

# A Sound and Complete Calculus for Probability Inequalities

Matthew Doty

December 9, 2023

### Abstract

We give a sound and complete multiple-conclusion calculus  $\$ \vdash$  for finitely additive probability inequalities. In particular, we show

$$\sim\Gamma\$ \vdash \sim\Phi \equiv \forall \mathcal{P} \in \text{probabilities}. \sum \phi \leftarrow \Phi. \mathcal{P}\phi \leq \sum \gamma \leftarrow \Gamma. \mathcal{P}\gamma$$

...where  $\sim\Gamma$  is the negation of all of the formulae in  $\Gamma$  (and similarly for  $\sim\Phi$ ). We prove this by using an abstract form of *MaxSAT*. We also show

$$\text{MaxSAT}(\sim\Gamma @ \Phi) + c \leq \text{length } \Gamma \equiv \forall \mathcal{P} \in \text{probabilities}. \left( \sum \phi \leftarrow \Phi. \mathcal{P}\phi \right) + c \leq \sum \gamma \leftarrow \Gamma. \mathcal{P}\gamma$$

Finally, we establish a *collapse theorem*, which asserts that  $(\sum \phi \leftarrow \Phi. \mathcal{P}\phi) + c \leq \sum \gamma \leftarrow \Gamma. \mathcal{P}\gamma$  holds for all probabilities  $\mathcal{P}$  if and only if  $(\sum \phi \leftarrow \Phi. \delta\phi) + c \leq \sum \gamma \leftarrow \Gamma. \delta\gamma$  holds for all binary-valued probabilities  $\delta$ .

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Measure Deduction and Counting Deduction</b>	<b>4</b>
2.1	Definition of Measure Deduction . . . . .	4
2.2	Definition of the Stronger Theory Relation . . . . .	5
2.3	The Stronger Theory Relation is a Preorder . . . . .	6
2.4	The Stronger Theory Relation is a Subrelation of of Measure Deduction . . . . .	10
2.5	Measure Deduction is a Preorder . . . . .	20
2.6	Measure Deduction Cancellation Rules . . . . .	86
2.7	Measure Deduction Substitution Rules . . . . .	93
2.8	Measure Deduction Sum Rules . . . . .	94
2.9	Measure Deduction Exchange Rule . . . . .	95
2.10	Definition of Counting Deduction . . . . .	96
2.11	Converting Back and Forth from Counting Deduction to Mea- sure Deduction . . . . .	96
2.12	Measure Deduction Soundness . . . . .	99
<b>3</b>	<b>MaxSAT</b>	<b>106</b>
3.1	Definition of Relative Maximal Clause Collections . . . . .	106
3.2	Definition of MaxSAT . . . . .	111
3.3	Reducing Counting Deduction to MaxSAT . . . . .	113
<b>4</b>	<b>Inequality Completeness For Probability Logic</b>	<b>156</b>
4.1	Limited Counting Deduction Completeness . . . . .	156
4.2	Measure Deduction Completeness . . . . .	159
4.3	Counting Deduction Completeness . . . . .	161
4.4	Collapse Theorem For Probability Logic . . . . .	162
4.5	MaxSAT Completeness For Probability Logic . . . . .	167

# Chapter 1

## Introduction

```
theory Probability-Inequality-Completeness
imports
  Suppes-Theorem.Probability-Logic
begin
```

```
no-notation FuncSet.funcset (infixr  $\rightarrow$  60)
```

We introduce a novel logical calculus and prove completeness for probability inequalities. This is a vast generalization of *Suppes' Theorem* which lays the foundation for this theory.

We provide two new logical judgements: *measure deduction* ( $\$ \vdash$ ) and *counting deduction* ( $\# \vdash$ ). Both judgements capture a notion of *measure* or *quantity*. In both cases premises must be partially or completely *consumed* in sense to prove multiple conclusions. That is to say, a portion of the premises must be used to prove each conclusion which cannot be reused. Counting deduction counts the number of times a particular conclusion can be proved (as the name implies), while measure deduction includes multiple, different conclusions which must be proven via the premises.

We also introduce an abstract notion of MaxSAT, which is the maximal number of clauses in a list of clauses that can be simultaneously satisfied.

We show the following are equivalent:

- $\sim \Gamma \ \$ \vdash \sim \Phi$
- $(\sim \Gamma \ @ \ \Phi) \ # \vdash (\text{length } \Phi) \perp$
- $\text{MaxSAT } (\sim \Gamma \ @ \ \Phi) \leq \text{length } \Gamma$
- $\forall \delta \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \delta \varphi) \leq (\sum \gamma \leftarrow \Gamma. \delta \gamma)$
- $\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$

In the special case of MaxSAT, we show the following are equivalent:

- $MaxSAT(\sim \Gamma @ \Phi) + c \leq length \Gamma$
- $\forall \delta \in \text{dirac-measures. } (\sum_{\varphi \leftarrow \Phi} \delta \varphi) + c \leq (\sum_{\gamma \leftarrow \Gamma} \delta \gamma)$
- $\forall \mathcal{P} \in \text{probabilities. } (\sum_{\varphi \leftarrow \Phi} \mathcal{P} \varphi) + c \leq (\sum_{\gamma \leftarrow \Gamma} \mathcal{P} \gamma)$

## Chapter 2

# Measure Deduction and Counting Deduction

### 2.1 Definition of Measure Deduction

To start, we introduce a common combinator for modifying functions that take two arguments.

**definition** *uncurry* :: ('a ⇒ 'b ⇒ 'c) ⇒ 'a × 'b ⇒ 'c  
where *uncurry-def [simp]*: *uncurry* f = (λ (x, y). f x y)

Our new logical calculus is a recursively defined relation ( $\$ \vdash$ ) using *list deduction* ( $\vdash$ ).

We call our new logical relation *measure deduction*:

**primrec** (in *classical-logic*)  
*measure-deduction* :: 'a list ⇒ 'a list ⇒ bool (infix  $\$ \vdash$  60)  
where  
   $\Gamma \ \$ \vdash \ [] = True$   
  |  $\Gamma \ \$ \vdash (\varphi \# \Phi) =$   
     $(\exists \Psi. mset (map \text{snd } \Psi) \subseteq\# mset \Gamma$   
       $\wedge map (uncurry (\sqcup)) \Psi \vdash \varphi$   
       $\wedge map (uncurry (\rightarrow)) \Psi @ \Gamma \ominus (map \text{snd } \Psi) \ \$ \vdash \Phi)$

Let us briefly analyze what the above definition is saying.

From the above we must find a special list-of-pairs  $\Psi$ , which we refer to as a *witness*, in order to establish  $\Gamma \ \$ \vdash \varphi \# \Phi$ .

We may motivate measure deduction as follows. In the simplest case we know  $\mathcal{P} \varphi \leq \mathcal{P} \psi + \Sigma$  if and only if  $\mathcal{P} (\chi \sqcup \varphi) + \mathcal{P} (\sim \chi \sqcup \varphi) \leq \mathcal{P} \psi + \Sigma$ , or equivalently  $\mathcal{P} (\chi \sqcup \varphi) + \mathcal{P} (\chi \rightarrow \varphi) \leq \mathcal{P} \psi + \Sigma$ . So it suffices to prove  $\mathcal{P} (\chi \sqcup \varphi) \leq \mathcal{P} \psi$  and  $\mathcal{P} (\chi \rightarrow \varphi) \leq \Sigma$ . Here  $[(\chi, \varphi)]$  is like the *witness* in our recursive definition, which reflects the  $\exists \Psi. \dots$  formula is our definition. The fact that measure deduction reflects proving theorems

in the theory of inequalities of probability logic is the elementary intuition behind the soundness theorem we will ultimately prove in §2.12.

A key difference from the simple motivation above is that, as in the case of Suppes' Theorem where we prove  $\sim \Gamma \vdash \sim \varphi$  if and only if  $\mathcal{P} \varphi \leq (\sum \gamma \leftarrow \Gamma . \mathcal{P} \gamma)$  for all  $\mathcal{P}$ , soundness in this context means  $\sim \Gamma \text{\$}\vdash \sim \Phi$  implies  $\forall \mathcal{P}. (\sum \gamma \leftarrow \Gamma . \mathcal{P} \gamma) \geq (\sum \varphi \leftarrow \Phi . \mathcal{P} \varphi)$ .

Another way of thinking about measure deduction is to think of  $\Gamma$  and  $\Sigma$  as bags of balls of soft clay and  $\Gamma \text{\$}\vdash \Sigma$  meaning that we have shown  $\Gamma$  is *heavier than*  $\Sigma$  (ignoring, for the moment, that  $(\text{\$}\vdash)$  is not totally ordered). We have a scale  $(\vdash)$  that lets us weigh several things on the left and one thing on the right at a time. We go through each clay ball  $\sigma$  in  $\Sigma$  one at a time without replacement, putting  $\sigma$  on the right of the scale. Then, we take a bunch of clay balls from  $\Gamma$ , cut them up as necessary (that is the  $\psi \sqcup \gamma$  trick using the witness  $\Psi$ ), and show they are heavier using our scale. We take the parts  $\psi \rightarrow \gamma$  that we didn't use and put them back in our bag  $\Gamma$ . We will be able to reuse them later. If we can do this trick for every element  $\sigma$  in  $\Sigma$  successively using combinations of split leftovers in  $\Gamma$ , then we can show  $\Gamma$  is heavier than  $\Sigma$  (i.e.,  $\Gamma \text{\$}\vdash \Sigma$ ).

## 2.2 Definition of the Stronger Theory Relation

We next turn to looking at a subrelation of  $(\text{\$}\vdash)$ , which we call the *stronger theory* relation  $(\preceq)$ . Here we construe a *theory* as a list of propositions. We say theory  $\Gamma$  is *stronger than*  $\Sigma$  where, for each element  $\sigma$  in  $\Sigma$ , we can take an element  $\gamma$  of  $\Gamma$  *without replacement* such that  $\vdash \gamma \rightarrow \sigma$ .

To motivate this notion, let's reuse the metaphor that  $\Gamma$  and  $\Sigma$  are bags of balls of clay, and we need to show  $\Gamma$  is heavier without simply weighing the two bags. A sufficient (but incomplete) approach is to take each ball of clay  $\sigma$  in  $\Sigma$  and find another ball of clay  $\gamma$  in  $\Gamma$  (without replacement) that is heavier. This simple approach avoids the complexity of iteratively cutting up balls of clay.

**definition (in implication-logic)**

*stronger-theory-relation* :: 'a list  $\Rightarrow$  'a list  $\Rightarrow$  bool (**infix**  $\preceq$  100)

**where**

$$\begin{aligned} \Sigma \preceq \Gamma = & \\ & (\exists \Phi. \text{map snd } \Phi = \Sigma \\ & \wedge \text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } \Gamma \\ & \wedge (\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma)) \end{aligned}$$

**abbreviation (in implication-logic)**

*stronger-theory-relation-op* :: 'a list  $\Rightarrow$  'a list  $\Rightarrow$  bool (**infix**  $\succeq$  100)

**where**

$$\Gamma \succeq \Sigma \equiv \Sigma \preceq \Gamma$$

## 2.3 The Stronger Theory Relation is a Preorder

Next, we show that  $(\preceq)$  is a preorder by establishing reflexivity and transitivity.

We first prove the following lemma with respect to multisets and stronger theories.

**lemma** (in *implication-logic*) *msub-stronger-theory-intro*:

**assumes**  $mset \Sigma \subseteq\# mset \Gamma$

**shows**  $\Sigma \preceq \Gamma$

**proof** –

**let**  $?\Delta\Sigma = map (\lambda x. (x,x)) \Sigma$

**have**  $map\ snd\ ?\Delta\Sigma = \Sigma$

**by** (*induct*  $\Sigma$ , *simp*, *simp*)

**moreover have**  $map\ fst\ ?\Delta\Sigma = \Sigma$

**by** (*induct*  $\Sigma$ , *simp*, *simp*)

**hence**  $mset (map\ fst\ ?\Delta\Sigma) \subseteq\# mset \Gamma$

**using** *assms* **by** *simp*

**moreover have**  $\forall (\gamma,\sigma) \in set\ ?\Delta\Sigma. \vdash \gamma \rightarrow \sigma$

**by** (*induct*  $\Sigma$ , *simp*, *simp*,

*metis list-implication.simps(1) list-implication-axiom-k*)

**ultimately show** *?thesis* **using** *stronger-theory-relation-def* **by** (*simp*, *blast*)

**qed**

The *reflexive* property immediately follows:

**lemma** (in *implication-logic*) *stronger-theory-reflexive* [*simp*]:  $\Gamma \preceq \Gamma$

**using** *msub-stronger-theory-intro* **by** *auto*

**lemma** (in *implication-logic*) *weakest-theory* [*simp*]:  $[] \preceq \Gamma$

**using** *msub-stronger-theory-intro* **by** *auto*

**lemma** (in *implication-logic*) *stronger-theory-empty-list-intro* [*simp*]:

**assumes**  $\Gamma \preceq []$

**shows**  $\Gamma = []$

**using** *assms* *stronger-theory-relation-def* **by** *simp*

Next, we turn to proving transitivity. We first prove two permutation theorems.

**lemma** (in *implication-logic*) *stronger-theory-right-permutation*:

**assumes**  $\Gamma \rightleftharpoons \Delta$

**and**  $\Sigma \preceq \Gamma$

**shows**  $\Sigma \preceq \Delta$

**proof** –

**from** *assms(1)* **have**  $mset \Gamma = mset \Delta$

**by** *simp*

**thus** *?thesis*

**using** *assms(2)* *stronger-theory-relation-def*

**by** *fastforce*



qed

**lemma** (in *implication-logic*) *stronger-theory-left-permutation*:

assumes  $\Sigma \rightleftharpoons \Delta$

and  $\Sigma \preceq \Gamma$

shows  $\Delta \preceq \Gamma$

**proof** –

have  $\forall \Sigma \Gamma. \Sigma \rightleftharpoons \Delta \longrightarrow \Sigma \preceq \Gamma \longrightarrow \Delta \preceq \Gamma$

**proof** (*induct*  $\Delta$ )

case *Nil*

then show *?case* by *simp*

**next**

case (*Cons*  $\delta$   $\Delta$ )

{

fix  $\Sigma \Gamma$

assume  $\Sigma \rightleftharpoons (\delta \# \Delta) \Sigma \preceq \Gamma$

from *this* obtain  $\Phi$  where  $\Phi$ :

map *snd*  $\Phi = \Sigma$

mset (map *fst*  $\Phi$ )  $\subseteq\#$  mset  $\Gamma$

$\forall (\gamma, \delta) \in \text{set } \Phi. \vdash \gamma \rightarrow \delta$

using *stronger-theory-relation-def* by *fastforce*

with  $\langle \Sigma \rightleftharpoons (\delta \# \Delta) \rangle$  have  $\delta \in\#$  mset (map *snd*  $\Phi$ )

by *fastforce*

from *this* obtain  $\gamma$  where  $\gamma: (\gamma, \delta) \in\#$  mset  $\Phi$

by (*induct*  $\Phi$ , *fastforce+*)

let  $\text{?}\Phi_0 = \text{remove1 } (\gamma, \delta) \Phi$

let  $\text{?}\Sigma_0 = \text{map snd } \text{?}\Phi_0$

from  $\gamma \Phi(2)$  have mset (map *fst*  $\text{?}\Phi_0$ )  $\subseteq\#$  mset (*remove1*  $\gamma$   $\Gamma$ )

by (*metis ex-mset*

*list-subtract-monotonic*

*list-subtract-mset-homomorphism*

*mset-remove1*

*remove1-pairs-list-projections-fst*)

moreover have mset  $\text{?}\Phi_0 \subseteq\#$  mset  $\Phi$  by *simp*

with  $\Phi(3)$  have  $\forall (\gamma, \delta) \in \text{set } \text{?}\Phi_0. \vdash \gamma \rightarrow \delta$  by *fastforce*

ultimately have  $\text{?}\Sigma_0 \preceq \text{remove1 } \gamma \Gamma$

unfolding *stronger-theory-relation-def* by *blast*

moreover have  $\Delta \rightleftharpoons (\text{remove1 } \delta \Sigma)$  using  $\langle \Sigma \rightleftharpoons (\delta \# \Delta) \rangle$

by (*metis perm-remove-perm perm-sym remove-hd*)

moreover from  $\gamma \Phi(1)$  have mset  $\text{?}\Sigma_0 = \text{mset } (\text{remove1 } \delta \Sigma)$

using *remove1-pairs-list-projections-snd*

by *fastforce*

hence  $\text{?}\Sigma_0 \rightleftharpoons \text{remove1 } \delta \Sigma$

by *blast*

ultimately have  $\Delta \preceq \text{remove1 } \gamma \Gamma$  using *Cons*

by *presburger*

from *this* obtain  $\Psi_0$  where  $\Psi_0$ :

map *snd*  $\Psi_0 = \Delta$

mset (map *fst*  $\Psi_0$ )  $\subseteq\#$  mset (*remove1*  $\gamma$   $\Gamma$ )

```

     $\forall (\gamma, \delta) \in \text{set } \Psi_0. \vdash \gamma \rightarrow \delta$ 
    using stronger-theory-relation-def by fastforce
  let ? $\Psi = (\gamma, \delta) \# \Psi_0$ 
  have map_snd ? $\Psi = (\delta \# \Delta)$ 
    by (simp add:  $\Psi_0(1)$ )
  moreover have mset (map fst ? $\Psi$ )  $\subseteq\#$  mset ( $\gamma \# (\text{remove1 } \gamma \Gamma)$ )
    using  $\Psi_0(2)$  by auto
  moreover from  $\gamma \Phi(3) \Psi_0(3)$  have  $\forall (\gamma, \sigma) \in \text{set } ?\Psi. \vdash \gamma \rightarrow \sigma$  by auto
  ultimately have  $(\delta \# \Delta) \preceq (\gamma \# (\text{remove1 } \gamma \Gamma))$ 
    unfolding stronger-theory-relation-def by metis
  moreover from  $\gamma \Phi(2)$  have  $\gamma \in\#$  mset  $\Gamma$ 
    using mset-subset-eqD by fastforce
  hence  $(\gamma \# (\text{remove1 } \gamma \Gamma)) \doteq \Gamma$ 
    by auto
  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 implication-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  $\Sigma$ )

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  $\Phi$  where  $\Phi$ :

map\_snd  $\Phi = \sigma \# \Sigma$

mset (map fst  $\Phi$ )  $\subseteq\#$  mset  $\Delta$

$\forall (\delta, \sigma) \in \text{set } \Phi. \vdash \delta \rightarrow \sigma$

using stronger-theory-relation-def by (simp, metis)

let  $?\delta = \text{fst } (\text{hd } \Phi)$

from  $\Phi(1)$  have  $\Phi \neq []$  by (induct  $\Phi$ , simp+)

hence  $?\delta \in\#$  mset (map fst  $\Phi$ ) by (induct  $\Phi$ , simp+)

with  $\Phi(2)$  have  $?\delta \in\#$  mset  $\Delta$  by (meson mset-subset-eqD)

hence mset (map fst (remove1 (hd  $\Phi$ )  $\Phi$ ))  $\subseteq\#$  mset (remove1  $?\delta \Delta$ )

using  $\langle \Phi \neq [] \rangle \Phi(2)$

by (simp,

metis

diff-single-eq-union

```

      hd-in-set
      image-mset-add-mset
      insert-subset-eq-iff
      set-mset-mset)
moreover have remove1 (hd  $\Phi$ )  $\Phi = tl \Phi$ 
  using  $\langle \Phi \neq [] \rangle$ 
  by (induct  $\Phi$ , simp+)
moreover from  $\Phi(1)$  have map snd (tl  $\Phi$ ) =  $\Sigma$ 
  by (simp add: map-tl)
moreover from  $\Phi(3)$  have  $\forall (\delta, \sigma) \in set (tl \Phi). \vdash \delta \rightarrow \sigma$ 
  by (simp add:  $\langle \Phi \neq [] \rangle list.set-sel(2)$ )
ultimately have  $\Sigma \preceq remove1 ?\delta \Delta$ 
  using stronger-theory-relation-def by auto
from  $\langle ?\delta \in \# mset \Delta \rangle$  have  $?\delta \# (remove1 ?\delta \Delta) \doteq \Delta$ 
  by fastforce
with  $\langle \Delta \preceq \Gamma \rangle$  have  $(?\delta \# (remove1 ?\delta \Delta)) \preceq \Gamma$ 
  using stronger-theory-left-permutation perm-sym by blast
from this obtain  $\Psi$  where  $\Psi$ :
  map snd  $\Psi = (?\delta \# (remove1 ?\delta \Delta))$ 
  mset (map fst  $\Psi$ )  $\subseteq \# mset \Gamma$ 
   $\forall (\gamma, \delta) \in set \Psi. \vdash \gamma \rightarrow \delta$ 
  using stronger-theory-relation-def by (simp, metis)
let  $? \gamma = fst (hd \Psi)$ 
from  $\Psi(1)$  have  $\Psi \neq []$  by (induct  $\Psi$ , simp+)
hence  $? \gamma \in \# mset (map fst \Psi)$  by (induct  $\Psi$ , simp+)
with  $\Psi(2)$  have  $? \gamma \in \# mset \Gamma$  by (meson mset-subset-eqD)
hence mset (map fst (remove1 (hd  $\Psi$ )  $\Psi$ ))  $\subseteq \# mset (remove1 ? \gamma \Gamma)$ 
  using  $\langle \Psi \neq [] \rangle \Psi(2)$ 
  by (simp,
    metis
      diff-single-eq-union
      hd-in-set
      image-mset-add-mset
      insert-subset-eq-iff
      set-mset-mset)
moreover from  $\langle \Psi \neq [] \rangle$  have remove1 (hd  $\Psi$ )  $\Psi = tl \Psi$ 
  by (induct  $\Psi$ , simp+)
moreover from  $\Psi(1)$  have map snd (tl  $\Psi$ ) = (remove1 ?\delta  $\Delta$ )
  by (simp add: map-tl)
moreover from  $\Psi(3)$  have  $\forall (\gamma, \delta) \in set (tl \Psi). \vdash \gamma \rightarrow \delta$ 
  by (simp add:  $\langle \Psi \neq [] \rangle list.set-sel(2)$ )
ultimately have remove1 ?\delta  $\Delta \preceq remove1 ? \gamma \Gamma$ 
  using stronger-theory-relation-def by auto
with  $\langle \Sigma \preceq remove1 ?\delta \Delta \rangle$  Cons.hyps have  $\Sigma \preceq remove1 ? \gamma \Gamma$ 
  by blast
from this obtain  $\Omega_0$  where  $\Omega_0$ :
  map snd  $\Omega_0 = \Sigma$ 
  mset (map fst  $\Omega_0$ )  $\subseteq \# mset (remove1 ? \gamma \Gamma)$ 
   $\forall (\gamma, \sigma) \in set \Omega_0. \vdash \gamma \rightarrow \sigma$ 

```

```

    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 force
  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

```

## 2.4 The Stronger Theory Relation is a Subrelation of Measure Deduction

Next, we show that  $\Gamma \succeq \Sigma$  implies  $\Gamma \text{ \$}\vdash \Sigma$ . Before doing so we establish several helpful properties regarding the stronger theory relation ( $\succeq$ ).

**lemma (in implication-logic) stronger-theory-witness:**

```

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

```

**moreover**  
**let**  $?\Phi_0 = \text{remove1 } (\gamma, \sigma) \Phi$   
**let**  $? \Sigma_0 = \text{map snd } ?\Phi_0$   
**from**  $\gamma \Phi(2)$  **have**  $\text{mset } (\text{map fst } ?\Phi_0) \subseteq\# \text{mset } (\text{remove1 } \gamma \Gamma)$   
**by** (*metis*  
*ex-mset*  
*list-subtract-monotonic*  
*list-subtract-mset-homomorphism*  
*remove1-pairs-list-projections-fst*  
*mset-remove1*)  
**moreover have**  $\text{mset } ?\Phi_0 \subseteq\# \text{mset } \Phi$  **by** *simp*  
**with**  $\Phi(3)$  **have**  $\forall (\gamma, \sigma) \in \text{set } ?\Phi_0. \vdash \gamma \rightarrow \sigma$  **by** *fastforce*  
**ultimately have**  $? \Sigma_0 \preceq \text{remove1 } \gamma \Gamma$   
**unfolding** *stronger-theory-relation-def* **by** *blast*  
**moreover from**  $\gamma \Phi(1)$  **have**  $\text{mset } ? \Sigma_0 = \text{mset } (\text{remove1 } \sigma \Sigma)$   
**using** *remove1-pairs-list-projections-snd*  
**by** *fastforce*  
**hence**  $? \Sigma_0 \Rightarrow \text{remove1 } \sigma \Sigma$   
**by** *linarith*  
**ultimately have**  $\text{remove1 } \sigma \Sigma \preceq \text{remove1 } \gamma \Gamma$   
**using** *stronger-theory-left-permutation*  
**by** *blast*  
**moreover from**  $\gamma \Phi(3)$  **have**  $\vdash \gamma \rightarrow \sigma$  **by** (*simp, fast*)  
**moreover from**  $\gamma \Phi(2)$  **have**  $\gamma \in\# \text{mset } \Gamma$   
**using** *mset-subset-eqD* **by** *fastforce*  
**ultimately show**  $\exists \gamma \in \text{set } \Gamma. \vdash \gamma \rightarrow \sigma \wedge (\text{remove1 } \sigma \Sigma) \preceq (\text{remove1 } \gamma \Gamma)$  **by**  
*auto*  
**next**  
**assume**  $\exists \gamma \in \text{set } \Gamma. \vdash \gamma \rightarrow \sigma \wedge (\text{remove1 } \sigma \Sigma) \preceq (\text{remove1 } \gamma \Gamma)$   
**from this obtain**  $\Phi \gamma$  **where**  $\gamma: \gamma \in \text{set } \Gamma \vdash \gamma \rightarrow \sigma$   
**and**  $\Phi: \text{map snd } \Phi = (\text{remove1 } \sigma \Sigma)$   
 $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } (\text{remove1 } \gamma \Gamma)$   
 $\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma$   
**unfolding** *stronger-theory-relation-def* **by** *blast*  
**let**  $? \Phi = (\gamma, \sigma) \# \Phi$   
**from**  $\Phi(1)$  **have**  $\text{map snd } ? \Phi = \sigma \# (\text{remove1 } \sigma \Sigma)$  **by** *simp*  
**moreover from**  $\Phi(2) \gamma(1)$  **have**  $\text{mset } (\text{map fst } ? \Phi) \subseteq\# \text{mset } \Gamma$   
**by** (*simp add: insert-subset-eq-iff*)  
**moreover from**  $\Phi(3) \gamma(2)$  **have**  $\forall (\gamma, \sigma) \in \text{set } ? \Phi. \vdash \gamma \rightarrow \sigma$   
**by** *auto*  
**ultimately have**  $(\sigma \# (\text{remove1 } \sigma \Sigma)) \preceq \Gamma$   
**unfolding** *stronger-theory-relation-def* **by** *metis*  
**moreover from** *assms* **have**  $\sigma \# (\text{remove1 } \sigma \Sigma) \Rightarrow \Sigma$   
**by** *force*  
**ultimately show**  $\Sigma \preceq \Gamma$   
**using** *stronger-theory-left-permutation* **by** *blast*  
**qed**

lemma (in *implication-logic*) *stronger-theory-cons-witness*:

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

**lemma** (in *implication-logic*) *stronger-theory-left-cons*:

**assumes**  $(\sigma \# \Sigma) \preceq \Gamma$

**shows**  $\Sigma \preceq \Gamma$

**proof** –

**from** *assms* **obtain**  $\Phi$  **where**  $\Phi$ :

*map snd*  $\Phi = \sigma \# \Sigma$

*mset (map fst*  $\Phi) \subseteq \# \text{mset } \Gamma$

$\forall (\delta, \sigma) \in \text{set } \Phi. \vdash \delta \rightarrow \sigma$

**using** *stronger-theory-relation-def* **by** (*simp, metis*)

**let**  $?\Phi' = \text{remove1 } (\text{hd } \Phi) \Phi$

**from**  $\Phi(1)$  **have** *map snd*  $?\Phi' = \Sigma$  **by** (*induct*  $\Phi$ , *simp+*)

**moreover from**  $\Phi(2)$  **have** *mset (map fst*  $?\Phi') \subseteq \# \text{mset } \Gamma$

**by** (*metis diff-subset-eq-self*

*list-subtract.simps(1)*

*list-subtract.simps(2)*

*list-subtract-mset-homomorphism*

*map-monotonic*

*subset-mset.dual-order.trans*)

**moreover from**  $\Phi(3)$  **have**  $\forall (\delta, \sigma) \in \text{set } ?\Phi'. \vdash \delta \rightarrow \sigma$  **by** *fastforce*

**ultimately show** *?thesis* **unfolding** *stronger-theory-relation-def* **by** *blast*

**qed**

**lemma** (in *implication-logic*) *stronger-theory-right-cons*:

**assumes**  $\Sigma \preceq \Gamma$

**shows**  $\Sigma \preceq (\gamma \# \Gamma)$

**proof** –

**from** *assms* **obtain**  $\Phi$  **where**  $\Phi$ :

*map snd*  $\Phi = \Sigma$

*mset (map fst*  $\Phi) \subseteq \# \text{mset } \Gamma$

$\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma$

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

**by** *auto*

**hence** *mset (map fst*  $\Phi) \subseteq \# \text{mset } (\gamma \# \Gamma)$

**by** (*metis Diff-eq-empty-iff-mset*

*list-subtract.simps(2)*

*list-subtract-mset-homomorphism*

*mset-zero-iff remove1.simps(1)*)

**with**  $\Phi(1)$   $\Phi(3)$  **show** *?thesis*

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

**by** *auto*

qed

**lemma** (in *implication-logic*) *stronger-theory-left-right-cons*:

**assumes**  $\vdash \gamma \rightarrow \sigma$   
**and**  $\Sigma \preceq \Gamma$   
**shows**  $(\sigma \# \Sigma) \preceq (\gamma \# \Gamma)$

**proof** –

**from** *assms*(2) **obtain**  $\Phi$  **where**  $\Phi$ :  
   $\text{map snd } \Phi = \Sigma$   
   $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } \Gamma$   
   $\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma$   
**unfolding** *stronger-theory-relation-def*  
**by** *auto*  
**let**  $?\Phi = (\gamma, \sigma) \# \Phi$   
**from** *assms*(1)  $\Phi$  **have**  
   $\text{map snd } ?\Phi = \sigma \# \Sigma$   
   $\text{mset } (\text{map fst } ?\Phi) \subseteq\# \text{mset } (\gamma \# \Gamma)$   
   $\forall (\gamma, \sigma) \in \text{set } ?\Phi. \vdash \gamma \rightarrow \sigma$   
**by** *fastforce+*  
**thus** *?thesis*  
**unfolding** *stronger-theory-relation-def*  
**by** *metis*

qed

**lemma** (in *implication-logic*) *stronger-theory-relation-alt-def*:

$\Sigma \preceq \Gamma = (\exists \Phi. \text{mset } (\text{map snd } \Phi) = \text{mset } \Sigma \wedge$   
   $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } \Gamma \wedge$   
   $(\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma))$

**proof** (*induct*  $\Gamma$  *arbitrary*:  $\Sigma$ )

**case** *Nil*  
  **then show** *?case*  
    **using** *stronger-theory-empty-list-intro*  
      *stronger-theory-reflexive*  
    **by** (*simp, blast*)

**next**

**case** (*Cons*  $\gamma$   $\Gamma$ )  
**have**  $\Sigma \preceq (\gamma \# \Gamma) = (\exists \Phi. \text{mset } (\text{map snd } \Phi) = \text{mset } \Sigma \wedge$   
   $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } (\gamma \# \Gamma) \wedge$   
   $(\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma))$

**proof** (*rule iffI*)

**assume**  $\Sigma \preceq (\gamma \# \Gamma)$   
**thus**  $\exists \Phi. \text{mset } (\text{map snd } \Phi) = \text{mset } \Sigma \wedge$   
   $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } (\gamma \# \Gamma) \wedge$   
   $(\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma)$   
**unfolding** *stronger-theory-relation-def*  
**by** *metis*

**next**

**assume**  $\exists \Phi. \text{mset } (\text{map snd } \Phi) = \text{mset } \Sigma \wedge$   
   $\text{mset } (\text{map fst } \Phi) \subseteq\# \text{mset } (\gamma \# \Gamma) \wedge$

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



