ ⊖ Ω) = 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 $1 > length\ (\Gamma \ominus \Omega)$
  using $\diamondsuit$
  by *simp*
with $\langle\neg\ \Gamma :\vdash \varphi\rangle\ \Omega$ **show** *?thesis*
  by *fastforce*
qed
qed

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

abbreviation (**in** *Minimal-Logic-With-Falsum*) *MaxSAT* :: $'a\ list \Rightarrow nat$
where
$MaxSAT\ \Gamma \equiv |\ \Gamma\ |_\bot$

definition (**in** *Minimal-Logic*) *complement-core-size* :: $'a\ list \Rightarrow 'a \Rightarrow nat$ ($(\|\ \text{-}\ \|_{\text{-}}$ [45])
where
$(\|\ \Gamma\ \|_\varphi) = length\ \Gamma - |\ \Gamma\ |_\varphi$

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

lemma (**in** *Minimal-Logic*) *complement-core-size-intro*:
assumes $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
shows $length\ (\Gamma \ominus \Phi) = \|\ \Gamma\ \|_\varphi$
proof −
have $mset\ \Phi \subseteq\#\ mset\ \Gamma$
  using *assms*
  unfolding *unproving-core-def*
  by *auto*

moreover from *this* **have** $length\ (\Gamma \ominus \Phi) = length\ \Gamma - length\ \Phi$
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*:
$length\ \Gamma = (|\ \Gamma\ |_{\varphi}) + \|\ \Gamma\ \|_{\varphi}$
proof (*cases* $\mathcal{C}\ \Gamma\ \varphi = \{\}$)
case *True*
then show *?thesis*
unfolding *core-size-def*
complement-core-size-def
by *simp*
next
case *False*
from *this* **obtain** Φ **where** $\Phi \in \mathcal{C}\ \Gamma\ \varphi$
by *fast*
moreover from *this* **have** $mset\ \Phi \subseteq\#\ mset\ \Gamma$
unfolding *unproving-core-def*
by *auto*
moreover from *this* **have** $length\ (\Gamma \ominus \Phi) = length\ \Gamma - length\ \Phi$
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
$\mathfrak{V}\ [] = []$
$|\ \mathfrak{V}\ (\psi\ \#\ \Psi) = (\Psi, \psi)\ \#\ \mathfrak{V}\ \Psi$

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

lemma *core-optimal-pre-witness-nonelement*:
assumes $length\ \Delta \geq length\ \Psi$
shows $(\Delta,\delta) \notin set\ (\mathfrak{V}\ \Psi)$
using *assms*
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* $\psi\ \Psi$)
hence $\Psi \neq \Delta$ **by** *auto*

then show *?case* **using** *Cons* **by** *simp*
qed

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

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

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

lemma *mset-remdups-set-sub-iff*:
(*mset* (*remdups A*) $\subseteq$# *mset* (*remdups B*)) = (*set A* $\subseteq$ *set B*)
proof –
have $\forall$ *B*. (*mset* (*remdups A*) $\subseteq$# *mset* (*remdups B*)) = (*set A* $\subseteq$ *set B*)
proof (*induct A*)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons a A*)
then show *?case*
proof (*cases a* $\in$ *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* $\Sigma \subseteq\#$ *mset* ($\mathfrak{V}$ Ψ)
and $\Sigma \neq []$
shows $\exists$ (Δ, δ) $\in$ *set* Σ. *length* Δ + *1* $\geq$ *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* $\circ$ *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** σ = (Δ, δ)
by *fastforce*
hence (Δ, δ) $\in$ *set* ($\mathfrak{V}$ Ψ)
using *Cons.prems mset-subset-eq-insertD*
by *fastforce*
hence $\forall$ (Σ,σ) $\in$ *set* ($\mathfrak{V}$ Ψ). (*length* Δ = *length* Σ) = ((Δ, δ) = (Σ, σ))
using *core-optimal-pre-witness-length-iff-eq* [**where** Ψ=Ψ]
by *fastforce*
hence $\forall$ (Σ,σ) $\in$ *set* Σ. (*length* Δ = *length* Σ) = ((Δ, δ) = (Σ, σ))
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*) ‹σ = (Δ, δ)›
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 $\mathfrak{W}\ \varphi\ \Xi \equiv map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi,\ \psi))\ (\mathfrak{V}\ \Xi)$

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

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

lemma (**in** *Classical-Propositional-Logic*) *core-optimal-witness-conjunction-identity*:
$\vdash \bigsqcap\ (\mathfrak{W}_{\sqcup}\ \varphi\ \Psi) \leftrightarrow (\varphi \sqcup \bigsqcap\ \Psi)$
proof (*induct* Ψ)
case *Nil*
then show *?case*
unfolding *biconditional-def*
disjunction-def
using *Axiom-1*
Modus-Ponens
verum-tautology
by (*simp*, *blast*)
next
case (*Cons* ψ Ψ)
have $\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcap\ \Psi \rightarrow \varphi)$
by (*simp add*: *list-curry-uncurry*)
hence $\vdash \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Psi)))$
$\leftrightarrow ((\bigsqcap\ \Psi \rightarrow \varphi \sqcup \psi) \sqcap \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ \Psi)))$
unfolding *biconditional-def*
using *conjunction-monotonic*
disjunction-monotonic
by *simp*
moreover have $\vdash ((\bigsqcap\ \Psi \rightarrow \varphi \sqcup \psi) \sqcap \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ \Psi)))$
$\leftrightarrow ((\bigsqcap\ \Psi \rightarrow \varphi \sqcup \psi) \sqcap (\varphi \sqcup \bigsqcap\ \Psi))$
using *Cons.hyps biconditional-conjunction-weaken-rule*
by *blast*
moreover
{
fix $\varphi\ \psi\ \chi$
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 $\vdash \bigsqcap\ (map\ (uncurry\ (\sqcup))\ (\mathfrak{W}\ \varphi\ (\psi\ \#\ \Psi))) \leftrightarrow (\varphi \sqcup (\psi \sqcap \bigsqcap$
$\Psi))$
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 $\vdash \mathfrak{W}_{\sqcup}\ \varphi\ \Psi :\rightarrow \varphi \leftrightarrow (\bigsqcap\ (\mathfrak{W}_{\sqcup}\ \varphi\ \Psi) \rightarrow \varphi)$
by (*simp add*: *list-curry-uncurry*)
moreover
{
fix $\alpha\ \beta\ \gamma$
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 $\vdash ((\varphi \sqcup \psi) \to \varphi) \to ((\varphi \to \psi) \to \varphi) \to \varphi$
proof $-$
let $?\varphi = ((\langle\varphi\rangle \sqcup \langle\psi\rangle) \to \langle\varphi\rangle) \to ((\langle\varphi\rangle \to \langle\psi\rangle) \to \langle\varphi\rangle) \to \langle\varphi\rangle$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
let $?A = \mathfrak{W}_{\sqcup}\ \varphi\ \Xi :\to \varphi$
let $?B = \mathfrak{W}_{\to}\ \varphi\ \Xi :\to \varphi$
let $?X = \Xi :\to \varphi$
from *Cons.hyps* **have** $\vdash ((?X \sqcup \psi) \to ?A) \to ((?X \to \psi) \to ?B) \to ?X$ **by** *simp*
moreover
have $\vdash (((?X \sqcup \psi) \to ?A) \to ((?X \to \psi) \to ?B) \to ?X)$
$\to ((\xi \to ?X \sqcup \psi) \to (?X \sqcup \xi) \to ?A) \to (((\xi \to ?X) \to \psi) \to (?X \to \xi) \to ?B) \to \xi \to ?X$
proof $-$
let $?\varphi =(((\langle ?X\rangle \sqcup \langle\psi\rangle) \to \langle ?A\rangle) \to ((\langle ?X\rangle \to \langle\psi\rangle) \to \langle ?B\rangle) \to \langle ?X\rangle) \to$
$((\langle\xi\rangle \to \langle ?X\rangle \sqcup \langle\psi\rangle) \to (\langle ?X\rangle \sqcup \langle\xi\rangle) \to \langle ?A\rangle) \to$
$(((\langle\xi\rangle \to \langle ?X\rangle) \to \langle\psi\rangle) \to (\langle ?X\rangle \to \langle\xi\rangle) \to \langle ?B\rangle) \to$
$\langle\xi\rangle \to$
$\langle ?X\rangle$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately
have $\vdash ((\xi \to ?X \sqcup \psi) \to (?X \sqcup \xi) \to ?A) \to (((\xi \to ?X) \to \psi) \to (?X \to \xi) \to ?B) \to \xi \to ?X$
using *Modus-Ponens*
by *blast*
thus *?case* **by** *simp*
qed

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

lemma (**in** *Classical-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 ... $= (\vdash \sigma :\rightarrow \varphi \wedge \vdash \bigsqcup\ (map \bigsqcap\ \Sigma) \rightarrow \varphi)$ **using** *Cons.hyps list-deduction-def* **by** *simp*
also have ... $= (\vdash \bigsqcap\ \sigma \rightarrow \varphi \wedge \vdash \bigsqcup\ (map \bigsqcap\ \Sigma) \rightarrow \varphi)$
using *list-curry-uncurry weak-biconditional-weaken* **by** *blast*
also have ... $= (\vdash \bigsqcap\ \sigma \rightarrow \varphi \sqcap \bigsqcup\ (map \bigsqcap\ \Sigma) \rightarrow \varphi)$ **by** *simp*
also have ... $= (\vdash (\bigsqcap\ \sigma \sqcup \bigsqcup\ (map \bigsqcap\ \Sigma)) \rightarrow \varphi)$
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* $((\sqcap)\ \varphi)\ \Psi) \leftrightarrow$ $(\varphi \sqcap \bigsqcup\ \Psi)$
proof (*induct* Ψ)
  case *Nil*
  then show *?case*
   by (*simp add*: *Ex-Falso-Quodlibet biconditional-def conjunction-right-elimination*)
next
  case (*Cons* $\psi\ \Psi$)
  have $\vdash \bigsqcup$ (*map* $((\sqcap)\ \varphi)\ \Psi) \leftrightarrow (\varphi \sqcap \bigsqcup\ \Psi)$
    $\to ((\varphi \sqcap \psi) \sqcup \bigsqcup$ (*map* $((\sqcap)\ \varphi)\ \Psi)) \leftrightarrow (\varphi \sqcap (\psi \sqcup \bigsqcup\ \Psi))$
  proof –
    let $?\varphi = \langle\bigsqcup$ (*map* $((\sqcap)\ \varphi)\ \Psi)\rangle \leftrightarrow (\langle\varphi\rangle \sqcap \langle\bigsqcup\ \Psi\rangle)$
      $\to ((\langle\varphi\rangle \sqcap \langle\psi\rangle) \sqcup \langle\bigsqcup$ (*map* $((\sqcap)\ \varphi)\ \Psi)\rangle) \leftrightarrow (\langle\varphi\rangle \sqcap (\langle\psi\rangle \sqcup \langle\bigsqcup\ \Psi\rangle))$
    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*) *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* $\sigma\ \Sigma$)
  moreover have
    $\vdash\ \ \bigsqcup$ (*map* $\bigsqcap$ (*map* $((@)\ \Delta)\ \Sigma)) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcup$ (*map* $\bigsqcap\ \Sigma))$
     $\to \bigsqcap\ (\Delta\ @\ \sigma) \leftrightarrow (\bigsqcap\ \Delta \sqcap \bigsqcap\ \sigma)$
     $\to (\bigsqcap\ (\Delta\ @\ \sigma) \sqcup \bigsqcup$ (*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* $\bigsqcap\ \Sigma)\rangle)$
     $\to \langle\bigsqcap\ (\Delta\ @\ \sigma)\rangle \leftrightarrow (\langle\bigsqcap\ \Delta\rangle \sqcap \langle\bigsqcap\ \sigma\rangle)$
     $\to (\langle\bigsqcap\ (\Delta\ @\ \sigma)\rangle \sqcup \langle\bigsqcup$ (*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* $\bigsqcap$

Σ)))
  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* $\circ$ (@) Δ)) Ψ) $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap$ $\bigsqcup$ (*map* ($\bigsqcap$ $\circ$ *map snd*) Ψ))
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* Δ
 let $?\chi' =$ *snd* χ
 let $?\Pi = \lambda\varphi.$ $\bigsqcap$ (*map snd* φ)
 let $?\Pi\Delta = \lambda\varphi.$ $\bigsqcap$ ($?\Delta'$ @ *map snd* φ)
 from *Cons* **have**
  $\vdash \bigsqcup$ (*map* $?\Pi\Delta$ Ψ) $\leftrightarrow$ ($\bigsqcap$ $?\Delta'$ $\sqcap$ $\bigsqcup$ (*map* $?\Pi$ Ψ))
  by *auto*
 moreover have $\star$: *map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) = *map* (($\sqcap$) $?\chi'$) $\circ$ *map* $?\Pi\Delta$
  by *fastforce*
 have $\bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ) = $\bigsqcup$ (*map* (($\sqcap$) $?\chi'$) (*map* $?\Pi\Delta$ Ψ))
  by (*simp add*: $\star$)
 hence
  $\vdash \bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ) $\leftrightarrow$ ($?\chi' \sqcap \bigsqcup$ (*map* ($\lambda\varphi.$ $?\Pi\Delta$ φ) Ψ))
  using *conj-extract* **by** *presburger*
 moreover have
  $\vdash \bigsqcup$ (*map* $?\Pi\Delta$ Ψ) $\leftrightarrow$ ($\bigsqcap$ $?\Delta'$ $\sqcap$ $\bigsqcup$ (*map* $?\Pi$ Ψ))
  $\to \bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ) $\leftrightarrow$ ($?\chi' \sqcap \bigsqcup$ (*map* $?\Pi\Delta$ Ψ))
  $\to \bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ) $\leftrightarrow$ (($?\chi' \sqcap \bigsqcap$ $?\Delta'$) $\sqcap$ $\bigsqcup$ (*map* $?\Pi$ Ψ))
 proof –
  let $?\varphi = \langle\bigsqcup$ (*map* $?\Pi\Delta$ Ψ)$\rangle \leftrightarrow (\langle\bigsqcap$ $?\Delta'\rangle \sqcap \langle\bigsqcup$ (*map* $?\Pi$ Ψ)$\rangle$)
     $\to \langle\bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ)$\rangle \leftrightarrow (\langle ?\chi'\rangle \sqcap \langle\bigsqcup$ (*map* $?\Pi\Delta$ Ψ)$\rangle$)
      $\to \langle\bigsqcup$ (*map* ($\lambda\varphi.$ $?\chi' \sqcap ?\Pi\Delta$ φ) Ψ)$\rangle \leftrightarrow ((\langle ?\chi'\rangle \sqcap \langle\bigsqcap$ $?\Delta'\rangle) \sqcap \langle\bigsqcup$ (*map* $?\Pi$ Ψ)$\rangle$)
  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* φ)) Ψ)
     $\leftrightarrow$ (($?\chi' \sqcap \bigsqcap$ $?\Delta'$) $\sqcap$ $\bigsqcup$ (*map* ($\lambda\varphi.$ $\bigsqcap$ (*map snd* φ)) Ψ))
  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* $\circ$ (@) Δ)) Ψ) $\leftrightarrow$
($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ))
proof −
have $\forall\ \Psi. \vdash \bigsqcup$ (*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))
proof (*induct* Δ)
case *Nil*
then show *?case*
by (*simp*,
meson Modus-Ponens
biconditional-introduction
conjunction-introduction
conjunction-right-elimination
verum-tautology)
next
case (*Cons* δ Δ)
{
fix Ψ
have $\vdash\ \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ($\bigsqcap$ (*map snd* (δ # Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ))
proof (*cases* $\delta \in$ *set* Δ)
assume $\delta \in$ *set* Δ
have
$\vdash\ \bigsqcap$ (*map snd* Δ) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
$\to \bigsqcup$ (*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))
$\to \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) Ψ)
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*))
Ψ))
proof −
let *?*φ = $\langle\bigsqcap$ (*map snd* Δ)$\rangle \leftrightarrow$ ($\langle$*snd* $\delta\rangle \sqcap \langle\bigsqcap$ (*map snd* Δ)$\rangle$)
$\to \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) Ψ)$\rangle$
$\leftrightarrow$ ($\langle\bigsqcap$ (*map snd* Δ)$\rangle \sqcap \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*))
Ψ)$\rangle$)
$\to \langle\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) 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)$\rangle$)
have $\forall\mathfrak{M}.\ \mathfrak{M} \models_{prop}$ *?*φ **by** *fastforce*
hence $\vdash$ ⦇ *?*φ ⦈ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
moreover have $\vdash \bigsqcap$ (*map snd* Δ) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ))
by (*simp add*: ‹$\delta \in$ *set* Δ› *conj-absorption*)
ultimately have
$\vdash\ \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) Δ)) Ψ)

$\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
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* Ψ)
case *Nil*
then show *?case*
by (*simp*, *metis list.simps*(*8*) *Arbitrary-Disjunction.simps*(*1*) *conj-extract*)
next
case (*Cons* ψ Ψ)
have $\vdash \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ [\psi])$
$\leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ [\psi]))$
using ‹$\forall \Psi.\ \vdash\ \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ \Delta))\ \Psi)$
$\leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$›
by *blast*
hence
$\vdash\ (\bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi))) \sqcup \bot)$
$\leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcap\ (map\ snd\ (remdups\ \psi)) \sqcup \bot)$
by *simp*
hence ⋆:
$\vdash \bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi))) \leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcap\ (map\ snd\ (remdups\ \psi)))$
by (*metis* (*no-types*, *hide-lams*)
biconditional-conjunction-weaken-rule
biconditional-symmetry-rule
biconditional-transitivity-rule
disjunction-def
double-negation-biconditional
negation-def)
have $\vdash\ \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)$
$\leftrightarrow (\bigsqcap\ (map\ snd\ (\delta\ \#\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$
using *Cons* **by** *blast*
hence ◇: $\vdash\ \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)$
$\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi))$

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 ♠: $\vdash \bigsqcap$ (*map snd* (*remdups* ψ)) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* ψ)))
using *conj-absorption* **by** *blast*
have
$\vdash$ ($\bigsqcap$ (*map snd* (*remdups* ψ)) $\leftrightarrow$ (*snd* $\delta \sqcap \bigsqcap$ (*map snd* (*remdups* ψ))))
$\rightarrow$ ($\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups* $\circ$ (@) (δ # Δ))) Ψ)
$\leftrightarrow$ ((*snd* $\delta \sqcap \bigsqcap$ (*map snd* Δ)) $\sqcap \bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)))
$\rightarrow$ ($\bigsqcap$ (*map snd* (*remdups* (Δ @ ψ))) $\leftrightarrow$ ($\bigsqcap$ (*map snd* Δ) $\sqcap \bigsqcap$ (*map snd* (*remdups* ψ))))
$\rightarrow$ ($\bigsqcap$ (*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* =
(⟨$\bigsqcap$ (*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* δ⟩ $\sqcap$ ⟨$\bigsqcap$ (*map snd* Δ)⟩)
$\sqcap$ (⟨$\bigsqcap$ (*map snd* (*remdups* ψ))⟩ $\sqcup$ ⟨$\bigsqcup$ (*map* ($\bigsqcap \circ$ (*map snd* $\circ$ *remdups*)) Ψ)⟩)))
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 ⋆ ♢ ♠ *Modus-Ponens* **by** *blast*
thus *?thesis* **using** ‹$\delta \notin$ *set* Δ› ‹$\delta \in$ *set* ψ›
by (*simp add*: †)

next
assume $\delta \notin set\ \psi$
have
$\vdash \quad (\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)$
$\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta)) \sqcap \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi)))$
$\rightarrow (\bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi))) \leftrightarrow (\bigsqcap\ (map\ snd\ \Delta) \sqcap \bigsqcap\ (map\ snd\ (remdups\ \psi))))$
$\rightarrow \quad ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi))))$
$\sqcup \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi))$
$\leftrightarrow ((snd\ \delta \sqcap \bigsqcap\ (map\ snd\ \Delta))$
$\sqcap (\bigsqcap\ (map\ snd\ (remdups\ \psi)) \sqcup \bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi)))$
proof $-$
let $?\varphi =$
$(\langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)\rangle$
$\leftrightarrow (((\langle snd\ \delta\rangle \sqcap \langle\bigsqcap\ (map\ snd\ \Delta)\rangle) \sqcap \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi)\rangle))$
$\rightarrow \quad (\langle\bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi)))\rangle$
$\leftrightarrow (\langle\bigsqcap\ (map\ snd\ \Delta)\rangle \sqcap \langle\bigsqcap\ (map\ snd\ (remdups\ \psi))\rangle))$
$\rightarrow \quad ((\langle snd\ \delta\rangle \sqcap \langle\bigsqcap\ (map\ snd\ (remdups\ (\Delta\ @\ \psi)))\rangle)$
$\sqcup \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups \circ (@)\ (\delta\ \#\ \Delta)))\ \Psi)\rangle)$
$\leftrightarrow ((\langle snd\ \delta\rangle \sqcap \langle\bigsqcap\ (map\ snd\ \Delta)\rangle)$
$\sqcap (\langle\bigsqcap\ (map\ snd\ (remdups\ \psi))\rangle \sqcup \langle\bigsqcup\ (map\ (\bigsqcap \circ (map\ snd \circ remdups))\ \Psi)\rangle)))$
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** ‹$\delta \notin set\ \psi$› ‹$\delta \notin set\ \Delta$› **by** (*simp add*: †)
qed
qed
qed
}
then show *?case* **by** *fastforce*
qed
thus *?thesis* **by** *blast*
qed

lemma *remove1-remdups-removeAll*: *remove1 x* (*remdups A*) = *remdups* (*removeAll x A*)
proof (*induct A*)

case *Nil*
then show *?case* **by** *simp*
next
case (*Cons a A*)
then show *?case*
by (*cases a* = *x*, (*simp add*: *Cons*)+)
qed

lemma *mset-remdups*:
assumes *mset A* = *mset B*
shows *mset* (*remdups A*) = *mset* (*remdups B*)
proof −
have ∀ *B*. *mset A* = *mset B* ⟶ *mset* (*remdups A*) = *mset* (*remdups B*)
proof (*induct A*)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons a A*)
{
fix *B*
assume *mset* (*a* # *A*) = *mset B*
hence *mset A* = *mset* (*remove1 a B*)
by (*metis add-mset-add-mset-same-iff*
list.set-intros(*1*)
mset.simps(*2*)
mset-eq-perm
mset-eq-setD
perm-remove)
hence *mset* (*remdups A*) = *mset* (*remdups* (*remove1 a B*))
using *Cons.hyps* **by** *blast*
hence *mset* (*remdups* (*a* # (*remdups A*))) = *mset* (*remdups* (*a* # (*remdups* (*remove1 a B*))))
by (*metis mset-eq-setD set-eq-iff-mset-remdups-eq list.simps*(*15*))
hence *mset* (*remdups* (*a* # (*removeAll a* (*remdups A*))))
= *mset* (*remdups* (*a* # (*removeAll a* (*remdups* (*remove1 a B*)))))
by (*metis insert-Diff-single list.set*(*2*) *set-eq-iff-mset-remdups-eq set-removeAll*)
hence *mset* (*remdups* (*a* # (*remdups* (*removeAll a A*))))
= *mset* (*remdups* (*a* # (*remdups* (*removeAll a* (*remove1 a B*)))))
by (*metis distinct-remdups distinct-remove1-removeAll remove1-remdups-removeAll*)
hence *mset* (*remdups* (*remdups* (*a* # *A*))) = *mset* (*remdups* (*remdups* (*a* # (*remove1 a B*))))
by (*metis* ‹*mset A* = *mset* (*remove1 a B*)›
list.set(*2*)
mset-eq-setD
set-eq-iff-mset-remdups-eq)
hence *mset* (*remdups* (*a* # *A*)) = *mset* (*remdups* (*a* # (*remove1 a B*)))
by (*metis remdups-remdups*)
hence *mset* (*remdups* (*a* # *A*)) = *mset* (*remdups B*)
using ‹*mset* (*a* # *A*) = *mset B*› *mset-eq-setD set-eq-iff-mset-remdups-eq* **by**

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

lemma mset-mset-map-snd-remdups:
  assumes mset (map mset A) = mset (map mset B)
  shows mset (map (mset ∘ (map snd) ∘ remdups) A) = mset (map (mset ∘ (map
snd) ∘ remdups) B)
proof −
  {
    fix B :: ('a × 'b) list list
    fix b :: ('a × 'b) list
    assume b ∈ 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* ∈ *set B* ∧ *a* ∉ *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*)) ⊆# *mset* (*remdups* (*B* @ *A*)))
= (*mset* (*remdups A*) ⊆# *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* Ξ ⊆# *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 :: $('a\ list \times 'a)\ list$ **where** Ψ_0:
mset $\Psi_0 \subseteq\#$ *mset* $(\mathfrak{V}\ \Xi)$
$map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ \Psi_0 = \Psi$
using *mset-sub-map-list-exists* **by** *fastforce*
let $?\Pi_C = \lambda\ (\Delta,\delta)\ \Sigma.\ (map\ ((\#)\ (\Delta,\ \delta))\ \Sigma)\ @\ (map\ ((@)\ (\mathfrak{V}\ \Delta))\ \Sigma)$
let $?T_\Sigma = \lambda\ \Psi.\ foldr\ ?\Pi_C\ \Psi\ [[]]$
let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi_0)$
have I: $\vdash (\Psi :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
proof −
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi_0)$
let $?\Psi' = map\ (\lambda(\Psi,\psi).\ (\Psi :\rightarrow \varphi \rightarrow \psi))\ \Psi_0$
{
fix Ψ :: $('a\ list \times 'a)\ list$
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi)$
let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi)$
have $\vdash (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Sigma) \rightarrow \varphi)$
proof (*induct* Ψ)
case *Nil*
then show *?case* **by** (*simp add*: *biconditional-reflection*)
next
case (*Cons* $\Delta\delta\ \Psi$)
let $?\Delta = fst\ \Delta\delta$
let $?\delta = snd\ \Delta\delta$
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi)$
let $?\Sigma = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi)$
let $?\Sigma_\alpha' = map\ (map\ snd)\ (?T_\Sigma\ ((?\Delta,?\delta)\ \#\ \Psi))$
let $?\Sigma' = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ ((?\Delta,?\delta)\ \#\ \Psi))$
{
fix Δ :: $'a\ list$
fix δ :: $'a$
let $?\Sigma_\alpha' = map\ (map\ snd)\ (?T_\Sigma\ ((\Delta,\delta)\ \#\ \Psi))$
let $?\Sigma' = map\ (map\ snd \circ remdups)\ (?T_\Sigma\ ((\Delta,\delta)\ \#\ \Psi))$
let $?\Phi = map\ (map\ snd \circ (@)\ [(\Delta,\ \delta)])\ (?T_\Sigma\ \Psi)$
let $?\Psi = map\ (map\ snd \circ (@)\ (\mathfrak{V}\ \Delta))\ (?T_\Sigma\ \Psi)$
let $?\Delta = map\ (map\ snd \circ remdups \circ (@)\ [(\Delta,\ \delta)])\ (?T_\Sigma\ \Psi)$
let $?\Omega = map\ (map\ snd \circ remdups \circ (@)\ (\mathfrak{V}\ \Delta))\ (?T_\Sigma\ \Psi)$
have $\vdash (\bigsqcup\ (map\ \bigsqcap\ ?\Phi\ @\ map\ \bigsqcap\ ?\Psi) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Phi) \sqcup \bigsqcup\ (map\ \bigsqcap\ ?\Psi))) \rightarrow$
$(\bigsqcup\ (map\ \bigsqcap\ ?\Delta\ @\ map\ \bigsqcap\ ?\Omega) \leftrightarrow (\bigsqcup\ (map\ \bigsqcap\ ?\Delta) \sqcup \bigsqcup\ (map\ \bigsqcap\ ?\Omega))) \rightarrow$