```

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

lemma (in implication-logic) stronger-theory-deduction-monotonic:
  assumes  $\Sigma \preceq \Gamma$ 
    and  $\Sigma \vdash \varphi$ 
    shows  $\Gamma \vdash \varphi$ 
using assms
proof (induct  $\Sigma$  arbitrary:  $\varphi$ )
  case Nil
    then show ?case
      by (simp add: list-deduction-weaken)
  next
    case (Cons  $\sigma$   $\Sigma$ )
    assume  $(\sigma \# \Sigma) \preceq \Gamma$   $(\sigma \# \Sigma) \vdash \varphi$ 
    hence  $\Sigma \vdash \sigma \rightarrow \varphi$   $\Sigma \preceq \Gamma$ 
      using
        list-deduction-theorem
        stronger-theory-left-cons
      by (blast, metis)
    with Cons have  $\Gamma \vdash \sigma \rightarrow \varphi$  by blast
    moreover
    have  $\sigma \in \text{set } (\sigma \# \Sigma)$  by auto
    with  $\langle \sigma \# \Sigma \rangle \preceq \Gamma$  obtain  $\gamma$  where  $\gamma \in \text{set } \Gamma \vdash \gamma \rightarrow \sigma$ 
      using stronger-theory-witness by blast
    hence  $\Gamma \vdash \sigma$ 
      using
        list-deduction-modus-ponens
        list-deduction-reflection
        list-deduction-weaken
      by blast
    ultimately have  $\Gamma \vdash \varphi$ 

```

```

    using list-deduction-modus-ponens by blast
  then show ?case by blast
qed

lemma (in classical-logic) measure-msub-left-monotonic:
  assumes mset  $\Sigma \subseteq\#$  mset  $\Gamma$ 
    and  $\Sigma \ \$\vdash \Phi$ 
  shows  $\Gamma \ \$\vdash \Phi$ 
  using assms
proof (induct  $\Phi$  arbitrary:  $\Sigma \ \Gamma$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\varphi \ \Phi$ )
  from this obtain  $\Psi$  where  $\Psi$ :
    mset (map snd  $\Psi$ )  $\subseteq\#$  mset  $\Sigma$ 
    map (uncurry ( $\sqcup$ ))  $\Psi \ \vdash \varphi$ 
    map (uncurry ( $\rightarrow$ ))  $\Psi \ @ \ \Sigma \ \ominus \ (\text{map snd } \Psi) \ \$\vdash \Phi$ 
  using measure-deduction.simps(2) by blast
  let ? $\Psi$  = map snd  $\Psi$ 
  let ? $\Psi'$  = map (uncurry ( $\rightarrow$ ))  $\Psi$ 
  let ? $\Sigma'$  = ? $\Psi' \ @ \ (\Sigma \ \ominus \ ?\Psi)$ 
  let ? $\Gamma'$  = ? $\Psi' \ @ \ (\Gamma \ \ominus \ ?\Psi)$ 
  from  $\Psi$  have mset ? $\Psi \subseteq\#$  mset  $\Gamma$ 
    using  $\langle$ mset  $\Sigma \subseteq\#$  mset  $\Gamma \rangle$  subset-mset.trans by blast
  moreover have mset ( $\Sigma \ \ominus \ ?\Psi$ )  $\subseteq\#$  mset ( $\Gamma \ \ominus \ ?\Psi$ )
    by (metis  $\langle$ mset  $\Sigma \subseteq\#$  mset  $\Gamma \rangle$  list-subtract-monotonic)
  hence mset ? $\Sigma' \subseteq\#$  mset ? $\Gamma'$ 
    by simp
  with Cons.hyps  $\Psi(3)$  have ? $\Gamma' \ \$\vdash \Phi$  by blast
  ultimately have  $\Gamma \ \$\vdash (\varphi \ \# \ \Phi)$ 
    using  $\Psi(2)$  by fastforce
  then show ?case
    by simp
qed

```

```

lemma (in classical-logic) witness-weaker-theory:
  assumes mset (map snd  $\Sigma$ )  $\subseteq\#$  mset  $\Gamma$ 
  shows map (uncurry ( $\sqcup$ ))  $\Sigma \ \preceq \ \Gamma$ 
proof -
  have  $\forall \ \Gamma. \ \text{mset} \ (\text{map snd } \Sigma) \ \subseteq\# \ \text{mset } \Gamma \ \longrightarrow \ \text{map} \ (\text{uncurry } (\sqcup)) \ \Sigma \ \preceq \ \Gamma$ 
  proof (induct  $\Sigma$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\sigma \ \Sigma$ )
    {
      fix  $\Gamma$ 
      assume mset (map snd ( $\sigma \ \# \ \Sigma$ ))  $\subseteq\#$  mset  $\Gamma$ 

```

```

hence  $mset (map\ snd\ \Sigma) \subseteq\# mset (remove1 (snd\ \sigma)\ \Gamma)$ 
  by (simp add: insert-subset-eq-iff)
with Cons have  $map (uncurry (\sqcup))\ \Sigma \preceq remove1 (snd\ \sigma)\ \Gamma$  by blast
moreover have  $uncurry (\sqcup) = (\lambda\ \sigma.\ fst\ \sigma \sqcup snd\ \sigma)$  by fastforce
hence  $uncurry (\sqcup)\ \sigma = fst\ \sigma \sqcup snd\ \sigma$  by simp
moreover have  $\vdash\ snd\ \sigma \rightarrow (fst\ \sigma \sqcup snd\ \sigma)$ 
  unfolding disjunction-def
  by (simp add: axiom-k)
ultimately have  $map (uncurry (\sqcup)) (\sigma\ \#\ \Sigma) \preceq (snd\ \sigma\ \#\ (remove1 (snd\ \sigma)\ \Gamma))$ 
 $\Gamma))$ 
  by (simp add: stronger-theory-left-right-cons)
moreover have  $mset (snd\ \sigma\ \#\ (remove1 (snd\ \sigma)\ \Gamma)) = mset\ \Gamma$ 
  using  $\langle mset (map\ snd (\sigma\ \#\ \Sigma)) \subseteq\# mset\ \Gamma \rangle$ 
  by (simp, meson insert-DiffM mset-subset-eq-insertD)
ultimately have  $map (uncurry (\sqcup)) (\sigma\ \#\ \Sigma) \preceq \Gamma$ 
  unfolding stronger-theory-relation-alt-def
  by simp
}
then show ?case by blast
qed
with assms show ?thesis by simp
qed

```

**lemma** (in *implication-logic*) *stronger-theory-combine*:

```

assumes  $\Phi \preceq \Delta$ 
and  $\Psi \preceq \Gamma$ 
shows  $(\Phi\ @\ \Psi) \preceq (\Delta\ @\ \Gamma)$ 

```

**proof** –

```

have  $\forall\ \Phi.\ \Phi \preceq \Delta \longrightarrow (\Phi\ @\ \Psi) \preceq (\Delta\ @\ \Gamma)$ 

```

```

proof (induct  $\Delta$ )

```

```

case Nil

```

```

then show ?case

```

```

  using assms(2) stronger-theory-empty-list-intro by fastforce

```

**next**

```

case (Cons  $\delta\ \Delta$ )

```

```

{

```

```

  fix  $\Phi$ 

```

```

  assume  $\Phi \preceq (\delta\ \#\ \Delta)$ 

```

```

  from this obtain  $\Sigma$  where  $\Sigma$ :

```

```

     $map\ snd\ \Sigma = \Phi$ 

```

```

     $mset (map\ fst\ \Sigma) \subseteq\# mset (\delta\ \#\ \Delta)$ 

```

```

     $\forall (\delta, \varphi) \in set\ \Sigma.\ \vdash\ \delta \rightarrow \varphi$ 

```

```

    unfolding stronger-theory-relation-def

```

```

    by blast

```

```

  have  $(\Phi\ @\ \Psi) \preceq ((\delta\ \#\ \Delta)\ @\ \Gamma)$ 

```

```

  proof (cases  $\exists\ \varphi.\ (\delta, \varphi) \in set\ \Sigma$ )

```

```

    assume  $\exists\ \varphi.\ (\delta, \varphi) \in set\ \Sigma$ 

```

```

    from this obtain  $\varphi$  where  $\varphi: (\delta, \varphi) \in set\ \Sigma$  by auto

```

```

    let  $?\Sigma = remove1 (\delta, \varphi)\ \Sigma$ 

```

**from**  $\varphi \Sigma(1)$  **have**  $mset (map\ snd\ ?\Sigma) = mset (remove1\ \varphi\ \Phi)$   
**using** *remove1-pairs-list-projections-snd* **by** *fastforce*  
**moreover from**  $\varphi$  **have**  $mset (map\ fst\ ?\Sigma) = mset (remove1\ \delta (map\ fst\ \Sigma))$   
**using** *remove1-pairs-list-projections-fst* **by** *fastforce*  
**hence**  $mset (map\ fst\ ?\Sigma) \subseteq\# mset\ \Delta$   
**using**  $\Sigma(2)$  *mset.simps(1)* *subset-eq-diff-conv* **by** *force*  
**moreover from**  $\Sigma(3)$  **have**  $\forall (\delta, \varphi) \in set\ ?\Sigma. \vdash \delta \rightarrow \varphi$  **by** *auto*  
**ultimately have**  $remove1\ \varphi\ \Phi \preceq\ \Delta$   
**unfolding** *stronger-theory-relation-alt-def* **by** *blast*  
**hence**  $(remove1\ \varphi\ \Phi @ \Psi) \preceq (\Delta @ \Gamma)$  **using** *Cons* **by** *auto*  
**from this obtain**  $\Omega$  **where**  $\Omega$ :  
 $map\ snd\ \Omega = (remove1\ \varphi\ \Phi) @ \Psi$   
 $mset (map\ fst\ \Omega) \subseteq\# mset (\Delta @ \Gamma)$   
 $\forall (\alpha, \beta) \in set\ \Omega. \vdash \alpha \rightarrow \beta$   
**unfolding** *stronger-theory-relation-def*  
**by** *blast*  
**let**  $?\Omega = (\delta, \varphi) \# \Omega$   
**have**  $map\ snd\ ?\Omega = \varphi \# remove1\ \varphi\ \Phi @ \Psi$   
**using**  $\Omega(1)$  **by** *simp*  
**moreover have**  $mset (map\ fst\ ?\Omega) \subseteq\# mset ((\delta \# \Delta) @ \Gamma)$   
**using**  $\Omega(2)$  **by** *simp*  
**moreover have**  $\vdash \delta \rightarrow \varphi$   
**using**  $\Sigma(3)$   $\varphi$  **by** *blast*  
**hence**  $\forall (\alpha, \beta) \in set\ ?\Omega. \vdash \alpha \rightarrow \beta$  **using**  $\Omega(3)$  **by** *auto*  
**ultimately have**  $(\varphi \# remove1\ \varphi\ \Phi @ \Psi) \preceq ((\delta \# \Delta) @ \Gamma)$   
**by** (*metis stronger-theory-relation-def*)  
**moreover have**  $\varphi \in set\ \Phi$   
**using**  $\Sigma(1)$   $\varphi$  **by** *force*  
**hence**  $(\varphi \# remove1\ \varphi\ \Phi) \equiv \Phi$   
**by** *force*  
**hence**  $(\varphi \# remove1\ \varphi\ \Phi @ \Psi) \equiv \Phi @ \Psi$   
**by** (*metis append-Cons perm-append2*)  
**ultimately show**  $?thesis$   
**using** *stronger-theory-left-permutation* **by** *blast*

**next**  
**assume**  $\nexists \varphi. (\delta, \varphi) \in set\ \Sigma$   
**hence**  $\delta \notin set (map\ fst\ \Sigma)$   
 $mset\ \Delta + add\ mset\ \delta (mset\ []) = mset (\delta \# \Delta)$   
**by** *auto*  
**hence**  $mset (map\ fst\ \Sigma) \subseteq\# mset\ \Delta$   
**by** (*metis (no-types) <mset (map fst Σ) ⊆# mset (δ # Δ)>*  
*diff-single-trivial*  
*mset.simps(1)*  
*set-mset-mset*  
*subset-eq-diff-conv*)  
**with**  $\Sigma(1)$   $\Sigma(3)$  **have**  $\Phi \preceq\ \Delta$   
**unfolding** *stronger-theory-relation-def*  
**by** *blast*  
**hence**  $(\Phi @ \Psi) \preceq (\Delta @ \Gamma)$  **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

```

We now turn to proving that  $(\succeq)$  is a subrelation of  $(:\vdash)$ .

**lemma** (in *classical-logic*) *stronger-theory-to-measure-deduction*:

```

  assumes  $\Gamma \succeq \Sigma$ 
  shows  $\Gamma \text{ \$}\vdash \Sigma$ 
  proof -
    have  $\forall \Gamma. \Sigma \preceq \Gamma \longrightarrow \Gamma \text{ \$}\vdash \Sigma$ 
    proof (induct  $\Sigma$ )
      case Nil
      then show ?case by fastforce
    next
      case (Cons  $\sigma \Sigma$ )
      {
        fix  $\Gamma$ 
        assume  $(\sigma \# \Sigma) \preceq \Gamma$ 
        from this obtain  $\gamma$  where  $\gamma: \gamma \in \text{set } \Gamma \vdash \gamma \rightarrow \sigma \Sigma \preceq (\text{remove1 } \gamma \Gamma)$ 
          using stronger-theory-cons-witness by blast
        let  $?\Phi = [(\gamma, \gamma)]$ 
        from  $\gamma$  Cons have  $(\text{remove1 } \gamma \Gamma) \text{ \$}\vdash \Sigma$  by blast
        moreover have  $\text{mset } (\text{remove1 } \gamma \Gamma) \subseteq\# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Phi @ \Gamma$ 
           $\ominus (\text{map } \text{snd } \text{ ?}\Phi))$ 
          by simp
        ultimately have  $\text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Phi @ \Gamma \ominus (\text{map } \text{snd } \text{ ?}\Phi) \text{ \$}\vdash \Sigma$ 
          using measure-msub-left-monotonic by blast
        moreover have  $\text{map } (\text{uncurry } (\sqcup)) \text{ ?}\Phi \text{ :\vdash } \sigma$ 
          by (simp, metis  $\gamma(2)$ 
              Peirces-law
              disjunction-def
              list-deduction-def
              list-deduction-modus-ponens
              list-deduction-weaken
              list-implication.simps(1)
              list-implication.simps(2))
        moreover from  $\gamma(1)$  have  $\text{mset } (\text{map } \text{snd } \text{ ?}\Phi) \subseteq\# \text{mset } \Gamma$  by simp
        ultimately have  $\Gamma \text{ \$}\vdash (\sigma \# \Sigma)$ 
          using measure-deduction.simps(2) by blast
      }
    then show ?case by blast
  qed
  thus ?thesis using assms by blast
  qed

```

## 2.5 Measure Deduction is a Preorder

We next show that measure deduction is a preorder.

Reflexivity follows immediately because  $(\preceq)$  is a subrelation and is itself reflexive.

**theorem** (in *classical-logic*) *measure-reflexive*:  $\Gamma \text{ \$}\vdash \Gamma$   
 by (*simp add: stronger-theory-to-measure-deduction*)

Transitivity is complicated. It requires constructing many witnesses and involves a lot of metatheorems. Below we provide various witness constructions that allow us to establish  $\llbracket \Gamma \text{ \$}\vdash \Lambda; \Lambda \text{ \$}\vdash \Delta \rrbracket \implies \Gamma \text{ \$}\vdash \Delta$ .

**primrec** (in *implication-logic*)

*first-component* ::  $('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list}$  ( $\mathfrak{A}$ )

**where**

$\mathfrak{A} \Psi [] = []$

|  $\mathfrak{A} \Psi (\delta \# \Delta) =$

(*case find*  $(\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi$  of

*None*  $\Rightarrow \mathfrak{A} \Psi \Delta$

| *Some*  $\psi \Rightarrow \psi \# (\mathfrak{A} (\text{remove1 } \psi \Psi) \Delta))$ )

**primrec** (in *implication-logic*)

*second-component* ::  $('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list}$  ( $\mathfrak{B}$ )

**where**

$\mathfrak{B} \Psi [] = []$

|  $\mathfrak{B} \Psi (\delta \# \Delta) =$

(*case find*  $(\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi$  of

*None*  $\Rightarrow \mathfrak{B} \Psi \Delta$

| *Some*  $\psi \Rightarrow \delta \# (\mathfrak{B} (\text{remove1 } \psi \Psi) \Delta))$ )

**lemma** (in *implication-logic*) *first-component-second-component-mset-connection*:

$mset (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{A} \Psi \Delta)) = mset (\text{map } \text{snd } (\mathfrak{B} \Psi \Delta))$

**proof** –

**have**  $\forall \Psi. mset (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{A} \Psi \Delta)) = mset (\text{map } \text{snd } (\mathfrak{B} \Psi \Delta))$

**proof** (*induct*  $\Delta$ )

**case** *Nil*

**then show** *?case* by *simp*

**next**

**case** (*Cons*  $\delta \Delta$ )

{

**fix**  $\Psi$

**have**  $mset (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{A} \Psi (\delta \# \Delta))) =$

$mset (\text{map } \text{snd } (\mathfrak{B} \Psi (\delta \# \Delta)))$

**proof** (*cases find*  $(\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi = \text{None}$ )

**case** *True*

**then show** *?thesis* using *Cons* by *simp*

**next**

**case** *False*

```

    from this obtain  $\psi$  where
      find  $(\lambda\psi. \text{uncurry } (\rightarrow) \psi = \text{snd } \delta) \Psi = \text{Some } \psi$ 
      uncurry  $(\rightarrow) \psi = \text{snd } \delta$ 
      using find-Some-predicate
      by fastforce
    then show ?thesis using Cons by simp
  qed
}
then show ?case by blast
qed
thus ?thesis by blast
qed

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

lemma (in implication-logic) first-component-msub:
   $\text{mset } (\mathfrak{A} \ \Psi \ \Delta) \subseteq\# \text{mset } \Psi$ 
proof -
  have  $\forall \Psi. \text{mset } (\mathfrak{A} \ \Psi \ \Delta) \subseteq\# \text{mset } \Psi$ 
  proof (induct  $\Delta$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\delta \ \Delta$ )
    {
      fix  $\Psi$ 
      have  $\text{mset } (\mathfrak{A} \ \Psi \ (\delta \# \ \Delta)) \subseteq\# \text{mset } \Psi$ 
      proof (cases find  $(\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi = \text{None}$ )
        case True
        then show ?thesis using Cons by simp
      next
        case False
        from this obtain  $\psi$  where
           $\psi: \text{find } (\lambda\psi. \text{uncurry } (\rightarrow) \psi = \text{snd } \delta) \Psi = \text{Some } \psi$ 
           $\psi \in \text{set } \Psi$ 
          using find-Some-set-membership
          by fastforce
        have  $\text{mset } (\mathfrak{A} \ (\text{remove1 } \psi \ \Psi) \ \Delta) \subseteq\# \text{mset } (\text{remove1 } \psi \ \Psi)$ 
          using Cons by metis
        thus ?thesis using  $\psi$  by (simp add: insert-subset-eq-iff)
      qed
    }
    then show ?case by blast
  qed
  thus ?thesis by blast
qed

```

```

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

lemma (in implication-logic) second-component-snd-projection-msub:
  mset (map snd (B Ψ Δ)) ⊆# mset (map (uncurry (→)) Ψ)
proof –
  have ∀ Ψ. mset (map snd (B Ψ Δ)) ⊆# mset (map (uncurry (→)) Ψ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have mset (map snd (B Ψ (δ # Δ))) ⊆# mset (map (uncurry (→)) Ψ)
      proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
        case True
        then show ?thesis
          using Cons by simp
      next
        case False
        from this obtain ψ where ψ:
          find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = Some ψ
          by auto
        hence B Ψ (δ # Δ) = δ # (B (remove1 ψ Ψ) Δ)
          using ψ by fastforce
        with Cons have mset (map snd (B Ψ (δ # Δ))) ⊆#

```



```

      mset ((snd δ) # map (uncurry (→)) (remove1 ψ Ψ))
    by (simp, metis mset-map mset-remove1)
  moreover from ψ have snd δ = (uncurry (→)) ψ
    using find-Some-predicate by fastforce
  ultimately have
    mset (map snd (ℬ Ψ (δ # Δ))) ⊆#
      mset (map (uncurry (→)) (ψ # (remove1 ψ Ψ)))
    by simp
  thus ?thesis
    by (metis
        first-component-msub
        first-component-second-component-mset-connection
        map-monotonic)
  qed
}
thus ?case by blast
qed
thus ?thesis by blast
qed

lemma (in implication-logic) second-component-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
      }
    }
  qed

```

**moreover have**  
 $mset (map\ snd\ \Delta)$   
 $\subseteq\# mset (map (uncurry (\rightarrow)) \Psi @ \Gamma \ominus map\ snd\ \Psi) - \{\#snd\ \delta\# \}$   
**using**  $\diamond$  *insert-subset-eq-iff* **by** *fastforce*

**ultimately have**  
 $mset (map\ snd\ \Delta)$   
 $\subseteq\# mset (map (uncurry (\rightarrow)) \Psi @ (remove1 (snd\ \delta)\ \Gamma) \ominus map\ snd\ \Psi)$   
**by** (*metis* (*no-types*)  
*mset-remove1*  
*union-code*  
*list-subtract.simps(2)*  
*list-subtract-remove1-cons-perm*  
*remove1-append*)

**hence**  $B: mset (map\ snd (\Delta \ominus (\mathfrak{B}\ \Psi\ \Delta))) \subseteq\# mset (remove1 (snd\ \delta)\ \Gamma \ominus (map\ snd\ \Psi))$   
**using** *Cons* **by** *blast*

**have**  $C: snd\ \delta \in\# mset (snd\ \delta \# map\ snd\ \Delta @ (map (uncurry (\rightarrow)) \Psi @ \Gamma \ominus map\ snd\ \Psi) \ominus (snd\ \delta \# map\ snd\ \Delta))$   
**by** (*meson* *in-multiset-in-set* *list.set-intros(1)*)

**have**  $mset (map\ snd (\delta \# \Delta))$   
 $+ (mset (map (uncurry (\rightarrow)) \Psi @ \Gamma \ominus map\ snd\ \Psi) - mset (map\ snd (\delta \# \Delta)))$   
 $= mset (map (uncurry (\rightarrow)) \Psi @ \Gamma \ominus map\ snd\ \Psi)$   
**using**  $\diamond$  *subset-mset.add-diff-inverse* **by** *blast*

**then have**  $snd\ \delta \in\# mset (map (uncurry (\rightarrow)) \Psi) + (mset\ \Gamma - mset (map\ snd\ \Psi))$   
**using**  $C$  **by** *simp*

**with**  $A$  **have**  $snd\ \delta \in set\ \Gamma$   
**by** (*metis* (*no-types*) *diff-subset-eq-self*  
*in-multiset-in-set*  
*subset-mset.add-diff-inverse*  
*union-iff*)

**have**  $D: \mathfrak{B}\ \Psi\ \Delta = \mathfrak{B}\ \Psi (\delta \# \Delta)$   
**using**  $\langle find (\lambda\psi. uncurry (\rightarrow) \psi = snd\ \delta) \Psi = None \rangle$   
**by** *simp*

**obtain**  $diff :: 'a\ list \Rightarrow 'a\ list \Rightarrow 'a\ list$  **where**  
 $\forall x0\ x1. (\exists v2. x1 @ v2 \rightleftharpoons x0) = (x1 @ diff\ x0\ x1 \rightleftharpoons x0)$   
**by** *moura*

**then have**  $E:$   
 $mset (map\ snd (\mathfrak{B}\ \Psi (\delta \# \Delta)))$   
 $@\ diff (map (uncurry (\rightarrow)) \Psi) (map\ snd (\mathfrak{B}\ \Psi (\delta \# \Delta)))$   
 $= mset (map (uncurry (\rightarrow)) \Psi)$   
**by** (*meson* *second-component-snd-projection-msub* *mset-le-perm-append*)

**have**  $F: \forall a\ m\ ma. (add-mset (a::'a) m \subseteq\# ma) = (a \in\# ma \wedge m \subseteq\# ma - \{\#a\# \})$   
**using** *insert-subset-eq-iff* **by** *blast*

**then have**  $snd\ \delta \in\# mset (map\ snd (\mathfrak{B}\ \Psi (\delta \# \Delta)))$

```

    @ diff (map (uncurry (→)) Ψ) (map snd (ℬ Ψ (δ #
Δ))))
      + mset (Γ ⊖ map snd Ψ)
    using E ◇ by force
  then have snd δ ∈# mset (Γ ⊖ map snd Ψ)
    using A E by (metis (no-types) in-multiset-in-set union-iff)
  then have G: add-mset (snd δ) (mset (map snd (Δ ⊖ ℬ Ψ Δ))) ⊆# mset
(Γ ⊖ map snd Ψ)
    using B F by force
  have H: ∀ ps psa f. ¬ mset (ps::('a × 'a) list) ⊆# mset psa ∨
      mset ((map f psa::'a list) ⊖ map f ps) = mset (map f (psa
⊖ ps))
    using map-list-subtract-mset-equivalence by blast
  have snd δ ∉# mset (map snd (ℬ Ψ (δ # Δ)))
    + mset (diff (map (uncurry (→)) Ψ) (map snd (ℬ Ψ (δ # Δ))))
    using A E by auto
  then have add-mset (snd δ) (mset (map snd (Δ ⊖ ℬ Ψ Δ)))
    = mset (map snd (δ # Δ) ⊖ map snd (ℬ Ψ (δ # Δ)))
    using D H second-component-msub by auto
  then show ?thesis
    using G H by (metis (no-types) second-component-msub)
next
case False
from this obtain ψ where ψ: find (λψ. uncurry (→) ψ = snd δ) Ψ = Some
ψ
  by auto
let ?Ψ' = remove1 ψ Ψ
let ?Γ' = remove1 (snd ψ) Γ
have snd δ = uncurry (→) ψ
  ψ ∈ set Ψ
  mset ((δ # Δ) ⊖ ℬ Ψ (δ # Δ)) =
  mset (Δ ⊖ ℬ ?Ψ' Δ)
  using ψ find-Some-predicate find-Some-set-membership
  by fastforce+
moreover
have mset (Γ ⊖ map snd Ψ) = mset (?Γ' ⊖ map snd ?Ψ')
  by (simp, metis ⟨ψ ∈ set Ψ⟩ image-mset-add-mset in-multiset-in-set
insert-DiffM)
moreover
obtain search :: ('a × 'a) list ⇒ ('a × 'a ⇒ bool) ⇒ 'a × 'a where
  ∀ xs P. (∃ x. x ∈ set xs ∧ P x) = (search xs P ∈ set xs ∧ P (search xs P))
  by moura
then have ∀ p ps. (find p ps ≠ None ∨ (∀ pa. pa ∉ set ps ∨ ¬ p pa))
  ∧ (find p ps = None ∨ search ps p ∈ set ps ∧ p (search ps p))
  by (metis (full-types) find-None-iff)
then have (find (λp. uncurry (→) p = snd δ) Ψ ≠ None
  ∨ (∀ p. p ∉ set Ψ ∨ uncurry (→) p ≠ snd δ))
  ∧ (find (λp. uncurry (→) p = snd δ) Ψ = None
  ∨ search Ψ (λp. uncurry (→) p = snd δ) ∈ set Ψ

```



```

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

```

**lemma** (in *classical-logic*) *second-component-merge-witness-stronger-theory*:  
 $(\text{map } (\text{uncurry } (\rightarrow)) \Delta @ \text{map } (\text{uncurry } (\rightarrow)) \Psi \ominus \text{map } \text{snd } (\mathfrak{B} \Psi \Delta)) \preceq$   
 $\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta)$

**proof** –

**have**  $\forall \Psi. (\text{map } (\text{uncurry } (\rightarrow)) \Delta @$   
 $\text{map } (\text{uncurry } (\rightarrow)) \Psi \ominus \text{map } \text{snd } (\mathfrak{B} \Psi \Delta)) \preceq$   
 $\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta)$

**proof** (*induct*  $\Delta$ )

**case** *Nil*

**then show** *?case*

**by** *simp*

**next**

**case** (*Cons*  $\delta \Delta$ )

{

**fix**  $\Psi$

**have**  $\vdash (\text{uncurry } (\rightarrow)) \delta \rightarrow (\text{uncurry } (\rightarrow)) \delta$

**using** *axiom-k modus-ponens implication-absorption* **by** *blast*

**have**

$(\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$   
 $\text{map } (\text{uncurry } (\rightarrow)) \Psi \ominus \text{map } \text{snd } (\mathfrak{B} \Psi (\delta \# \Delta))) \preceq$   
 $\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi (\delta \# \Delta))$

**proof** (*cases find*  $(\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi = \text{None}$ )

**case** *True*

**thus** *?thesis*

**using** *Cons*

$\langle \vdash (\text{uncurry } (\rightarrow)) \delta \rightarrow (\text{uncurry } (\rightarrow)) \delta \rangle$

**by** (*simp, metis stronger-theory-left-right-cons*)

**next**

**case** *False*

**from this obtain**  $\psi$  **where**  $\psi: \text{find } (\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta) \Psi = \text{Some}$