$(\bigsqcup\ (map \prod\ ?\Phi) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Psi) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Delta) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma))) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Omega) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma))) \rightarrow$
$((\bigsqcup\ (map \prod\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Sigma) \rightarrow \varphi)) \rightarrow$
$((\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \rightarrow \varphi))$

proof −
let $?\varphi =$
$(\langle\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi)\rangle \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Phi)\rangle \sqcup \langle\bigsqcup\ (map \prod\ ?\Psi)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega)\rangle \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Delta)\rangle \sqcup \langle\bigsqcup\ (map \prod\ ?\Omega)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Phi)\rangle \leftrightarrow (\langle\prod\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Psi)\rangle \leftrightarrow (\langle\prod\ \Delta\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Delta)\rangle \leftrightarrow (\langle\prod\ [\delta]\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle)) \rightarrow$
$(\langle\bigsqcup\ (map \prod\ ?\Omega)\rangle \leftrightarrow (\langle\prod\ \Delta\rangle \sqcap \langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle)) \rightarrow$
$((\langle\bigsqcup\ (map \prod\ ?\Sigma_\alpha)\rangle \rightarrow \langle\varphi\rangle) \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Sigma)\rangle \rightarrow \langle\varphi\rangle)) \rightarrow$
$((\langle\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi)\rangle \rightarrow \langle\varphi\rangle) \leftrightarrow (\langle\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega)\rangle \rightarrow \langle\varphi\rangle))$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
moreover
have *map snd* $(\mathfrak{V}\ \Delta) = \Delta$ **by** (*induct* Δ, *auto*)
hence $\vdash \bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Phi) \sqcup \bigsqcup\ (map \prod\ ?\Psi))$
$\vdash \bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta) \sqcup \bigsqcup\ (map \prod\ ?\Omega))$
$\vdash \bigsqcup\ (map \prod\ ?\Phi) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))$
$\vdash \bigsqcup\ (map \prod\ ?\Psi) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma_\alpha))$
$\vdash \bigsqcup\ (map \prod\ ?\Delta) \leftrightarrow (\prod\ [\delta] \sqcap \bigsqcup\ (map \prod\ ?\Sigma))$
$\vdash \bigsqcup\ (map \prod\ ?\Omega) \leftrightarrow (\prod\ \Delta \sqcap \bigsqcup\ (map \prod\ ?\Sigma))$
using *arbitrary-disj-concat-equiv*
extract-inner-concat [**where** $\Delta = [(\Delta, \delta)]$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat [**where** $\Delta = \mathfrak{V}\ \Delta$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat-remdups [**where** $\Delta = [(\Delta, \delta)]$ **and** $\Psi = ?T_\Sigma\ \Psi$]
extract-inner-concat-remdups [**where** $\Delta = \mathfrak{V}\ \Delta$ **and** $\Psi = ?T_\Sigma\ \Psi$]
by *auto*
ultimately have
$\vdash ((\bigsqcup\ (map \prod\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Sigma) \rightarrow \varphi)) \rightarrow$
$(\bigsqcup\ (map \prod\ ?\Phi\ @\ map \prod\ ?\Psi) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Delta\ @\ map \prod\ ?\Omega) \rightarrow \varphi)$
using *Modus-Ponens* **by** *blast*
moreover have $(\#)\ (\Delta, \delta) = (@)\ [(\Delta, \delta)]$ **by** *fastforce*
ultimately have
$\vdash ((\bigsqcup\ (map \prod\ ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \prod\ ?\Sigma) \rightarrow \varphi)) \rightarrow$

$((\bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha{}') \to \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ ?\Sigma') \to \varphi))$
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 $\vdash (\!|\ ?\varphi\ |\!)$ **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 $?\Psi' = map\ (\lambda(\Psi,\psi).\ (\Psi :\to \varphi \to \psi))\ \Psi_0$
let $?\Sigma_\alpha = map\ (map\ snd)\ (?T_\Sigma\ \Psi_0)$
{
fix $\Xi :: 'a\ list$
have $map\ snd\ (\mathfrak{V}\ \Xi) = \Xi$ **by** (*induct* 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 †:
$map\ (map\ snd)\ (?T_\Sigma\ (\psi_0\ \#\ \Psi_0)) = map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha\ @\ map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha$
$map\ (\lambda(\Psi,\psi).\ (\Psi :\to \varphi \to \psi))\ (\psi_0\ \#\ \Psi_0) = ?\Xi :\to \varphi \to ?\delta\ \#\ ?\Psi'$
by (*simp add*: *case-prod-beta'*)+
have A: $\vdash (?\Psi' :\to \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha) \to \varphi)$ **using** *Cons.hyps* **by** *auto*
have B: $\vdash (?\Xi :\to \varphi) \leftrightarrow (\bigsqcap\ ?\Xi \to \varphi)$
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
$\vdash \quad (?\Psi' :\rightarrow \varphi) \leftrightarrow (\bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha) \rightarrow \varphi)$
$\rightarrow \quad (?\Xi :\rightarrow \varphi) \leftrightarrow (\bigsqcap\ ?\Xi \rightarrow \varphi)$
$\rightarrow \quad \bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))$
$\leftrightarrow (\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \sqcup \bigsqcup\ (map \bigsqcap\ (map\ ((@)\ ?\Xi)$
$?\Sigma_\alpha)))$
$\rightarrow \quad \bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)) \leftrightarrow (?\delta \sqcap \bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha))$
$\rightarrow \quad \bigsqcup\ (map \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha)) \leftrightarrow (\bigsqcap\ ?\Xi \sqcap \bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha))$
$\rightarrow \quad ((?\Xi :\rightarrow \varphi \rightarrow ?\delta) \rightarrow ?\Psi' :\rightarrow \varphi)$
$\leftrightarrow (\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))$
$\rightarrow \varphi)$
proof $-$
let $?\varphi =$
$\langle ?\Psi' :\rightarrow \varphi\rangle \leftrightarrow (\langle\bigsqcup\ (map \bigsqcap\ ?\Sigma_\alpha)\rangle \rightarrow \langle\varphi\rangle)$
$\rightarrow \quad \langle(?\Xi :\rightarrow \varphi)\rangle \leftrightarrow (\langle\bigsqcap\ ?\Xi\rangle \rightarrow \langle\varphi\rangle)$
$\rightarrow \quad \langle\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map \bigsqcap\ (map\ ((@)\ ?\Xi)$
$?\Sigma_\alpha))\rangle$
$\leftrightarrow (\langle\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha))\rangle \sqcup \langle\bigsqcup\ (map \bigsqcap\ (map\ ((@)$
$?\Xi)\ ?\Sigma_\alpha))\rangle)$
$\rightarrow \quad \langle\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha))\rangle \leftrightarrow (\langle ?\delta\rangle \sqcap \langle\bigsqcup\ (map \bigsqcap$
$?\Sigma_\alpha)\rangle)$
$\rightarrow \quad \langle\bigsqcup\ (map \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))\rangle \leftrightarrow (\langle\bigsqcap\ ?\Xi\rangle \sqcap \langle\bigsqcup\ (map$
$\bigsqcap\ ?\Sigma_\alpha)\rangle)$
$\rightarrow \quad ((\langle ?\Xi :\rightarrow \varphi\rangle \rightarrow \langle ?\delta\rangle) \rightarrow \langle ?\Psi' :\rightarrow \varphi\rangle)$
$\leftrightarrow (\langle\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map \bigsqcap\ (map\ ((@)\ ?\Xi)$
$?\Sigma_\alpha))\rangle \rightarrow \langle\varphi\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
hence
$\vdash \quad ((?\Xi :\rightarrow \varphi \rightarrow ?\delta) \rightarrow ?\Psi' :\rightarrow \varphi)$
$\leftrightarrow (\bigsqcup\ (map \bigsqcap\ (map\ ((\#)\ ?\delta)\ ?\Sigma_\alpha)\ @\ map \bigsqcap\ (map\ ((@)\ ?\Xi)\ ?\Sigma_\alpha))$
$\rightarrow \varphi)$
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* Ψ_0 = *mset* (*?*$\mathcal{S}$ Ψ_0) **by** *simp*

have $\forall$ Φ. *mset* Ψ_0 = *mset* Φ $\longrightarrow$ *mset* (*map mset* (*?*T_Σ Ψ_0)) = *mset* (*map mset* (*?*T_Σ Φ))
proof (*induct* Ψ_0)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ψ Ψ_0)
obtain Δ δ **where** ψ = (Δ,δ) **by** *fastforce*
{
fix Φ
assume *mset* (ψ # Ψ_0) = *mset* Φ
hence *mset* Ψ_0 = *mset* (*remove1* ψ Φ)
by (*simp add*: *union-single-eq-diff*)
have ψ $\in$ *set* Φ **using** ‹*mset* (ψ # Ψ_0) = *mset* Φ›
using *mset-eq-setD* **by** *fastforce*
hence *mset* (*map mset* (*?*T_Σ Φ)) = *mset* (*map mset* (*?*T_Σ (ψ # (*remove1* ψ Φ))))
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
then show *?case* **proof** (*cases* φ = ψ)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
let *?*Σ' = *?*T_Σ (ψ # (*remove1* ψ Φ))
have †: *mset* (*map mset ?*Σ') = *mset* (*map mset* (*?*T_Σ Φ))
using *Cons False* **by** *simp*
obtain Δ' δ'
where φ = (Δ',δ')
by *fastforce*
let *?*Σ = *?*T_Σ (*remove1* ψ Φ)
let *?*$\mathfrak{m}$ = *image-mset mset*
have
mset (*map mset* (*?*T_Σ (ψ # *remove1* ψ (φ # Φ)))) =
mset (*map mset* (*?*Π_C ψ (*?*Π_C φ *?*Σ)))
using *False* **by** *simp*
hence *mset* (*map mset* (*?*T_Σ (ψ # *remove1* ψ (φ # Φ)))) =
(*?*$\mathfrak{m}$ $\circ$ (*image-mset* ((#) ψ) $\circ$ *image-mset* ((#) φ))) (*mset ?*Σ) +
(*?*$\mathfrak{m}$ $\circ$ (*image-mset* ((#) ψ) $\circ$ *image-mset* ((@) ($\mathfrak{V}$ Δ')))) (*mset ?*Σ) +
(*?*$\mathfrak{m}$ $\circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ)) $\circ$ *image-mset* ((#) φ))) (*mset ?*Σ) +
(*?*$\mathfrak{m}$ $\circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ)) $\circ$ *image-mset* ((@) ($\mathfrak{V}$ Δ')))) (*mset ?*Σ)

using ‹$\psi = (\Delta,\delta)$› ‹$\varphi = (\Delta',\delta')$›
by (*simp add*: *multiset.map-comp*)
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
($?\mathfrak{m} \circ$ (*image-mset* ((#) φ) $\circ$ *image-mset* ((#) ψ))) (*mset* $?\Sigma$) +
($?\mathfrak{m} \circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ')) $\circ$ *image-mset* ((#) ψ))) (*mset* $?\Sigma$) +
($?\mathfrak{m} \circ$ (*image-mset* ((#) φ) $\circ$ *image-mset* ((@) ($\mathfrak{V}$ Δ)))) (*mset* $?\Sigma$) +
($?\mathfrak{m} \circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ')) $\circ$ *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)))) =
($?\mathfrak{m} \circ$ (*image-mset* ((#) φ))) (*mset* $?\Sigma'$) +
($?\mathfrak{m} \circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ')))) (*mset* $?\Sigma'$)
using ‹$\psi = (\Delta,\delta)$›
by (*simp add*: *multiset.map-comp*)
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
image-mset ((+) {#φ#}) (*mset* (*map mset* $?\Sigma'$)) +
image-mset ((+) (*mset* ($\mathfrak{V}$ Δ'))) (*mset* (*map mset* $?\Sigma'$))
by (*simp add*: *image-mset-cons-homomorphism*
image-mset-append-homomorphism)
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
image-mset ((+) {#φ#}) (*mset* (*map mset* ($?T_\Sigma$ Φ))) +
image-mset ((+) (*mset* ($\mathfrak{V}$ Δ'))) (*mset* (*map mset* ($?T_\Sigma$ Φ)))
using † **by** *auto*
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # *remove1* ψ (φ # Φ)))) =
($?\mathfrak{m} \circ$ (*image-mset* ((#) φ))) (*mset* ($?T_\Sigma$ Φ)) +
($?\mathfrak{m} \circ$ (*image-mset* ((@) ($\mathfrak{V}$ Δ')))) (*mset* ($?T_\Sigma$ Φ))
by (*simp add*: *image-mset-cons-homomorphism*
image-mset-append-homomorphism)
thus *?thesis* **using** ‹$\varphi = (\Delta',\delta')$› **by** (*simp add*: *multiset.map-comp*)
qed
qed
hence *image-mset mset* (*image-mset* ((#) ψ) (*mset* ($?T_\Sigma$ (*remove1* ψ Φ)))) +
image-mset mset (*image-mset* ((@) ($\mathfrak{V}$ Δ)) (*mset* ($?T_\Sigma$ (*remove1* ψ Φ))))
= *image-mset mset* (*mset* ($?T_\Sigma$ Φ))
by (*simp add*: ‹$\psi = (\Delta,\delta)$› *multiset.map-comp*)
hence
image-mset ((+) {# ψ #}) (*image-mset mset* (*mset* ($?T_\Sigma$ (*remove1* ψ Φ)))) +
image-mset ((+) (*mset* ($\mathfrak{V}$ Δ))) (*image-mset mset* (*mset* ($?T_\Sigma$ (*remove1* ψ Φ))))
= *image-mset mset* (*mset* ($?T_\Sigma$ Φ))

by (*simp add*: *image-mset-cons-homomorphism image-mset-append-homomorphism*)
hence
image-mset ((+) {# ψ #}) (*image-mset mset* (*mset* ($?T_\Sigma$ Ψ_0))) +
image-mset ((+) (*mset* ($\mathfrak{V}$ Δ))) (*image-mset mset* (*mset* ($?T_\Sigma$ Ψ_0)))
= *image-mset mset* (*mset* ($?T_\Sigma$ Φ))
using *Cons* ‹*mset* Ψ_0 = *mset* (*remove1* ψ Φ)›
by *fastforce*
hence
image-mset mset (*image-mset* ((#) ψ) (*mset* ($?T_\Sigma$ Ψ_0))) +
image-mset mset (*image-mset* ((@) ($\mathfrak{V}$ Δ)) (*mset* ($?T_\Sigma$ Ψ_0)))
= *image-mset mset* (*mset* ($?T_\Sigma$ Φ))
by (*simp add*: *image-mset-cons-homomorphism image-mset-append-homomorphism*)
hence *mset* (*map mset* ($?T_\Sigma$ (ψ # Ψ_0))) = *mset* (*map mset* ($?T_\Sigma$ Φ))
by (*simp add*: ‹ψ = (Δ,δ)› *multiset.map-comp*)
}
then show *?case* **by** *blast*
qed
hence *mset* (*map mset* ($?T_\Sigma$ Ψ_0)) = *mset* (*map mset* ($?T_\Sigma$ ($?\mathcal{S}$ Ψ_0)))
using ‹*mset* Ψ_0 = *mset* ($?\mathcal{S}$ Ψ_0)› **by** *blast*
hence *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) ($?T_\Sigma$ Ψ_0))
= *mset* (*map* (*mset* ∘ (*map snd*) ∘ *remdups*) ($?T_\Sigma$ ($?\mathcal{S}$ Ψ_0)))
using *mset-mset-map-snd-remdups* **by** *blast*
hence *mset* (*map mset* $?\Sigma$) = *mset* (*map mset* $?\Sigma'$)
by (*simp add*: *fun.map-comp*)
hence *set* (*map mset* $?\Sigma$) = *set* (*map mset* $?\Sigma'$)
using *mset-eq-setD* **by** *blast*
hence $\forall$ $\sigma \in$ *set* $?\Sigma$. $\exists$ $\sigma' \in$ *set* $?\Sigma'$. *mset* σ = *mset* σ'
by *fastforce*
hence $\forall$ $\sigma \in$ *set* $?\Sigma$. $\exists$ $\sigma' \in$ *set* $?\Sigma'$. *length* σ = *length* σ'
using *mset-eq-length* **by** *blast*
have *mset* ($?\mathcal{S}$ Ψ_0) $\subseteq$# *mset* ($\mathfrak{V}$ Ξ)
by (*simp add*: Ψ_0(*1*))
{
fix *n*
have $\forall$ Ψ. *mset* Ψ $\subseteq$# *mset* ($\mathfrak{V}$ Ξ) $\longrightarrow$
sorted (*map* (− $?\mathcal{M}$) Ψ) $\longrightarrow$
length Ψ = *n* $\longrightarrow$
($\forall$ $\sigma' \in$ *set* (*map* (*map snd* ∘ *remdups*) ($?T_\Sigma$ Ψ)). *length* σ' + *1*
$\geq$ *n*)
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Ψ :: ('*a list* × '*a*) *list*
assume *A*: *mset* Ψ $\subseteq$# *mset* ($\mathfrak{V}$ Ξ)
and *B*: *sorted* (*map* (− $?\mathcal{M}$) Ψ)
and *C*: *length* Ψ = *n* + *1*

obtain Δ δ **where** $(\Delta, \delta) = hd\ \Psi$
  using *prod.collapse* **by** *blast*
let $?\Psi' = tl\ \Psi$
have $mset\ ?\Psi' \subseteq\#\ mset\ (\mathfrak{V}\ \Xi)$ **using** *A*
by (*induct* Ψ, *simp*, *simp*, *meson mset-subset-eq-insertD subset-mset-def*)
moreover
have $sorted\ (map\ (- ?\mathcal{M})\ (tl\ \Psi))$
  using *B*
  by (*simp add*: *map-tl sorted-tl*)
moreover have $length\ ?\Psi' = n$ **using** *C*
  by *simp*
ultimately have $\star$: $\forall\ \sigma' \in set\ (map\ (map\ snd \circ remdups)\ (?T_\Sigma\ ?\Psi'))$.
$length\ \sigma' + 1 \geq n$
  using *Suc*
  by *blast*
from *C* **have** $\Psi = (\Delta, \delta)\ \#\ ?\Psi'$
  by (*metis* $\langle(\Delta, \delta) = hd\ \Psi\rangle$
    One-nat-def
    add-is-0
    list.exhaust-sel
    list.size(*3*)
    nat.simps(*3*))
have $distinct\ ((\Delta, \delta)\ \#\ ?\Psi')$
  using *A* $\langle\Psi = (\Delta, \delta)\ \#\ ?\Psi'\rangle$
    core-optimal-pre-witness-distinct
    mset-distinct-msub-down
  by *fastforce*
hence $set\ ((\Delta, \delta)\ \#\ ?\Psi') \subseteq set\ (\mathfrak{V}\ \Xi)$
  by (*metis A* $\langle\Psi = (\Delta, \delta)\ \#\ ?\Psi'\rangle$
    Un-iff
    mset-le-perm-append
    perm-set-eq set-append
    subsetI)
hence $\forall\ (\Delta', \delta') \in set\ ?\Psi'.\ (\Delta, \delta) \neq (\Delta', \delta')$
  $\forall\ (\Delta', \delta') \in set\ (\mathfrak{V}\ \Xi).\ ((\Delta, \delta) \neq (\Delta', \delta')) \longrightarrow (length\ \Delta \neq length$
$\Delta')$
  $set\ ?\Psi' \subseteq set\ (\mathfrak{V}\ \Xi)$
  using *core-optimal-pre-witness-length-iff-eq* [**where** Ψ=Ξ]
    $\langle distinct\ ((\Delta, \delta)\ \#\ ?\Psi')\rangle$
  by *auto*
hence $\forall\ (\Delta', \delta') \in set\ ?\Psi'.\ length\ \Delta \neq length\ \Delta'$
  $sorted\ (map\ (- ?\mathcal{M})\ ((\Delta, \delta)\ \#\ ?\Psi'))$
  using *B* $\langle\Psi = (\Delta, \delta)\ \#\ ?\Psi'\rangle$
  by (*fastforce*, *auto*)
hence $\forall\ (\Delta', \delta') \in set\ ?\Psi'.\ length\ \Delta > length\ \Delta'$
  by *fastforce*
{
  fix σ' :: *'a list*
  assume $\sigma' \in set\ (map\ (map\ snd \circ remdups)\ (?T_\Sigma\ \Psi))$

hence $\sigma' \in$ *set* (*map* (*map snd* $\circ$ *remdups*) ($?T_\Sigma$ ((Δ, δ) # $?\Psi'$)))
  using ‹Ψ = (Δ, δ) # $?\Psi'$›
  by *simp*
from *this* **obtain** ψ **where** ψ:
  $\psi \in$ *set* ($?T_\Sigma$ $?\Psi'$)
  σ' = (*map snd* $\circ$ *remdups* $\circ$ (#) (Δ, δ)) ψ $\vee$
   σ' = (*map snd* $\circ$ *remdups* $\circ$ (@) ($\mathfrak{V}$ Δ)) ψ
  by *fastforce*
hence *length* $\sigma' \geq n$
proof (*cases* σ' = (*map snd* $\circ$ *remdups* $\circ$ (#) (Δ, δ)) ψ)
  case *True*
  {
    fix Ψ :: ($'a$ *list* $\times$ $'a$) *list*
    fix n :: *nat*
    assume $\forall$ (Δ, δ) $\in$ *set* Ψ. $n >$ *length* Δ
    hence $\forall$ $\sigma \in$ *set* ($?T_\Sigma$ Ψ). $\forall$ (Δ, δ) $\in$ *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*: $\forall$ $\sigma \in$ *set* ($?T_\Sigma$ Ψ). $\forall$ (Δ', δ') $\in$ *set* σ. $n >$ *length* Δ'
        using *Cons* **by** *simp*
      {
        fix σ :: ($'a$ *list* $\times$ $'a$) *list*
        fix ψ' :: $'a$ *list* $\times$ $'a$
        assume *1*: $\sigma \in$ *set* ($?T_\Sigma$ (ψ # Ψ))
           and *2*: $\psi' \in$ *set* σ
        obtain Δ' δ' **where** ψ' = (Δ', δ')
          by *fastforce*
         have *3*: $\sigma \in$ (#) (Δ, δ) ‘ *set* ($?T_\Sigma$ Ψ) $\vee$ $\sigma \in$ (@) ($\mathfrak{V}$ Δ) ‘ *set*
($?T_\Sigma$ Ψ)
          using *1* ‹ψ = (Δ, δ)› **by** *simp*
        have $n >$ *length* Δ'
        proof (*cases* $\sigma \in$ (#) (Δ, δ) ‘ *set* ($?T_\Sigma$ Ψ))
          case *True*
          from *this* **obtain** σ' **where**
            set σ = *insert* (Δ, δ) (*set* σ')
            $\sigma' \in$ *set* ($?T_\Sigma$ Ψ)
            by *auto*
          then show *?thesis*
            using *0* ‹$\psi' \in$ *set* σ› ‹ψ' = (Δ', δ')› ‹$n >$ *length* Δ›
            by *auto*
        next
          case *False*
          from *this* **and** *3* **obtain** σ' **where** σ':