$\psi$

**by** *auto*

**from**  $\psi$  **have**  $\text{snd } \delta = \text{uncurry } (\rightarrow) \psi$

**using** *find-Some-predicate* **by** *fastforce*

**from**  $\psi \langle \text{snd } \delta = \text{uncurry } (\rightarrow) \psi \rangle$  **have**

$\text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$   
 $\text{map } (\text{uncurry } (\rightarrow)) \Psi \ominus \text{map } \text{snd } (\mathfrak{B} \Psi (\delta \# \Delta))) =$   
 $\text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$   
 $\text{map } (\text{uncurry } (\rightarrow)) (\text{remove1 } \psi \Psi) \ominus$   
 $\text{map } \text{snd } (\mathfrak{B} (\text{remove1 } \psi \Psi) \Delta))$

**by** (*simp add: find-Some-set-membership image-mset-Diff*)

**hence**

$(\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$   
 $\text{map } (\text{uncurry } (\rightarrow)) \Psi \ominus \text{map } \text{snd } (\mathfrak{B} \Psi (\delta \# \Delta))) \preceq$   
 $(\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$   
 $\text{map } (\text{uncurry } (\rightarrow)) (\text{remove1 } \psi \Psi) \ominus \text{map } \text{snd } (\mathfrak{B} (\text{remove1 } \psi \Psi) \Delta))$

**by** (*simp add: msub-stronger-theory-intro*)

**with** *Cons*  $\langle \vdash (\text{uncurry } (\rightarrow)) \delta \rightarrow (\text{uncurry } (\rightarrow)) \delta \rangle$  **have**

$(\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) @$

```

      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-logic) merge-witness-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 Ψ))

```

```

have mset (map snd (⋈ Ψ (δ # Δ))) ⊆# mset Γ
proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
  case True
  hence snd δ ∉ set (map (uncurry (→)) Ψ)
  proof (induct Ψ)
    case Nil
    then show ?case by simp
  next
  case (Cons ψ Ψ)
  hence uncurry (→) ψ ≠ snd δ by fastforce
  with Cons show ?case by fastforce
qed
with ◇(2) have snd δ ∈# mset (Γ ⊖ map snd Ψ)
  using mset-subset-eq-insertD by fastforce
with ◇(1) have mset (map snd Ψ) ⊆# mset (remove1 (snd δ) Γ)
  by (metis list-subtract-mset-homomorphism
    mset-remove1
    single-subset-iff
    subset-mset.add-diff-assoc
    subset-mset.add-diff-inverse
    subset-mset.le-iff-add)
moreover
have add-mset (snd δ) (mset (Γ ⊖ map snd Ψ) - {#snd δ#}) = mset (Γ
⊖ map snd Ψ)
  by (meson ⟨snd δ ∈# mset (Γ ⊖ map snd Ψ)⟩ insert-DiffM)
  then have image-mset snd (mset Δ) - (mset Γ - add-mset (snd δ)
(image-mset snd (mset Ψ)))
    ⊆# {#x → y. (x, y) ∈# mset Ψ#}
  using ◇(2) by (simp, metis add-mset-diff-bothsides
    list-subtract-mset-homomorphism
    mset-map subset-eq-diff-conv)
hence mset (map snd Δ)
  ⊆# mset (map (uncurry (→)) Ψ @ (remove1 (snd δ) Γ) ⊖ (map snd Ψ))
  using subset-eq-diff-conv by (simp, blast)
ultimately have mset (map snd (⋈ Ψ Δ)) ⊆# mset (remove1 (snd δ) Γ)
  using Cons by blast
hence mset (map snd (δ # (⋈ Ψ Δ))) ⊆# mset Γ
  by (simp, metis ⟨snd δ ∈# mset (Γ ⊖ map snd Ψ)⟩
    cancel-ab-semigroup-add-class.diff-right-commute
    diff-single-trivial
    insert-subset-eq-iff
    list-subtract-mset-homomorphism
    multi-drop-mem-not-eq)
with ⟨find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None⟩
show ?thesis
  by simp
next
case False
from this obtain ψ where ψ:

```



```

    find ( $\lambda\psi. \text{uncurry } (\rightarrow) \psi = \text{snd } \delta$ )  $\Psi = \text{Some } \psi$ 
  by fastforce
let  $?x = \text{fst } \psi$ 
let  $?y = \text{snd } \psi$ 
have  $\text{uncurry } (\rightarrow) = (\lambda \psi. \text{fst } \psi \rightarrow \text{snd } \psi)$ 
  by fastforce
moreover
from this have  $\text{uncurry } (\rightarrow) \psi = ?x \rightarrow ?y$  by fastforce
with  $\psi$  have  $A: (?x, ?y) \in \text{set } \Psi$ 
  and  $B: \text{snd } \delta = ?x \rightarrow ?y$ 
  using find-Some-predicate
  by (simp add: find-Some-set-membership, fastforce)
let  $?x' = \text{remove1 } (?x, ?y) \Psi$ 
from  $B \diamond(2)$  have
   $\text{mset } (\text{map } \text{snd } \Delta) \subseteq\# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \Psi @ \Gamma \ominus \text{map } \text{snd } \Psi)$ 
-  $\{\# ?x \rightarrow ?y \#\}$ 
  by (simp add: insert-subset-eq-iff)
moreover
have  $\text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \Psi)$ 
  =  $\text{add-mset } (\text{case } (\text{fst } \psi, \text{snd } \psi) \text{ of } (x, xa) \Rightarrow x \rightarrow xa)$ 
  ( $\text{image-mset } (\text{uncurry } (\rightarrow)) (\text{mset } (\text{remove1 } (\text{fst } \psi, \text{snd } \psi) \Psi))$ )
  by (metis (no-types)
    A
    image-mset-add-mset
    in-multiset-in-set
    insert-DiffM
    mset-map
    mset-remove1
    uncurry-def)
ultimately have
   $\text{mset } (\text{map } \text{snd } \Delta) \subseteq\# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) ?x' @ \Gamma \ominus \text{map } \text{snd } \Psi)$ 
  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  $A B \diamond$ 
have  $\text{mset } (\Gamma \ominus \text{map } \text{snd } \Psi) = \text{mset}((\text{remove1 } ?y \Gamma) \ominus (\text{remove1 } ?y (\text{map } \text{snd } \Psi)))$ 
  using
    image-eqI
    prod.sel(2)
    set-map
  by force
with  $A$  have

```

$mset (\Gamma \ominus \text{map snd } \Psi) = mset((\text{remove1 } ?\gamma \Gamma) \ominus (\text{map snd } ?\Psi'))$   
**by** (*metis*  
*remove1-pairs-list-projections-snd*  
*in-multiset-in-set*  
*list-subtract-mset-homomorphism*  
*mset-remove1*)

**ultimately have**  
 $mset (\text{map snd } \Delta) \subseteq\# mset (\text{map } (\text{uncurry } (\rightarrow)) ?\Psi'$   
 $\quad \text{@ } (\text{remove1 } ?\gamma \Gamma)$   
 $\quad \ominus \text{map snd } ?\Psi')$

**by** *simp*

**hence**  $mset (\text{map snd } (\mathfrak{J} ?\Psi' \Delta)) \subseteq\# mset (\text{remove1 } ?\gamma \Gamma)$   
**using** *Cons*  $\diamond(1)$  *A*  
**by** (*metis* (*no-types*, *lifting*)  
*image-mset-add-mset*  
*in-multiset-in-set*  
*insert-DiffM*  
*insert-subset-eq-iff*  
*mset-map mset-remove1*  
*prod.collapse*)

**with**  $\diamond(1)$  *A* **have**  $mset (\text{map snd } (\mathfrak{J} ?\Psi' \Delta)) + \{\# ?\gamma \#\} \subseteq\# mset \Gamma$   
**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**  $mset (\text{map snd } ((\text{fst } \delta \sqcap ?\chi, ?\gamma) \# (\mathfrak{J} ?\Psi' \Delta))) \subseteq\# mset \Gamma$   
**by** *simp*

**moreover from**  $\psi$  **have**  
 $\mathfrak{J} \Psi (\delta \# \Delta) = (\text{fst } \delta \sqcap ?\chi, ?\gamma) \# (\mathfrak{J} ?\Psi' \Delta)$   
**by** *simp*

**ultimately show** *?thesis* **by** *simp*

**qed**  
**}**  
**thus** *?case* **by** *blast*

**qed**  
**with** *assms* **show** *?thesis* **by** *blast*

**qed**

**lemma** (*in classical-logic*) *right-merge-witness-stronger-theory*:  
 $\text{map } (\text{uncurry } (\sqcup)) \Delta \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{J} \Psi \Delta)$

**proof** –

**have**  $\forall \Psi. \text{map } (\text{uncurry } (\sqcup)) \Delta \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{J} \Psi \Delta)$

**proof** (*induct*  $\Delta$ )

**case** *Nil*

**then show** *?case* **by** *simp*

```

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

```

```

lemma (in classical-logic) left-merge-witness-stronger-theory:
  map (uncurry (⊔)) Ψ ≲ map (uncurry (⊔)) (⋈ Ψ Δ)
proof –
  have ∀ Ψ. map (uncurry (⊔)) Ψ ≲ map (uncurry (⊔)) (⋈ 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 (⊔)) (⋈ Ψ (δ # Δ))
  proof (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None)
  case True
  then show ?thesis
  using Cons stronger-theory-right-cons
  by auto
next
  case False
  from this obtain ψ where ψ:
    find (λ ψ. 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)
    uncurry (⊔) ψ = ?χ ⊔ ?γ
  using ψ find-Some-predicate
  by fastforce+
moreover
  {
  fix μ χ γ
  have ⊢ ((μ ⊓ χ) ⊔ γ) → (χ ⊔ γ)
  proof –
  have ∀ ℳ. ℳ ⊨prop ((⟨μ⟩ ⊓ ⟨χ⟩) ⊔ ⟨γ⟩) → (⟨χ⟩ ⊔ ⟨γ⟩)
  by fastforce
  hence ⊢ ⟨ ((μ) ⊓ (χ)) ⊔ (γ) ⟩ → (⟨χ⟩ ⊔ ⟨γ⟩)
  using propositional-semantic by blast
  thus ?thesis
  by simp
  }
  qed
  }
  ultimately have

```

```

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

```

**lemma** (in *classical-logic*) *measure-empty-deduction*:

```

[] $⊢ Φ = (∀ φ ∈ set Φ. ⊢ φ)
by (induct Φ, simp, rule iffI, fastforce+)

```

**lemma** (in *classical-logic*) *measure-stronger-theory-left-monotonic*:

```

assumes Σ ≼ Γ
and Σ $⊢ Φ
shows Γ $⊢ Φ
using assms
proof (induct Φ arbitrary: Σ Γ)
  case Nil
  then show ?case by simp
next
  case (Cons φ Φ)
  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⟩

```

$\langle \text{map snd } \Delta = \Sigma \rangle$   
**obtain**  $\Omega$  **where**  $\Omega$ :  
 $\text{map } (\lambda (\psi, \sigma, -). (\psi, \sigma)) \Omega = \Psi$   
 $\text{mset } (\text{map } (\lambda (-, \sigma, \gamma). (\gamma, \sigma)) \Omega) \subseteq\# \text{mset } \Delta$   
**using** *triple-list-exists* **by** *blast*  
**let**  $?\Theta = \text{map } (\lambda (\psi, -, \gamma). (\psi, \gamma)) \Omega$   
**have**  $\text{map snd } ?\Theta = \text{map fst } (\text{map } (\lambda (-, \sigma, \gamma). (\gamma, \sigma)) \Omega)$   
**by** *auto*  
**hence**  $\text{mset } (\text{map snd } ?\Theta) \subseteq\# \text{mset } \Gamma$   
**using**  $\Omega(2)$   $\Delta(2)$  *map-monotonic subset-mset.order-trans*  
**by** *metis*  
**moreover** **have**  $\text{map } (\text{uncurry } (\sqcup)) \Psi \preceq \text{map } (\text{uncurry } (\sqcup)) ?\Theta$   
**proof** –  
**let**  $?\Phi = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \sqcup \gamma, \psi \sqcup \sigma)) \Omega$   
**have**  $\text{map snd } ?\Phi = \text{map } (\text{uncurry } (\sqcup)) \Psi$   
**using**  $\Omega(1)$  **by** *fastforce*  
**moreover** **have**  $\text{map fst } ?\Phi = \text{map } (\text{uncurry } (\sqcup)) ?\Theta$   
**by** *fastforce*  
**hence**  $\text{mset } (\text{map fst } ?\Phi) \subseteq\# \text{mset } (\text{map } (\text{uncurry } (\sqcup)) ?\Theta)$   
**by** (*metis subset-mset.dual-order.refl*)  
**moreover**  
**have**  $\text{mset } (\text{map } (\lambda(\psi, \sigma, -). (\psi, \sigma)) \Omega) \subseteq\# \text{mset } \Psi$   
**using**  $\Omega(1)$  **by** *simp*  
**hence**  $\forall (\varphi, \chi) \in \text{set } ?\Phi. \vdash \varphi \rightarrow \chi$  **using**  $\Omega(2)$   
**proof** (*induct*  $\Omega$ )  
**case** *Nil*  
**then show** *?case* **by** *simp*  
**next**  
**case** (*Cons*  $\omega$   $\Omega$ )  
**let**  $?\Phi = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \sqcup \gamma, \psi \sqcup \sigma)) (\omega \# \Omega)$   
**let**  $?\Phi' = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \sqcup \gamma, \psi \sqcup \sigma)) \Omega$   
**have**  $\text{mset } (\text{map } (\lambda(\psi, \sigma, -). (\psi, \sigma)) \Omega) \subseteq\# \text{mset } \Psi$   
 $\text{mset } (\text{map } (\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \Omega) \subseteq\# \text{mset } \Delta$   
**using** *Cons.prem1* *Cons.prem2* *subset-mset.dual-order.trans* **by** *fast-*  
*force+*  
**with** *Cons* **have**  $\forall (\varphi, \chi) \in \text{set } ?\Phi'. \vdash \varphi \rightarrow \chi$  **by** *fastforce*  
**moreover**  
**let**  $? \psi = (\lambda (\psi, -, -). \psi) \omega$   
**let**  $? \sigma = (\lambda (-, \sigma, -). \sigma) \omega$   
**let**  $? \gamma = (\lambda (-, -, \gamma). \gamma) \omega$   
**have**  $(\lambda(-, \sigma, \gamma). (\gamma, \sigma)) = (\lambda \omega. ((\lambda (-, -, \gamma). \gamma) \omega, (\lambda (-, \sigma, -). \sigma) \omega))$  **by** *auto*  
**hence**  $(\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \omega = (? \gamma, ? \sigma)$  **by** *metis*  
**hence**  $\vdash ? \gamma \rightarrow ? \sigma$   
**using** *Cons.prem2* *mset-subset-eqD*  $\Delta(3)$   
**by** *fastforce*  
**hence**  $\vdash (? \psi \sqcup ? \gamma) \rightarrow (? \psi \sqcup ? \sigma)$   
**unfolding** *disjunction-def*  
**using** *modus-ponens* *hypothetical-syllogism*  
**by** *blast*

**moreover have**  
 $(\lambda(\psi, \sigma, \gamma). (\psi \sqcup \gamma, \psi \sqcup \sigma)) =$   
 $(\lambda \omega. (((\lambda (\psi, -, -). \psi) \omega) \sqcup ((\lambda (-, -, \gamma). \gamma) \omega),$   
 $((\lambda (\psi, -, -). \psi) \omega) \sqcup ((\lambda (-, \sigma, -). \sigma) \omega)))$   
**by auto**  
**hence**  $(\lambda(\psi, \sigma, \gamma). (\psi \sqcup \gamma, \psi \sqcup \sigma)) \omega = ((? \psi \sqcup ? \gamma), (? \psi \sqcup ? \sigma))$  **by metis**  
**ultimately show** *?case by simp*  
**qed**  
**ultimately show** *?thesis*  
**unfolding** *stronger-theory-relation-def*  
**by blast**  
**qed**  
**hence**  $\text{map } (\text{uncurry } (\sqcup)) \text{ ?}\Theta \vdash \varphi$   
**using**  $\Psi(2)$   
*stronger-theory-deduction-monotonic*  
**[where**  $\Sigma = \text{map } (\text{uncurry } (\sqcup)) \Psi$   
**and**  $\Gamma = \text{map } (\text{uncurry } (\sqcup)) \text{ ?}\Theta$   
**and**  $\varphi = \varphi]$   
**by metis**  
**moreover have**  
 $(\text{map } (\text{uncurry } (\rightarrow)) \Psi @ \Sigma @ (\text{map } \text{snd } \Psi)) \preceq$   
 $(\text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Theta @ \Gamma @ (\text{map } \text{snd } \text{ ?}\Theta))$   
**proof** –  
**have**  $\text{map } (\text{uncurry } (\rightarrow)) \Psi \preceq \text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Theta$   
**proof** –  
**let**  $? \Phi = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \rightarrow \gamma, \psi \rightarrow \sigma)) \Omega$   
**have**  $\text{map } \text{snd } ? \Phi = \text{map } (\text{uncurry } (\rightarrow)) \Psi$   
**using**  $\Omega(1)$  **by fastforce**  
**moreover have**  $\text{map } \text{fst } ? \Phi = \text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Theta$   
**by fastforce**  
**hence**  $\text{mset } (\text{map } \text{fst } ? \Phi) \subseteq \# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Theta)$   
**by** *(metis subset-mset.dual-order.refl)*  
**moreover**  
**have**  $\text{mset } (\text{map } (\lambda(\psi, \sigma, -). (\psi, \sigma)) \Omega) \subseteq \# \text{mset } \Psi$   
**using**  $\Omega(1)$  **by simp**  
**hence**  $\forall (\varphi, \chi) \in \text{set } ? \Phi. \vdash \varphi \rightarrow \chi$  **using**  $\Omega(2)$   
**proof** *(induct  $\Omega$ )*  
**case Nil**  
**then show** *?case by simp*  
**next**  
**case**  $(\text{Cons } \omega \Omega)$   
**let**  $? \Phi = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \rightarrow \gamma, \psi \rightarrow \sigma)) (\omega \# \Omega)$   
**let**  $? \Phi' = \text{map } (\lambda (\psi, \sigma, \gamma). (\psi \rightarrow \gamma, \psi \rightarrow \sigma)) \Omega$   
**have**  $\text{mset } (\text{map } (\lambda(\psi, \sigma, -). (\psi, \sigma)) \Omega) \subseteq \# \text{mset } \Psi$   
 $\text{mset } (\text{map } (\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \Omega) \subseteq \# \text{mset } \Delta$   
**using** *Cons.prem1 Cons.prem2 subset-mset.dual-order.trans* **by**  
*fastforce+*  
**with Cons** **have**  $\forall (\varphi, \chi) \in \text{set } ? \Phi'. \vdash \varphi \rightarrow \chi$  **by fastforce**  
**moreover**

**let**  $?\psi = (\lambda (\psi, -, -). \psi) \omega$   
**let**  $?\sigma = (\lambda (-, \sigma, -). \sigma) \omega$   
**let**  $? \gamma = (\lambda (-, -, \gamma). \gamma) \omega$   
**have**  $(\lambda(-, \sigma, \gamma). (\gamma, \sigma)) = (\lambda \omega. ((\lambda (-, -, \gamma). \gamma) \omega, (\lambda (-, \sigma, -). \sigma) \omega))$  **by**  
*auto*  
**hence**  $(\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \omega = (? \gamma, ? \sigma)$  **by** *metis*  
**hence**  $\vdash ? \gamma \rightarrow ? \sigma$   
**using** *Cons.premis(2) mset-subset-eqD  $\Delta(3)$*   
**by** *fastforce*  
**hence**  $\vdash (? \psi \rightarrow ? \gamma) \rightarrow (? \psi \rightarrow ? \sigma)$   
**using** *modus-ponens hypothetical-syllogism*  
**by** *blast*  
**moreover have**  
 $(\lambda(\psi, \sigma, \gamma). (\psi \rightarrow \gamma, \psi \rightarrow \sigma)) =$   
 $(\lambda \omega. (((\lambda (\psi, -, -). \psi) \omega) \rightarrow ((\lambda (-, -, \gamma). \gamma) \omega),$   
 $((\lambda (\psi, -, -). \psi) \omega) \rightarrow ((\lambda (-, \sigma, -). \sigma) \omega)))$   
**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**  $(\Sigma \ominus (\text{map snd } \Psi)) \preceq (\Gamma \ominus (\text{map snd } ?\Theta))$   
**proof** –  
**let**  $? \Delta = \Delta \ominus (\text{map } (\lambda (-, \sigma, \gamma). (\gamma, \sigma)) \Omega)$   
**have**  $\text{mset } (\text{map fst } ? \Delta) \subseteq \# \text{mset } (\Gamma \ominus (\text{map snd } ? \Theta))$   
**using**  $\Delta(2)$   
**by** *(metis  $\Omega(2)$*   
 $\langle \text{map snd } (\text{map } (\lambda(\psi, -, \gamma). (\psi, \gamma)) \Omega) =$   
 $\text{map fst } (\text{map } (\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \Omega) \rangle$   
*list-subtract-monotonic*  
*map-list-subtract-mset-equivalence)*  
**moreover**  
**from**  $\Omega(2)$  **have**  $\text{mset } ? \Delta \subseteq \# \text{mset } \Delta$  **by** *simp*  
**hence**  $\forall (\gamma, \sigma) \in \text{set } ? \Delta. \vdash \gamma \rightarrow \sigma$   
**using**  $\Delta(3)$   
**by** *(metis mset-subset-eqD set-mset-mset)*  
**moreover**  
**have**  $\text{map snd } (\text{map } (\lambda(-, \sigma, \gamma). (\gamma, \sigma)) \Omega) = \text{map snd } \Psi$   
**using**  $\Omega(1)$   
**by** *(induct  $\Omega$ , simp, fastforce)*  
**hence**  $\text{mset } (\text{map snd } ? \Delta) = \text{mset } (\Sigma \ominus (\text{map snd } \Psi))$   
**by** *(metis  $\Delta(1)$   $\Omega(2)$  map-list-subtract-mset-equivalence)*  
**ultimately show** *?thesis*  
**by** *(metis stronger-theory-relation-alt-def)*  
**qed**



**ultimately show** *?thesis using stronger-theory-combine by blast*  
**qed**  
**hence**  $\text{map } (\text{uncurry } (\rightarrow)) \text{ ?}\Theta @ \Gamma \ominus (\text{map snd ?}\Theta) \$\vdash \Phi$   
**using**  $\Psi(3)$  *Cons by blast*  
**ultimately show** *?case*  
**by** *(metis measure-deduction.simps(2))*  
**qed**

**lemma** *(in classical-logic) merge-witness-measure-deduction-intro:*  
**assumes**  $\text{mset } (\text{map snd } \Delta) \subseteq\# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \Psi @ \Gamma \ominus (\text{map snd } \Psi))$   
**and**  $\text{map } (\text{uncurry } (\rightarrow)) \Delta @ (\text{map } (\text{uncurry } (\rightarrow)) \Psi @ \Gamma \ominus \text{map snd } \Psi) \ominus \text{map snd } \Delta \$\vdash \Phi$   
**(is**  $\text{?}\Gamma_0 \$\vdash \Phi$ **)**  
**shows**  $\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta) @ \Gamma \ominus \text{map snd } (\mathfrak{J} \Psi \Delta) \$\vdash \Phi$   
**(is**  $\text{?}\Gamma \$\vdash \Phi$ **)**

**proof** –  
**let**  $\text{?}\Sigma = \mathfrak{B} \Psi \Delta$   
**let**  $\text{?}A = \text{map } (\text{uncurry } (\rightarrow)) \Delta$   
**let**  $\text{?}B = \text{map } (\text{uncurry } (\rightarrow)) \Psi$   
**let**  $\text{?}C = \text{map snd } \text{?}\Sigma$   
**let**  $\text{?}D = \Gamma \ominus (\text{map snd } \Psi)$   
**let**  $\text{?}E = \text{map snd } (\Delta \ominus \text{?}\Sigma)$   
**have**  $\Sigma: \text{mset } \text{?}\Sigma \subseteq\# \text{mset } \Delta$   
 $\text{mset } \text{?}C \subseteq\# \text{mset } \text{?}B$   
 $\text{mset } \text{?}E \subseteq\# \text{mset } \text{?}D$   
**using** *assms(1)*  
*second-component-msub*  
*second-component-snd-projection-msub*  
*second-component-diff-msub*  
**by** *simp+*  
**moreover**  
**from** *calculation* **have**  
 $\text{image-mset snd } (\text{mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta))$   
 $\subseteq\# \text{mset } \Gamma - \text{image-mset snd } (\text{mset } \Psi)$   
**by** *simp*  
**hence**  $\text{mset } \Gamma - \text{image-mset snd } (\text{mset } \Psi)$   
 $- \text{image-mset snd } (\text{mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta))$   
 $+ \text{image-mset snd } (\text{mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta))$   
 $= \text{mset } \Gamma - \text{image-mset snd } (\text{mset } \Psi)$   
**using** *subset-mset.diff-add by blast*  
**then have**  $\text{image-mset snd } (\text{mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta))$   
 $+ (\{\#x \rightarrow y. (x, y) \in\# \text{mset } \Psi\# \}$   
 $+ (\text{mset } \Gamma - (\text{image-mset snd } (\text{mset } \Psi)$   
 $+ \text{image-mset snd } (\text{mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta))))))$   
 $= \{\#x \rightarrow y. (x, y) \in\# \text{mset } \Psi\# \} + (\text{mset } \Gamma - \text{image-mset snd } (\text{mset } \Psi))$   
**by** *(simp add: union-commute)*  
**with** *calculation* **have**  $\text{mset } \text{?}\Gamma_0 = \text{mset } (\text{?}A @ (\text{?}B \ominus \text{?}C) @ (\text{?}D \ominus \text{?}E))$   
**by** *(simp, metis (no-types) add-diff-cancel-left image-mset-union subset-mset.diff-add)*

**moreover have**  $(?A @ (?B \ominus ?C)) \preceq \text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta)$   
**using** *second-component-merge-witness-stronger-theory* **by** *simp*  
**moreover have**  $\text{mset } (?D \ominus ?E) = \text{mset } (\Gamma \ominus \text{map } \text{snd } (\mathfrak{J} \Psi \Delta))$   
**using** *second-component-merge-witness-snd-projection*  
**by** *simp*  
**with calculation have**  $(?A @ (?B \ominus ?C) @ (?D \ominus ?E)) \preceq ?\Gamma$   
**by** (*metis*  
*(no-types, lifting)*  
*stronger-theory-combine*  
*append.assoc*  
*list-subtract-mset-homomorphism*  
*mset-stronger-theory-intro*  
*map-list-subtract-mset-containment*  
*map-list-subtract-mset-equivalence*  
*mset-subset-eq-add-right*  
*subset-mset.add-diff-inverse*  
*subset-mset.diff-add-assoc2*)  
**ultimately have**  $?\Gamma_0 \preceq ?\Gamma$   
**unfolding** *stronger-theory-relation-alt-def*  
**by** *simp*  
**thus** *?thesis*  
**using** *assms(2) measure-stronger-theory-left-monotonic*  
**by** *blast*  
**qed**

**lemma (in classical-logic) measure-formula-right-split:**

$$\Gamma \text{ \$}\vdash (\psi \sqcup \varphi \# \psi \rightarrow \varphi \# \Phi) = \Gamma \text{ \$}\vdash (\varphi \# \Phi)$$

**proof** (*rule iffI*)

**assume**  $\Gamma \text{ \$}\vdash (\varphi \# \Phi)$

**from this obtain**  $\Psi$  **where**  $\Psi$ :

$$\text{mset } (\text{map } \text{snd } \Psi) \subseteq \# \text{mset } \Gamma$$

$$\text{map } (\text{uncurry } (\sqcup)) \Psi \text{ \$}\vdash \varphi$$

$$(\text{map } (\text{uncurry } (\rightarrow)) \Psi @ \Gamma \ominus (\text{map } \text{snd } \Psi)) \text{ \$}\vdash \Phi$$

**by** *auto*

**let**  $?\Psi_1 = \text{zip } (\text{map } (\lambda (\chi, \gamma). \psi \sqcup \chi) \Psi) (\text{map } \text{snd } \Psi)$

**let**  $?\Gamma_1 = \text{map } (\text{uncurry } (\rightarrow)) ?\Psi_1 @ \Gamma \ominus (\text{map } \text{snd } ?\Psi_1)$

**let**  $?\Psi_2 = \text{zip } (\text{map } (\lambda (\chi, \gamma). \psi \rightarrow \chi) \Psi) (\text{map } (\text{uncurry } (\rightarrow)) ?\Psi_1)$

**let**  $?\Gamma_2 = \text{map } (\text{uncurry } (\rightarrow)) ?\Psi_2 @ ?\Gamma_1 \ominus (\text{map } \text{snd } ?\Psi_2)$

**have**  $\text{map } (\text{uncurry } (\rightarrow)) \Psi \preceq \text{map } (\text{uncurry } (\rightarrow)) ?\Psi_2$

**proof** (*induct*  $\Psi$ )

**case** *Nil*

**then show** *?case* **by** *simp*

**next**

**case** (*Cons*  $\delta$   $\Psi$ )

**let**  $?\chi = \text{fst } \delta$

**let**  $?\gamma = \text{snd } \delta$

**let**  $?\Psi_1 = \text{zip } (\text{map } (\lambda (\chi, \gamma). \psi \sqcup \chi) \Psi) (\text{map } \text{snd } \Psi)$

**let**  $?\Psi_2 = \text{zip } (\text{map } (\lambda (\chi, \gamma). \psi \rightarrow \chi) \Psi) (\text{map } (\text{uncurry } (\rightarrow)) ?\Psi_1)$