set σ = *set* ($\mathfrak{V}$ Δ) $\cup$ (*set* σ')
$\sigma' \in$ *set* (*?T*$_\Sigma$ Ψ)
by *auto*
have $\forall$ (Δ', δ') $\in$ *set* ($\mathfrak{V}$ Δ). *length* Δ > *length* Δ'
by (*metis* (*mono-tags*, *lifting*)
case-prodI2
core-optimal-pre-witness-nonelement
not-le)
hence $\forall$ (Δ', δ') $\in$ *set* ($\mathfrak{V}$ Δ). *n* > *length* Δ'
using ‹*n* > *length* Δ› **by** *auto*
then show *?thesis* **using** *0* σ' ‹$\psi' \in$ *set* σ› ‹ψ' = (Δ', δ')› **by**
fastforce
qed
hence *n* > *length* (*fst* ψ') **using** ‹ψ' = (Δ', δ')› **by** *fastforce*
}
then show *?case* **by** *fastforce*
qed
}
hence $\forall$ $\sigma \in$ *set* (*?T*$_\Sigma$ *?*Ψ'). $\forall$ (Δ', δ') $\in$ *set* σ. *length* Δ > *length*
Δ'
using ‹$\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* Δ > *length* Δ'›
by *blast*
then show *?thesis* **using** *True* $\star$ ψ(*1*) **by** *fastforce*
next
case *False*
have $\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* $\Delta \geq$ *length* Δ'
using ‹$\forall$ (Δ', δ') $\in$ *set ?*Ψ'. *length* Δ > *length* Δ'›
by *auto*
hence $\forall$ (Δ', δ') $\in$ *set* Ψ. *length* $\Delta \geq$ *length* Δ'
using ‹Ψ = (Δ, δ) # *?*Ψ'›
by (*metis case-prodI2 eq-iff prod.sel*(*1*) *set-ConsD*)
hence *length* Δ + *1* $\geq$ *length* Ψ
using *A core-optimal-pre-witness-pigeon-hole*
by *fastforce*
hence *length* $\Delta \geq$ *n*
using *C*
by *simp*
have *length* Δ = *length* ($\mathfrak{V}$ Δ)
by (*induct* Δ, *simp*+)
hence *length* (*remdups* ($\mathfrak{V}$ Δ)) = *length* ($\mathfrak{V}$ Δ)
by (*simp add*: *core-optimal-pre-witness-distinct*)
hence *length* (*remdups* ($\mathfrak{V}$ Δ)) $\geq$ *n*
using ‹*length* Δ = *length* ($\mathfrak{V}$ Δ)› ‹*n* $\leq$ *length* Δ›
by *linarith*
have *mset* (*remdups* ($\mathfrak{V}$ Δ @ ψ)) = *mset* (*remdups* (ψ @ $\mathfrak{V}$ Δ))
by (*simp add*: *mset-remdups*)
hence *length* (*remdups* ($\mathfrak{V}$ Δ @ ψ)) $\geq$ *length* (*remdups* ($\mathfrak{V}$ Δ))
by (*metis le-cases length-sub-mset mset-remdups-append-msub*
size-mset)

```
          hence length (remdups (𝔙 Δ @ ψ)) ≥ n
            using ‹n ≤ length (remdups (𝔙 Δ))› dual-order.trans by blast
          thus ?thesis using False ψ(2)
            by simp
        qed
      }
      hence ∀ σ' ∈ set (map (map snd ∘ remdups) (?T_Σ Ψ)). length σ' ≥ n
        by blast
    }
    then show ?case by fastforce
  qed
}
hence ∀ σ' ∈ set ?Σ'. length σ' + 1 ≥ length (?𝒮 Ψ0)
  using ‹mset (?𝒮 Ψ0) ⊆# mset (𝔙 Ξ)›
  by fastforce
hence ∀ σ' ∈ set ?Σ'. length σ' + 1 ≥ length Ψ0 by simp
hence ∀ σ ∈ set ?Σ. length σ + 1 ≥ length Ψ0
  using ‹∀ σ ∈ set ?Σ. ∃ σ' ∈ 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 $\sigma \in (@)\ (\mathfrak{V}\ \Delta)$ ‘ *set* $(?T_\Sigma\ \Psi_0)$ **using** $\star$ **by** *simp*
moreover have *set* $(\mathfrak{V}\ \Delta) \subseteq$ *set* $(\mathfrak{V}\ \Xi)$
using *core-optimal-pre-witness-element-inclusion* ‹$(\Delta, \delta) \in$ *set* $(\mathfrak{V}\ \Xi)$›
by *fastforce*
ultimately show *?thesis*
using ‹$\forall\ \sigma \in$ *set* $(?T_\Sigma\ \Psi_0)$. *set* $\sigma \subseteq$ *set* $(\mathfrak{V}\ \Xi)$›
by *force*
qed
}
hence $\forall \sigma \in (\#)\ (\Delta, \delta)$ ‘ *set* $(?T_\Sigma\ \Psi_0) \cup (@)\ (\mathfrak{V}\ \Delta)$ ‘ *set* $(?T_\Sigma\ \Psi_0)$. *set* σ $\subseteq$ *set* $(\mathfrak{V}\ \Xi)$
by *auto*
thus *?case* **using** ‹$\psi = (\Delta, \delta)$› **by** *simp*
qed
hence $\forall\ \sigma \in$ *set* $(?T_\Sigma\ \Psi_0)$. *mset* (*remdups* σ) $\subseteq\#$ *mset* (*remdups* $(\mathfrak{V}\ \Xi)$)
using *mset-remdups-set-sub-iff* **by** *blast*
hence $\forall\ \sigma \in$ *set* $?\Sigma$. *mset* $\sigma \subseteq\#$ *mset* (*map snd* $(\mathfrak{V}\ \Xi)$)
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* Σ_0. *mset* $\sigma \subseteq\#$ *mset* $\Xi \wedge$ *length* $\sigma + 1 \geq$ *length* Ψ
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 $\vdash$ (Φ @ Ψ) $:\rightarrow \varphi \leftrightarrow$ ($\bigsqcup$ (*map* $\bigsqcap$ *?*Σ) $\rightarrow \varphi$)
using *Modus-Ponens biconditional-transitivity-rule*
by *blast*
moreover
{
fix σ
assume $\sigma \in$ *set ?*Σ
from *this* **obtain** σ_0 **where** σ_0: $\sigma = \Phi$ @ σ_0 $\sigma_0 \in$ *set* Σ_0 **by** (*simp*, *blast*)
hence *mset* $\sigma_0 \subseteq\#$ *mset* Ξ **using** $\Sigma_0(2)$ **by** *blast*
hence *mset* $\sigma \subseteq\#$ *mset* (Φ @ Ξ) **using** $\sigma_0(1)$ **by** *simp*
hence *mset* $\sigma \subseteq\#$ *mset* Γ **using** *assms*(*1*) *assms*(*2*)
by (*simp*, *meson subset-mset.dual-order.trans subset-mset.le-diff-conv2*)
moreover
have *length* σ + *1* $\geq$ *length* (Φ @ Ψ) **using** $\Sigma_0(2)$ σ_0 **by** *simp*
ultimately have *mset* $\sigma \subseteq\#$ *mset* Γ *length* σ + *1* $\geq$ *length* (Φ @ Ψ) **by** *auto*
}
ultimately
show *?thesis* **by** *blast*
qed

lemma (**in** *Classical-Propositional-Logic*) *unproving-core-optimal-witness*:
assumes $\neg \vdash \varphi$
shows *0* < ($\parallel \Gamma \parallel_\varphi$)
= ($\exists$ Σ. *mset* (*map snd* Σ) $\subseteq\#$ *mset* Γ $\wedge$
map (*uncurry* ($\sqcup$)) Σ $:\vdash \varphi$ $\wedge$
1 + ($\parallel$ *map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ $\parallel_\varphi$) = $\parallel \Gamma \parallel_\varphi$)
proof (*rule iffI*)
assume *0* < $\parallel \Gamma \parallel_\varphi$
from *this* **obtain** Ξ **where** Ξ: $\Xi \in \mathcal{C}$ Γ φ *length* Ξ < *length* Γ
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* < $\parallel \Gamma \parallel_\varphi$›
less-not-refl
list.exhaust
list.set-intros(*1*)
list.size(*3*)
complement-core-size-intro)
let *?*Σ = $\mathfrak{W}$ φ (ψ # Ξ)
let *?*Σ_A = $\mathfrak{W}_\sqcup$ φ (ψ # Ξ)
let *?*Σ_B = $\mathfrak{W}_\rightarrow$ φ (ψ # Ξ)
have $\diamondsuit$: *mset* (ψ # Ξ) $\subseteq\#$ *mset* Γ
ψ # Ξ $:\vdash \varphi$
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 ?*Σ = ψ # Ξ **by** (*induct* Ξ, *simp+*)
ultimately have ?Σ_A :$\vdash$ φ
            mset (*map snd ?*Σ) $\subseteq$# *mset* Γ
  using *core-optimal-witness-deduction*
        list-deduction-def weak-biconditional-weaken
  by (*metis+*)
moreover
{
  let ?Γ' = ?Σ_B @ Γ $\ominus$ *map snd ?*Σ
  have *A*: *length* ?Σ_B = *1* + *length* Ξ
    by (*induct* Ξ, *simp+*)
  have *B*: ?Σ_B $\in$ $\mathcal{C}$?Γ' φ
  proof −
    have $\neg$?Σ_B :$\vdash$ φ
      by (*metis* (*no-types*, *lifting*)
            Ξ(*1*) ‹?Σ_A :$\vdash$ φ›
            Modus-Ponens list-deduction-def
            optimal-witness-split-identity
            unproving-core-def
            mem-Collect-eq)
    moreover have *mset* ?Σ_B $\subseteq$# *mset* ?Γ'
      by *simp*
    hence $\forall$ Ψ. *mset* Ψ $\subseteq$# *mset* ?Γ' $\longrightarrow$ $\neg$ Ψ :$\vdash$ φ $\longrightarrow$ *length* Ψ $\leq$ *length* ?Σ_B
    proof −
      have $\forall$ Ψ $\in$ $\mathcal{C}$?Γ' φ. *length* Ψ = *length* ?Σ_B
      proof (*rule ccontr*)
        assume $\neg$ ($\forall$ Ψ $\in$ $\mathcal{C}$?Γ' φ. *length* Ψ = *length* ?Σ_B)
        from *this* **obtain** Ψ **where**
          Ψ: Ψ $\in$ $\mathcal{C}$?Γ' φ
             length Ψ $\neq$ *length* ?Σ_B
          by *blast*
        have *length* Ψ $\geq$ *length* ?Σ_B
          using Ψ(*1*)
              ‹$\neg$?Σ_B :$\vdash$ φ›
              ‹*mset* ?Σ_B $\subseteq$# *mset* ?Γ'›
          unfolding *unproving-core-def*
          by *blast*
        hence *length* Ψ > *length* ?Σ_B
          using Ψ(*2*)
          by *linarith*
        have *length* Ψ = *length* (Ψ $\ominus$?Σ_B) + *length* (Ψ $\cap$?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* $\sigma \subseteq\#$ *mset* Γ

length σ + *1* $\geq$ *length* (*?A* @ *?B*)

hence *length* σ + *1* $\geq$ *length* Ψ

using ‹*length* Ψ = *length ?A* + *length ?B*›

by *simp*

hence *length* σ + *1* > *length* $?\Sigma_B$

using ‹*length* Ψ > *length* $?\Sigma_B$› **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* $\sigma \leq$ *length* Ξ

using ‹*mset* $\sigma \subseteq\#$ *mset* Γ› Ξ(*1*)

unfolding *unproving-core-def*

by *blast*

thus *False* **using** ‹*length* σ > *length* Ξ› **by** *linarith*

qed

}

moreover

have *mset* $\Psi \subseteq\#$ *mset* $?\Gamma'$

¬ Ψ :⊢ φ

∀ Φ. *mset* $\Phi \subseteq\#$ *mset* $?\Gamma' \wedge$ ¬ Φ :⊢ $\varphi \longrightarrow$ *length* $\Phi \leq$ *length* Ψ

using Ψ(*1*) *unproving-core-def* **by** *blast+*

hence *mset ?A* $\subseteq\#$ *mset* ($\Gamma \ominus$ *map snd* $?\Sigma$)

by (*simp add*: *add.commute subset-eq-diff-conv*)

hence *mset ?A* $\subseteq\#$ *mset* ($\Gamma \ominus (\psi \# \Xi)$)

using ‹*map snd* $?\Sigma = \psi \# \Xi$› **by** *metis*

moreover

have *mset ?B* $\subseteq\#$ *mset* ($\mathfrak{W}_{\rightarrow}$ φ ($\psi \# \Xi$))

using *list-intersect-right-project* **by** *blast*

ultimately obtain Σ **where** Σ: ⊢ ((*?A* @ *?B*) :→ φ) ↔ ($\bigsqcup$ (*map* $\bigsqcap$ Σ) → φ)

∀ σ∈*set* Σ. σ :⊢ φ

using ◇ *optimal-witness-list-intersect-biconditional*

by *metis*

hence ⊢ $\bigsqcup$ (*map* $\bigsqcap$ 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** $\Psi(1)$ **unfolding** *unproving-core-def* **by** *blast*
qed
moreover have $\exists\ \Psi.\ \Psi \in \mathcal{C}\ ?\Gamma'\ \varphi$
using *assms unproving-core-existence* **by** *blast*
ultimately show *?thesis*
using *unproving-core-def*
by *fastforce*
qed
ultimately show *?thesis*
unfolding *unproving-core-def*
by *fastforce*
qed
have *C*: $\forall \Xi\ \Gamma\ \varphi.\ \Xi \in \mathcal{C}\ \Gamma\ \varphi \longrightarrow$ *length* $\Xi = |\ \Gamma\ |_\varphi$
using *core-size-intro* **by** *blast*
then have *D*: *length* $\Xi = |\ \Gamma\ |_\varphi$
using ‹$\Xi \in \mathcal{C}\ \Gamma\ \varphi$› **by** *blast*
have
$\forall (\Sigma :: 'a\ list)\ \Gamma\ n.$ ($\neg$ *mset* $\Sigma \subseteq\#$ *mset* $\Gamma \vee$ *length* $(\Gamma \ominus \Sigma) \neq n) \vee$ *length* Γ $= n +$ *length* Σ
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 ‹*map snd* $(\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi)) = \psi\ \#\ \Xi$› ‹*mset* $(\psi\ \#\ \Xi) \subseteq\#$ *mset* Γ› **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* ‹*map snd* $(\mathfrak{W}\ \varphi\ (\psi\ \#\ \Xi)) = \psi\ \#\ \Xi$› *complement-core-size-def* **by** *force*
}
ultimately
show $\exists\ \Sigma.$ *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma\ \wedge$
map (*uncurry* $(\sqcup)$) Σ :⊢ $\varphi\ \wedge$
$1 + (\|$ *map* (*uncurry* $(\rightarrow)$) Σ @ $\Gamma \ominus$ *map snd* $\Sigma\ \|_\varphi) = \|\ \Gamma\ \|_\varphi$
by *metis*
next
assume $\exists\ \Sigma.$ *mset* (*map snd* Σ) $\subseteq\#$ *mset* $\Gamma\ \wedge$
map (*uncurry* $(\sqcup)$) Σ :⊢ $\varphi\ \wedge$
$1 + (\|$ *map* (*uncurry* $(\rightarrow)$) Σ @ $\Gamma \ominus$ *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
$\mathfrak{U}$ - [] = []

| $\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi) = (\textit{case find}\ (\lambda\ \sigma.\ \xi = \textit{snd}\ \sigma)\ \Sigma\ \textit{of}$
$\quad\quad\quad\quad\quad\quad \textit{None} \Rightarrow \mathfrak{U}\ \Sigma\ \Xi$
$\quad\quad\quad\quad\quad\ \mid \textit{Some}\ \sigma \Rightarrow \sigma\ \#\ (\mathfrak{U}\ (\textit{remove1}\ \sigma\ \Sigma)\ \Xi))$

lemma (**in** *Minimal-Logic*) *core-witness-right-msub*:
  $\textit{mset}\ (\textit{map snd}\ (\mathfrak{U}\ \Sigma\ \Xi)) \subseteq\#\ \textit{mset}\ \Xi$
proof −
  have $\forall\ \Sigma.\ \textit{mset}\ (\textit{map snd}\ (\mathfrak{U}\ \Sigma\ \Xi)) \subseteq\#\ \textit{mset}\ \Xi$
  proof (*induct* Ξ)
    case *Nil*
    then show *?case* **by** *simp*
  next
    case (*Cons* ξ Ξ)
    {
      fix Σ
      have $\textit{mset}\ (\textit{map snd}\ (\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi))) \subseteq\#\ \textit{mset}\ (\xi\ \#\ \Xi)$
      proof (*cases find* $(\lambda\ \sigma.\ \xi = \textit{snd}\ \sigma)\ \Sigma$)
        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 $\sigma = \textit{this}$
        hence $\xi = \textit{snd}\ \sigma$
          by (*meson find-Some-predicate*)
        moreover
        have $\sigma \in \textit{set}\ \Sigma$
        using σ
        proof (*induct* Σ)
          case *Nil*
          then show *?case* **by** *simp*
        next
          case (*Cons* σ' Σ)
          then show *?case*
            by (*cases* $\xi = \textit{snd}\ \sigma'$, *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*:
  $\textit{mset}\ (\mathfrak{U}\ \Sigma\ \Xi) \subseteq\#\ \textit{mset}\ \Sigma$
proof −

have $\forall\ \Sigma.$ *mset* $(\mathfrak{U}\ \Sigma\ \Xi) \subseteq\#$ *mset* Σ
proof (*induct* Ξ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
{
fix Σ
have *mset* $(\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi)) \subseteq\#$ *mset* Σ
proof (*cases find* $(\lambda\ \sigma.\ \xi = snd\ \sigma)\ \Sigma$)
case *None*
then show *?thesis* **using** *Cons.hyps* **by** *simp*
next
case (*Some* σ)
note $\sigma = this$
hence $\sigma \in set\ \Sigma$
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ' Σ)
then show *?case*
by (*cases* $\xi = snd\ \sigma'$, *simp+*)
qed
moreover from *Cons.hyps* **have** *mset* $(\mathfrak{U}\ (remove1\ \sigma\ \Sigma)\ \Xi) \subseteq\#$ *mset* $(remove1\ \sigma\ \Sigma)$
by *blast*
hence *mset* $(\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi)) \subseteq\#$ *mset* $(\sigma\ \#\ remove1\ \sigma\ \Sigma)$ **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 $(map\ snd\ (\mathfrak{U}\ \Sigma\ \Xi)) =$ *mset* $((map\ snd\ \Sigma) \cap \Xi)$
proof −
have $\forall\ \Sigma.$ *mset* $(map\ snd\ (\mathfrak{U}\ \Sigma\ \Xi)) =$ *mset* $((map\ snd\ \Sigma) \cap \Xi)$
proof (*induct* Ξ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* ξ Ξ)
{
fix Σ
have *mset* $(map\ snd\ (\mathfrak{U}\ \Sigma\ (\xi\ \#\ \Xi))) =$ *mset* $(map\ snd\ \Sigma \cap \xi\ \#\ \Xi)$
proof (*cases find* $(\lambda\ \sigma.\ \xi = snd\ \sigma)\ \Sigma$)

case *None*
hence $\xi \notin set\ (map\ snd\ \Sigma)$
proof (*induct* Σ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* σ Σ)
have $find\ (\lambda\sigma.\ \xi = snd\ \sigma)\ \Sigma = None$
$\xi \neq snd\ \sigma$
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 $\sigma \in set\ \Sigma$ $\xi = snd\ \sigma$
by (*meson find-Some-predicate find-Some-set-membership*)+
moreover
from ‹$\sigma \in set\ \Sigma$› **have** $mset\ \Sigma = mset\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma))$
by *simp*
hence $mset\ (map\ snd\ \Sigma) = mset\ ((snd\ \sigma)\ \#\ (remove1\ (snd\ \sigma)\ (map\ snd\ \Sigma)))$
$mset\ (map\ snd\ \Sigma) = mset\ (map\ snd\ (\sigma\ \#\ (remove1\ \sigma\ \Sigma)))$
by (*simp add*: ‹$\sigma \in set\ \Sigma$›, *metis map-monotonic subset-mset.eq-iff*)
hence $mset\ (map\ snd\ (remove1\ \sigma\ \Sigma)) = mset\ (remove1\ (snd\ \sigma)\ (map\ snd\ \Sigma))$
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*:
$\vdash (map\ (uncurry\ (\sqcup))\ \Sigma :\rightarrow \varphi) \rightarrow \bigsqcap\ (map\ (\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)\ \Sigma) \rightarrow \varphi$
proof (*induct* Σ)
case *Nil*
then show *?case* **using** *Axiom-1* **by** *simp*
next
case (*Cons* σ Σ)
let $?\chi = fst\ \sigma$
let $?\xi = snd\ \sigma$
let $?\Sigma_A = map\ (uncurry\ (\sqcup))\ \Sigma$
let $?\Sigma_B = map\ (\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)\ \Sigma$
assume $\vdash ?\Sigma_A :\rightarrow \varphi \rightarrow \bigsqcap\ ?\Sigma_B \rightarrow \varphi$

moreover have
$\vdash (?\Sigma_A :\rightarrow \varphi \rightarrow \bigsqcap\ ?\Sigma_B \rightarrow \varphi)$
$\rightarrow ((?\chi \sqcup ?\xi) \rightarrow ?\Sigma_A :\rightarrow \varphi) \rightarrow (((?\chi \rightarrow ?\xi) \rightarrow \varphi) \sqcap \bigsqcap\ ?\Sigma_B) \rightarrow \varphi$
proof −
let $?\varphi = (\langle ?\Sigma_A :\rightarrow \varphi\rangle \rightarrow \langle \bigsqcap\ ?\Sigma_B\rangle \rightarrow \langle\varphi\rangle)$
$\rightarrow (((\langle ?\chi\rangle \sqcup \langle ?\xi\rangle) \rightarrow \langle ?\Sigma_A :\rightarrow \varphi\rangle) \rightarrow (((\langle ?\chi\rangle \rightarrow \langle ?\xi\rangle) \rightarrow \langle\varphi\rangle) \sqcap \langle \bigsqcap\ ?\Sigma_B\rangle) \rightarrow \langle\varphi\rangle)$
have $\forall \mathfrak{M}.\ \mathfrak{M} \models_{prop} ?\varphi$ **by** *fastforce*
hence $\vdash (\!|\ ?\varphi\ |\!)$ **using** *propositional-semantics* **by** *blast*
thus *?thesis* **by** *simp*
qed
ultimately have $\vdash ((?\chi \sqcup ?\xi) \rightarrow ?\Sigma_A :\rightarrow \varphi) \rightarrow (((?\chi \rightarrow ?\xi) \rightarrow \varphi) \sqcap \bigsqcap\ ?\Sigma_B) \rightarrow \varphi$
using *Modus-Ponens* **by** *blast*
moreover
have $(\lambda\ \sigma.\ (fst\ \sigma \rightarrow snd\ \sigma) \rightarrow \varphi) = (\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)$
$uncurry\ (\sqcup) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$
by *fastforce+*
hence $(\lambda\ (\chi, \xi).\ (\chi \rightarrow \xi) \rightarrow \varphi)\ \sigma = (?\chi \rightarrow ?\xi) \rightarrow \varphi$
$uncurry\ (\sqcup)\ \sigma = ?\chi \sqcup ?\xi$
by *metis+*
ultimately show *?case* **by** *simp*
qed

lemma (**in** *Classical-Propositional-Logic*) *witness-core-size-increase*:
assumes $\neg \vdash \varphi$
and $mset\ (map\ snd\ \Sigma) \subseteq\#\ mset\ \Gamma$
and $map\ (uncurry\ (\sqcup))\ \Sigma :\vdash \varphi$
shows $(|\ \Gamma\ |_\varphi) < (|\ map\ (uncurry\ (\rightarrow))\ \Sigma\ @\ \Gamma \ominus map\ snd\ \Sigma\ |_\varphi)$
proof −
from $\langle\neg \vdash \varphi\rangle$ **obtain** Ξ **where** Ξ: $\Xi \in \mathcal{C}\ \Gamma\ \varphi$
using *unproving-core-existence* **by** *blast*
let $?\Sigma' = \Sigma \ominus \mathfrak{U}\ \Sigma\ \Xi$
let $?\Sigma\Xi' = map\ (uncurry\ (\sqcup))\ (\mathfrak{U}\ \Sigma\ \Xi)\ @\ map\ (uncurry\ (\rightarrow))\ (\mathfrak{U}\ \Sigma\ \Xi)$
have $mset\ \Sigma = mset\ (\mathfrak{U}\ \Sigma\ \Xi\ @\ ?\Sigma')$ **by** (*simp add*: *core-witness-left-msub*)
hence $set\ (map\ (uncurry\ (\sqcup))\ \Sigma) = set\ (map\ (uncurry\ (\sqcup))\ ((\mathfrak{U}\ \Sigma\ \Xi)\ @\ ?\Sigma'))$
by (*metis mset-map mset-eq-setD*)
hence $map\ (uncurry\ (\sqcup))\ ((\mathfrak{U}\ \Sigma\ \Xi)\ @\ ?\Sigma') :\vdash \varphi$
using *list-deduction-monotonic assms*(*3*)
by *blast*
hence $map\ (uncurry\ (\sqcup))\ (\mathfrak{U}\ \Sigma\ \Xi)\ @\ map\ (uncurry\ (\sqcup))\ ?\Sigma' :\vdash \varphi$ **by** *simp*
moreover
{
fix $\Phi\ \Psi$
have $((\Phi\ @\ \Psi) :\rightarrow \varphi) = (\Phi :\rightarrow (\Psi :\rightarrow \varphi))$
by (*induct* Φ, *simp+*)
hence $(\Phi\ @\ \Psi) :\vdash \varphi = \Phi :\vdash (\Psi :\rightarrow \varphi)$
unfolding *list-deduction-def*
by (*induct* Φ, *simp+*)

}
ultimately have *map* (*uncurry* ($\sqcup$)) ($\mathfrak{U}$ Σ Ξ) :⊢ *map* (*uncurry* ($\sqcup$)) *?*Σ' :→ φ
  by *simp*
moreover have *set* (*map* (*uncurry* ($\sqcup$)) ($\mathfrak{U}$ Σ Ξ)) ⊆ *set* *?*$\Sigma\Xi'$
  by *simp*
ultimately have *?*$\Sigma\Xi'$:⊢ *map* (*uncurry* ($\sqcup$)) *?*Σ' :→ φ
  using *list-deduction-monotonic* **by** *blast*
hence *?*$\Sigma\Xi'$:⊢ $\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$
  using *list-deduction-modus-ponens*
        list-deduction-weaken
        witness-list-implication-rule
  by *blast*
hence *?*$\Sigma\Xi'$ \$⊢ [$\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$]
  using *segmented-deduction-one-collapse* **by** *metis*
hence
  ?$\Sigma\Xi'$ @ (*map snd* ($\mathfrak{U}$ Σ Ξ)) ⊖ (*map snd* ($\mathfrak{U}$ Σ Ξ))
    \$⊢ [$\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$]
  by *simp*
hence *map snd* ($\mathfrak{U}$ Σ Ξ) \$⊢ [$\prod$ (*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) \$⊢ [$\prod$ (*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) :⊢ $\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ') $\to \varphi$
  using *segmented-deduction-one-collapse* **by** *blast*
hence ⋆:
  map snd ($\mathfrak{U}$ Σ Ξ) @ Ξ ⊖ (*map snd* Σ) :⊢ $\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$)
?Σ') $\to \varphi$
  (**is** *?*Ξ_0 :⊢ -)
  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* (Ξ ⊖ (*map snd* Σ)) + *mset* (Ξ ∩ (*map snd* Σ))
  using *list-diff-intersect-comp* **by** *blast*
hence *mset* Ξ = *mset* ((*map snd* Σ) ∩ Ξ) + *mset* (Ξ ⊖ (*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* (Ξ ⊖ (*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 ¬ *?*Ξ_0 :⊢ $\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ')
proof (*rule notI*)
  assume *?*Ξ_0 :⊢ $\prod$ (*map* (λ (χ, γ). ($\chi \to \gamma$) $\to \varphi$) *?*Σ')
  hence *?*Ξ_0 :⊢ φ

using $\star$ *list-deduction-modus-ponens* **by** *blast*
hence $\Xi :\vdash \varphi$
using *list-deduction-monotonic* ‹*set* Ξ = *set* $?\Xi_0$› **by** *blast*
thus *False*
using Ξ *unproving-core-def* **by** *blast*
qed
moreover
have *mset* (*map snd* ($\mathfrak{U}\ \Sigma\ \Xi$)) $\subseteq\#$ *mset* $?\Xi_0$
mset (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{U}\ \Sigma\ \Xi$) @ $?\Xi_0 \ominus$ *map snd* ($\mathfrak{U}\ \Sigma\ \Xi$))
= *mset* (*map* (*uncurry* ($\rightarrow$)) ($\mathfrak{U}\ \Sigma\ \Xi$) @ $\Xi \ominus$ (*map snd* Σ))
(**is** - = *mset* $?\Xi_1$)
by *auto*
hence $?\Xi_1 \preceq ?\Xi_0$
by (*metis add.commute*
witness-stronger-theory
add-diff-cancel-right'
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
$\neg\ ?\Xi_1 :\vdash \prod$ (*map* (λ (χ, γ). ($\chi \rightarrow \gamma$) $\rightarrow \varphi$) $?\Sigma'$)
using *stronger-theory-deduction-monotonic* **by** *blast*
from *this* **obtain** χ γ **where**
(χ,γ) $\in$ *set* $?\Sigma'$
$\neg$ ($\chi \rightarrow \gamma$) # $?\Xi_1 :\vdash \varphi$
using *list-deduction-theorem*
by *fastforce*
have *mset* ($\chi \rightarrow \gamma$ # $?\Xi_1$) $\subseteq\#$ *mset* (*map* (*uncurry* ($\rightarrow$)) Σ @ $\Gamma \ominus$ *map snd* Σ)
proof −
let $?A$ = *map* (*uncurry* ($\rightarrow$)) Σ
let $?B$ = *map* (*uncurry* ($\rightarrow$)) ($\mathfrak{U}\ \Sigma\ \Xi$)
have (χ,γ) $\in$ (*set* Σ − *set* ($\mathfrak{U}\ \Sigma\ \Xi$))
proof −
from ‹(χ,γ) $\in$ *set* $?\Sigma'$› **have** $\gamma \in\#$ *mset* (*map snd* ($\Sigma \ominus \mathfrak{U}\ \Sigma\ \Xi$))
by (*metis set-mset-mset image-eqI set-map snd-conv*)
hence $\gamma \in\#$ *mset* (*map snd* $\Sigma \ominus$ *map snd* ($\mathfrak{U}\ \Sigma\ \Xi$))
by (*metis core-witness-left-msub map-listSubtract-mset-equivalence*)
hence $\gamma \in\#$ *mset* (*map snd* $\Sigma \ominus$ (*map snd* $\Sigma \cap \Xi$))
by (*metis core-witness-right-projection listSubtract-mset-homomorphism*)
hence $\gamma \in\#$ *mset* (*map snd* $\Sigma \ominus \Xi$)
by (*metis add-diff-cancel-right'*
listSubtract-mset-homomorphism
list-diff-intersect-comp)
moreover from *assms*(*2*) **have** *mset* (*map snd* $\Sigma \ominus \Xi$) $\subseteq\#$ *mset* ($\Gamma \ominus \Xi$)