**let**  $?\Gamma_1 = \lambda \Psi. \text{map } (\text{uncurry } (\rightarrow)) (\text{zip } (\text{map } (\lambda (\chi, \gamma). \psi \sqcup \chi) \Psi) (\text{map } \text{snd } \Psi))$

```

Ψ))
let ?T2 = λ Ψ. map (uncurry (→)) (zip (map (λ (χ,γ). ψ → χ) Ψ) (?T1 Ψ))
{
  fix δ :: 'a × 'a
  have (λ (χ,γ). ψ ⊔ χ) = (λ δ. ψ ⊔ (fst δ))
    (λ (χ,γ). ψ → χ) = (λ δ. ψ → (fst δ))
    by fastforce+
  note functional-identities = this
  have (λ (χ,γ). ψ ⊔ χ) δ = ψ ⊔ (fst δ)
    (λ (χ,γ). ψ → χ) δ = ψ → (fst δ)
    by (simp add: functional-identities)+
}
hence ?T2 (δ # Ψ) = ((ψ → ?χ) → (ψ ⊔ ?χ) → ?γ) # (map (uncurry (→))
?Ψ2)
  by simp
moreover have map (uncurry (→)) (δ # Ψ) = (?χ → ?γ) # map (uncurry
(→)) Ψ
  by (simp add: case-prod-beta)
moreover
{
  fix χ ψ γ
  have ⊢ ((ψ → χ) → (ψ ⊔ χ) → γ) ↔ (χ → γ)
  proof -
    have ∀ M. M ⊨prop (((ψ) → (χ)) → ((ψ) ⊔ (χ)) → (γ)) ↔ ((χ) → (γ))
    by fastforce
    hence ⊢ (⊢ (((ψ) → (χ)) → ((ψ) ⊔ (χ)) → (γ)) ↔ ((χ) → (γ))) ⊢
    using propositional-semantic by blast
    thus ?thesis by simp
  qed
}
hence identity: ⊢ ((ψ → ?χ) → (ψ ⊔ ?χ) → ?γ) → (?χ → ?γ)
  using biconditional-def by auto
assume map (uncurry (→)) Ψ ≲ map (uncurry (→)) ?Ψ2
with identity have ((?χ → ?γ) # map (uncurry (→)) Ψ) ≲
  (((ψ → ?χ) → (ψ ⊔ ?χ) → ?γ) # (map (uncurry (→)) ?Ψ2))
  using stronger-theory-left-right-cons by blast
ultimately show ?case by simp
qed
hence (map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)) ≲
  ((map (uncurry (→)) ?Ψ2) @ Γ ⊖ (map snd Ψ))
  using stronger-theory-combine stronger-theory-reflexive by blast
moreover have mset ?T2 = mset ((map (uncurry (→)) ?Ψ2) @ Γ ⊖ (map snd
?Ψ1))
  by simp
ultimately have (map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)) ≲ ?T2
  by (simp add: stronger-theory-relation-def)
hence ?T2 $⊢ Φ
  using Ψ(3) measure-stronger-theory-left-monotonic by blast
moreover

```

```

have (map (uncurry ( $\sqcup$ ))  $?\Psi_2$ )  $\vdash \psi \rightarrow \varphi$ 
proof –
  let  $?\Gamma = \text{map } (\lambda (\chi, \gamma). (\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma) \Psi$ 
  let  $?\Sigma = \text{map } (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) \Psi$ 
  have map (uncurry ( $\sqcup$ ))  $?\Psi_2 = ?\Gamma$ 
  proof (induct  $\Psi$ )
    case Nil
    then show  $?case$  by simp
  next
    case (Cons  $\chi \Psi$ )
    have  $(\lambda \varphi. (\text{case } \varphi \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup (\text{case } \varphi \text{ of } (\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow$ 
  snd  $\varphi) =$ 
       $(\lambda \varphi. (\text{case } \varphi \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma))$ 
    by fastforce
    hence  $(\text{case } \chi \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow \chi) \sqcup (\text{case } \chi \text{ of } (\chi, \gamma) \Rightarrow \psi \sqcup \chi) \rightarrow \text{snd } \chi =$ 
       $(\text{case } \chi \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma)$ 
    by metis
    with Cons show  $?case$  by simp
  qed
  moreover have  $?\Sigma \preceq ?\Gamma$ 
  proof (induct  $\Psi$ )
    case Nil
    then show  $?case$  by simp
  next
    case (Cons  $\delta \Psi$ )
    let  $?\alpha = (\lambda (\chi, \gamma). (\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma) \delta$ 
    let  $?\beta = (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) \delta$ 
    let  $?\chi = \text{fst } \delta$ 
    let  $?\gamma = \text{snd } \delta$ 
    have  $(\lambda \delta. (\text{case } \delta \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow \chi \sqcup (\psi \sqcup \chi) \rightarrow \gamma)) =$ 
       $(\lambda \delta. \psi \rightarrow \text{fst } \delta \sqcup (\psi \sqcup \text{fst } \delta) \rightarrow \text{snd } \delta)$ 
       $(\lambda \delta. (\text{case } \delta \text{ of } (\chi, \gamma) \Rightarrow \psi \rightarrow (\chi \sqcup \gamma))) = (\lambda \delta. \psi \rightarrow (\text{fst } \delta \sqcup \text{snd } \delta))$ 
    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  $\vdash ((\psi \rightarrow \chi) \sqcup (\psi \sqcup \chi) \rightarrow \gamma) \rightarrow (\psi \rightarrow (\chi \sqcup \gamma))$ 
    proof –
      have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ((\langle \psi \rangle \rightarrow \langle \chi \rangle) \sqcup (\langle \psi \rangle \sqcup \langle \chi \rangle) \rightarrow \langle \gamma \rangle) \rightarrow (\langle \psi \rangle \rightarrow$ 
   $(\langle \chi \rangle \sqcup \langle \gamma \rangle))$ 
      by fastforce
      hence  $\vdash (\langle (\langle \psi \rangle \rightarrow \langle \chi \rangle) \sqcup (\langle \psi \rangle \sqcup \langle \chi \rangle) \rightarrow \langle \gamma \rangle) \rightarrow (\langle \psi \rangle \rightarrow (\langle \chi \rangle \sqcup \langle \gamma \rangle)) \Downarrow$ 
      using propositional-semantics by blast
      thus  $?thesis$  by simp
    qed
  }
}

```

```

ultimately have  $\vdash ?\alpha \rightarrow ?\beta$  by simp
thus ?case
  using Cons
    stronger-theory-left-right-cons
  by simp
qed
moreover have  $\forall \varphi. (\text{map } (\text{uncurry } (\sqcup)) \Psi) \vdash \varphi \longrightarrow ?\Sigma \vdash \psi \rightarrow \varphi$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case
    using axiom-k modus-ponens
    by fastforce
next
  case (Cons  $\delta \Psi$ )
  let  $?\delta' = (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) \delta$ 
  let  $?\Sigma = \text{map } (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) \Psi$ 
  let  $?\Sigma' = \text{map } (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) (\delta \# \Psi)$ 
  {
    fix  $\varphi$ 
    assume  $\text{map } (\text{uncurry } (\sqcup)) (\delta \# \Psi) \vdash \varphi$ 
    hence  $\text{map } (\text{uncurry } (\sqcup)) \Psi \vdash (\text{uncurry } (\sqcup)) \delta \rightarrow \varphi$ 
      using list-deduction-theorem
      by simp
    hence  $?\Sigma \vdash \psi \rightarrow (\text{uncurry } (\sqcup)) \delta \rightarrow \varphi$ 
      using Cons
      by blast
  moreover
    {
      fix  $\alpha \beta \gamma$ 
      have  $\vdash (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow ((\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \gamma)$ 
        using axiom-s by auto
    }
  ultimately have  $?\Sigma \vdash (\psi \rightarrow (\text{uncurry } (\sqcup)) \delta) \rightarrow \psi \rightarrow \varphi$ 
    using list-deduction-weaken [where  $?\Gamma = ?\Sigma$ ]
      list-deduction-modus-ponens [where  $?\Gamma = ?\Sigma$ ]
    by metis
  moreover
    have  $(\lambda \delta. \psi \rightarrow (\text{uncurry } (\sqcup)) \delta) = (\lambda \delta. (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma)))) \delta$ 
      by fastforce
    ultimately have  $?\Sigma \vdash (\lambda (\chi, \gamma). (\psi \rightarrow (\chi \sqcup \gamma))) \delta \rightarrow \psi \rightarrow \varphi$ 
      by metis
    hence  $?\Sigma' \vdash \psi \rightarrow \varphi$ 
      using list-deduction-theorem
      by simp
  }
  then show ?case by simp
qed
with  $\Psi(2)$  have  $?\Sigma \vdash \psi \rightarrow \varphi$ 
  by blast

```

```

ultimately show ?thesis
  using stronger-theory-deduction-monotonic by auto
qed
moreover have mset (map snd ?Ψ2) ⊆# mset ?Γ1 by simp
ultimately have ?Γ1 $⊢ (ψ → φ # Φ) using measure-deduction.simps(2) by
blast
moreover have ⊢ (map (uncurry (⊔)) Ψ :→ φ) → (map (uncurry (⊔)) ?Ψ1) :→
(ψ ⊔ φ)
proof (induct Ψ)
  case Nil
  then show ?case
    unfolding disjunction-def
    using axiom-k modus-ponens
    by fastforce
next
case (Cons ν Ψ)
let ?Δ = map (uncurry (⊔)) Ψ
let ?Δ' = map (uncurry (⊔)) (ν # Ψ)
let ?Σ = map (uncurry (⊔)) (zip (map (λ (χ,γ). ψ ⊔ χ) Ψ) (map snd Ψ))
let ?Σ' = map (uncurry (⊔)) (zip (map (λ (χ,γ). ψ ⊔ χ) (ν # Ψ)) (map snd
(ν # Ψ)))
have ⊢ (?Δ' :→ φ) → (uncurry (⊔)) ν → ?Δ :→ φ
  by (simp, metis axiom-k axiom-s modus-ponens)
with Cons have ⊢ (?Δ' :→ φ) → (uncurry (⊔)) ν → ?Σ :→ (ψ ⊔ φ)
  using hypothetical-syllogism modus-ponens
  by blast
hence (?Δ' :→ φ) # ((uncurry (⊔)) ν) # ?Σ :⊢ ψ ⊔ φ
  by (simp add: list-deduction-def)
moreover have set ((?Δ' :→ φ) # ((uncurry (⊔)) ν) # ?Σ) =
  set (((uncurry (⊔)) ν) # (?Δ' :→ φ) # ?Σ)
  by fastforce
ultimately have ((uncurry (⊔)) ν) # (?Δ' :→ φ) # ?Σ :⊢ ψ ⊔ φ
  using list-deduction-monotonic by blast
hence (?Δ' :→ φ) # ?Σ :⊢ ((uncurry (⊔)) ν) → (ψ ⊔ φ)
  using list-deduction-theorem
  by simp
moreover
let ?χ = fst ν
let ?γ = snd ν
have (λ ν . (uncurry (⊔)) ν) = (λ ν . fst ν ⊔ snd ν)
  by fastforce
hence (uncurry (⊔)) ν = ?χ ⊔ ?γ by simp
ultimately have (?Δ' :→ φ) # ?Σ :⊢ (?χ ⊔ ?γ) → (ψ ⊔ φ) by simp
moreover
{
  fix α β δ γ
  have ⊢ ((β ⊔ α) → (γ ⊔ δ)) → ((γ ⊔ β) ⊔ α) → (γ ⊔ δ)
  proof -
    have ∀ M. M ⊢prop (((β) ⊔ (α)) → ((γ) ⊔ (δ))) → (((γ) ⊔ (β)) ⊔ (α))

```

$\rightarrow (\langle \gamma \rangle \sqcup \langle \delta \rangle)$   
**by** *fastforce*  
**hence**  $\vdash \langle (\langle \beta \rangle \sqcup \langle \alpha \rangle) \rightarrow (\langle \gamma \rangle \sqcup \langle \delta \rangle) \rangle \rightarrow (\langle \langle \gamma \rangle \sqcup \langle \beta \rangle \rangle \sqcup \langle \alpha \rangle) \rightarrow (\langle \gamma \rangle \sqcup \langle \delta \rangle)$   $\Downarrow$   
**using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**}**  
**hence**  $(? \Delta' : \rightarrow \varphi) \# ? \Sigma \vdash ((? \chi \sqcup ? \gamma) \rightarrow (\psi \sqcup \varphi)) \rightarrow ((\psi \sqcup ? \chi) \sqcup ? \gamma) \rightarrow (\psi \sqcup \varphi)$   
**using** *list-deduction-weaken* **by** *blast*  
**ultimately have**  $(? \Delta' : \rightarrow \varphi) \# ? \Sigma \vdash ((\psi \sqcup ? \chi) \sqcup ? \gamma) \rightarrow (\psi \sqcup \varphi)$   
**using** *list-deduction-modus-ponens* **by** *blast*  
**hence**  $((\psi \sqcup ? \chi) \sqcup ? \gamma) \# (? \Delta' : \rightarrow \varphi) \# ? \Sigma \vdash \psi \sqcup \varphi$   
**using** *list-deduction-theorem*  
**by** *simp*  
**moreover have**  $set ((\psi \sqcup ? \chi) \sqcup ? \gamma) \# (? \Delta' : \rightarrow \varphi) \# ? \Sigma = set ((? \Delta' : \rightarrow \varphi) \# ((\psi \sqcup ? \chi) \sqcup ? \gamma) \# ? \Sigma)$   
**by** *fastforce*  
**moreover have**  
 $map (uncurry (\sqcup)) (\nu \# \Psi) : \rightarrow \varphi$   
 $\# (\psi \sqcup fst \nu) \sqcup snd \nu$   
 $\# map (uncurry (\sqcup)) (zip (map (\lambda(-, a). \psi \sqcup a) \Psi) (map snd \Psi)) \vdash (\psi \sqcup fst \nu) \sqcup snd \nu$   
**by** (*meson list.set-intros*(1)  
*list-deduction-monotonic*  
*list-deduction-reflection*  
*set-subset-Cons*)  
**ultimately have**  $(? \Delta' : \rightarrow \varphi) \# ((\psi \sqcup ? \chi) \sqcup ? \gamma) \# ? \Sigma \vdash \psi \sqcup \varphi$   
**using** *list-deduction-modus-ponens list-deduction-monotonic* **by** *blast*  
**moreover**  
**have**  $(\lambda \nu. \psi \sqcup fst \nu) = (\lambda (\chi, \gamma). \psi \sqcup \chi)$   
**by** *fastforce*  
**hence**  $\psi \sqcup fst \nu = (\lambda (\chi, \gamma). \psi \sqcup \chi) \nu$   
**by** *metis*  
**hence**  $((\psi \sqcup ? \chi) \sqcup ? \gamma) \# ? \Sigma = ? \Sigma'$   
**by** *simp*  
**ultimately have**  $(? \Delta' : \rightarrow \varphi) \# ? \Sigma' \vdash \psi \sqcup \varphi$  **by** *simp*  
**then show** *?case* **by** (*simp add: list-deduction-def*)  
**qed**  
**with**  $\Psi(2)$  **have**  $map (uncurry (\sqcup)) ? \Psi_1 \vdash (\psi \sqcup \varphi)$   
**unfolding** *list-deduction-def*  
**using** *modus-ponens*  
**by** *blast*  
**moreover have**  $mset (map snd ? \Psi_1) \subseteq \# mset \Gamma$  **using**  $\Psi(1)$  **by** *simp*  
**ultimately show**  $\Gamma \S \vdash (\psi \sqcup \varphi \# \psi \rightarrow \varphi \# \Phi)$   
**using** *measure-deduction.simps*(2) **by** *blast*  
**next**  
**assume**  $\Gamma \S \vdash (\psi \sqcup \varphi \# \psi \rightarrow \varphi \# \Phi)$

```

from this obtain  $\Psi$  where  $\Psi$ :
  mset (map snd  $\Psi$ )  $\subseteq\#$  mset  $\Gamma$ 
  map (uncurry ( $\sqcup$ ))  $\Psi$   $\vdash$   $\psi \sqcup \varphi$ 
  map (uncurry ( $\rightarrow$ ))  $\Psi$  @  $\Gamma$   $\ominus$  (map snd  $\Psi$ )  $\$$  $\vdash$  ( $\psi \rightarrow \varphi \# \Phi$ )
  using measure-deduction.simps(2) by blast
let  $\mathcal{F}\Gamma' = \text{map (uncurry ( $\rightarrow$ ))  $\Psi$  @  $\Gamma$   $\ominus$  (map snd  $\Psi$ )}$ 
from  $\Psi$  obtain  $\Delta$  where  $\Delta$ :
  mset (map snd  $\Delta$ )  $\subseteq\#$  mset  $\mathcal{F}\Gamma'$ 
  map (uncurry ( $\sqcup$ ))  $\Delta$   $\vdash$   $\psi \rightarrow \varphi$ 
  (map (uncurry ( $\rightarrow$ ))  $\Delta$  @  $\mathcal{F}\Gamma' \ominus$  (map snd  $\Delta$ ))  $\$$  $\vdash$   $\Phi$ 
  using measure-deduction.simps(2) by blast
let  $\mathcal{F}\Omega = \mathfrak{J} \Psi \Delta$ 
have mset (map snd  $\mathcal{F}\Omega$ )  $\subseteq\#$  mset  $\Gamma$ 
  using  $\Delta(1)$   $\Psi(1)$  merge-witness-msub-intro
  by blast
moreover have map (uncurry ( $\sqcup$ ))  $\mathcal{F}\Omega$   $\vdash$   $\varphi$ 
proof –
  have map (uncurry ( $\sqcup$ ))  $\mathcal{F}\Omega$   $\vdash$   $\psi \sqcup \varphi$ 
    map (uncurry ( $\sqcup$ ))  $\mathcal{F}\Omega$   $\vdash$   $\psi \rightarrow \varphi$ 
    using  $\Psi(2)$   $\Delta(2)$ 
      stronger-theory-deduction-monotonic
      right-merge-witness-stronger-theory
      left-merge-witness-stronger-theory
    by blast+
moreover
have  $\vdash$  ( $\psi \sqcup \varphi$ )  $\rightarrow$  ( $\psi \rightarrow \varphi$ )  $\rightarrow$   $\varphi$ 
  unfolding disjunction-def
  using modus-ponens excluded-middle-elimination flip-implication
  by blast
ultimately show ?thesis
  using list-deduction-weaken list-deduction-modus-ponens
  by blast
qed
moreover have map (uncurry ( $\rightarrow$ ))  $\mathcal{F}\Omega$  @  $\Gamma$   $\ominus$  (map snd  $\mathcal{F}\Omega$ )  $\$$  $\vdash$   $\Phi$ 
  using  $\Delta(1)$   $\Delta(3)$   $\Psi(1)$  merge-witness-measure-deduction-intro by blast
ultimately show  $\Gamma$   $\$$  $\vdash$  ( $\varphi \# \Phi$ )
  using measure-deduction.simps(2) by blast
qed

primrec (in implication-logic)
  X-witness :: ('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 ( $\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta$ )  $\Psi$  of
      None  $\Rightarrow$   $\delta \# \mathfrak{X} \Psi \Delta$ 
    | Some  $\psi \Rightarrow$  (fst  $\psi \rightarrow$  fst  $\delta, \text{snd } \psi$ )  $\#$  ( $\mathfrak{X}$  (remove1  $\psi$   $\Psi$ )  $\Delta$ ))

```

**primrec (in implication-logic)**



*X-component* :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (x•)  
**where**  
 x• Ψ [] = []  
 | x• Ψ (δ # Δ) =  
   (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of  
     None ⇒ x• Ψ Δ  
     | Some ψ ⇒ (fst ψ → fst δ, snd ψ) # (x• (remove1 ψ Ψ) Δ))

**primrec** (in *implication-logic*)  
*Y-witness* :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (y)  
**where**  
 y Ψ [] = Ψ  
 | y Ψ (δ # Δ) =  
   (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of  
     None ⇒ y Ψ Δ  
     | Some ψ ⇒ (fst ψ, (fst ψ → fst δ) → snd ψ) #  
       (y (remove1 ψ Ψ) Δ))

**primrec** (in *implication-logic*)  
*Y-component* :: ('a × 'a) list ⇒ ('a × 'a) list ⇒ ('a × 'a) list (y•)  
**where**  
 y• Ψ [] = []  
 | y• Ψ (δ # Δ) =  
   (case find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ of  
     None ⇒ y• Ψ Δ  
     | Some ψ ⇒ (fst ψ, (fst ψ → fst δ) → snd ψ) #  
       (y• (remove1 ψ Ψ) Δ))

**lemma** (in *implication-logic*) *X-witness-right-empty* [simp]:  
 x [] Δ = Δ  
**by** (induct Δ, simp+)

**lemma** (in *implication-logic*) *Y-witness-right-empty* [simp]:  
 y [] Δ = []  
**by** (induct Δ, simp+)

**lemma** (in *implication-logic*) *X-witness-map-snd-decomposition*:  
 mset (map snd (x Ψ Δ)) = mset (map snd ((x Ψ Δ) @ (Δ ⊖ (y Ψ Δ))))

**proof** –  
**have** ∀ Ψ. mset (map snd (x Ψ Δ)) = mset (map snd ((x Ψ Δ) @ (Δ ⊖ (y Ψ Δ))))

**proof** (induct Δ)  
 case Nil  
 then show ?case by simp  
**next**  
 case (Cons δ Δ)  
 {  
 fix Ψ  
 have mset (map snd (x Ψ (δ # Δ)))

```

    = mset (map snd (A Ψ (δ # Δ) @ (δ # Δ) ⊖ B Ψ (δ # Δ)))
using Cons
by (cases find (λ ψ. (uncurry (→)) 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
    second-component-msub,
    fastforce)
  }
  then show ?case by blast
qed
thus ?thesis by blast
qed

lemma (in implication-logic) Y-witness-map-snd-decomposition:
  mset (map snd (Q Ψ Δ)) = mset (map snd ((Ψ ⊖ (A Ψ Δ)) @ (Q• Ψ Δ)))
proof –
  have ∀ Ψ. mset (map snd (Q Ψ Δ)) = mset (map snd ((Ψ ⊖ (A Ψ Δ)) @ (Q•
  Ψ Δ)))
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)
    {
      fix Ψ
      have mset (map snd (Q Ψ (δ # Δ))) = mset (map snd (Ψ ⊖ A Ψ (δ # Δ)
  @ Q• Ψ (δ # Δ)))
      using Cons
      by (cases find (λ ψ. (uncurry (→)) ψ = snd δ) Ψ = None, fastforce+)
    }
    then show ?case by blast
  qed
thus ?thesis by blast
qed

lemma (in implication-logic) X-witness-msub:
  assumes mset (map snd Ψ) ⊆# mset Γ
  and mset (map snd Δ) ⊆# mset (map (uncurry (→)) Ψ @ Γ ⊖ (map snd
  Ψ))
  shows mset (map snd (X Ψ Δ)) ⊆# mset Γ
proof –
  have mset (map snd (Δ ⊖ (B Ψ Δ))) ⊆# mset (Γ ⊖ (map snd Ψ))
  using assms second-component-diff-msub by blast
  moreover have mset (map snd (A Ψ Δ)) ⊆# mset (map snd Ψ)
  using first-component-msub

```

by (*simp add: image-mset-subseteq-mono*)  
**moreover have**  $mset ((map\ snd\ \Psi) @ (\Gamma \ominus map\ snd\ \Psi)) = mset\ \Gamma$   
**using** *assms(1)*  
**by simp**  
**moreover have**  $image\ mset\ snd\ (mset\ (\mathfrak{A}\ \Psi\ \Delta)) + image\ mset\ snd\ (mset\ (\Delta \ominus \mathfrak{B}\ \Psi\ \Delta))$   
 $= mset\ (map\ snd\ (\mathfrak{X}\ \Psi\ \Delta))$   
**using** *X-witness-map-snd-decomposition* **by force**  
**ultimately**  
**show** *?thesis*  
**by** (*metis (no-types) mset-append mset-map subset-mset.add-mono*)  
**qed**

**lemma** (*in implication-logic*) *Y-component-msub:*

$mset\ (map\ snd\ (\mathfrak{Y}\bullet\ \Psi\ \Delta)) \subseteq\# mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}\ \Psi\ \Delta))$   
**proof** –  
**have**  $\forall\ \Psi.\ mset\ (map\ snd\ (\mathfrak{Y}\bullet\ \Psi\ \Delta)) \subseteq\# mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}\ \Psi\ \Delta))$   
**proof** (*induct*  $\Delta$ )  
**case** *Nil*  
**then show** *?case by simp*  
**next**  
**case** (*Cons*  $\delta\ \Delta$ )  
 {  
**fix**  $\Psi$   
**have**  $mset\ (map\ snd\ (\mathfrak{Y}\bullet\ \Psi\ (\delta\ \# \Delta))) \subseteq\# mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}\ \Psi\ (\delta\ \# \Delta)))$   
**using** *Cons*  
**by** (*cases find*  $(\lambda\ \psi.\ (uncurry\ (\rightarrow))\ \psi = snd\ \delta)\ \Psi = None,$   
*simp, metis add-mset-add-single*  
*mset-subset-eq-add-left*  
*subset-mset.order-trans,*  
*fastforce*)  
 }  
**then show** *?case by blast*  
**qed**  
**thus** *?thesis by blast*  
**qed**

**lemma** (*in implication-logic*) *Y-witness-msub:*

**assumes**  $mset\ (map\ snd\ \Psi) \subseteq\# mset\ \Gamma$   
**and**  $mset\ (map\ snd\ \Delta) \subseteq\# mset\ (map\ (uncurry\ (\rightarrow))\ \Psi @ \Gamma \ominus (map\ snd\ \Psi))$   
**shows**  $mset\ (map\ snd\ (\mathfrak{Y}\ \Psi\ \Delta)) \subseteq\#$   
 $mset\ (map\ (uncurry\ (\rightarrow))\ (\mathfrak{X}\ \Psi\ \Delta) @ \Gamma \ominus map\ snd\ (\mathfrak{X}\ \Psi\ \Delta))$   
**proof** –  
**have** *A: image-mset snd (mset  $\Psi$ )  $\subseteq\#$  mset  $\Gamma$*  **using** *assms by simp*  
**have** *B: image-mset snd (mset  $(\mathfrak{A}\ \Psi\ \Delta)) + image-mset snd (mset\ \Delta - mset\ (\mathfrak{B}\ \Psi\ \Delta)) \subseteq\#$  mset  $\Gamma$*   
**using** *A X-witness-map-snd-decomposition assms(2) X-witness-msub by auto*

```

have mset  $\Gamma - \text{image-mset snd (mset } \Psi) = \text{mset } (\Gamma \ominus \text{map snd } \Psi)$ 
  by simp
then have C: mset (map snd ( $\Delta \ominus \mathfrak{B} \Psi \Delta$ )) + image-mset snd (mset  $\Psi$ )  $\subseteq\#$ 
mset  $\Gamma$ 
  using A by (metis (full-types) assms(2) second-component-diff-msub subset-mset.le-diff-conv2)
have image-mset snd (mset ( $\Psi \ominus \mathfrak{A} \Psi \Delta$ )) + image-mset snd (mset ( $\mathfrak{A} \Psi \Delta$ ))
= image-mset snd (mset  $\Psi$ )
  by (metis (no-types) image-mset-union
list-subtract-mset-homomorphism
first-component-msub
subset-mset.diff-add)
then have image-mset snd (mset  $\Psi - \text{mset } (\mathfrak{A} \Psi \Delta)$ )
+ (image-mset snd (mset ( $\mathfrak{A} \Psi \Delta$ )) + image-mset snd (mset  $\Delta - \text{mset } (\mathfrak{B} \Psi \Delta)$ ))
= mset (map snd ( $\Delta \ominus \mathfrak{B} \Psi \Delta$ )) + image-mset snd (mset  $\Psi$ )
  by (simp add: union-commute)
then have image-mset snd (mset  $\Psi - \text{mset } (\mathfrak{A} \Psi \Delta)$ )
 $\subseteq\#$  mset  $\Gamma - (\text{image-mset snd (mset } (\mathfrak{A} \Psi \Delta)) + \text{image-mset snd (mset } \Delta - \text{mset } (\mathfrak{B} \Psi \Delta)))$ 
  by (metis (no-types) B C subset-mset.le-diff-conv2)
hence mset (map snd ( $\Psi \ominus \mathfrak{A} \Psi \Delta$ ))  $\subseteq\#$  mset ( $\Gamma \ominus \text{map snd } (\mathfrak{X} \Psi \Delta)$ )
  using assms X-witness-map-snd-decomposition
  by simp
thus ?thesis
  using Y-component-msub
Y-witness-map-snd-decomposition
  by (simp add: mset-subset-eq-mono-add union-commute)
qed

```

**lemma** (in classical-logic) X-witness-right-stronger-theory:

map (uncurry ( $\sqcup$ ))  $\Delta \preceq$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{X} \Psi \Delta$ )

**proof** –

**have**  $\forall \Psi. \text{map (uncurry } (\sqcup)) \Delta \preceq \text{map (uncurry } (\sqcup)) (\mathfrak{X} \Psi \Delta)$

**proof** (induct  $\Delta$ )

case Nil

**then show** ?case **by** simp

**next**

case (Cons  $\delta \Delta$ )

{

**fix**  $\Psi$

**have** map (uncurry ( $\sqcup$ )) ( $\delta \# \Delta$ )  $\preceq$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{X} \Psi (\delta \# \Delta)$ )

**proof** (cases find ( $\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta$ )  $\Psi = \text{None}$ )

case True

**then show** ?thesis

**using** Cons

**by** (simp add: stronger-theory-left-right-cons
trivial-implication)

**next**

```

case False
from this obtain  $\psi$  where
   $\psi$ : find  $(\lambda\psi. \text{uncurry } (\rightarrow) \psi = \text{snd } \delta) \Psi = \text{Some } \psi$ 
   $\psi \in \text{set } \Psi$ 
   $(\text{fst } \psi \rightarrow \text{snd } \psi) = \text{snd } \delta$ 
using find-Some-set-membership
  find-Some-predicate
by fastforce
let  $?\Psi' = \text{remove1 } \psi \Psi$ 
let  $?\alpha = \text{fst } \psi$ 
let  $?\beta = \text{snd } \psi$ 
let  $?\gamma = \text{fst } \delta$ 
have  $\text{map } (\text{uncurry } (\sqcup)) \Delta \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{X} \ ?\Psi' \ \Delta)$ 
  using Cons by simp
moreover
have  $(\text{uncurry } (\sqcup)) = (\lambda \delta. \text{fst } \delta \sqcup \text{snd } \delta)$  by fastforce
hence  $(\text{uncurry } (\sqcup)) \delta = ?\gamma \sqcup (?\alpha \rightarrow ?\beta)$  using  $\psi(3)$  by fastforce
moreover
{
  fix  $\alpha \beta \gamma$ 
  have  $\vdash (\alpha \rightarrow \gamma \sqcup \beta) \rightarrow (\gamma \sqcup (\alpha \rightarrow \beta))$ 
  proof –
    let  $? \varphi = (\langle \alpha \rangle \rightarrow \langle \gamma \rangle \sqcup \langle \beta \rangle) \rightarrow (\langle \gamma \rangle \sqcup (\langle \alpha \rangle \rightarrow \langle \beta \rangle))$ 
    have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ? \varphi$  by fastforce
    hence  $\vdash (\!| \ ? \varphi \ !|)$  using propositional-semantic by blast
    thus thesis by simp
  qed
}
hence  $\vdash (? \alpha \rightarrow ? \gamma \sqcup ? \beta) \rightarrow (? \gamma \sqcup (? \alpha \rightarrow ? \beta))$  by simp
ultimately
show thesis using  $\psi$ 
  by (simp add: stronger-theory-left-right-cons)
qed
}
then show case by simp
qed
thus thesis by simp
qed

lemma (in classical-logic) Y-witness-left-stronger-theory:
   $\text{map } (\text{uncurry } (\sqcup)) \Psi \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{Y} \ \Psi \ \Delta)$ 
proof –
  have  $\forall \Psi. \text{map } (\text{uncurry } (\sqcup)) \Psi \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{Y} \ \Psi \ \Delta)$ 
proof (induct  $\Delta$ )
  case Nil
  then show case by simp
next
case (Cons  $\delta \ \Delta$ )
  {