```
        by (simp, metis 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 (χ, γ) ∉ set (𝔘 Σ (ξ # Ξ))
          proof (cases find (λσ. ξ = snd σ) Σ)
            case None
            then show ?thesis using Cons by simp
          next
            case (Some σ)
            moreover from this have snd σ = ξ
              using find-Some-predicate by fastforce
            with Cons.prems have σ ≠ (χ,γ) by fastforce
            ultimately show ?thesis using Cons by simp
          qed
        }
        then show ?case by blast
      qed
      moreover from ‹(χ,γ) ∈ set ?Σ'› have (χ,γ) ∈ set Σ
        by (meson listSubtract-set-trivial-upper-bound subsetCE)
      ultimately show ?thesis by fastforce
    qed
    with ‹(χ, γ) ∈ set ?Σ'› have mset ((χ,γ) # 𝔘 Σ Ξ) ⊆# mset Σ
      by (meson core-witness-left-msub msub-listSubtract-elem-cons-msub)
    hence mset (χ → γ # ?B) ⊆# mset (map (uncurry (→)) Σ)
      by (metis (no-types, lifting) ‹(χ, γ) ∈ set ?Σ'›
                                    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 Ξ ⊆# 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* $\Omega \geq (|\ \Gamma\ |_\varphi) + 1$
using *unproving-core-def*
by (*metis* (*no-types*, *lifting*) ‹$\neg\ \chi \to \gamma\ \#\ ?\Xi_1 :\vdash \varphi$› *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 $\neg \vdash \varphi$
shows $(\Gamma\ \#\vdash\ n\ \varphi) = (n \leq \|\ \Gamma\ \|_\varphi)$
proof −
have $\forall\ \Gamma.\ (\Gamma\ \#\vdash\ n\ \varphi) = (n \leq \|\ \Gamma\ \|_\varphi)$
proof (*induct n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
{
fix Γ
assume $\Gamma\ \#\vdash\ (Suc\ n)\ \varphi$
from *this* **obtain** Σ **where** Σ:
mset (*map snd* Σ) $\subseteq\#$ *mset* Γ
map (*uncurry* $(\sqcup)$) $\Sigma :\vdash \varphi$
map (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ (*map snd* Σ) $\#\vdash\ n\ \varphi$
by *fastforce*
let $?\Gamma' =$ *map* (*uncurry* $(\to)$) Σ @ $\Gamma \ominus$ (*map snd* Σ)
have *length* $\Gamma =$ *length* $?\Gamma'$
using $\Sigma(1)$ *listSubtract-msub-eq* **by** *fastforce*
hence $(\|\ \Gamma\ \|_\varphi) > (\|\ ?\Gamma'\ \|_\varphi)$
by (*metis* $\Sigma(1)$ $\Sigma(2)$ ‹$\neg \vdash \varphi$›

```
                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 Σ(3) ‹Suc n ≤ ‖ Γ ‖φ› by linarith
      hence ?Γ' #⊢ n φ using Suc by blast
      hence Γ #⊢ (Suc n) φ using Σ(1) Σ(2) by fastforce
    }
    ultimately show ?case by metis
  qed
  thus ?thesis by auto
qed

lemma (in Classical-Propositional-Logic) stratified-deduction-tautology-equiv:
  (∀ n. Γ #⊢ n φ) = ⊢ φ
proof (cases ⊢ φ)
  case True
  then show ?thesis
    by (simp add: stratified-deduction-tautology-weaken)
next
  case False
  have ¬ Γ #⊢ (1 + length Γ) φ
  proof (rule notI)
    assume Γ #⊢ (1 + length Γ) φ
    hence 1 + length Γ ≤ ‖ Γ ‖φ
      using ‹¬ ⊢ φ› unproving-core-stratified-deduction-lower-bound by blast
    hence 1 + length Γ ≤ length Γ
      using complement-core-size-def by fastforce
    thus False by linarith
  qed
  then show ?thesis
    using ‹¬ ⊢ φ› by blast
qed

lemma (in Classical-Propositional-Logic) unproving-core-max-stratified-deduction:
```