```

```

fix Ψ
have map (uncurry (⊔)) Ψ ≤ map (uncurry (⊔)) (⌈ 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 (λ ψ. 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 (⊔)) (⌈ ?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-semantic by blast
    thus ?thesis by simp
  qed
}
hence ⊢ ?φ → (uncurry (⊔)) ψ using ψ(β) by auto
ultimately
have map (uncurry (⊔)) (ψ # ?Ψ') ≤ (?φ # map (uncurry (⊔)) (⌈ ?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-map perm-remove)
ultimately show ?thesis
  using stronger-theory-relation-alt-def ψ(1) by auto
qed
}
then show ?case by blast
qed
thus ?thesis by blast
qed

```

**lemma** (in *implication-logic*) *X-witness-second-component-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  $\Delta$ )
  case Nil
  then show ?case by simp
next
case (Cons  $\delta \Delta$ )
{
  fix  $\Psi$ 
  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 second-component-msub subset-mset.diff-add-assoc2,
    fastforce)
}
then show ?case by blast
qed
thus ?thesis by blast
qed

```

**lemma (in implication-logic) Y-witness-first-component-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  $\Delta$ )
  case Nil
  then show ?case by simp
next
case (Cons  $\delta \Delta$ )
{
  fix  $\Psi$ 
  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 implication-logic) Y-witness-right-stronger-theory:**

```

 $map (uncurry (\rightarrow)) \Delta \preceq map (uncurry (\rightarrow)) (\mathfrak{Y} \Psi \Delta \ominus (\Psi \ominus \mathfrak{A} \Psi \Delta) @ (\Delta \ominus \mathfrak{B} \Psi \Delta))$ 
proof -
  let  $?f = \lambda \Psi \Delta. (\Psi \ominus \mathfrak{A} \Psi \Delta)$ 
  let  $?g = \lambda \Psi \Delta. (\Delta \ominus \mathfrak{B} \Psi \Delta)$ 
  have  $\forall \Psi. map (uncurry (\rightarrow)) \Delta \preceq map (uncurry (\rightarrow)) (\mathfrak{Y} \Psi \Delta \ominus ?f \Psi \Delta @ ?g \Psi \Delta)$ 

```

```

proof (induct  $\Delta$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\delta$   $\Delta$ )
  let  $?\delta = (\text{uncurry } (\rightarrow)) \delta$ 
  {
    fix  $\Psi$ 
    have  $\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta)$ 
       $\preceq \text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi (\delta \# \Delta) \ominus \mathfrak{F} \Psi (\delta \# \Delta) \textcircled{\small @} \mathfrak{G} \Psi (\delta \# \Delta))$ 
    proof (cases find ( $\lambda \psi. (\text{uncurry } (\rightarrow)) \psi = \text{snd } \delta$ )  $\Psi = \text{None}$ )
      case True
      moreover
      from Cons have
         $\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \Delta) \preceq \text{map } (\text{uncurry } (\rightarrow)) (\delta \# \mathfrak{J} \Psi \Delta \ominus \mathfrak{F} \Psi$ 
 $\Delta \textcircled{\small @} \mathfrak{G} \Psi \Delta)$ 
        by (simp add: stronger-theory-left-right-cons trivial-implication)
      moreover
      have  $\text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\delta \# \mathfrak{J} \Psi \Delta \ominus \mathfrak{F} \Psi \Delta \textcircled{\small @} \mathfrak{G} \Psi \Delta))$ 
         $= \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta \ominus \mathfrak{F} \Psi \Delta \textcircled{\small @} ((\delta \# \Delta) \ominus \mathfrak{B} \Psi \Delta)))$ 
        by (simp,
          metis (no-types, lifting)
          add-mset-add-single
          image-mset-single
          image-mset-union
          second-component-msub
          mset-subset-eq-multiset-union-diff-commute)
      moreover have
         $\forall \Psi \Phi. \Psi \preceq \Phi$ 
         $= (\exists \Sigma. \text{map } \text{snd } \Sigma = \Psi$ 
           $\wedge \text{mset } (\text{map } \text{fst } \Sigma) \subseteq \# \text{mset } \Phi$ 
           $\wedge (\forall \xi. \xi \notin \text{set } \Sigma \vee \vdash (\text{uncurry } (\rightarrow) \xi)))$ 
        by (simp add: Ball-def-raw stronger-theory-relation-def)
      moreover have
         $((\text{uncurry } (\rightarrow) \delta) \# \text{map } (\text{uncurry } (\rightarrow)) \Delta)$ 
         $\preceq ((\text{uncurry } (\rightarrow) \delta) \# \text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{J} \Psi \Delta \ominus (\mathfrak{F} \Psi \Delta)))$ 
         $\textcircled{\small @} \text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{G} \Psi \Delta)$ 
        using calculation by auto
      ultimately show ?thesis
        by (simp, metis union-mset-add-mset-right)
    }
next
  case False
  from this obtain  $\psi$  where
     $\psi: \text{find } (\lambda \psi. \text{uncurry } (\rightarrow) \psi = \text{snd } \delta) \Psi = \text{Some } \psi$ 
     $\text{uncurry } (\rightarrow) \psi = \text{snd } \delta$ 
  using find-Some-predicate
  by fastforce
  let  $?\alpha = \text{fst } \psi$ 
  let  $?\beta = \text{fst } \delta$ 

```



**let**  $?γ = \text{snd } ψ$   
**have**  $(λ \delta. \text{fst } \delta \rightarrow \text{snd } \delta) = \text{uncurry } (\rightarrow)$  **by** *fastforce*  
**hence**  $?β \rightarrow ?α \rightarrow ?γ = \text{uncurry } (\rightarrow) \delta$  **using**  $ψ(2)$  **by** *metis*  
**moreover**  
**let**  $?A = \mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta$   
**let**  $?B = \mathfrak{X} (\text{remove1 } ψ \Psi) \Delta$   
**let**  $?C = \mathfrak{B} (\text{remove1 } ψ \Psi) \Delta$   
**let**  $?D = ?A \ominus ((\text{remove1 } ψ \Psi) \ominus ?B)$   
**have**  $\text{mset } ((\text{remove1 } ψ \Psi) \ominus ?B) \subseteq\# \text{mset } ?A$   
**using** *Y-witness-first-component-diff-decomposition* **by** *simp*  
**{**  
**assume**  $\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)) \subseteq\# \text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta)$   
**moreover have**  $B: \forall \Phi \Psi. \exists \Delta. \Psi \subseteq\# \Phi \longrightarrow \Psi + \Delta = \Phi$   
**by** *(metis subset-mset.le-iff-add)*  
**moreover obtain**  $f$  **where**  
 $A: \text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta)$   
 $\quad - (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad = f (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad \quad (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
**by** *blast*  
**ultimately obtain**  $g$  **where**  
 $B: \forall p. \text{add-mset } p (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad - (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad = \text{add-mset } p$   
 $\quad \quad (g (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad \quad (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta))))$   
**by** *(metis add-diff-cancel-left' union-mset-add-mset-right)*  
**have**  $g (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad = \text{add-mset } (\text{fst } ψ, (\text{fst } ψ \rightarrow \text{fst } \delta) \rightarrow \text{snd } ψ)$   
 $\quad \quad (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad - (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad - \{\#(\text{fst } ψ, (\text{fst } ψ \rightarrow \text{fst } \delta) \rightarrow \text{snd } ψ)\#$   
**by** *(simp add: B)*  
**then have**  $C:$   
 $g (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad = \text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta)$   
 $\quad - (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
**by** *simp*  
**let**  $?S_1 =$   
 $\{\# x \rightarrow y.$   
 $\quad (x, y) \in\# \text{add-mset } (\text{fst } ψ, (\text{fst } ψ \rightarrow \text{fst } \delta) \rightarrow \text{snd } ψ)$   
 $\quad \quad (\text{mset } (\mathfrak{Y} (\text{remove1 } ψ \Psi) \Delta))$   
 $\quad \quad - (\text{mset } \Psi - \text{add-mset } ψ (\text{mset } (\mathfrak{X} (\text{remove1 } ψ \Psi) \Delta)))$   
 $\quad \#$   
**let**  $?S_2 =$   
 $\text{add-mset}$

```

      (fst  $\psi \rightarrow$  (fst  $\psi \rightarrow$  fst  $\delta$ )  $\rightarrow$  snd  $\psi$ )
      {#  $x \rightarrow y$ .
        ( $x, y$ )  $\in$  # mset ( $\mathfrak{Q}$ ) (remove1  $\psi$   $\Psi$ )  $\Delta$ )
          - (mset  $\Psi$ 
            - add-mset  $\psi$  (mset ( $\mathfrak{A}$ ) (remove1  $\psi$   $\Psi$ )  $\Delta$ )))
      #}
    have ? $S_1 = ?S_2$ 
      using  $A C$  by (simp add:  $B$ )
  }
  hence mset (map (uncurry ( $\rightarrow$ ))
    (((? $\alpha$ , (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ ) # ? $A$ )  $\ominus$  remove1  $\psi$  ( $\Psi \ominus ?B$ )
      @ (remove1  $\delta$  (( $\delta$  #  $\Delta$ )  $\ominus ?C$ ))))
    = mset ((? $\alpha \rightarrow$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ ) # map (uncurry ( $\rightarrow$ )) (? $D$  @ ( $\Delta \ominus$ 
? $C$ )))
  using
    add-mset-add-single
    image-mset-add-mset
    prod.simps(2)
    subset-mset.diff-add-assoc2
     $\langle$  mset (remove1  $\psi$   $\Psi \ominus \mathfrak{A}$  (remove1  $\psi$   $\Psi$ )  $\Delta$ )  $\subseteq$  # mset ( $\mathfrak{Q}$ ) (remove1  $\psi$ 
 $\Psi$ )  $\Delta$   $\rangle$ 
    by fastforce
  moreover
  have  $\vdash$  (? $\alpha \rightarrow$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ )  $\rightarrow ?\beta \rightarrow ?\alpha \rightarrow ?\gamma$ 
  proof -
    let ? $\Gamma =$  [(? $\alpha \rightarrow$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ ), ? $\beta$ , ? $\alpha$ ]
    have ? $\Gamma$  : $\vdash$  ? $\alpha \rightarrow$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ 
      ? $\Gamma$  : $\vdash$  ? $\alpha$ 
      by (simp add: list-deduction-reflection)+
    hence ? $\Gamma$  : $\vdash$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$ 
      using list-deduction-modus-ponens by blast
    moreover have ? $\Gamma$  : $\vdash$  ? $\beta$ 
      by (simp add: list-deduction-reflection)
    hence ? $\Gamma$  : $\vdash$  ? $\alpha \rightarrow ?\beta$ 
      using axiom-k list-deduction-modus-ponens list-deduction-weaken by blast
    ultimately have ? $\Gamma$  : $\vdash$  ? $\gamma$ 
      using list-deduction-modus-ponens by blast
    thus ?thesis
      unfolding list-deduction-def by simp
  qed
  hence (? $\beta \rightarrow ?\alpha \rightarrow ?\gamma$  # map (uncurry ( $\rightarrow$ ))  $\Delta$ )  $\preceq$ 
    (? $\alpha \rightarrow$  (? $\alpha \rightarrow ?\beta$ )  $\rightarrow ?\gamma$  # map (uncurry ( $\rightarrow$ )) (? $D$  @ ( $\Delta \ominus ?C$ )))
    using Cons stronger-theory-left-right-cons by blast
  ultimately show ?thesis
    using  $\psi$  by (simp add: stronger-theory-relation-alt-def)
  qed
}
then show ?case by blast
qed

```

thus *?thesis* by *blast*  
qed

**lemma** (in *implication-logic*) *xcomponent-ycomponent-connection*:

$map (uncurry (\rightarrow)) (\mathfrak{X}\bullet \Psi \Delta) = map\ snd (\mathfrak{Y}\bullet \Psi \Delta)$

**proof** –

**have**  $\forall \Psi. map (uncurry (\rightarrow)) (\mathfrak{X}\bullet \Psi \Delta) = map\ snd (\mathfrak{Y}\bullet \Psi \Delta)$

**proof** (*induct*  $\Delta$ )

**case** *Nil*

**then show** *?case* by *simp*

**next**

**case** (*Cons*  $\delta \Delta$ )

    {

**fix**  $\Psi$

**have**  $map (uncurry (\rightarrow)) (\mathfrak{X}\bullet \Psi (\delta \# \Delta)) = map\ snd (\mathfrak{Y}\bullet \Psi (\delta \# \Delta))$

**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 *classical-logic*) *xwitness-ywitness-measure-deduction-intro*:

**assumes**  $mset (map\ snd \Psi) \subseteq\# mset \Gamma$

**and**  $mset (map\ snd \Delta) \subseteq\# mset (map (uncurry (\rightarrow)) \Psi @ \Gamma @ (map\ snd \Psi))$

**and**  $map (uncurry (\rightarrow)) \Delta @ (map (uncurry (\rightarrow)) \Psi @ \Gamma @ map\ snd \Psi) @ map\ snd \Delta \ \$\vdash \Phi$

**(is**  $? \Gamma_0 \ \$\vdash \Phi$ )

**shows**  $map (uncurry (\rightarrow)) (\mathfrak{Y} \Psi \Delta) @$

$(map (uncurry (\rightarrow)) (\mathfrak{X} \Psi \Delta) @ \Gamma @ map\ snd (\mathfrak{X} \Psi \Delta)) @$

$map\ snd (\mathfrak{Y} \Psi \Delta) \ \$\vdash \Phi$

**(is**  $? \Gamma \ \$\vdash \Phi$ )

**proof** –

**let**  $?A = map (uncurry (\rightarrow)) (\mathfrak{Y} \Psi \Delta)$

**let**  $?B = map (uncurry (\rightarrow)) (\mathfrak{X} \Psi \Delta)$

**let**  $?C = \Psi @ \mathfrak{A} \Psi \Delta$

**let**  $?D = map (uncurry (\rightarrow)) ?C$

**let**  $?E = \Delta @ \mathfrak{B} \Psi \Delta$

**let**  $?F = map (uncurry (\rightarrow)) ?E$

**let**  $?G = map\ snd (\mathfrak{B} \Psi \Delta)$

**let**  $?H = map (uncurry (\rightarrow)) (\mathfrak{X}\bullet \Psi \Delta)$

**let**  $?I = \mathfrak{A} \Psi \Delta$

**let**  $?J = map\ snd (\mathfrak{X} \Psi \Delta)$

**let**  $?K = map\ snd (\mathfrak{Y} \Psi \Delta)$

**have**  $mset (map (uncurry (\rightarrow)) (\mathfrak{Y} \Psi \Delta @ ?C @ ?E)) = mset (?A @ ?D @ ?F)$

**by** (*simp add: Y-witness-first-component-diff-decomposition*)

**hence**  $(map (uncurry (\rightarrow)) \Delta) \preceq (?A @ ?D @ ?F)$

**using** *Y-witness-right-stronger-theory*  
*stronger-theory-relation-alt-def*  
**by** (*simp*, *metis* (*no-types*, *lifting*))  
**hence**  $? \Gamma_0 \preceq ((?A \ominus ?D @ ?F) @ (\text{map} (\text{uncurry} (\rightarrow)) \Psi @ \Gamma \ominus \text{map} \text{snd} \Psi) \ominus \text{map} \text{snd} \Delta)$   
**using** *stronger-theory-combine stronger-theory-reflexive* **by** *blast*  
**moreover**  
**have**  $\spadesuit$ :  $\text{mset } ?G \subseteq\# \text{mset} (\text{map} (\text{uncurry} (\rightarrow)) \Psi)$   
 $\text{mset} (\mathfrak{B} \Psi \Delta) \subseteq\# \text{mset} \Delta$   
 $\text{mset} (\text{map} \text{snd} ?E) \subseteq\# \text{mset} (\Gamma \ominus \text{map} \text{snd} \Psi)$   
 $\text{mset} (\text{map} (\text{uncurry} (\rightarrow)) \Psi \ominus ?G) = \text{mset } ?D$   
 $\text{mset } ?D \subseteq\# \text{mset } ?A$   
 $\text{mset} (\text{map} \text{snd} ?I) \subseteq\# \text{mset} (\text{map} \text{snd} \Psi)$   
 $\text{mset} (\text{map} \text{snd} ?I) \subseteq\# \text{mset} \Gamma$   
 $\text{mset} (\text{map} \text{snd} (?I @ ?E)) = \text{mset } ?J$   
**using** *second-component-msub*  
*second-component-diff-msub*  
*second-component-snd-projection-msub*  
*first-component-second-component-mset-connection*  
*X-witness-map-snd-decomposition*  
  
**by** (*simp*,  
*simp*,  
*metis* *assms*(2),  
*simp* *add: image-mset-Diff first-component-msub*,  
*simp* *add: Y-witness-first-component-diff-decomposition*,  
*simp* *add: image-mset-subseteq-mono first-component-msub*,  
*metis* *assms*(1) *first-component-msub map-monotonic subset-mset.dual-order.trans*,  
*simp*)  
**hence**  $\text{mset } \Delta - \text{mset} (\mathfrak{B} \Psi \Delta) + \text{mset} (\mathfrak{B} \Psi \Delta) = \text{mset } \Delta$   
**by** *simp*  
**hence**  $\heartsuit$ :  $\{\#x \rightarrow y. (x, y) \in\# \text{mset } \Psi\# \} + (\text{mset } \Gamma - \text{image-mset} \text{snd} (\text{mset} \Psi))$   
 $\quad - \text{image-mset} \text{snd} (\text{mset } \Delta)$   
 $= \{\#x \rightarrow y. (x, y) \in\# \text{mset } \Psi\# \} + (\text{mset } \Gamma - \text{image-mset} \text{snd} (\text{mset } \Psi))$   
 $\quad - \text{image-mset} \text{snd} (\text{mset } \Delta - \text{mset} (\mathfrak{B} \Psi \Delta))$   
 $\quad - \text{image-mset} \text{snd} (\text{mset} (\mathfrak{B} \Psi \Delta))$   
 $\quad \text{image-mset} \text{snd} (\text{mset } \Psi - \text{mset} (\mathfrak{A} \Psi \Delta)) + \text{image-mset} \text{snd} (\text{mset} (\mathfrak{A} \Psi \Delta))$   
 $= \text{image-mset} \text{snd} (\text{mset } \Psi)$   
**using**  $\spadesuit$   
**by** (*metis* (*no-types*) *diff-diff-add-mset image-mset-union*,  
*metis* (*no-types*) *image-mset-union first-component-msub subset-mset.diff-add*)  
**then have**  $\text{mset } \Gamma - \text{image-mset} \text{snd} (\text{mset } \Psi)$   
 $\quad - \text{image-mset} \text{snd} (\text{mset } \Delta - \text{mset} (\mathfrak{B} \Psi \Delta))$   
 $= \text{mset } \Gamma - (\text{image-mset} \text{snd} (\text{mset } \Psi - \text{mset} (\mathfrak{A} \Psi \Delta))$   
 $\quad + \text{image-mset} \text{snd} (\text{mset} (\mathfrak{X} \Psi \Delta)))$   
**using**  $\spadesuit$  **by** (*simp*, *metis* (*full-types*) *diff-diff-add-mset*)  
**hence**  $\text{mset} ((\text{map} (\text{uncurry} (\rightarrow)) \Psi @ \Gamma \ominus \text{map} \text{snd} \Psi) \ominus \text{map} \text{snd} \Delta)$

=  $mset (?D @ (\Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
**using**  $\heartsuit \spadesuit$  **by** (*simp*, *metis* (*no-types*) *add.commute subset-mset.add-diff-assoc*)  
**ultimately have**  $? \Gamma_0 \preceq ((?A \ominus ?D @ ?F) @ ?D @ (\Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
**unfolding** *stronger-theory-relation-alt-def*  
**by** *simp*  
**moreover**  
**have**  $mset\ ?F = mset\ (?B \ominus ?H)$   
 $mset\ ?D \subseteq\# mset\ ?A$   
 $mset\ (map\ snd\ (\Psi \ominus ?I)) \subseteq\# mset\ (\Gamma \ominus ?J)$   
**by** (*simp add: X-witness-second-component-diff-decomposition*,  
*simp add: Y-witness-first-component-diff-decomposition*,  
*simp, metis* (*no-types, lifting*)  
 $\heartsuit(2) \spadesuit(8)$  *add.assoc assms(1) assms(2) image-mset-union*  
*X-witness-msub merge-witness-msub-intro*  
*second-component-merge-witness-snd-projection*  
*mset-map*  
*subset-mset.le-diff-conv2*  
*union-code*)  
**hence**  $mset\ ((?A \ominus ?D @ ?F) @ ?D @ (\Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
 =  $mset\ (?A @ (?B \ominus ?H @ \Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
 $mset\ ?H \subseteq\# mset\ ?B$   
 $\{\#x \rightarrow y. (x, y) \in\# mset\ (\mathfrak{X} \bullet \Psi \Delta)\#\} = mset\ (map\ snd\ (\mathfrak{Y} \bullet \Psi \Delta))$   
**by** (*simp add: subset-mset.diff-add-assoc*,  
*simp add: X-witness-second-component-diff-decomposition*,  
*metis xcomponent-ycomponent-connection mset-map uncurry-def*)  
**hence**  $mset\ ((?A \ominus ?D @ ?F) @ ?D @ (\Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
 =  $mset\ (?A @ (?B @ \Gamma \ominus ?J) \ominus (?H @ map\ snd\ ?C))$   
 $\{\#x \rightarrow y. (x, y) \in\# mset\ (\mathfrak{X} \bullet \Psi \Delta)\#\} + image-mset\ snd\ (mset\ \Psi - mset\ (\mathfrak{A} \Psi \Delta))$   
 =  $mset\ (map\ snd\ (\mathfrak{Y} \Psi \Delta))$   
**using** *Y-witness-map-snd-decomposition*  
**by** (*simp add: subset-mset.diff-add-assoc, force*)  
**hence**  $mset\ ((?A \ominus ?D @ ?F) @ ?D @ (\Gamma \ominus ?J) \ominus map\ snd\ ?C)$   
 =  $mset\ (?A @ (?B @ \Gamma \ominus ?J) \ominus ?K)$   
**by** (*simp*)  
**ultimately have**  $? \Gamma_0 \preceq (?A @ (?B @ \Gamma \ominus ?J) \ominus ?K)$   
**unfolding** *stronger-theory-relation-alt-def*  
**by** *metis*  
**thus** *?thesis*  
**using** *assms(3) measure-stronger-theory-left-monotonic*  
**by** *blast*  
**qed**

**lemma** (in *classical-logic*) *measure-cons-cons-right-permute*:

**assumes**  $\Gamma \mathcal{S} \vdash (\varphi \# \psi \# \Phi)$

**shows**  $\Gamma \mathcal{S} \vdash (\psi \# \varphi \# \Phi)$

**proof** –

**from** *assms* **obtain**  $\Psi$  **where**  $\Psi$ :

$mset\ (map\ snd\ \Psi) \subseteq\# mset\ \Gamma$

```

    map (uncurry (⊔)) Ψ ⊢ φ
    map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ) $⊢ (ψ # Φ)
  by fastforce
let ?Γ0 = map (uncurry (→)) Ψ @ Γ ⊖ (map snd Ψ)
from Ψ(3) obtain Δ where Δ:
  mset (map snd Δ) ⊆# mset ?Γ0
  map (uncurry (⊔)) Δ ⊢ ψ
  (map (uncurry (→)) Δ @ ?Γ0 ⊖ (map snd Δ)) $⊢ Φ
  using measure-deduction.simps(2) by blast
let ?Ψ' = ⋈ Ψ Δ
let ?Γ1 = map (uncurry (→)) ?Ψ' @ Γ ⊖ (map snd ?Ψ')
let ?Δ' = ⋈ Ψ Δ
have (map (uncurry (→)) ?Δ' @ ?Γ1 ⊖ (map snd ?Δ')) $⊢ Φ
  map (uncurry (⊔)) Ψ ⋖ map (uncurry (⊔)) ?Δ'
  using Ψ(1) Δ(1) Δ(3)
  xwitness-ywitness-measure-deduction-intro
  Y-witness-left-stronger-theory
  by auto
hence ?Γ1 $⊢ (φ # Φ)
  using Ψ(1) Ψ(2) Δ(1)
  Y-witness-msub measure-deduction.simps(2)
  stronger-theory-deduction-monotonic
  by blast
thus ?thesis
  using Ψ(1) Δ(1) Δ(2)
  X-witness-msub
  X-witness-right-stronger-theory
  measure-deduction.simps(2)
  stronger-theory-deduction-monotonic
  by blast
qed

```

**lemma** (in *classical-logic*) *measure-cons-remove1*:

```

  assumes φ ∈ set Φ
  shows Γ $⊢ Φ = Γ $⊢ (φ # (remove1 φ Φ))
proof -
  from ⟨φ ∈ set Φ⟩
  have ∀ Γ. Γ $⊢ Φ = Γ $⊢ (φ # (remove1 φ Φ))
  proof (induct Φ)
    case Nil
    then show ?case by simp
  next
    case (Cons χ Φ)
    {
      fix Γ
      have Γ $⊢ (χ # Φ) = Γ $⊢ (φ # (remove1 φ (χ # Φ)))
      proof (cases χ = φ)
        case True
        then show ?thesis by simp
      }
    }
  }

```

```

next
  case False
  hence  $\varphi \in \text{set } \Phi$ 
  using Cons.prems by simp
  with Cons.hyps have  $\Gamma \ \$\vdash (\chi \ \# \ \Phi) = \Gamma \ \$\vdash (\chi \ \# \ \varphi \ \# \ (\text{remove1 } \varphi \ \Phi))$ 
  by fastforce
  hence  $\Gamma \ \$\vdash (\chi \ \# \ \Phi) = \Gamma \ \$\vdash (\varphi \ \# \ \chi \ \# \ (\text{remove1 } \varphi \ \Phi))$ 
  using measure-cons-cons-right-permute by blast
  then show ?thesis using  $\langle \chi \neq \varphi \rangle$  by simp
qed
}
then show ?case by blast
qed
thus ?thesis using assms by blast
qed

lemma (in classical-logic) witness-stronger-theory:
  assumes  $\text{mset } (\text{map } \text{snd } \Psi) \subseteq\# \text{mset } \Gamma$ 
  shows  $(\text{map } (\text{uncurry } (\rightarrow))) \Psi \ @ \ \Gamma \ \ominus \ (\text{map } \text{snd } \Psi) \preceq \Gamma$ 
proof -
  have  $\forall \Gamma. \text{mset } (\text{map } \text{snd } \Psi) \subseteq\# \text{mset } \Gamma \longrightarrow (\text{map } (\text{uncurry } (\rightarrow))) \Psi \ @ \ \Gamma \ \ominus \ (\text{map } \text{snd } \Psi) \preceq \Gamma$ 
  proof (induct  $\Psi$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\psi \ \Psi$ )
    let  $? \gamma = \text{snd } \psi$ 
    {
      fix  $\Gamma$ 
      assume  $\text{mset } (\text{map } \text{snd } (\psi \ \# \ \Psi)) \subseteq\# \text{mset } \Gamma$ 
      hence  $\text{mset } (\text{map } \text{snd } \Psi) \subseteq\# \text{mset } (\text{remove1 } (\text{snd } \psi) \ \Gamma)$ 
      by (simp add: insert-subset-eq-iff)
      with Cons have
         $(\text{map } (\text{uncurry } (\rightarrow))) \Psi \ @ \ (\text{remove1 } (\text{snd } \psi) \ \Gamma) \ \ominus \ (\text{map } \text{snd } \Psi) \preceq (\text{remove1 } ? \gamma \ \Gamma)$ 
      by blast
      hence  $(\text{map } (\text{uncurry } (\rightarrow))) \Psi \ @ \ \Gamma \ \ominus \ (\text{map } \text{snd } (\psi \ \# \ \Psi)) \preceq (\text{remove1 } ? \gamma \ \Gamma)$ 
      by (simp add: stronger-theory-relation-alt-def)
      moreover
      have  $(\text{uncurry } (\rightarrow)) = (\lambda \psi. \text{fst } \psi \rightarrow \text{snd } \psi)$ 
      by fastforce
      hence  $\vdash ? \gamma \rightarrow \text{uncurry } (\rightarrow) \ \psi$ 
      using axiom-k by simp
      ultimately have
         $(\text{map } (\text{uncurry } (\rightarrow))) (\psi \ \# \ \Psi) \ @ \ \Gamma \ \ominus \ (\text{map } \text{snd } (\psi \ \# \ \Psi)) \preceq (? \gamma \ \# \ (\text{remove1 } ? \gamma \ \Gamma))$ 
      using stronger-theory-left-right-cons by auto
      hence  $(\text{map } (\text{uncurry } (\rightarrow))) (\psi \ \# \ \Psi) \ @ \ \Gamma \ \ominus \ (\text{map } \text{snd } (\psi \ \# \ \Psi)) \preceq \Gamma$ 
    }
  qed

```

```

    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-logic) measure-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 measure-deduction.simps(2)
          measure-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⟩
          measure-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  $\Psi \subseteq\#$  mset  $\Phi$ 
      by (metis (no-types) False
          diff-single-trivial
          in-multiset-in-set mset.simps(1)
          subset-eq-diff-conv)
  then show ?thesis
    using  $\langle \Gamma \ \$\vdash \ \Phi \rangle$  Cons
    by blast
qed
}
then show ?case by blast
qed
with assms show ?thesis by blast
qed

```

lemma (in *classical-logic*) *measure-stronger-theory-right-antitonic*:

```

assumes  $\Psi \preceq \Phi$ 
  and  $\Gamma \ \$\vdash \ \Phi$ 
shows  $\Gamma \ \$\vdash \ \Psi$ 

```

proof –

```

have  $\forall \Psi \ \Gamma. \ \Psi \preceq \Phi \longrightarrow \Gamma \ \$\vdash \ \Phi \longrightarrow \Gamma \ \$\vdash \ \Psi$ 

```

proof (induct  $\Phi$ )

case Nil

then show ?case

```

  using measure-deduction.simps(1)

```

```

  stronger-theory-empty-list-intro

```

by blast

next

```

case (Cons  $\varphi \ \Phi$ )

```

```

{

```

```

  fix  $\Psi \ \Gamma$ 

```

```

  assume  $\Gamma \ \$\vdash \ (\varphi \ \#\ \Phi)$ 

```

```

     $\Psi \preceq (\varphi \ \#\ \Phi)$ 

```

from this obtain  $\Sigma$  where

```

 $\Sigma: \text{map snd } \Sigma = \Psi$ 

```

```

 $\text{mset } (\text{map fst } \Sigma) \subseteq\# \text{mset } (\varphi \ \#\ \Phi)$ 

```