$\Gamma \; \#\vdash \; n \; \varphi = (\forall \; \Phi \in \mathcal{C} \; \Gamma \; \varphi. \; n \leq \textit{length} \; (\Gamma \ominus \Phi))$

proof (*cases* $\vdash \varphi$)
  case *True*
  from ‹$\vdash \varphi$› **have** $\Gamma \; \#\vdash \; n \; \varphi$
    using *stratified-deduction-tautology-weaken*
    by *blast*
  moreover from ‹$\vdash \varphi$› **have** $\mathcal{C} \; \Gamma \; \varphi = \{\}$
    using *unproving-core-existence* **by** *auto*
  hence $\forall \; \Phi \in \mathcal{C} \; \Gamma \; \varphi. \; n \leq \textit{length} \; (\Gamma \ominus \Phi)$ **by** *blast*
  ultimately show *?thesis* **by** *meson*
next
  case *False*
  from ‹$\neg \vdash \varphi$› **have** $(\Gamma \; \#\vdash \; n \; \varphi) = (n \leq \| \; \Gamma \; \|_{\varphi})$
    by (*simp add*: *unproving-core-stratified-deduction-lower-bound*)
  moreover have $(n \leq \| \; \Gamma \; \|_{\varphi}) = (\forall \; \Phi \in \mathcal{C} \; \Gamma \; \varphi. \; n \leq \textit{length} \; (\Gamma \ominus \Phi))$
  proof (*rule iffI*)
    assume $n \leq \| \; \Gamma \; \|_{\varphi}$
    {
      fix Φ
      assume $\Phi \in \mathcal{C} \; \Gamma \; \varphi$
      hence $n \leq \textit{length} \; (\Gamma \ominus \Phi)$
        using ‹$n \leq \| \; \Gamma \; \|_{\varphi}$› *complement-core-size-intro* **by** *auto*
    }
    thus $\forall \Phi \in \mathcal{C} \; \Gamma \; \varphi. \; n \leq \textit{length} \; (\Gamma \ominus \Phi)$ **by** *blast*
  next
    assume $\forall \Phi \in \mathcal{C} \; \Gamma \; \varphi. \; n \leq \textit{length} \; (\Gamma \ominus \Phi)$
    with ‹$\neg \vdash \varphi$› **obtain** Φ **where**
      $\Phi \in \mathcal{C} \; \Gamma \; \varphi$
      $n \leq \textit{length} \; (\Gamma \ominus \Phi)$
      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*:
  $(\sum \gamma \leftarrow \Gamma. \; \textit{Pr} \; \gamma) \leq \textit{real} \; (\textit{length} \; \Gamma)$
proof (*induct* Γ)
  case *Nil*
  then show *?case* **by** *simp*
next
  case (*Cons* γ Γ)
  moreover have $\textit{Pr} \; \gamma \leq \textit{1}$ **using** *unity-upper-bound* **by** *blast*
  ultimately have $\textit{Pr} \; \gamma + (\sum \gamma \leftarrow \Gamma. \; \textit{Pr} \; \gamma) \leq \textit{1} + \textit{real} \; (\textit{length} \; \Gamma)$ **by** *linarith*
  then show *?case* **by** *simp*
qed

theorem (**in** *Classical-Propositional-Logic*) *binary-limited-stratified-deduction-completeness*:
  $(\forall\ Pr \in \textit{Dirac-Measures}.\ \textit{real}\ n * Pr\ \varphi \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)) = \sim \Gamma\ \#\vdash\ n\ (\sim \varphi)$
proof −
  {
    fix $Pr :: 'a \Rightarrow \textit{real}$
    assume $Pr \in \textit{Dirac-Measures}$
    from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
      unfolding *Dirac-Measures-def*
      by *auto*
    assume $\sim \Gamma\ \#\vdash\ n\ (\sim \varphi)$
    moreover have $\textit{replicate}\ n\ (\sim \varphi) = \sim (\textit{replicate}\ n\ \varphi)$
      by (*induct n*, *auto*)
    ultimately have $\sim \Gamma\ \$\vdash\ \sim (\textit{replicate}\ n\ \varphi)$
      using *stratified-segmented-deduction-replicate* **by** *metis*
    hence $(\sum \varphi \leftarrow (\textit{replicate}\ n\ \varphi).\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
      using *segmented-deduction-summation-introduction*
      by *blast*
    moreover have $(\sum \varphi \leftarrow (\textit{replicate}\ n\ \varphi).\ Pr\ \varphi) = \textit{real}\ n * Pr\ \varphi$
      by (*induct n*, *simp*, *simp add*: *semiring-normalization-rules*(*3*))
    ultimately have $\textit{real}\ n * Pr\ \varphi \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
      by *simp*
  }
  moreover
  {
    assume $\neg \sim \Gamma\ \#\vdash\ n\ (\sim \varphi)$
    have $\exists\ Pr \in \textit{Dirac-Measures}.\ \textit{real}\ n * Pr\ \varphi > (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
    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: $(\sim \Phi) \in \mathcal{C}\ (\sim \Gamma)\ (\sim \varphi)\ \textit{mset}\ \Phi \subseteq\#\ \textit{mset}\ \Gamma$
        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: $\textit{MCS}\ \Omega\ \varphi \in \Omega\ \bigsqcup\ \Phi \notin \Omega$
        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* φ = *0*
using Φ(*1*) Ω(*1*) Ω(*3*) *arbitrary-disjunction-exclusion-MCS* **by** *auto*
with Φ(*2*) **have** ($\sum\gamma{\leftarrow}\Gamma$. *?Pr* γ) = ($\sum\gamma{\leftarrow}(\Gamma \ominus \Phi)$. *?Pr* γ)
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
then show *?case*
proof –
obtain ω :: *'a* **where**
ω: $\neg$ *mset* $\Phi \subseteq\#$ *mset* Γ
$\vee\ \omega \in$ *set* $\Phi \wedge \omega \in \Omega$
$\vee$ ($\sum\gamma{\leftarrow}\Gamma$. *?Pr* γ) = ($\sum\gamma{\leftarrow}\Gamma \ominus \Phi$. *?Pr* γ)
using *Cons.hyps* **by** *fastforce*
have *A*:
$\forall$(*f* :: *'a* $\Rightarrow$ *real*) (Γ ::*'a list*) Φ.
$\neg$ *mset* $\Phi \subseteq\#$ *mset* Γ
$\vee$ *sum-list* (($\sum\varphi{\leftarrow}\Phi$. *f* φ) # *map f* ($\Gamma \ominus \Phi$)) = ($\sum\gamma{\leftarrow}\Gamma$. *f* γ)
using *listSubstract-multisubset-list-summation* **by** *auto*
have *B*: $\forall$ *rs*. *sum-list* ((*0*::*real*) # *rs*) = *sum-list rs*
by *auto*
have *C*: $\forall$ *r rs*. (*0*::*real*) = *r* $\vee$ *sum-list* (*r* # *rs*) $\neq$ *sum-list rs*
by *simp*
have *D*: $\forall$ *f*. *sum-list* (*sum-list* (*map f* (φ # Φ)) # *map f* ($\Gamma \ominus$ (φ # Φ)))
= (*sum-list* (*map f* Γ)::*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* Φ)
$\vee$ *sum-list* (*map f* Γ) $\neq$ *sum-list* (*map f* ($\Gamma \ominus \Phi$))
using *C A* **by** (*metis* (*no-types*))
then have *G*: ($\sum\varphi'{\leftarrow}(\varphi\ \#\ \Phi)$. *?Pr* φ') = *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* γ)
$\vee\ \omega \in$ *set* Φ
$\vee$ *r* $\neq$ ($\sum\gamma{\leftarrow}(\varphi\ \#\ \Gamma)$. *?Pr* γ)
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 $H\ F\ E\ D\ B\ \omega$ **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* Φ(*1*) $\langle\neg (\sim \Gamma)\ \#\!\vdash n\ (\sim \varphi)\rangle$
unproving-core-max-stratified-deduction
unproving-listSubtract-length-equiv)
hence *real* (*length* $(\sim \Gamma \ominus \sim \Phi)$) $<$ *real n*
by *simp*
with Ω(*2*) **have** *real* (*length* $(\sim \Gamma \ominus \sim \Phi)$) $<$ *real n* $*$ *?Pr* φ
by *simp*
moreover
have $(\sim (\Gamma \ominus \Phi))$ <~~> $(\sim \Gamma \ominus \sim \Phi)$
by (*metis* Φ(*2*) *map-listSubtract-mset-equivalence mset-eq-perm*)
with *perm-length* **have** *length* $(\Gamma \ominus \Phi)$ = *length* $(\sim \Gamma \ominus \sim \Phi)$
by *fastforce*
hence *real* (*length* $(\Gamma \ominus \Phi)$) = *real* (*length* $(\sim \Gamma \ominus \sim \Phi)$)
by *simp*
ultimately show *?thesis*
by *force*
qed
}
ultimately show *?thesis* **by** *fastforce*
qed

lemma (**in** *Classical-Propositional-Logic*) *binary-segmented-deduction-completeness*:
$(\forall\ Pr \in$ *Dirac-Measures*. $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)) = \sim \Gamma\ \$\!\vdash\ \sim \Phi$
proof $-$
{
fix $Pr :: 'a \Rightarrow real$
assume $Pr \in$ *Dirac-Measures*
from *this* **interpret** *Logical-Probability* $(\lambda\ \varphi.\ \vdash \varphi)\ (\rightarrow)\ \bot\ Pr$
unfolding *Dirac-Measures-def*
by *auto*
assume $\sim \Gamma\ \$\!\vdash\ \sim \Phi$
hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
using *segmented-deduction-summation-introduction*

```
    by blast
  }
  moreover
  {
    assume ¬ ∼ Γ $⊢ ∼ Φ
    have ∃ Pr ∈ Dirac-Measures. (∑φ←Φ. Pr φ) > (∑γ←Γ. Pr γ)
    proof −
      from ‹¬ ∼ Γ $⊢ ∼ Φ› have ¬ ∼ (∼ Φ) @ ∼ Γ #⊢ (length (∼ Φ)) ⊥
        using segmented-stratified-falsum-equiv by blast
      moreover
      have ∼ (∼ Φ) @ ∼ Γ #⊢ (length (∼ Φ)) ⊥ = ∼ (∼ Φ) @ ∼ Γ #⊢ (length
Φ) ⊥
        by (induct Φ, auto)
      moreover have ⊢ ∼ ⊤ → ⊥
        by (simp add: negation-def)
      ultimately have ¬ ∼ (∼ Φ @ Γ) #⊢ (length Φ) (∼ ⊤)
        using stratified-deduction-implication by fastforce
      from this obtain Pr where Pr:
        Pr ∈ Dirac-Measures
        real (length Φ) ∗ Pr ⊤ > (∑γ← (∼ Φ @ Γ). Pr γ)
        using binary-limited-stratified-deduction-completeness
        by fastforce
      from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
        unfolding Dirac-Measures-def
        by auto
      from Pr(2) have real (length Φ) > (∑γ← ∼ Φ. Pr γ) + (∑γ← Γ. Pr γ)
        by (simp add: Unity)
      moreover have (∑γ← ∼ Φ. Pr γ) = real (length Φ) − (∑γ← 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:
  (∀ Pr ∈ Logical-Probabilities. (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)) = ∼ Γ $⊢ ∼
Φ
proof −
  {
    fix Pr :: 'a ⇒ real
    assume Pr ∈ Logical-Probabilities
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Logical-Probabilities-def
      by auto
    assume ∼ Γ $⊢ ∼ Φ
    hence (∑φ←Φ. Pr φ) ≤ (∑γ←Γ. 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*:
$(\forall\ Pr \in \textit{Logical-Probabilities}.\ (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum \varphi{\leftarrow}\Phi.\ Pr\ \varphi) \leq (\sum \gamma{\leftarrow}\Gamma.\ Pr\ \gamma))$
by (*simp add*: *binary-segmented-deduction-completeness*
segmented-deduction-completeness)

lemma (**in** *Classical-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*:
$(\forall\ Pr \in \textit{Dirac-Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)) = (\textit{length}\ \Phi \leq \|$
$\sim \Gamma\ @\ \Phi\ \|_\perp)$
proof (*cases* $\vdash \perp$)
case *True*
hence $\mathcal{C}\ (\sim \Gamma\ @\ \Phi)\ \perp = \{\}$
using *unproving-core-existence* **by** *auto*
hence $\textit{length}\ (\sim \Gamma\ @\ \Phi) = \|\ \sim \Gamma\ @\ \Phi\ \|_\perp$
unfolding *complement-core-size-def core-size-def* **by** *presburger*
then show *?thesis*
using *True stratified-deduction-completeness stratified-deduction-tautology-weaken*
by *auto*
next
case *False*
then show *?thesis*
using *stratified-deduction-completeness unproving-core-stratified-deduction-lower-bound*
by *blast*
qed

lemma (**in** *Classical-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**
$(\forall \Phi\ \Gamma.\ \varrho\ \Phi\ \Gamma \in \textit{Dirac-Measures} \land \neg\ (\sum\varphi\leftarrow\Phi.\ (\varrho\ \Phi\ \Gamma)\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ (\varrho$
$\Phi\ \Gamma)\ \gamma)$
$\lor\ \textit{length}\ \Phi \leq \|\ \sim \Gamma\ @\ \Phi\ \|_\perp)$
$\land\ (\forall \Phi\ \Gamma.\ \textit{length}\ \Phi \leq (\|\ \sim \Gamma\ @\ \Phi\ \|_\perp)$
$\longrightarrow (\forall Pr \in \textit{Dirac-Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)))$
using *complement-core-completeness* **by** *moura*
moreover have $\forall \Gamma\ \varphi\ n.\ \textit{length}\ \Gamma - n \leq (\|\ \Gamma\ \|_\varphi) \lor (|\ \Gamma\ |_\varphi) - n \neq \mathit{0}$
by (*metis add-diff-cancel-right′*
cancel-ab-semigroup-add-class.diff-right-commute
diff-is-0-eq length-core-decomposition)
moreover have $\forall\ \Gamma\ \Phi\ n.\ \textit{length}\ (\Gamma\ @\ \Phi) - n \leq \textit{length}\ \Gamma \lor \textit{length}\ \Phi - n \neq \mathit{0}$
by *force*
ultimately have
$(Pr \in \textit{Dirac-Measures} \longrightarrow (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$\land\ (|\ \sim \Gamma\ @\ \Phi\ |_\perp) \leq \textit{length}\ (\sim \Gamma)$
$\lor\qquad \neg\ (|\ \sim \Gamma\ @\ \Phi\ |_\perp) \leq \textit{length}\ (\sim \Gamma)$

$\wedge$ ($\exists$ *Pr*. *Pr* $\in$ *Dirac-Measures* $\wedge$ $\neg$ ($\sum\varphi\leftarrow\Phi$. *Pr* φ) $\leq$ ($\sum\gamma\leftarrow\Gamma$. *Pr* γ))
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* $\in$ *Dirac-Measures*. $\exists$ *n* :: *nat*. *real n* = ($\sum\varphi\leftarrow\Phi$. *Pr* φ)
proof (*induct* Φ)
case *Nil*
then show *?case* **by** *simp*
next
case (*Cons* φ Φ)
{
fix *Pr* :: *'a* $\Rightarrow$ *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* = ($\sum\varphi'\leftarrow(\varphi$ # Φ). *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 ¬ ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
moreover
obtain x :: *int*
and y :: *int*
and z :: *int*
where *xyz*: x = ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$)
$y = \lceil c\rceil$
z = ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
using *nat-binary-probability*
by (*metis* ‹*Pr* ∈ *Dirac-Measures*› *of-int-of-nat-eq*)
ultimately have $x + y - 1 \geq z$ **by** *linarith*
hence ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c > ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$) **using** *xyz* **by** *linarith*
thus *False* **using** *assm* **by** *simp*
qed
next
assume ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
thus ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
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 ⊤) @ Φ. *Pr* φ)
using *Unity*
by (*induct n*, *auto*)

lemma (**in** *Classical-Propositional-Logic*) *dirac-collapse*:
(∀ *Pr* ∈ *Logical-Probabilities*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$))
= (∀ *Pr* ∈ *Dirac-Measures*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$))
proof
assume ∀ *Pr* ∈ *Logical-Probabilities*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
hence ∀ *Pr* ∈ *Dirac-Measures*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
using *Dirac-Measures-subset* **by** *fastforce*
thus ∀ *Pr* ∈ *Dirac-Measures*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
using *dirac-ceiling* **by** *blast*
next
assume *assm*: ∀ *Pr* ∈ *Dirac-Measures*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + $\lceil c\rceil$ ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr$
γ)
show ∀ *Pr* ∈ *Logical-Probabilities*. ($\sum\varphi\leftarrow\Phi.\ Pr\ \varphi$) + c ≤ ($\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma$)
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 ∈ 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 assm ‹Pr ∈ Dirac-Measures› by blast
    hence (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
      using ‹real n = ⌈c⌉›
          probability-replicate-verum [where Φ=Φ and n=n]
      by metis
  }
  hence ∀ Pr ∈ Dirac-Measures. (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ.
Pr γ)
    by blast
  hence †: ∀ Pr ∈ Logical-Probabilities.
        (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
    using weakly-additive-completeness-collapse by blast
  {
    fix Pr
    assume Pr ∈ Logical-Probabilities
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Logical-Probabilities-def
      by auto
    have (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
      using † ‹Pr ∈ Logical-Probabilities› by blast
    hence (∑φ←Φ. Pr φ) + c ≤ (∑γ←Γ. Pr γ)
      using ‹real n = ⌈c⌉›
          probability-replicate-verum [where Φ=Φ and n=n]
      by linarith
  }
  then show ?thesis by blast
 next
  case False
  hence ⌈c⌉ ≤ 0 by auto
   from this obtain n :: nat where real n = − ⌈c⌉ by (metis neg-0-le-iff-le
of-nat-nat)
  {
    fix Pr
    assume Pr ∈ Dirac-Measures
    from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
      unfolding Dirac-Measures-def
      by auto
    have (∑φ←Φ. Pr φ) + ⌈c⌉ ≤ (∑γ←Γ. Pr γ)
```