```

 $\forall (\varphi, \psi) \in \text{set } \Sigma. \vdash \varphi \rightarrow \psi$ 

```

unfolding stronger-theory-relation-def

by auto

hence  $\Gamma \ \$\vdash \ \Psi$

proof (cases  $\varphi \in \text{set } (\text{map fst } \Sigma)$ )

case True

from this obtain  $\psi$  where  $(\varphi, \psi) \in \text{set } \Sigma$

by (induct  $\Sigma$ , simp, fastforce)

hence A:  $\text{mset } (\text{map snd } (\text{remove1 } (\varphi, \psi) \ \Sigma)) = \text{mset } (\text{remove1 } \psi \ \Psi)$

and B:  $\text{mset } (\text{map fst } (\text{remove1 } (\varphi, \psi) \ \Sigma)) \subseteq\# \text{mset } \Phi$

using  $\Sigma$  remove1-pairs-list-projections-snd

```

  remove1-pairs-list-projections-fst

```

```

  subset-eq-diff-conv

```

```

    by fastforce+
  have  $\forall (\varphi, \psi) \in \text{set } (\text{remove1 } (\varphi, \psi) \Sigma). \vdash \varphi \rightarrow \psi$ 
    using  $\Sigma(3)$  by fastforce+
  hence  $(\text{remove1 } \psi \Psi) \preceq \Phi$ 
    unfolding stronger-theory-relation-alt-def using A B by blast
  moreover
  from  $\langle \Gamma \ \$\vdash (\varphi \# \Phi) \rangle$  obtain  $\Delta$  where
     $\Delta: \text{mset } (\text{map } \text{snd } \Delta) \subseteq \# \text{mset } \Gamma$ 
     $\text{map } (\text{uncurry } (\sqcup)) \Delta \vdash \varphi$ 
     $(\text{map } (\text{uncurry } (\rightarrow)) \Delta @ \Gamma \ominus (\text{map } \text{snd } \Delta)) \ \$\vdash \Phi$ 
    by auto
  ultimately have  $(\text{map } (\text{uncurry } (\rightarrow)) \Delta @ \Gamma \ominus (\text{map } \text{snd } \Delta)) \ \$\vdash \text{remove1}$ 
 $\psi \Psi$ 
    using Cons by blast
  moreover have  $\text{map } (\text{uncurry } (\sqcup)) \Delta \vdash \psi$ 
    using  $\Delta(2) \Sigma(3) \langle (\varphi, \psi) \in \text{set } \Sigma \rangle$ 
    list-deduction-weaken
    list-deduction-modus-ponens
    by blast
  ultimately have  $\langle \Gamma \ \$\vdash (\psi \# (\text{remove1 } \psi \Psi)) \rangle$ 
    using  $\Delta(1)$  by auto
  moreover from  $\langle (\varphi, \psi) \in \text{set } \Sigma \rangle \Sigma(1)$  have  $\psi \in \text{set } \Psi$ 
    by force
  hence  $\text{mset } \Psi \subseteq \# \text{mset } (\psi \# (\text{remove1 } \psi \Psi))$ 
    by auto
  ultimately show ?thesis using measure-msub-weaken by blast
next
case False
  hence  $\text{mset } (\text{map } \text{fst } \Sigma) \subseteq \# \text{mset } \Phi$ 
    using  $\Sigma(2)$ 
    by (simp,
      metis add-mset-add-single
      diff-single-trivial
      mset-map set-mset-mset
      subset-eq-diff-conv)
  hence  $\Psi \preceq \Phi$ 
    using  $\Sigma(1) \Sigma(3)$ 
    unfolding stronger-theory-relation-def
    by auto
  moreover from  $\langle \Gamma \ \$\vdash (\varphi \# \Phi) \rangle$  have  $\Gamma \ \$\vdash \Phi$ 
    using measure-deduction.simps(2)
    measure-stronger-theory-left-monotonic
    witness-stronger-theory
    by blast
  ultimately show ?thesis using Cons by blast
qed
}
then show ?case by blast
qed

```

thus *?thesis using assms by blast*  
qed

**lemma** (in *classical-logic*) *measure-witness-right-split*:

**assumes**  $mset (map\ snd\ \Psi) \subseteq\# mset\ \Phi$

**shows**  $\Gamma \ \$\vdash (map\ (uncurry\ (\sqcup))\ \Psi @ map\ (uncurry\ (\rightarrow))\ \Psi @ \Phi \ominus (map\ snd\ \Psi)) = \Gamma \ \$\vdash\ \Phi$

**proof** –

**have**  $\forall\ \Gamma\ \Phi. mset (map\ snd\ \Psi) \subseteq\# mset\ \Phi \longrightarrow$

$\Gamma \ \$\vdash\ \Phi = \Gamma \ \$\vdash (map\ (uncurry\ (\sqcup))\ \Psi @ map\ (uncurry\ (\rightarrow))\ \Psi @ \Phi \ominus (map\ snd\ \Psi))$

**proof** (*induct*  $\Psi$ )

**case** *Nil*

**then show** *?case by simp*

**next**

**case** (*Cons*  $\psi\ \Psi$ )

{

**fix**  $\Gamma\ \Phi$

**let**  $?X = fst\ \psi$

**let**  $?Y = snd\ \psi$

**let**  $?X' = map\ (uncurry\ (\sqcup))\ (\psi\ \# \Psi) @$   
 $map\ (uncurry\ (\rightarrow))\ (\psi\ \# \Psi) @$   
 $\Phi \ominus map\ snd\ (\psi\ \# \Psi)$

**let**  $?X_0 = map\ (uncurry\ (\sqcup))\ \Psi @$   
 $map\ (uncurry\ (\rightarrow))\ \Psi @$   
 $(remove1\ ?Y\ \Phi) \ominus map\ snd\ \Psi$

**assume**  $mset (map\ snd\ (\psi\ \# \Psi)) \subseteq\# mset\ \Phi$

**hence**  $mset (map\ snd\ \Psi) \subseteq\# mset (remove1\ ?Y\ \Phi)$

$mset\ (?Y\ \# remove1\ ?Y\ \Phi) = mset\ \Phi$

**by** (*simp add: insert-subset-eq-iff*)**+**

**hence**  $\Gamma \ \$\vdash\ \Phi = \Gamma \ \$\vdash\ (?Y\ \# remove1\ ?Y\ \Phi)$

$\forall\ \Gamma. \Gamma \ \$\vdash (remove1\ ?Y\ \Phi) = \Gamma \ \$\vdash\ ?X_0$

**by** (*metis list.set-intros(1) measure-cons-remove1 set-mset-mset,*  
*metis Cons.hyps*)

**moreover**

**have**  $(uncurry\ (\sqcup)) = (\lambda\ \psi. fst\ \psi\ \sqcup\ snd\ \psi)$

$(uncurry\ (\rightarrow)) = (\lambda\ \psi. fst\ \psi\ \rightarrow\ snd\ \psi)$

**by** *fastforce***+**

**hence**  $mset\ ?X' \subseteq\# mset\ (?X\ \sqcup\ ?Y\ \#\ ?X\ \rightarrow\ ?Y\ \#\ ?X_0)$

$mset\ (?X\ \sqcup\ ?Y\ \#\ ?X\ \rightarrow\ ?Y\ \#\ ?X_0) \subseteq\# mset\ ?X'$

(**is**  $mset\ ?X \subseteq\# mset\ ?Y$ )

**by** *fastforce***+**

**hence**  $\Gamma \ \$\vdash\ ?X' = \Gamma \ \$\vdash\ (?Y\ \#\ ?X_0)$

**using** *measure-formula-right-split*

*measure-msub-weaken*

**by** *blast*

**ultimately have**  $\Gamma \ \$\vdash\ \Phi = \Gamma \ \$\vdash\ ?X'$

**by** *fastforce*

}

```

    then show ?case by blast
  qed
  with assms show ?thesis by blast
qed

```

```

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

```

**lemma** (in *classical-logic*) *submerge-witness-stronger-theory-left*:

```

  map (uncurry (⊔)) Σ ≤ map (uncurry (⊔)) (⊔ Σ Δ)

```

**proof** –

```

  have ∀ Σ. map (uncurry (⊔)) Σ ≤ map (uncurry (⊔)) (⊔ 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 (⊔)) (⊔ Σ [])
```

```
  by (induct Σ,
```

```
    simp,
```

```
    simp add: stronger-theory-left-right-cons tautology)
```

```
}
```

```
  then show ?case by auto
```

**next**

```
case (Cons δ Δ)
```

```
{
```

```
  fix Σ
```

```
  have map (uncurry (⊔)) Σ ≤ map (uncurry (⊔)) (⊔ Σ (δ # Δ))
```

```
  proof (cases find (λ σ. (uncurry (→)) σ = snd δ) Σ = None)
```

```
    case True
```

```
    then show ?thesis using Cons by simp
```

**next**

```
case False
```

```
  from this obtain σ where
```

```
    σ: find (λσ. uncurry (→) σ = snd δ) Σ = Some σ
```

```
    uncurry (→) σ = snd δ
```

```
    σ ∈ set Σ
```

```

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

```

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

**lemma** (in *classical-logic*) *submerge-witness-msub*:

$mset (map\ snd\ (\mathfrak{E}\ \Sigma\ \Delta)) \subseteq\# mset (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta))$

**proof** –

**have**  $\forall\ \Sigma.\ mset (map\ snd\ (\mathfrak{E}\ \Sigma\ \Delta)) \subseteq\# mset (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta))$

**proof** (*induct*  $\Delta$ )

**case** *Nil*

{

**fix**  $\Sigma$

**have**  $mset (map\ snd\ (\mathfrak{E}\ \Sigma\ [])) \subseteq\#$

$mset (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ []))$

**by** (*induct*  $\Sigma$ , *simp+*)

}

**then show** *?case* **by** *blast*

**next**

**case** (*Cons*  $\delta\ \Delta$ )

{

**fix**  $\Sigma$

**have**  $mset (map\ snd\ (\mathfrak{E}\ \Sigma\ (\delta\ \#\ \Delta))) \subseteq\#$

$mset (map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ (\delta\ \#\ \Delta)))$

**using** *Cons*

**by** (*cases find* ( $\lambda\ \sigma.\ (uncurry\ (\rightarrow))\ \sigma = snd\ \delta$ )  $\Sigma = None$ ,

*simp*,

*meson diff-subset-eq-self*

*insert-subset-eq-iff*

*mset-subset-eq-add-mset-cancel*

*subset-mset.dual-order.trans*,

*fastforce*)

}

**then show** *?case* **by** *blast*

**qed**

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

**qed**

**lemma** (in *classical-logic*) *submerge-witness-stronger-theory-right*:

$map\ (uncurry\ (\sqcup))\ \Delta$

$\preceq (map\ (uncurry\ (\rightarrow))\ (\mathfrak{E}\ \Sigma\ \Delta) @ map\ (uncurry\ (\sqcup))\ (\mathfrak{J}\ \Sigma\ \Delta) \ominus map\ snd\ (\mathfrak{E}\ \Sigma\ \Delta))$

**proof** –

**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*  $\Delta$ )

**case** *Nil*

**then show** *?case* **by** *simp*

**next**

**case** (*Cons*  $\delta\ \Delta$ )

```

{
  fix  $\Sigma$ 
  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 ( $\rightarrow$ ))  $\sigma = \text{snd } \delta$ )  $\Sigma = \text{None}$ )
    case True
    from Cons obtain  $\Phi$  where  $\Phi$ :
      map snd  $\Phi = \text{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 ( $\mathfrak{E} \Sigma \Delta$ ))
       $\forall (\gamma, \sigma) \in \text{set } \Phi. \vdash \gamma \rightarrow \sigma$ 
    unfolding stronger-theory-relation-def
    by fastforce
  let  $?\Phi' = (\text{uncurry } (\sqcup) \delta, (\text{uncurry } (\sqcup) \delta) \# \Phi$ 
  have map snd  $?\Phi' = \text{map (uncurry } (\sqcup)) (\delta \# \Delta)$  using  $\Phi(1)$  by simp
  moreover
  from  $\Phi(2)$  have A:
    image-mset fst (mset  $\Phi$ )
   $\subseteq\# \{ \#x \rightarrow y. (x, y) \in\# \text{mset } (\mathfrak{E} \Sigma \Delta) \# \}$ 
    + ( $\{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma \Delta) \# \} - \text{image-mset snd (mset } (\mathfrak{E} \Sigma$ 
 $\Delta))$ )
    by simp
  have image-mset snd (mset ( $\mathfrak{E} \Sigma \Delta$ ))  $\subseteq\# \{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma$ 
 $\Delta) \# \}$ 
    using submerge-witness-msub by force
  then have B:  $\{ \# \text{case } \delta \text{ of } (x, xa) \Rightarrow x \sqcup xa \# \}$ 
     $\subseteq\# \text{add-mset (case } \delta \text{ of } (x, xa) \Rightarrow x \sqcup xa$ 
       $\{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma \Delta) \# \} - \text{image-mset snd$ 
 $(\text{mset } (\mathfrak{E} \Sigma \Delta))$ 
    by (metis add-mset-add-single subset-mset.le-add-diff)
  have add-mset (case  $\delta$  of  $(x, xa) \Rightarrow x \sqcup xa$ )  $\{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma$ 
 $\Delta) \# \}$ 
     $- \text{image-mset snd (mset } (\mathfrak{E} \Sigma \Delta)) - \{ \# \text{case } \delta \text{ of } (x, xa) \Rightarrow x \sqcup xa \# \}$ 
    =  $\{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma \Delta) \# \} - \text{image-mset snd (mset } (\mathfrak{E} \Sigma$ 
 $\Delta))$ 
    by force
  then have add-mset (case  $\delta$  of  $(x, xa) \Rightarrow x \sqcup xa$ ) (image-mset fst (mset
 $\Phi$ ))
     $- (\text{add-mset (case } \delta \text{ of } (x, xa) \Rightarrow x \sqcup xa) \{ \#x \sqcup y. (x, y) \in\# \text{mset}$ 
 $(\mathfrak{J} \Sigma \Delta) \# \}$ 
       $- \text{image-mset snd (mset } (\mathfrak{E} \Sigma \Delta))$ )
     $\subseteq\# \{ \#x \rightarrow y. (x, y) \in\# \text{mset } (\mathfrak{E} \Sigma \Delta) \# \}$ 
    using A B by (metis (no-types) add-mset-add-single
      subset-eq-diff-conv
      subset-mset.diff-diff-right)
  hence add-mset (case  $\delta$  of  $(x, xa) \Rightarrow x \sqcup xa$ ) (image-mset fst (mset  $\Phi$ ))

```

```

     $\subseteq\# \{ \#x \rightarrow y. (x, y) \in\# \text{mset } (\mathfrak{E} \Sigma \Delta)\# \}$ 
    + (add-mset (case  $\delta$  of  $(x, xa) \Rightarrow x \sqcup xa$ )  $\{ \#x \sqcup y. (x, y) \in\# \text{mset } (\mathfrak{J} \Sigma \Delta)\# \}$ 
      - image-mset snd (mset ( $\mathfrak{E} \Sigma \Delta$ )))
    using subset-eq-diff-conv by blast
  hence
    mset (map fst ? $\Phi'$ )  $\subseteq\#$ 
      mset (map (uncurry ( $\rightarrow$ )) ( $\mathfrak{E} \Sigma (\delta \# \Delta)$ )
        @ map (uncurry ( $\sqcup$ )) ( $\mathfrak{J} \Sigma (\delta \# \Delta)$ )
         $\ominus$  map snd ( $\mathfrak{E} \Sigma (\delta \# \Delta)$ ))
    using True  $\Phi(2)$ 
    by simp
  moreover have  $\forall (\gamma, \sigma) \in \text{set } ?\Phi'. \vdash \gamma \rightarrow \sigma$ 
    using  $\Phi(3)$  trivial-implication by auto
  ultimately show ?thesis
    unfolding stronger-theory-relation-def
    by blast
next
case False
from this obtain  $\sigma$  where
   $\sigma$ : find ( $\lambda \sigma. \text{uncurry } (\rightarrow) \sigma = \text{snd } \delta$ )  $\Sigma = \text{Some } \sigma$ 
  uncurry ( $\rightarrow$ )  $\sigma = \text{snd } \delta$ 
  using find-Some-predicate
  by fastforce
moreover from Cons have
  map (uncurry ( $\sqcup$ ))  $\Delta \preceq$ 
  (map (uncurry ( $\rightarrow$ )) ( $\mathfrak{E} (\text{remove1 } \sigma \Sigma) \Delta$ ) @
    remove1 ((fst  $\delta \sqcap$  fst  $\sigma$ )  $\sqcup$  snd  $\sigma$ )
      (((fst  $\delta \sqcap$  fst  $\sigma$ )  $\sqcup$  snd  $\sigma \#$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{J} (\text{remove1 } \sigma \Sigma) \Delta$ ))
         $\ominus$  map snd ( $\mathfrak{E} (\text{remove1 } \sigma \Sigma) \Delta$ )))
  unfolding stronger-theory-relation-alt-def
  by simp
moreover
{
  fix  $\alpha \beta \gamma$ 
  have  $\vdash (\alpha \rightarrow ((\gamma \sqcap \alpha) \sqcup \beta)) \rightarrow (\gamma \sqcup (\alpha \rightarrow \beta))$ 
  proof -
    let ? $\varphi = (\langle \alpha \rangle \rightarrow ((\langle \gamma \rangle \sqcap \langle \alpha \rangle) \sqcup \langle \beta \rangle)) \rightarrow (\langle \gamma \rangle \sqcup (\langle \alpha \rangle \rightarrow \langle \beta \rangle))$ 
    have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  by fastforce
    hence  $\vdash (\langle ?\varphi \rangle)$  using propositional-semantic by blast
    thus ?thesis by simp
  qed
}
note tautology = this
let ? $\alpha = \text{fst } \sigma$ 
let ? $\beta = \text{snd } \sigma$ 
let ? $\gamma = \text{fst } \delta$ 
have  $(\lambda \delta. \text{uncurry } (\sqcup) \delta) = (\lambda \delta. \text{fst } \delta \sqcup \text{snd } \delta)$ 
   $(\lambda \sigma. \text{uncurry } (\rightarrow) \sigma) = (\lambda \sigma. \text{fst } \sigma \rightarrow \text{snd } \sigma)$  by fastforce+

```



**hence**  $(\text{uncurry } (\sqcup) \delta) = (? \gamma \sqcup (? \alpha \rightarrow ? \beta))$  **using**  $\sigma(2)$  **by** *simp*  
**hence**  $\vdash (? \alpha \rightarrow ((? \gamma \sqcap ? \alpha) \sqcup ? \beta)) \rightarrow (\text{uncurry } (\sqcup) \delta)$  **using** *tautology* **by**  
*auto*  
**ultimately show** *?thesis*  
**using** *stronger-theory-left-right-cons*  
**by** *fastforce*  
**qed**  
**}**  
**then show** *?case* **by** *auto*  
**qed**  
**thus** *?thesis* **by** *simp*  
**qed**

**lemma** (in *classical-logic*) *merge-witness-cons-measure-deduction*:

**assumes**  $\text{map } (\text{uncurry } (\sqcup)) \Sigma \vdash \varphi$   
**and**  $\text{mset } (\text{map } \text{snd } \Delta) \subseteq \# \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus \text{map } \text{snd } \Sigma)$   
**and**  $\text{map } (\text{uncurry } (\sqcup)) \Delta \ \$\vdash \Phi$   
**shows**  $\text{map } (\text{uncurry } (\sqcup)) (\mathfrak{J} \Sigma \Delta) \ \$\vdash (\varphi \# \Phi)$   
**proof** –  
**let**  $? \Sigma' = \mathfrak{E} \Sigma \Delta$   
**let**  $? \Gamma = \text{map } (\text{uncurry } (\rightarrow)) ? \Sigma' @ \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{J} \Sigma \Delta) \ominus \text{map } \text{snd } ? \Sigma'$   
**have**  $? \Gamma \ \$\vdash \Phi$   
**using** *assms(3)*  
*submerge-witness-stronger-theory-right*  
*measure-stronger-theory-left-monotonic*  
**by** *blast*  
**moreover have**  $\text{map } (\text{uncurry } (\sqcup)) ? \Sigma' \vdash \varphi$   
**using** *assms(1)*  
*stronger-theory-deduction-monotonic*  
*submerge-witness-stronger-theory-left*  
**by** *blast*  
**ultimately show** *?thesis*  
**using** *submerge-witness-msub*  
**by** *fastforce*  
**qed**

**primrec** (in *classical-logic*)

*recover-witness-A* ::  $('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list } (\mathfrak{P})$

**where**

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

**primrec** (in *classical-logic*)

*recover-complement-A* ::  $('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list} \Rightarrow ('a \times 'a) \text{ list } (\mathfrak{P}^C)$

**where**

$\mathfrak{P}^C \Sigma [] = []$

```

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

```

**primrec** (in *classical-logic*)

```

recover-witness-B :: ('a × 'a) list  $\Rightarrow$  ('a × 'a) list  $\Rightarrow$  ('a × 'a) list ( $\Omega$ )

```

**where**

```

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

```

**lemma** (in *classical-logic*) *recover-witness-A-left-stronger-theory*:

```

map (uncurry ( $\sqcup$ ))  $\Sigma \preceq$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{P} \Sigma \Delta$ )

```

**proof** –

```

have  $\forall \Sigma. \text{map } (\text{uncurry } (\sqcup)) \Sigma \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{P} \Sigma \Delta)$ 

```

**proof** (*induct*  $\Delta$ )

**case** *Nil*

```

{
  fix  $\Sigma$ 
  have map (uncurry ( $\sqcup$ ))  $\Sigma \preceq$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{P} \Sigma []$ )
  by(induct  $\Sigma$ , simp+)
}

```

**then show** *?case by auto*

**next**

**case** (*Cons*  $\delta \Delta$ )

```

{
  fix  $\Sigma$ 
  have map (uncurry ( $\sqcup$ ))  $\Sigma \preceq$  map (uncurry ( $\sqcup$ )) ( $\mathfrak{P} \Sigma (\delta \# \Delta)$ )
  proof (cases find ( $\lambda \sigma. \text{snd } \sigma = \text{uncurry } (\sqcup) \delta$ )  $\Sigma = \text{None}$ )
    case True
    then show ?thesis using Cons by simp
  next

```

**case** *False*

**from this obtain**  $\sigma$  **where**

```

 $\sigma: \text{find } (\lambda \sigma. \text{snd } \sigma = \text{uncurry } (\sqcup) \delta) \Sigma = \text{Some } \sigma$ 
   $\text{snd } \sigma = \text{uncurry } (\sqcup) \delta$ 
   $\sigma \in \text{set } \Sigma$ 

```

**using** *find-Some-predicate*

*find-Some-set-membership*

**by** *fastforce*

**let**  $?\alpha = \text{fst } \sigma$

**let**  $?\beta = \text{fst } \delta$

**let**  $?\gamma = \text{snd } \delta$

**have**  $\text{uncurry } (\sqcup) = (\lambda \delta. \text{fst } \delta \sqcup \text{snd } \delta)$  **by** *fastforce*

**hence**  $\vdash ((?\alpha \sqcup ?\beta) \sqcup ?\gamma) \rightarrow \text{uncurry } (\sqcup) \sigma$

**using**  $\sigma(2)$  *biconditional-def disjunction-associativity*

```

    by auto
  moreover
  have map (uncurry (⊔)) (remove1 σ Σ)
    ≤ map (uncurry (⊔)) (℘ (remove1 σ Σ) Δ)
    using Cons by simp
  ultimately have map (uncurry (⊔)) (σ # (remove1 σ Σ))
    ≤ map (uncurry (⊔)) (℘ Σ (δ # Δ))
    using σ(1)
    by (simp, metis stronger-theory-left-right-cons)
  moreover
  from σ(β) have mset Σ = mset (σ # (remove1 σ Σ))
    by simp
    hence mset (map (uncurry (⊔)) Σ) = mset (map (uncurry (⊔)) (σ #
(remove1 σ Σ)))
    by (metis mset-map)
    hence map (uncurry (⊔)) Σ ≤ map (uncurry (⊔)) (σ # (remove1 σ Σ))
    by (simp add: msub-stronger-theory-intro)
    ultimately show ?thesis
    using stronger-theory-transitive by blast
  qed
}
then show ?case by blast
qed
thus ?thesis by auto
qed

lemma (in classical-logic) recover-witness-A-mset-equiv:
  assumes mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
  shows mset (map snd (℘ Σ Δ @ ℘C Σ Δ)) = mset (map snd Δ)
proof -
  have ∀ Σ. mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
    → mset (map snd (℘ Σ Δ @ ℘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 (℘ Σ (δ # Δ) @ ℘C Σ (δ # Δ))) = mset (map snd (δ
# Δ))
      proof (cases find (λ σ. snd σ = uncurry (⊔) δ) Σ = None)
        case True
        hence uncurry (⊔) δ ∉ set (map snd Σ)
        proof (induct Σ)
          case Nil
          then show ?case by simp
        next

```

```

      case (Cons σ Σ)
      then show ?case
        by (cases (uncurry (⊔)) δ = snd σ, fastforce+)
    qed
    moreover have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ) +
  {#uncurry (⊔) δ#}
      using * by fastforce
    ultimately have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) 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 (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
        snd σ = uncurry (⊔) δ
        σ ∈ set Σ
    using find-Some-predicate
        find-Some-set-membership
    by fastforce
  have A: mset (map snd Σ)
    ⊆# mset (map (uncurry (⊔)) Δ) + add-mset (uncurry (⊔) δ) (mset [])
    using * by auto
  have (fst σ, uncurry (⊔) δ) ∈# 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 (⊔) δ) Σ))
    = mset (map snd Σ) - {#uncurry (⊔) δ#}
    by (meson remove1-pairs-list-projections-snd)
  have (fst σ, uncurry (⊔) δ) = σ
    by (metis σ(2) prod.collapse)
  then have mset (map snd Σ) - add-mset (uncurry (⊔) δ) (mset [])
    = mset (map snd (remove1 σ Σ))
    using B by simp
  hence mset (map snd (remove1 σ Σ)) ⊆# mset (map (uncurry (⊔)) Δ)
    using A by (metis (no-types) subset-eq-diff-conv)
  with σ(1) Cons show ?thesis by simp
}
}
then show ?case by simp
qed
with assms show ?thesis by blast
qed

lemma (in classical-logic) recover-witness-B-stronger-theory:
  assumes mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
  shows (map (uncurry (→)) Σ @ map (uncurry (⊔)) Δ ⊖ map snd Σ)
    ≤ map (uncurry (⊔)) (⊔ Σ Δ)
proof -

```

```

have  $\forall \Sigma. \text{mset} (\text{map snd } \Sigma) \subseteq\# \text{mset} (\text{map} (\text{uncurry } (\sqcup)) \Delta)$ 
   $\rightarrow (\text{map} (\text{uncurry } (\rightarrow)) \Sigma @ \text{map} (\text{uncurry } (\sqcup)) \Delta \ominus \text{map snd } \Sigma)$ 
   $\preceq \text{map} (\text{uncurry } (\sqcup)) (\Omega \Sigma \Delta)$ 
proof(induct  $\Delta$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\delta \Delta$ )
  {
  fix  $\Sigma :: ('a \times 'a) \text{ list}$ 
  assume  $\star: \text{mset} (\text{map snd } \Sigma) \subseteq\# \text{mset} (\text{map} (\text{uncurry } (\sqcup)) (\delta \# \Delta))$ 
  have  $(\text{map} (\text{uncurry } (\rightarrow)) \Sigma @ \text{map} (\text{uncurry } (\sqcup)) (\delta \# \Delta) \ominus \text{map snd } \Sigma)$ 
     $\preceq \text{map} (\text{uncurry } (\sqcup)) (\Omega \Sigma (\delta \# \Delta))$ 
  proof (cases find  $(\lambda \sigma. \text{snd } \sigma = \text{uncurry } (\sqcup) \delta) \Sigma = \text{None}$ )
    case True
    hence  $\text{uncurry } (\sqcup) \delta \notin \text{set} (\text{map snd } \Sigma)$ 
    proof (induct  $\Sigma$ )
      case Nil
      then show ?case by simp
    next
      case (Cons  $\sigma \Sigma$ )
      then show ?case
        by (cases uncurry  $(\sqcup) \delta = \text{snd } \sigma, \text{fastforce+}$ )
    qed
  hence  $\text{mset} (\text{map} (\text{uncurry } (\rightarrow)) \Sigma @ (\text{map} (\text{uncurry } (\sqcup)) (\delta \# \Delta)) \ominus \text{map}$ 
snd  $\Sigma)$ 
     $= \text{mset} (\text{uncurry } (\sqcup) \delta \# \text{map} (\text{uncurry } (\rightarrow)) \Sigma$ 
       $@ \text{map} (\text{uncurry } (\sqcup)) \Delta \ominus \text{map snd } \Sigma)$ 
     $\text{mset} (\text{map snd } \Sigma) \subseteq\# \text{mset} (\text{map} (\text{uncurry } (\sqcup)) \Delta)$ 
  using  $\star$ 
  by (simp, simp,
    metis add-mset-add-single
    diff-single-trivial
    image-set
    mset-map
    set-mset-mset
    subset-eq-diff-conv)
  moreover from this have
     $(\text{map} (\text{uncurry } (\rightarrow)) \Sigma @ \text{map} (\text{uncurry } (\sqcup)) \Delta \ominus \text{map snd } \Sigma)$ 
     $\preceq \text{map} (\text{uncurry } (\sqcup)) (\Omega \Sigma \Delta)$ 
  using Cons
  by auto
  hence  $(\text{uncurry } (\sqcup) \delta \# \text{map} (\text{uncurry } (\rightarrow)) \Sigma @ \text{map} (\text{uncurry } (\sqcup)) \Delta \ominus$ 
map snd  $\Sigma)$ 
     $\preceq \text{map} (\text{uncurry } (\sqcup)) (\Omega \Sigma (\delta \# \Delta))$ 
  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 ?Γ = map (uncurry (→)) Σ @ (map (uncurry (⊔)) (δ # Δ)) ⊖ map snd
Σ
from False obtain σ where
  σ: find (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
      snd σ = uncurry (⊔) δ
      σ ∈ set Σ
using find-Some-predicate
      find-Some-set-membership
by fastforce
let ?Γ₀ = map (uncurry (→)) (remove1 σ Σ)
      @ (map (uncurry (⊔)) Δ) ⊖ map snd (remove1 σ Σ)
let ?α = fst σ
let ?β = fst δ
let ?γ = snd δ
have uncurry (⊔) = (λ σ. fst σ ⊔ snd σ)
      uncurry (→) = (λ σ. fst σ → snd σ)
by fastforce+
hence uncurry (→) σ = ?α → (?β ⊔ ?γ)
using σ(2)
by simp
from σ(3) have mset (σ # (remove1 σ Σ)) = mset Σ by simp
hence ♠: mset (map snd (σ # (remove1 σ Σ))) = mset (map snd Σ)
      mset (map (uncurry (→)) (σ # (remove1 σ Σ))) = mset (map
(uncurry (→)) Σ)
by (metis mset-map)+
hence mset ?Γ = mset (map (uncurry (→)) (σ # (remove1 σ Σ))
      @ (uncurry (⊔) δ # map (uncurry (⊔)) Δ)
      ⊖ map snd (σ # (remove1 σ Σ)))

by simp
hence ?Γ ≼ (?α → (?β ⊔ ?γ) # ?Γ₀)
using σ(2) ⟨uncurry (→) σ = ?α → (?β ⊔ ?γ)⟩
by (simp add: msub-stronger-theory-intro)
moreover have mset (map snd (remove1 σ Σ)) ⊆# mset (map (uncurry
(⊔)) Δ)
using ♠(1)
by (simp,
      metis (no-types, lifting)
      * σ(2)
      list.simps(9)
      mset.simps(2)
      mset-map
      uncurry-def
      mset-subset-eq-add-mset-cancel)
with Cons have ♥: ?Γ₀ ≼ map (uncurry (⊔)) (Ω (remove1 σ Σ) Δ) by
simp
{

```