using *assm* ‹*Pr* ∈ *Dirac-Measures*› **by** *blast*
hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow (replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
using ‹*real n* = − ⌈*c*⌉›
probability-replicate-verum [**where** Φ=Γ **and** *n*=*n*]
by *linarith*
}
hence ∀ *Pr* ∈ *Dirac-Measures*. $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow (replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
by *blast*
hence ‡: ∀ *Pr* ∈ *Logical-Probabilities*.
$(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) \leq (\sum \gamma \leftarrow (replicate\ n\ \top)\ @\ \Gamma.\ Pr\ \gamma)$
using *weakly-additive-completeness-collapse* **by** *blast*
{
fix *Pr*
assume *Pr* ∈ *Logical-Probabilities*
from *this* **interpret** *Logical-Probability* (λ φ. ⊢ φ) (→) ⊥ *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* ∈ *Logical-Probabilities*› **by** *blast*
hence $(\sum \varphi \leftarrow \Phi.\ Pr\ \varphi) + c \leq (\sum \gamma \leftarrow \Gamma.\ Pr\ \gamma)$
using ‹*real n* = − ⌈*c*⌉›
probability-replicate-verum [**where** Φ=Γ **and** *n*=*n*]
by *linarith*
}
then show *?thesis* **by** *blast*
qed
qed

lemma (**in** *Classical-Propositional-Logic*) *dirac-strict-floor*:
∀ *Pr* ∈ *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* ⇒ *real*
let *?Pr′* = (λ φ. ⌊ *Pr* φ ⌋) :: ′*a* ⇒ *int*
assume *Pr* ∈ *Dirac-Measures*
hence ∀ φ. *Pr* φ = *?Pr′* φ
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 ‹∀ φ. *Pr* φ = *?Pr′* φ› **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 ... $= ((\sum \varphi \leftarrow \Phi.\ ?Pr'\ \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma.\ ?Pr'\ \gamma))$

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*:
$(\forall\ Pr \in \mathit{Logical\text{-}Probabilities}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
$= (\forall\ Pr \in \mathit{Dirac\text{-}Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
assume $\forall\ Pr \in \mathit{Logical\text{-}Probabilities}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
hence $\forall\ Pr \in \mathit{Dirac\text{-}Measures}.\ (\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
using *Dirac-Measures-subset* **by** *blast*
thus $\forall\ Pr \in \mathit{Dirac\text{-}Measures}.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
using *dirac-strict-floor* **by** *blast*
next
assume $\forall\ Pr \in \mathit{Dirac\text{-}Measures}.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1 \leq (\sum\gamma\leftarrow\Gamma.\ Pr$
$\gamma))$
moreover have $\lfloor c\rfloor + 1 = \lceil (\lfloor c\rfloor + 1) :: \mathit{real}\rceil$
by *simp*
ultimately have $\star$: $\forall\ Pr \in \mathit{Logical\text{-}Probabilities}.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + \lfloor c\rfloor + 1$
$\leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
using *dirac-collapse* $[\mathit{of}\ \Phi\ \lfloor c\rfloor + 1\ \Gamma]$
by *auto*
show $\forall\ Pr \in \mathit{Logical\text{-}Probabilities}.\ ((\sum\varphi\leftarrow\Phi.\ Pr\ \varphi) + c < (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma))$
proof
fix $Pr :: 'a \Rightarrow \mathit{real}$
assume $Pr \in \mathit{Logical\text{-}Probabilities}$
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 $(|\ \mathit{replicate}\ n\ \top\ @\ \Phi\ |_\varphi) = n + (|\ \Phi\ |_\varphi)$
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 $\top \in \mathit{set}\ \Sigma$
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 ¬ (*remove1* $\top$ Σ :⊢ φ)
by (*meson* ‹$\Sigma \in \mathcal{C}$ ($\top$ # Φ) φ›
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* Σ ⊆# *mset* ($\top$ # Φ)
using ‹$\Sigma \in \mathcal{C}$ ($\top$ # Φ) φ› *unproving-core-def* **by** *blast*
hence *mset* (*remove1* $\top$ Σ) ⊆# *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 ¬ (| 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 ¬ ($\top$ # Δ) :⊢ φ
using *list-deduction-modus-ponens*
list-deduction-theorem
list-deduction-weaken
unproving-core-def
verum-tautology
by *blast*
moreover have *mset* ($\top$ # Δ) ⊆# *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 (| ⊤ # Φ |φ) = (1 + | Φ |φ)
      by (metis Suc-eq-plus1 Suc-le-eq ‹Σ ∈ 𝒞 (⊤ # 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 (∼ ⊤) @ Φ |φ) = (| Φ |φ)
proof (induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  {
    fix Φ
    have (| (∼ ⊤) # Φ |φ) = (| Φ |φ)
    proof (cases ⊢ φ)
      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 Σ ∈ 𝒞 ((∼ ⊤) # Φ) φ
        using unproving-core-existence by fastforce
      have [(∼ ⊤)] :⊢ φ
        by (metis Modus-Ponens
                  Peirces-law
                  The-Principle-of-Pseudo-Scotus
                  list-deduction-theorem
                  list-deduction-weaken
                  negation-def
                  verum-def)
      hence ∼ ⊤ ∉ set Σ
        by (meson ‹Σ ∈ 𝒞 (∼ ⊤ # Φ) φ›
                  list.set-intros(1)
                  list-deduction-base-theory
                  list-deduction-theorem
                  list-deduction-weaken
                  unproving-core-complement-equiv)
      hence remove1 (∼ ⊤) Σ = Σ
        by (simp add: remove1-idem)
      moreover have mset Σ ⊆# mset ((∼ ⊤) # Φ)
        using ‹Σ ∈ 𝒞 (∼ ⊤ # Φ) φ› unproving-core-def by blast
      ultimately have mset Σ ⊆# mset Φ
```

by (*metis add-mset-add-single mset.simps*(*2*) *mset-remove1 subset-eq-diff-conv*)
moreover have $\neg$ (Σ :$\vdash$ φ)
  using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *unproving-core-def* **by** *blast*
ultimately have ($\mid \Phi \mid_\varphi$) $\geq$ *length* Σ
  by (*metis* (*no-types*, *lifting*)
      core-size-intro
      list-deduction-weaken
      unproving-core-def
      unproving-core-existence
      mem-Collect-eq)
hence ($\mid \Phi \mid_\varphi$) $\geq$ ($\mid$ ($\sim \top$) # $\Phi \mid_\varphi$)
  using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *core-size-intro* **by** *auto*
moreover
have ($\mid \Phi \mid_\varphi$) $\leq$ ($\mid$ ($\sim \top$) # $\Phi \mid_\varphi$)
proof −
  obtain Δ **where** $\Delta \in \mathcal{C}$ Φ φ
    using *False unproving-core-existence* **by** *blast*
  hence
    $\neg$ Δ :$\vdash$ φ
    mset Δ $\subseteq$# *mset* (($\sim \top$) # Φ)
    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* Δ $\leq$ *length* Σ
    using ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *unproving-core-def* **by** *blast*
  thus *?thesis*
    using ‹$\Delta \in \mathcal{C}$ Φ φ› ‹$\Sigma \in \mathcal{C}$ ($\sim \top$ # Φ) φ› *core-size-intro* **by** *auto*
qed
ultimately show *?thesis*
  using *le-antisym* **by** *blast*
qed
}
thus *?case* **using** *Suc* **by** *simp*
qed

lemma (**in** *Consistent-Classical-Logic*) *binary-inequality-elim*:
  assumes $\forall$ *Pr* $\in$ *Dirac-Measures*. ($\sum \varphi \leftarrow \Phi$. *Pr* φ) + (c :: *real*) $\leq$ ($\sum \gamma \leftarrow \Gamma$. *Pr* γ)
  shows (($\mid \sim \Gamma$ @ $\Phi \mid_\bot$) + (c :: *real*) $\leq$ *length* Γ)
proof (*cases* $c \geq 0$)
  case *True*
  from *this* **obtain** n :: *nat* **where** *real* n = $\lceil c \rceil$

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

```
  hence (| ∼ 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 ¬ ⊢ ⊥
  then show ?thesis
  proof (cases c ≥ 0)
    assume c ≥ 0
    from this obtain n :: nat where real n = ⌈c⌉
      by (metis ceiling-mono ceiling-zero of-nat-nat)
    hence n + (| ∼ Γ @ Φ |⊥) ≤ length Γ
      using assms by linarith
    hence (| replicate n ⊤ @ ∼ Γ @ Φ |⊥) ≤ length Γ
      by (simp add: ‹¬ ⊢ ⊥› unproving-core-verum-extract)
    moreover have mset (replicate n ⊤ @ ∼ Γ @ Φ) = mset (∼ Γ @ replicate n
⊤ @ Φ)
      by simp
    ultimately have (| ∼ Γ @ replicate n ⊤ @ Φ |⊥) ≤ length Γ
      unfolding core-size-def unproving-core-def
      by metis
   hence ∀ Pr ∈ Dirac-Measures. (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ.
Pr γ)
      using binary-core-partial-completeness by blast
    {
      fix Pr
      assume Pr ∈ Dirac-Measures
      from this interpret Logical-Probability (λ φ. ⊢ φ) (→) ⊥ Pr
        unfolding Dirac-Measures-def
        by auto
      have (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ. Pr γ)
        using ‹Pr ∈ Dirac-Measures›
            ‹∀ Pr ∈ Dirac-Measures. (∑φ←(replicate n ⊤) @ Φ. Pr φ) ≤ (∑γ←Γ.
```

$Pr\ \gamma$))
by *blast*
hence $(\sum\varphi\leftarrow\ \Phi.\ Pr\ \varphi) + n \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
by (*simp add*: *probability-replicate-verum*)
hence $(\sum\varphi\leftarrow\ \Phi.\ Pr\ \varphi) + c \leq (\sum\gamma\leftarrow\Gamma.\ Pr\ \gamma)$
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 $(\mid \sim \Gamma\ @\ \Phi \mid_{\bot}) \leq n +$ *length* Γ
using *assms* **by** *linarith*
hence $(\mid \sim$ (*replicate* $n\ \top\ @\ \Gamma)\ @\ \Phi \mid_{\bot}) \leq$ *length* (*replicate* $n\ \top\ @\ \Gamma$)
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* Γ)
using *binary-inequality-elim binary-inequality-intro consistency* **by** *auto*

end