```

fix  $\alpha \beta \gamma$ 
have  $\vdash (\beta \sqcup (\alpha \sqcup \beta) \rightarrow \gamma) \rightarrow (\alpha \rightarrow (\beta \sqcup \gamma))$ 
proof -
  let  $?\varphi = (\langle \beta \rangle \sqcup (\langle \alpha \rangle \sqcup \langle \beta \rangle) \rightarrow \langle \gamma \rangle) \rightarrow (\langle \alpha \rangle \rightarrow (\langle \beta \rangle \sqcup \langle \gamma \rangle))$ 
  have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  by fastforce
  hence  $\vdash (\langle ?\varphi \rangle)$  using propositional-semantic by blast
  thus  $?thesis$  by simp
qed
}
hence  $\vdash (? \beta \sqcup (? \alpha \sqcup ? \beta) \rightarrow ? \gamma) \rightarrow (? \alpha \rightarrow (? \beta \sqcup ? \gamma))$ 
by simp
hence  $(? \alpha \rightarrow (? \beta \sqcup ? \gamma)) \# ?\Gamma_0 \preceq \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{Q} \Sigma (\delta \# \Delta))$ 
using  $\sigma(1) \heartsuit$ 
by (simp, metis stronger-theory-left-right-cons)
ultimately show  $?thesis$ 
using stronger-theory-transitive by blast
qed
}
then show  $?case$  by simp
qed
thus  $?thesis$  using assms by blast
qed

lemma (in classical-logic) recover-witness-B-mset-equiv:
assumes  $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } (\text{map } (\text{uncurry } (\sqcup)) \Delta)$ 
shows  $\text{mset } (\text{map } \text{snd } (\mathfrak{Q} \Sigma \Delta))$ 
   $= \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{P} \Sigma \Delta) @ \text{map } \text{snd } \Delta \ominus \text{map } \text{snd } (\mathfrak{P} \Sigma \Delta))$ 
proof -
have  $\forall \Sigma. \text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } (\text{map } (\text{uncurry } (\sqcup)) \Delta)$ 
   $\rightarrow \text{mset } (\text{map } \text{snd } (\mathfrak{Q} \Sigma \Delta)) = \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{P} \Sigma \Delta) @ \text{map}$ 
   $\text{snd } (\mathfrak{P}^C \Sigma \Delta))$ 
proof (induct  $\Delta$ )
  case Nil
  then show  $?case$  by simp
next
case (Cons  $\delta \Delta$ )
  {
  fix  $\Sigma :: ('a \times 'a) \text{ list}$ 
assume  $*$ :  $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } (\text{map } (\text{uncurry } (\sqcup)) (\delta \# \Delta))$ 
have  $\text{mset } (\text{map } \text{snd } (\mathfrak{Q} \Sigma (\delta \# \Delta)))$ 
   $= \text{mset } (\text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{P} \Sigma (\delta \# \Delta)) @ \text{map } \text{snd } (\mathfrak{P}^C \Sigma (\delta \# \Delta)))$ 
proof (cases find  $(\lambda \sigma. \text{snd } \sigma = \text{uncurry } (\sqcup) \delta) \Sigma = \text{None}$ )
  case True
  hence  $\text{uncurry } (\sqcup) \delta \notin \text{set } (\text{map } \text{snd } \Sigma)$ 
proof (induct  $\Sigma$ )
  case Nil
  then show  $?case$  by simp
next
case (Cons  $\sigma \Sigma$ )

```

```

    then show ?case
      by (cases (uncurry (⊔)) δ = snd σ, fastforce+)
  qed
  moreover have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ) +
  {#uncurry (⊔) δ#}
    using * by force
  ultimately have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) 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 (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
      snd σ = uncurry (⊔) δ
      σ ∈ set Σ
  using find-Some-predicate
      find-Some-set-membership
  by fastforce
hence (fst σ, uncurry (⊔) δ) ∈# mset Σ
  by (metis (full-types) prod.collapse set-mset-mset)
then have mset (map snd (remove1 (fst σ, uncurry (⊔) δ) Σ))
  = mset (map snd Σ) - {#uncurry (⊔) δ#}
  by (meson remove1-pairs-list-projections-snd)
moreover have
  mset (map snd Σ)
  ⊆# mset (map (uncurry (⊔)) Δ) + add-mset (uncurry (⊔) δ) (mset [])
    using * by force
  ultimately have mset (map snd (remove1 σ Σ))
    ⊆# mset (map (uncurry (⊔)) Δ)
    by (metis (no-types) σ(2) mset.simps(1) prod.collapse subset-eq-diff-conv)
  with σ(1) Cons show ?thesis by simp
qed
}
then show ?case by blast
qed
thus ?thesis
  using assms recover-witness-A-mset-equiv
  by (simp, metis add-diff-cancel-left')
qed

lemma (in classical-logic) recover-witness-B-right-stronger-theory:
  map (uncurry (→)) Δ ≲ map (uncurry (→)) (Ω Σ Δ)
proof -
  have ∀ Σ. map (uncurry (→)) Δ ≲ map (uncurry (→)) (Ω Σ Δ)
  proof (induct Δ)
    case Nil
    then show ?case by simp
  next
    case (Cons δ Δ)

```



```

{
  fix  $\Sigma$ 
  have map (uncurry ( $\rightarrow$ )) ( $\delta \# \Delta$ )  $\preceq$  map (uncurry ( $\rightarrow$ )) ( $\Omega \Sigma (\delta \# \Delta)$ )
  proof (cases find ( $\lambda \sigma$ . snd  $\sigma = \text{uncurry } (\sqcup) \delta$ )  $\Sigma = \text{None}$ )
    case True
    then show ?thesis
      using Cons
      by (simp add: stronger-theory-left-right-cons trivial-implication)
  next
  case False
  from this obtain  $\sigma$  where  $\sigma$ :
    find ( $\lambda \sigma$ . snd  $\sigma = \text{uncurry } (\sqcup) \delta$ )  $\Sigma = \text{Some } \sigma$ 
  by fastforce
  let ? $\alpha = \text{fst } \delta$ 
  let ? $\beta = \text{snd } \delta$ 
  let ? $\gamma = \text{fst } \sigma$ 
  have uncurry ( $\rightarrow$ ) = ( $\lambda \delta$ . fst  $\delta \rightarrow \text{snd } \delta$ ) by fastforce
  hence uncurry ( $\rightarrow$ )  $\delta = ?\alpha \rightarrow ?\beta$  by auto
  moreover have  $\vdash (?\alpha \rightarrow (?\gamma \sqcup ?\alpha) \rightarrow ?\beta) \rightarrow ?\alpha \rightarrow ?\beta$ 
    unfolding disjunction-def
    using axiom-k axiom-s modus-ponens flip-implication
    by blast
  ultimately show ?thesis
    using Cons  $\sigma$ 
    by (simp add: stronger-theory-left-right-cons)
  qed
}
then show ?case by simp
qed
thus ?thesis by simp
qed

lemma (in classical-logic) recoverWitnesses-mset-equiv:
  assumes mset (map snd  $\Delta$ )  $\subseteq\#$  mset  $\Gamma$ 
  and mset (map snd  $\Sigma$ )  $\subseteq\#$  mset (map (uncurry ( $\sqcup$ ))  $\Delta$ )
  shows mset ( $\Gamma \ominus \text{map snd } \Delta$ )
    = mset ((map (uncurry ( $\rightarrow$ )) ( $\mathfrak{P} \Sigma \Delta$ ) @  $\Gamma \ominus \text{map snd } (\mathfrak{P} \Sigma \Delta)$ )  $\ominus \text{map}$ 
  snd ( $\Omega \Sigma \Delta$ ))
  proof -
  have mset ( $\Gamma \ominus \text{map snd } \Delta$ ) = mset ( $\Gamma \ominus \text{map snd } (\mathfrak{P}^C \Sigma \Delta) \ominus \text{map snd } (\mathfrak{P} \Sigma \Delta)$ )
  using assms(2) recover-witness-A-mset-equiv
  by (simp add: union-commute)
  moreover have  $\forall \Sigma$ . mset (map snd  $\Sigma$ )  $\subseteq\#$  mset (map (uncurry ( $\sqcup$ ))  $\Delta$ )
     $\rightarrow$  mset ( $\Gamma \ominus \text{map snd } (\mathfrak{P}^C \Sigma \Delta)$ )
    = (mset ((map (uncurry ( $\rightarrow$ )) ( $\mathfrak{P} \Sigma \Delta$ ) @  $\Gamma$ )  $\ominus \text{map snd } (\Omega \Sigma$ 
   $\Delta$ )))
  using assms(1)
  proof (induct  $\Delta$ )

```

```

    case Nil
  then show ?case by simp
next
case (Cons δ Δ)
from Cons.prem1 have snd δ ∈ set Γ
  using mset-subset-eqD by fastforce
from Cons.prem2 have ∅: mset (map snd Δ) ⊆# mset Γ
  using subset-mset.dual-order.trans
  by fastforce
{
  fix Σ :: ('a × 'a) list
  assume *: mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) (δ # Δ))
  have mset (Γ ⊖ map snd (℘C Σ (δ # Δ)))
    = mset ((map (uncurry (→)) (℘ Σ (δ # Δ)) @ Γ) ⊖ map snd (Ω Σ (δ #
Δ)))
  proof (cases find (λ σ. snd σ = uncurry (⊔) δ) Σ = None)
  case True
  hence uncurry (⊔) δ ∉ set (map snd Σ)
  proof (induct Σ)
  case Nil
  then show ?case by simp
  next
  case (Cons σ Σ)
  then show ?case
    by (cases (uncurry (⊔)) δ = snd σ, fastforce+)
  qed
  moreover have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ) +
  {#uncurry (⊔) δ#}
    using * by auto
  ultimately have mset (map snd Σ) ⊆# mset (map (uncurry (⊔)) Δ)
  by (metis (full-types) diff-single-trivial in-multiset-in-set subset-eq-diff-conv)
  with Cons.hyps ∅ have mset (Γ ⊖ map snd (℘C Σ Δ))
    = mset ((map (uncurry (→)) (℘ Σ Δ) @ Γ) ⊖ map snd
(Ω Σ Δ))
    by simp
  thus ?thesis using True ⟨snd δ ∈ set Γ⟩ by simp
next
case False
from this obtain σ where σ:
  find (λσ. snd σ = uncurry (⊔) δ) Σ = Some σ
  snd σ = uncurry (⊔) δ
  σ ∈ set Σ
  using find-Some-predicate
  find-Some-set-membership
  by fastforce
with * have mset (map snd (remove1 σ Σ)) ⊆# mset (map (uncurry (⊔))
Δ)
  by (simp, metis (no-types, lifting)
    add-mset-remove-trivial-eq

```

*image-mset-add-mset*  
*in-multiset-in-set*  
*mset-subset-eq-add-mset-cancel*  
**with** *Cons.hyps* **have**  $mset (\Gamma \ominus map\ snd (\mathfrak{P}^C (remove1\ \sigma\ \Sigma)\ \Delta))$   
 $= mset ((map\ (uncurry\ (\rightarrow))) (\mathfrak{P} (remove1\ \sigma\ \Sigma)\ \Delta) @ \Gamma)$   
 $\ominus map\ snd (\mathfrak{Q} (remove1\ \sigma\ \Sigma)\ \Delta))$   
**using**  $\heartsuit$  **by** *blast*  
**then show** *?thesis* **using**  $\sigma$  **by** *simp*  
**qed**  
**}**  
**then show** *?case* **by** *blast*  
**qed**  
**moreover have**  $image\ mset\ snd (mset (\mathfrak{P}^C\ \Sigma\ \Delta)) = mset (map\ snd\ \Delta \ominus map\ snd (\mathfrak{P}\ \Sigma\ \Delta))$   
**using** *assms(2)* *recover-witness-A-mset-equiv*  
**by** (*simp*, *metis (no-types) diff-union-cancelL list-subtract-mset-homomorphism mset-map*)  
**then have**  $mset\ \Gamma - (image\ mset\ snd (mset (\mathfrak{P}^C\ \Sigma\ \Delta)) + image\ mset\ snd (mset (\mathfrak{P}\ \Sigma\ \Delta)))$   
 $= \{\#x \rightarrow y. (x, y) \in\# mset (\mathfrak{P}\ \Sigma\ \Delta)\#\}$   
 $+ (mset\ \Gamma - image\ mset\ snd (mset (\mathfrak{P}\ \Sigma\ \Delta))) - image\ mset\ snd (mset (\mathfrak{Q}\ \Sigma\ \Delta))$   
**using** *calculation*  
*assms(2)*  
*recover-witness-A-mset-equiv*  
*recover-witness-B-mset-equiv*  
**by** *fastforce*  
**ultimately**  
**show** *?thesis*  
**using** *assms recover-witness-A-mset-equiv*  
**by** *simp*  
**qed**

**theorem** (*in classical-logic*) *measure-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** –

$$\mathbf{have} \ \forall \Gamma \Psi. \Gamma \ \$\vdash (\Phi @ \Psi) = (\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge map\ (uncurry\ (\sqcup))\ \Sigma \ \$\vdash \Phi \wedge (map\ (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus (map\ snd\ \Sigma)) \ \$\vdash \Psi)$$

**proof** (*induct*  $\Phi$ )

**case** *Nil*

**{**

**fix**  $\Gamma \Psi$

**have**  $\Gamma \ \$\vdash (\sqcup @ \Psi) = (\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge$

$$map\ (uncurry\ (\sqcup))\ \Sigma \ \$\vdash \sqcup \wedge$$

$$map\ (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \ \$\vdash \Psi)$$

**proof** (*rule iffI*)

```

assume  $\Gamma \Vdash (\Box @ \Psi)$ 
moreover
have  $\Gamma \Vdash (\Box @ \Psi) = (mset (map\ snd\ \Box) \subseteq\# mset\ \Gamma \wedge$ 
       $map (uncurry\ (\sqcup))\ \Box \Vdash \Box \wedge$ 
       $map (uncurry\ (\rightarrow))\ \Box @ \Gamma \ominus (map\ snd\ \Box) \Vdash \Psi)$ 
  by simp
ultimately show  $\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge$ 
       $map (uncurry\ (\sqcup))\ \Sigma \Vdash \Box \wedge$ 
       $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \Vdash \Psi$ 
  by metis
next
assume  $\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge$ 
       $map (uncurry\ (\sqcup))\ \Sigma \Vdash \Box \wedge$ 
       $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \Vdash \Psi$ 
from this obtain  $\Sigma$  where
   $\Sigma: mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma$ 
   $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \Vdash (\Box @ \Psi)$ 
  by fastforce
hence  $(map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma) \preceq \Gamma$ 
  using witness-stronger-theory by auto
with  $\Sigma(2)$  show  $\Gamma \Vdash (\Box @ \Psi)$ 
  using measure-stronger-theory-left-monotonic by blast
qed
}
then show ?case by blast
next
case  $(Cons\ \varphi\ \Phi)$ 
{
  fix  $\Gamma\ \Psi$ 
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  $\Sigma$  where
   $\Sigma: mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma$ 
   $map (uncurry\ (\sqcup))\ \Sigma \Vdash \varphi$ 
   $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus (map\ snd\ \Sigma) \Vdash (\Phi @ \Psi)$ 
  (is  $? \Gamma_0 \Vdash (\Phi @ \Psi)$ )
  by auto
from this(3) obtain  $\Delta$  where
   $\Delta: mset (map\ snd\ \Delta) \subseteq\# mset\ ? \Gamma_0$ 
   $map (uncurry\ (\sqcup))\ \Delta \Vdash \Phi$ 
   $map (uncurry\ (\rightarrow))\ \Delta @ ? \Gamma_0 \ominus (map\ snd\ \Delta) \Vdash \Psi$ 
  using Cons
  by auto
let  $? \Sigma' = \mathfrak{J}\ \Sigma\ \Delta$ 
have  $map (uncurry\ (\sqcup))\ ? \Sigma' \Vdash (\varphi\ \# \Phi)$ 
  using  $\Delta(1)\ \Delta(2)\ \Sigma(2)$  merge-witness-cons-measure-deduction by blast

```

**moreover have**  $mset (map\ snd\ ?\Sigma') \subseteq\# mset\ \Gamma$   
**using**  $\Delta(1)\ \Sigma(1)\ merge-witness-msub-intro$  **by** *blast*  
**moreover have**  $map (uncurry\ (\rightarrow))\ ?\Sigma' @ \Gamma \ominus map\ snd\ ?\Sigma' \$\vdash\ \Psi$   
**using**  $\Delta(1)\ \Delta(3)\ merge-witness-measure-deduction-intro$  **by** *blast*  
**ultimately show**  
 $\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge$   
 $map (uncurry\ (\sqcup))\ \Sigma \$\vdash (\varphi \# \Phi) \wedge$   
 $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \$\vdash\ \Psi$   
**by** *fast*  
**next**  
**assume**  $\exists \Sigma. mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma \wedge$   
 $map (uncurry\ (\sqcup))\ \Sigma \$\vdash (\varphi \# \Phi) \wedge$   
 $map (uncurry\ (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \$\vdash\ \Psi$   
**from this obtain**  $\Delta$  **where**  $\Delta$ :  
 $mset (map\ snd\ \Delta) \subseteq\# mset\ \Gamma$   
 $map (uncurry\ (\sqcup))\ \Delta \$\vdash (\varphi \# \Phi)$   
 $map (uncurry\ (\rightarrow))\ \Delta @ \Gamma \ominus map\ snd\ \Delta \$\vdash\ \Psi$   
**by** *auto*  
**from this obtain**  $\Sigma$  **where**  $\Sigma$ :  
 $mset (map\ snd\ \Sigma) \subseteq\# mset (map (uncurry\ (\sqcup))\ \Delta)$   
 $map (uncurry\ (\sqcup))\ \Sigma :\vdash\ \varphi$   
 $map (uncurry\ (\rightarrow))\ \Sigma @ (map (uncurry\ (\sqcup))\ \Delta) \ominus map\ snd\ \Sigma \$\vdash\ \Phi$   
**by** *auto*  
**let**  $?\Omega = \mathfrak{P}\ \Sigma\ \Delta$   
**let**  $?\Xi = \mathfrak{Q}\ \Sigma\ \Delta$   
**let**  $?T_0 = map (uncurry\ (\rightarrow))\ ?\Omega @ \Gamma \ominus map\ snd\ ?\Omega$   
**let**  $?T_1 = map (uncurry\ (\rightarrow))\ ?\Xi @ ?T_0 \ominus map\ snd\ ?\Xi$   
**have**  $mset (\Gamma \ominus map\ snd\ \Delta) = mset (?T_0 \ominus map\ snd\ ?\Xi)$   
**using**  $\Delta(1)\ \Sigma(1)\ recoverWitnesses-mset-equiv$  **by** *blast*  
**hence**  $(\Gamma \ominus map\ snd\ \Delta) \preceq (?T_0 \ominus map\ snd\ ?\Xi)$   
**by** (*simp add: msub-stronger-theory-intro*)  
**hence**  $?T_1 \$\vdash\ \Psi$   
**using**  $\Delta(3)\ measure-stronger-theory-left-monotonic$   
*stronger-theory-combine*  
*recover-witness-B-right-stronger-theory*  
**by** *blast*  
**moreover**  
**have**  $mset (map\ snd\ ?\Xi) \subseteq\# mset\ ?T_0$   
**using**  $\Sigma(1)\ \Delta(1)\ recover-witness-B-mset-equiv$   
**by** (*simp,*  
*metis list-subtract-monotonic*  
*list-subtract-mset-homomorphism*  
*mset-map*)  
**moreover**  
**have**  $map (uncurry\ (\sqcup))\ ?\Xi \$\vdash\ \Phi$   
**using**  $\Sigma(1)\ recover-witness-B-stronger-theory$   
 $\Sigma(3)\ measure-stronger-theory-left-monotonic$  **by** *blast*  
**ultimately have**  $?T_0 \$\vdash (\Phi @ \Psi)$   
**using** *Cons* **by** *fast*

```

moreover
have  $mset (map\ snd\ ?\Omega) \subseteq\# mset (map\ snd\ \Delta)$ 
  using  $\Sigma(1)$  recover-witness-A-mset-equiv
  by (simp, metis mset-subset-eq-add-left)
hence  $mset (map\ snd\ ?\Omega) \subseteq\# mset\ \Gamma$  using  $\Delta(1)$  by simp
moreover
have  $map (uncurry (\sqcup))\ ?\Omega \vdash \varphi$ 
  using  $\Sigma(2)$ 
    recover-witness-A-left-stronger-theory
    stronger-theory-deduction-monotonic
  by blast
ultimately show  $\Gamma \ \$\vdash ((\varphi \# \Phi) @ \Psi)$ 
  by (simp, blast)
qed
}
then show ?case by metis
qed
thus ?thesis by blast
qed

```

**lemma** (in *classical-logic*) *measure-list-deduction-antitonic*:

```

assumes  $\Gamma \ \$\vdash \Psi$ 
  and  $\Psi \vdash \varphi$ 
shows  $\Gamma \vdash \varphi$ 
using assms
proof (induct  $\Psi$  arbitrary:  $\Gamma \ \varphi$ )
case Nil
then show ?case
  using list-deduction-weaken
  by simp
next
case (Cons  $\psi \ \Psi$ )
hence  $\Psi \vdash \psi \rightarrow \varphi$ 
  using list-deduction-theorem by blast
from  $\langle \Gamma \ \$\vdash (\psi \# \Psi) \rangle$  obtain  $\Sigma$  where  $\Sigma$ :
   $mset (map\ snd\ \Sigma) \subseteq\# mset\ \Gamma$ 
   $map (uncurry (\sqcup))\ \Sigma \vdash \psi$ 
   $map (uncurry (\rightarrow))\ \Sigma @ \Gamma \ominus map\ snd\ \Sigma \ \$\vdash \Psi$ 
by auto
hence  $\Gamma \vdash \psi \rightarrow \varphi$ 
using
  measure-stronger-theory-left-monotonic
  witness-stronger-theory
   $\langle \Psi \vdash \psi \rightarrow \varphi \rangle$ 
  Cons
by blast
moreover
have  $\Gamma \vdash \psi$ 
using  $\Sigma(1)$   $\Sigma(2)$ 

```

```

      stronger-theory-deduction-monotonic
      witness-weaker-theory
    by blast
  ultimately show ?case using list-deduction-modus-ponens by auto
qed

```

Finally, we may establish that ( $\$ \vdash$ ) is transitive.

```

theorem (in classical-logic) measure-transitive:
  assumes  $\Gamma \ \$ \vdash \ \Lambda$ 
    and  $\Lambda \ \$ \vdash \ \Delta$ 
  shows  $\Gamma \ \$ \vdash \ \Delta$ 
  using assms
proof (induct  $\Delta$  arbitrary:  $\Gamma \ \Lambda$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\delta \ \Delta$ )
  from this obtain  $\Sigma$  where  $\Sigma$ :
    mset (map snd  $\Sigma$ )  $\subseteq\#$  mset  $\Lambda$ 
    map (uncurry ( $\sqcup$ ))  $\Sigma \ \vdash \ \delta$ 
    map (uncurry ( $\rightarrow$ ))  $\Sigma \ @ \ \Lambda \ \ominus \ \text{map snd } \Sigma \ \$ \vdash \ \Delta$ 
  by auto
  hence  $\Gamma \ \$ \vdash \ (\text{map } (\text{uncurry } (\sqcup)) \ \Sigma \ @ \ \text{map } (\text{uncurry } (\rightarrow)) \ \Sigma \ @ \ \Lambda \ \ominus \ (\text{map snd } \Sigma))$ 
  using Cons measure-witness-right-split
  by simp
  from this obtain  $\Psi$  where  $\Psi$ :
    mset (map snd  $\Psi$ )  $\subseteq\#$  mset  $\Gamma$ 
    map (uncurry ( $\sqcup$ ))  $\Psi \ \$ \vdash \ \text{map } (\text{uncurry } (\sqcup)) \ \Sigma$ 
    map (uncurry ( $\rightarrow$ ))  $\Psi \ @ \ \Gamma \ \ominus \ \text{map snd } \Psi \ \$ \vdash \ (\text{map } (\text{uncurry } (\rightarrow)) \ \Sigma \ @ \ \Lambda \ \ominus \ \text{map snd } \Sigma)$ 
  using measure-deduction-generalized-witness
  by fastforce
  have map (uncurry ( $\rightarrow$ ))  $\Psi \ @ \ \Gamma \ \ominus \ \text{map snd } \Psi \ \$ \vdash \ \Delta$ 
  using  $\Sigma(3) \ \Psi(3) \ \text{Cons}$ 
  by auto
  moreover
  have map (uncurry ( $\sqcup$ ))  $\Psi \ \vdash \ \delta$ 
  using  $\Psi(2) \ \Sigma(2) \ \text{measure-list-deduction-antitonic}$ 
  by blast
  ultimately show ?case
  using  $\Psi(1)$ 
  by fastforce
qed

```

## 2.6 Measure Deduction Cancellation Rules

In this chapter we go over how to cancel formulae occurring in measure deduction judgements.

The first observation is that tautologies can always be canceled on either side of the turnstile.

**lemma** (in *classical-logic*) *measure-tautology-right-cancel*:

```

assumes  $\vdash \varphi$ 
shows  $\Gamma \ \$\vdash (\varphi \# \Phi) = \Gamma \ \$\vdash \Phi$ 
proof (rule iffI)
assume  $\Gamma \ \$\vdash (\varphi \# \Phi)$ 
from this obtain  $\Sigma$  where  $\Sigma$ :
   $mset (map \ snd \ \Sigma) \subseteq\# \ mset \ \Gamma$ 
   $map (uncurry (\sqcup)) \ \Sigma \ :\vdash \ \varphi$ 
   $map (uncurry (\rightarrow)) \ \Sigma \ @ \ \Gamma \ \ominus \ map \ snd \ \Sigma \ \$\vdash \ \Phi$ 
by auto
thus  $\Gamma \ \$\vdash \Phi$ 
using measure-stronger-theory-left-monotonic
       witness-stronger-theory
by blast
next
assume  $\Gamma \ \$\vdash \Phi$ 
hence  $map (uncurry (\rightarrow)) \ [] \ @ \ \Gamma \ \ominus \ map \ snd \ [] \ \$\vdash \Phi$ 
        $mset (map \ snd \ []) \subseteq\# \ mset \ \Gamma$ 
        $map (uncurry (\sqcup)) \ [] \ :\vdash \ \varphi$ 
using assms
by simp+
thus  $\Gamma \ \$\vdash (\varphi \# \Phi)$ 
using measure-deduction.simps(2)
by blast
qed

```

**lemma** (in *classical-logic*) *measure-tautology-left-cancel* [*simp*]:

```

assumes  $\vdash \gamma$ 
shows  $(\gamma \# \Gamma) \ \$\vdash \Phi = \Gamma \ \$\vdash \Phi$ 
proof (rule iffI)
assume  $(\gamma \# \Gamma) \ \$\vdash \Phi$ 
moreover have  $\Gamma \ \$\vdash \Gamma$ 
  by (simp add: stronger-theory-to-measure-deduction)
hence  $\Gamma \ \$\vdash (\gamma \# \Gamma)$ 
using assms measure-tautology-right-cancel
by simp
ultimately show  $\Gamma \ \$\vdash \Phi$ 
using measure-transitive by blast
next
assume  $\Gamma \ \$\vdash \Phi$ 
moreover have  $mset \ \Gamma \subseteq\# \ mset (\gamma \# \Gamma)$ 
by simp

```



```

hence ( $\gamma \# \Gamma$ )  $\$ \vdash \Gamma$ 
  using msub-stronger-theory-intro
    stronger-theory-to-measure-deduction
  by blast
ultimately show ( $\gamma \# \Gamma$ )  $\$ \vdash \Phi$ 
  using measure-transitive by blast
qed

```

**lemma** (in *classical-logic*) *measure-deduction-one-collapse*:

```

 $\Gamma \ \$ \vdash [\varphi] = \Gamma \ : \vdash \varphi$ 
proof (rule iffI)
  assume  $\Gamma \ \$ \vdash [\varphi]$ 
  from this obtain  $\Sigma$  where
     $\Sigma$ :  $mset (map \text{snd } \Sigma) \subseteq \# mset \Gamma$ 
     $map (uncurry (\sqcup)) \Sigma \ : \vdash \varphi$ 
  by auto
  hence  $map (uncurry (\sqcup)) \Sigma \preceq \Gamma$ 
  using witness-weaker-theory by blast
  thus  $\Gamma \ : \vdash \varphi$  using  $\Sigma(2)$ 
  using stronger-theory-deduction-monotonic by blast
next
  assume  $\Gamma \ : \vdash \varphi$ 
  let  $? \Sigma = map (\lambda \gamma. (\perp, \gamma)) \Gamma$ 
  have  $\Gamma \preceq map (uncurry (\sqcup)) ? \Sigma$ 
  proof (induct  $\Gamma$ )
    case Nil
    then show ?case by simp
  next
  case (Cons  $\gamma \Gamma$ )
  have  $\vdash (\perp \sqcup \gamma) \rightarrow \gamma$ 
  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  $map (uncurry (\sqcup)) ? \Sigma \ : \vdash \varphi$ 
  using  $\langle \Gamma \ : \vdash \varphi \rangle$  stronger-theory-deduction-monotonic by blast
  moreover have  $mset (map \text{snd } ? \Sigma) \subseteq \# mset \Gamma$  by (induct  $\Gamma$ , simp+)
  ultimately show  $\Gamma \ \$ \vdash [\varphi]$ 
  using measure-deduction.simps(1)
    measure-deduction.simps(2)
  by blast
qed

```

*Split cases*, which are occurrences of  $\psi \sqcup \varphi \# \psi \rightarrow \varphi \# \dots$ , also cancel and simplify to just  $\varphi \# \dots$ . We previously established  $\Gamma \ \$ \vdash \psi \sqcup \varphi \# \psi \rightarrow \varphi \# \Phi = \Gamma \ \$ \vdash \varphi \# \Phi$  as part of the proof of transitivity.

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

**lemma** (in *classical-logic*) *measure-witness-left-split [simp]*:  
**assumes**  $mset (map \text{snd } \Sigma) \subseteq\# mset \Gamma$   
**shows**  $(map (uncurry (\sqcup)) \Sigma @ map (uncurry (\rightarrow)) \Sigma @ \Gamma \ominus (map \text{snd } \Sigma)) \ \$\vdash \Phi = \Gamma \ \$\vdash \Phi$   
**using** *assms*  
**proof** (*induct*  $\Sigma$  *arbitrary: \Gamma*)  
**case** *Nil*  
**then show** *?case* **by** *simp*  
**next**  
**case** (*Cons*  $\sigma \Sigma$ )  
**let**  $?x = \text{fst } \sigma$   
**let**  $?y = \text{snd } \sigma$   
**let**  $?T_0 = map (uncurry (\sqcup)) \Sigma @ map (uncurry (\rightarrow)) \Sigma @ \Gamma \ominus map \text{snd } (\sigma \# \Sigma)$   
**let**  $?T' = map (uncurry (\sqcup)) (\sigma \# \Sigma) @ map (uncurry (\rightarrow)) (\sigma \# \Sigma) @ \Gamma \ominus map \text{snd } (\sigma \# \Sigma)$   
**assume**  $mset (map \text{snd } (\sigma \# \Sigma)) \subseteq\# mset \Gamma$   
**hence** *A*:  $add\text{-}mset (\text{snd } \sigma) (image\text{-}mset \text{snd } (mset \Sigma)) \subseteq\# mset \Gamma$  **by** *simp*  
**hence** *B*:  $image\text{-}mset \text{snd } (mset \Sigma) + (mset \Gamma - image\text{-}mset \text{snd } (mset \Sigma))$   
 $= add\text{-}mset (\text{snd } \sigma) (image\text{-}mset \text{snd } (mset \Sigma))$   
 $+ (mset \Gamma - add\text{-}mset (\text{snd } \sigma) (image\text{-}mset \text{snd } (mset \Sigma)))$   
**by** (*metis* (*no-types*) *mset-subset-eq-insertD* *subset-mset.add-diff-inverse subset-mset-def*)  
**have**  $\{\#x \rightarrow y. (x, y) \in\# mset \Sigma\# \}$   
 $+ mset \Gamma - add\text{-}mset (\text{snd } \sigma) (image\text{-}mset \text{snd } (mset \Sigma))$

$= \{\#x \rightarrow y. (x, y) \in \# \text{mset } \Sigma \#\}$   
 $+ (\text{mset } \Gamma - \text{add-mset } (\text{snd } \sigma) (\text{image-mset } \text{snd } (\text{mset } \Sigma)))$   
**using** *A subset-mset.diff-add-assoc* **by** *blast*  
**hence**  $\{\#x \rightarrow y. (x, y) \in \# \text{mset } \Sigma \#\} + (\text{mset } \Gamma - \text{image-mset } \text{snd } (\text{mset } \Sigma))$   
 $= \text{add-mset } (\text{snd } \sigma) (\{\#x \rightarrow y. (x, y) \in \# \text{mset } \Sigma \#\}$   
 $+ \text{mset } \Gamma - \text{add-mset } (\text{snd } \sigma) (\text{image-mset } \text{snd } (\text{mset } \Sigma)))$   
**using** *B* **by** *auto*  
**hence** *C*:  
 $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } \Gamma$   
 $\text{mset } (\text{map } (\text{uncurry } (\sqcup)) \Sigma @ \text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus \text{map } \text{snd } \Sigma)$   
 $= \text{mset } (? \gamma \# ? \Gamma_0)$   
**using**  $\langle \text{mset } (\text{map } \text{snd } (\sigma \# \Sigma)) \subseteq \# \text{mset } \Gamma \rangle$   
*subset-mset.dual-order.trans*  
**by** *(fastforce+)*  
**hence**  $\Gamma \text{ \$} \vdash \Phi = (? \chi \sqcup ? \gamma \# ? \chi \rightarrow ? \gamma \# ? \Gamma_0) \text{ \$} \vdash \Phi$   
**proof** –  
**have**  $\forall \Gamma \Delta. \neg \text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } \Gamma$   
 $\vee \neg \Gamma \text{ \$} \vdash \Phi$   
 $\vee \neg \text{mset } (\text{map } (\text{uncurry } (\sqcup)) \Sigma$   
 $@ \text{map } (\text{uncurry } (\rightarrow)) \Sigma$   
 $@ \Gamma \ominus \text{map } \text{snd } \Sigma)$   
 $\subseteq \# \text{mset } \Delta$   
 $\vee \Delta \text{ \$} \vdash \Phi$   
**using** *Cons.hyps measure-msub-left-monotonic* **by** *blast*  
**moreover**  
{  
**assume**  $\neg \Gamma \text{ \$} \vdash \Phi$   
**then have**  $\exists \Delta. \text{mset } (\text{snd } \sigma \# \text{map } (\text{uncurry } (\sqcup)) \Sigma$   
 $@ \text{map } (\text{uncurry } (\rightarrow)) \Sigma$   
 $@ \Gamma \ominus \text{map } \text{snd } (\sigma \# \Sigma))$   
 $\subseteq \# \text{mset } \Delta$   
 $\wedge \neg \Gamma \text{ \$} \vdash \Phi$   
 $\wedge \neg \Delta \text{ \$} \vdash \Phi$   
**by** *(metis (no-types) Cons.hyps C subset-mset.dual-order.refl)*  
**then have** *?thesis*  
**using** *measure-formula-left-split measure-msub-left-monotonic* **by** *blast*  
}  
**ultimately show** *?thesis*  
**by** *(metis (full-types) C measure-formula-left-split subset-mset.dual-order.refl)*  
**qed**  
**moreover**  
**have**  $(\text{uncurry } (\sqcup)) = (\lambda \psi. \text{fst } \psi \sqcup \text{snd } \psi)$   
 $(\text{uncurry } (\rightarrow)) = (\lambda \psi. \text{fst } \psi \rightarrow \text{snd } \psi)$   
**by** *fastforce+*  
**hence**  $\text{mset } ? \Gamma' = \text{mset } (? \chi \sqcup ? \gamma \# ? \chi \rightarrow ? \gamma \# ? \Gamma_0)$   
**by** *fastforce*  
**hence**  $(? \chi \sqcup ? \gamma \# ? \chi \rightarrow ? \gamma \# ? \Gamma_0) \text{ \$} \vdash \Phi = ? \Gamma' \text{ \$} \vdash \Phi$   
**by** *(metis*  
*(mono-tags, lifting)*)

```

      measure-msub-left-monotonic
      subset-mset.dual-order.refl)
ultimately have  $\Gamma \text{\$}\vdash \Phi = ?\Gamma' \text{\$}\vdash \Phi$ 
  by fastforce
then show ?case by blast
qed

```

We now have enough to establish the cancellation rule for ( $\text{\$}\vdash$ ).

**lemma** (in *classical-logic*) *measure-cancel*:  $(\Delta @ \Gamma) \text{\$}\vdash (\Delta @ \Phi) = \Gamma \text{\$}\vdash \Phi$

**proof** –

```

{
  fix  $\Delta \Gamma \Phi$ 
  assume  $\Gamma \text{\$}\vdash \Phi$ 
  hence  $(\Delta @ \Gamma) \text{\$}\vdash (\Delta @ \Phi)$ 
  proof (induct  $\Delta$ )
    case Nil
      then show ?case by simp
    next
      case (Cons  $\delta \Delta$ )
        let  $?\Sigma = [(\delta, \delta)]$ 
        have map (uncurry ( $\sqcup$ ))  $?\Sigma \text{\$}\vdash \delta$ 
          unfolding disjunction-def list-deduction-def
          by (simp add: Peirces-law)
        moreover have mset (map snd  $?\Sigma$ )  $\subseteq\#$  mset ( $\delta \# \Delta$ ) by simp
        moreover have map (uncurry ( $\rightarrow$ ))  $?\Sigma @ ((\delta \# \Delta) @ \Gamma) \ominus$  map snd  $?\Sigma \text{\$}\vdash$ 
          ( $\Delta @ \Phi$ )
          using Cons
          by (simp add: trivial-implication)
        moreover have map snd  $[(\delta, \delta)] = [\delta]$  by force
        ultimately show ?case
          by (metis (no-types) measure-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  $(\Delta @ \Gamma) \text{\$}\vdash (\Delta @ \Phi)$ 
  hence  $\Gamma \text{\$}\vdash \Phi$ 
  proof (induct  $\Delta$ )
    case Nil
      then show ?case by simp
    next
      case (Cons  $\delta \Delta$ )
        have mset  $((\delta \# \Delta) @ \Phi) =$  mset  $((\Delta @ \Phi) @ [\delta])$  by simp
        with Cons.premis have  $((\delta \# \Delta) @ \Gamma) \text{\$}\vdash ((\Delta @ \Phi) @ [\delta])$ 

```

```

    by (metis measure-msub-weaken
        subset-mset.dual-order.refl)
from this obtain  $\Sigma$  where  $\Sigma$ :
  mset (map snd  $\Sigma$ )  $\subseteq\#$  mset (( $\delta \# \Delta$ ) @  $\Gamma$ )
  map (uncurry ( $\sqcup$ ))  $\Sigma$   $\$ \vdash$  ( $\Delta$  @  $\Phi$ )
  map (uncurry ( $\rightarrow$ ))  $\Sigma$  @ (( $\delta \# \Delta$ ) @  $\Gamma$ )  $\ominus$  map snd  $\Sigma$   $\$ \vdash$  [ $\delta$ ]
  by (metis append-assoc measure-deduction-generalized-witness)
show ?case
proof (cases find ( $\lambda \sigma. \text{snd } \sigma = \delta$ )  $\Sigma = \text{None}$ )
  case True
  hence  $\delta \notin \text{set } (\text{map snd } \Sigma)$ 
  proof (induct  $\Sigma$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\sigma$   $\Sigma$ )
    then show ?case by (cases snd  $\sigma = \delta, \text{simp+}$ )
  qed
with  $\Sigma(1)$  have mset (map snd  $\Sigma$ )  $\subseteq\#$  mset ( $\Delta$  @  $\Gamma$ )
  by (simp, metis add-mset-add-single
      diff-single-trivial
      mset-map
      set-mset-mset
      subset-eq-diff-conv)
thus ?thesis
  using measure-stronger-theory-left-monotonic
      witness-weaker-theory
      Cons.hyps  $\Sigma(2)$ 
  by blast
next
case False
from this obtain  $\sigma \chi$  where
   $\sigma = (\chi, \delta)$ 
   $\sigma \in \text{set } \Sigma$ 
  using find-Some-predicate
      find-Some-set-membership
  by fastforce
let  $?\Sigma' = \text{remove1 } \sigma \Sigma$ 
let  $?\Sigma_A = \text{map } (\text{uncurry } (\sqcup)) \ ?\Sigma'$ 
let  $?\Sigma_B = \text{map } (\text{uncurry } (\rightarrow)) \ ?\Sigma'$ 
have mset  $\Sigma = \text{mset } (?\Sigma' @ [(\chi, \delta)])$ 
  mset  $\Sigma = \text{mset } ((\chi, \delta) \# ?\Sigma')$ 
  using  $\sigma$  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 measure-msub-left-monotonic subset-mset.dual-order.refl, simp*)  
**moreover**  
**have**  $\vdash ((\chi \rightarrow \delta) \rightarrow \delta) \rightarrow (\chi \sqcup \delta)$   
**unfolding** *disjunction-def*  
**using** *modus-ponens*  
*pseudo-scotus*  
*flip-hypothetical-syllogism*  
**by** *blast*  
**ultimately have**  $(? \Sigma_A @ ? \Sigma_B @ (\Delta @ \Gamma) \ominus map\ snd\ ? \Sigma') \$\vdash (\Delta @ \Phi)$   
**using** *measure-deduction-one-collapse*  
*list-deduction-theorem*  
*list-deduction-modus-ponens*  
*list-deduction-weaken*  
*forward-direction*  
*measure-transitive*  
**by** *meson*  
**moreover**  
**have**  $\delta = snd\ \sigma$   
 $snd\ \sigma \in set (map\ snd\ \Sigma)$   
**by** (*simp add: \sigma(1), simp add: \sigma(2)*)  
**with**  $\Sigma(1)$  **have**  $mset (map\ snd (remove1\ \sigma\ \Sigma)) \subseteq\# mset (remove1\ \delta ((\delta$   
 $\# \Delta) @ \Gamma))$   
**by** (*metis insert-DiffM*  
*insert-subset-eq-iff*  
*mset-remove1*  
 $\sigma(1)\ \sigma(2)$   
*remove1-pairs-list-projections-snd*  
*set-mset-mset*)  
**hence**  $mset (map\ snd (remove1\ \sigma\ \Sigma)) \subseteq\# mset (\Delta @ \Gamma)$  **by** *simp*  
**ultimately show** *?thesis*  
**using** *measure-witness-left-split Cons.hyps*  
**by** *blast*  
**qed**  
**qed**  
**}**  
**with** *forward-direction* **show** *?thesis* **by** *auto*  
**qed**

**lemma** (in *classical-logic*) *measure-biconditional-cancel*:

**assumes**  $\vdash \gamma \leftrightarrow \varphi$

**shows**  $(\gamma \# \Gamma) \$\vdash (\varphi \# \Phi) = \Gamma \$\vdash \Phi$

**proof** –  
**from** *assms* **have**  $(\gamma \# \Phi) \preceq (\varphi \# \Phi) (\varphi \# \Phi) \preceq (\gamma \# \Phi)$   
**unfolding** *biconditional-def*  
**by** (*simp add: stronger-theory-left-right-cons*)  
**hence**  $(\gamma \# \Phi) \$\vdash (\varphi \# \Phi)$   
 $(\varphi \# \Phi) \$\vdash (\gamma \# \Phi)$   
**using** *stronger-theory-to-measure-deduction* **by** *blast+*  
**moreover**  
**have**  $\Gamma \$\vdash \Phi = (\gamma \# \Gamma) \$\vdash (\gamma \# \Phi)$   
**by** (*metis append-Cons append-Nil measure-cancel*)  
**ultimately**  
**have**  $\Gamma \$\vdash \Phi \implies \gamma \# \Gamma \$\vdash (\varphi \# \Phi)$   
 $\gamma \# \Gamma \$\vdash (\varphi \# \Phi) \implies \Gamma \$\vdash \Phi$   
**using** *measure-transitive* **by** *blast+*  
**thus** *?thesis* **by** *blast*  
**qed**

## 2.7 Measure Deduction Substitution Rules

Just like conventional deduction, if two formulae are equivalent then they may be substituted for one another.

**lemma** (in *classical-logic*) *right-measure-sub*:

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

**lemma** (in *classical-logic*) *left-measure-sub*:

**assumes**  $\vdash \gamma \leftrightarrow \chi$   
**shows**  $(\gamma \# \Gamma) \$\vdash \Phi = (\chi \# \Gamma) \$\vdash \Phi$   
**proof** –  
**have**  $(\gamma \# \Gamma) \$\vdash \Phi = (\chi \# \gamma \# \Gamma) \$\vdash (\chi \# \Phi)$   
**using** *measure-cancel* [**where**  $\Delta=[\chi]$  **and**  $\Gamma=(\gamma \# \Gamma)$  **and**  $\Phi=\Phi$ ] **by** *simp*  
**also have**  $\dots = (\gamma \# \chi \# \Gamma) \$\vdash (\chi \# \Phi)$   
**using**  
*measure-cons-cons-right-permute*  
*stronger-theory-to-measure-deduction*  
*measure-transitive*  
*stronger-theory-reflexive*  
**by** *blast*  
**also have**  $\dots = (\chi \# \Gamma) \$\vdash \Phi$

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

## 2.8 Measure Deduction Sum Rules

We next establish analogues of the rule in probability that  $\mathcal{P} \alpha + \mathcal{P} \beta = \mathcal{P} (\alpha \sqcup \beta) + \mathcal{P} (\alpha \sqcap \beta)$ . This equivalence holds for both sides of the ( $\$$ ) turnstile.

**lemma (in classical-logic) right-measure-sum-rule:**

$$\Gamma \$\vdash (\alpha \# \beta \# \Phi) = \Gamma \$\vdash (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Phi)$$

**proof** –

$$\text{have } A: \text{mset } (\alpha \sqcup \beta \# \beta \rightarrow \alpha \# \beta \# \Phi) = \text{mset } (\beta \rightarrow \alpha \# \beta \# \alpha \sqcup \beta \# \Phi)$$

**by simp**

$$\text{have } B: \vdash (\beta \rightarrow \alpha) \leftrightarrow (\beta \rightarrow (\alpha \sqcap \beta))$$

**proof** –

$$\text{let } ?\varphi = (\langle \beta \rangle \rightarrow \langle \alpha \rangle) \leftrightarrow (\langle \beta \rangle \rightarrow (\langle \alpha \rangle \sqcap \langle \beta \rangle))$$

$$\text{have } \forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi \text{ by fastforce}$$

$$\text{hence } \vdash (\langle ?\varphi \rangle) \text{ using propositional-semantic by blast}$$

$$\text{thus } ?thesis \text{ by simp}$$

**qed**

$$\text{have } C: \vdash \beta \leftrightarrow (\beta \sqcup (\alpha \sqcap \beta))$$

**proof** –

$$\text{let } ?\varphi = \langle \beta \rangle \leftrightarrow (\langle \beta \rangle \sqcup (\langle \alpha \rangle \sqcap \langle \beta \rangle))$$

$$\text{have } \forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi \text{ by fastforce}$$

$$\text{hence } \vdash (\langle ?\varphi \rangle) \text{ using propositional-semantic by blast}$$

$$\text{thus } ?thesis \text{ by simp}$$

**qed**

$$\text{have } \Gamma \$\vdash (\alpha \# \beta \# \Phi) = \Gamma \$\vdash (\beta \sqcup \alpha \# \beta \rightarrow \alpha \# \beta \# \Phi)$$

$$\text{using measure-formula-right-split by blast}$$

$$\text{also have } \dots = \Gamma \$\vdash (\alpha \sqcup \beta \# \beta \rightarrow \alpha \# \beta \# \Phi)$$

$$\text{using disjunction-commutativity right-measure-sub by blast}$$

$$\text{also have } \dots = \Gamma \$\vdash (\beta \rightarrow \alpha \# \beta \# \alpha \sqcup \beta \# \Phi)$$

$$\text{by (metis A measure-msub-weaken subset-mset.dual-order.refl)}$$

$$\text{also have } \dots = \Gamma \$\vdash (\beta \rightarrow (\alpha \sqcap \beta) \# \beta \# \alpha \sqcup \beta \# \Phi)$$

$$\text{using B right-measure-sub by blast}$$

$$\text{also have } \dots = \Gamma \$\vdash (\beta \# \beta \rightarrow (\alpha \sqcap \beta) \# \alpha \sqcup \beta \# \Phi)$$

$$\text{using measure-cons-cons-right-permute by blast}$$

$$\text{also have } \dots = \Gamma \$\vdash (\beta \sqcup (\alpha \sqcap \beta) \# \beta \rightarrow (\alpha \sqcap \beta) \# \alpha \sqcup \beta \# \Phi)$$

$$\text{using C right-measure-sub by blast}$$

$$\text{also have } \dots = \Gamma \$\vdash (\alpha \sqcap \beta \# \alpha \sqcup \beta \# \Phi)$$

$$\text{using measure-formula-right-split by blast}$$

**finally show** *?thesis*

$$\text{using measure-cons-cons-right-permute by blast}$$

**qed**

**lemma (in classical-logic) left-measure-sum-rule:**

$$(\alpha \# \beta \# \Gamma) \$\vdash \Phi = (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma) \$\vdash \Phi$$



**proof** –

**have**  $\star$ :  $mset (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \alpha \# \beta \# \Gamma) = mset (\alpha \# \beta \# \alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma)$  **by** *simp*

**have**  $(\alpha \# \beta \# \Gamma) \$\vdash \Phi = (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \alpha \# \beta \# \Gamma) \$\vdash (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Phi)$

**using** *measure-cancel* [**where**  $\Delta=[\alpha \sqcup \beta, \alpha \sqcap \beta]$  **and**  $\Gamma=(\alpha \# \beta \# \Gamma)$  **and**  $\Phi=\Phi]$  **by** *simp*

**also have**  $\dots = (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \alpha \# \beta \# \Gamma) \$\vdash (\alpha \# \beta \# \Phi)$

**using** *right-measure-sum-rule* **by** *blast*

**also have**  $\dots = (\alpha \# \beta \# \alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma) \$\vdash (\alpha \# \beta \# \Phi)$

**by** (*metis*  $\star$  *measure-msub-left-monotonic subset-mset.dual-order.refl*)

**also have**  $\dots = (\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma) \$\vdash \Phi$

**using** *measure-cancel* [**where**  $\Delta=[\alpha, \beta]$  **and**  $\Gamma=(\alpha \sqcup \beta \# \alpha \sqcap \beta \# \Gamma)$  **and**  $\Phi=\Phi]$  **by** *simp*

**finally show** *?thesis* .

**qed**

## 2.9 Measure Deduction Exchange Rule

As we will see, a key result is that we can move formulae from the right hand side of the ( $\$\vdash$ ) turnstile to the left.

We observe a novel logical principle, which we call *exchange*. This principle follows immediately from the split rules and cancellation rules.

**lemma** (in *classical-logic*) *measure-exchange*:

$(\gamma \# \Gamma) \$\vdash (\varphi \# \Phi) = (\varphi \rightarrow \gamma \# \Gamma) \$\vdash (\gamma \rightarrow \varphi \# \Phi)$

**proof** –

**have**  $(\gamma \# \Gamma) \$\vdash (\varphi \# \Phi) = (\varphi \sqcup \gamma \# \varphi \rightarrow \gamma \# \Gamma) \$\vdash (\gamma \sqcup \varphi \# \gamma \rightarrow \varphi \# \Phi)$

**using** *measure-formula-left-split*

*measure-formula-right-split*

**by** *blast+*

**thus** *?thesis*

**using** *measure-biconditional-cancel*

*disjunction-commutativity*

**by** *blast*

**qed**

The exchange rule allows us to prove an analogue of the rule in classical logic that  $\Gamma \vdash \varphi = (\sim \varphi \# \Gamma) \vdash \perp$  for measure deduction.

**theorem** (in *classical-logic*) *measure-negation-swap*:

$\Gamma \$\vdash (\varphi \# \Phi) = (\sim \varphi \# \Gamma) \$\vdash (\perp \# \Phi)$

**proof** –

**have**  $\Gamma \$\vdash (\varphi \# \Phi) = (\perp \# \Gamma) \$\vdash (\perp \# \varphi \# \Phi)$

**by** (*metis* *append-Cons append-Nil measure-cancel*)

**also have**  $\dots = (\perp \# \Gamma) \$\vdash (\varphi \# \perp \# \Phi)$

**using** *measure-cons-cons-right-permute* **by** *blast*

**also have**  $\dots = (\sim \varphi \# \Gamma) \$\vdash (\perp \rightarrow \varphi \# \perp \# \Phi)$

**unfolding** *negation-def*

```

    using measure-exchange
    by blast
  also have ... = ( $\sim \varphi \# \Gamma$ )  $\$ \vdash (\perp \# \Phi)$ 
    using ex-falso-quodlibet
           measure-tautology-right-cancel
    by blast
  finally show ?thesis .
qed

```

## 2.10 Definition of Counting Deduction

The theorem  $\Gamma \ \$ \vdash \varphi \# \Phi = \sim \varphi \# \Gamma \ \$ \vdash \perp \# \Phi$  gives rise to another kind of judgement: *how many times can a list of premises  $\Gamma$  prove a formula  $\varphi$ ?* We call this kind of judgment *counting deduction*. As with measure deduction, bits of  $\Gamma$  get "used up" with each dispatched conclusion.

```

primrec (in classical-logic)
  counting-deduction :: 'a list  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  bool (-  $\# \vdash$  - - [60,100,59] 60)
  where
     $\Gamma \ \# \vdash 0 \ \varphi = \text{True}$ 
  |  $\Gamma \ \# \vdash (\text{Suc } n) \ \varphi = (\exists \Psi. \text{mset } (\text{map } \text{snd } \Psi) \subseteq \# \text{mset } \Gamma \wedge$ 
                                $\text{map } (\text{uncurry } (\sqcup)) \Psi \ \$ \vdash \varphi \wedge$ 
                                $\text{map } (\text{uncurry } (\rightarrow)) \Psi \ @ \ \Gamma \ominus (\text{map } \text{snd } \Psi) \ \# \vdash n \ \varphi)$ 

```

## 2.11 Converting Back and Forth from Counting Deduction to Measure Deduction

We next show how to convert back and forth from counting deduction to measure deduction.

First, we show that trivially counting deduction is a special case of measure deduction.

**lemma** (in classical-logic) *counting-deduction-to-measure-deduction:*

```

 $\Gamma \ \# \vdash n \ \varphi = \Gamma \ \$ \vdash (\text{replicate } n \ \varphi)$ 
  by (induct n arbitrary:  $\Gamma$ , simp+)

```

We next prove a few helpful lemmas regarding counting deduction.

**lemma** (in classical-logic) *counting-deduction-tautology-weaken:*

```

  assumes  $\vdash \varphi$ 
  shows  $\Gamma \ \# \vdash n \ \varphi$ 
proof (induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  hence  $\Gamma \ \$ \vdash (\varphi \# \text{replicate } n \ \varphi)$ 

```

**using** *assms*  
     *counting-deduction-to-measure-deduction*  
     *measure-tautology-right-cancel*  
**by** *blast*  
**hence**  $\Gamma \text{\$}\vdash \text{replicate } (\text{Suc } n) \varphi$   
**by** *simp*  
**then show** *?case*  
     **using** *counting-deduction-to-measure-deduction*  
     **by** *blast*  
**qed**

**lemma** (in *classical-logic*) *counting-deduction-weaken*:  
**assumes**  $n \leq m$   
     **and**  $\Gamma \# \vdash m \varphi$   
**shows**  $\Gamma \# \vdash n \varphi$   
**proof** –  
     **have**  $\Gamma \text{\$}\vdash \text{replicate } m \varphi$   
         **using** *assms(2)* *counting-deduction-to-measure-deduction*  
         **by** *blast*  
**hence**  $\Gamma \text{\$}\vdash \text{replicate } n \varphi$   
**by** (*metis append-Nil2*  
     *assms(1)*  
     *le-iff-add*  
     *measure-deduction.simps(1)*  
     *measure-deduction-generalized-witness*  
     *replicate-add*)  
**thus** *?thesis*  
     **using** *counting-deduction-to-measure-deduction*  
     **by** *blast*  
**qed**

**lemma** (in *classical-logic*) *counting-deduction-implication*:  
**assumes**  $\vdash \varphi \rightarrow \psi$   
     **and**  $\Gamma \# \vdash n \varphi$   
**shows**  $\Gamma \# \vdash n \psi$   
**proof** –  
     **have**  $\text{replicate } n \psi \preceq \text{replicate } n \varphi$   
         **using** *stronger-theory-left-right-cons* *assms(1)*  
         **by** (*induct n, auto*)  
**thus** *?thesis*  
     **using** *assms(2)*  
         *measure-stronger-theory-right-antitonic*  
         *counting-deduction-to-measure-deduction*  
     **by** *blast*  
**qed**

Finally, we use  $\Gamma \text{\$}\vdash \varphi \# \Phi = \sim \varphi \# \Gamma \text{\$}\vdash \perp \# \Phi$  to prove that measure deduction reduces to counting deduction.

**theorem** (in *classical-logic*) *measure-deduction-to-counting-deduction*:

```

Γ $⊢ Φ = (∼ Φ @ Γ) #⊢ (length Φ) ⊥
proof –
have ∀ Ψ. Γ $⊢ (Φ @ Ψ) = (∼ Φ @ Γ) $⊢ (replicate (length Φ) ⊥ @ Ψ)
proof (induct Φ arbitrary: Γ)
  case Nil
  then show ?case by simp
next
case (Cons φ Φ)
  {
  fix Ψ
  have Γ $⊢ ((φ # Φ) @ Ψ) = (∼ φ # Γ) $⊢ (⊥ # Φ @ Ψ)
    using measure-negation-swap by auto
  moreover have mset (Φ @ (⊥ # Ψ)) = mset (⊥ # Φ @ Ψ)
    by simp
  ultimately have Γ $⊢ ((φ # Φ) @ Ψ) = (∼ φ # Γ) $⊢ (Φ @ (⊥ # Ψ))
    by (metis measure-msub-weaken subset-mset.order-refl)
  hence
  Γ $⊢ ((φ # Φ) @ Ψ)
    = (∼ Φ @ (∼ φ # Γ)) $⊢ (replicate (length Φ) ⊥ @ (⊥ # Ψ))
    using Cons
    by blast
  moreover have
  mset (∼ Φ @ (∼ φ # Γ)) = mset (∼ (φ # Φ) @ Γ)
  mset (replicate (length Φ) ⊥ @ (⊥ # Ψ))
    = mset (replicate (length (φ # Φ)) ⊥ @ Ψ)
    by simp+
  ultimately have
  Γ $⊢ ((φ # Φ) @ Ψ) = ∼ (φ # Φ) @ Γ $⊢ (replicate (length (φ # Φ)) ⊥ @
Ψ)
  by (metis
    append.assoc
    append-Cons
    append-Nil
    length-Cons
    replicate-append-same
    list-subtract.simps(1)
    map-ident replicate-Suc
    measure-msub-left-monotonic
    map-list-subtract-mset-containment)
  }
  then show ?case by blast
qed
thus ?thesis
by (metis append-Nil2 counting-deduction-to-measure-deduction)
qed

```

## 2.12 Measure Deduction Soundness

The last major result for measure deduction we have to show is *soundness*. That is, judgments in measure deduction of lists of formulae can be translated into tautologies for inequalities of finitely additive probability measures over those same formulae (using the same underlying classical logic).

**lemma** (in *classical-logic*) *negated-measure-deduction*:

$$\begin{aligned} & \sim \Gamma \ \$\vdash (\varphi \# \Phi) = \\ & (\exists \Psi. \text{mset} (\text{map fst } \Psi) \subseteq\# \text{mset } \Gamma \wedge \\ & \quad \sim (\text{map} (\text{uncurry } (\backslash)) \Psi) \vdash \varphi \wedge \\ & \quad \sim (\text{map} (\text{uncurry } (\cap)) \Psi) \ @ \ \Gamma \ominus (\text{map fst } \Psi)) \ \$\vdash \Phi) \end{aligned}$$

**proof** (*rule iff1*)

**assume**  $\sim \Gamma \ \$\vdash (\varphi \# \Phi)$   
**from this obtain**  $\Psi$  **where**  $\Psi$ :  
 $\text{mset} (\text{map snd } \Psi) \subseteq\# \text{mset} (\sim \Gamma)$   
 $\text{map} (\text{uncurry } (\sqcup)) \Psi \vdash \varphi$   
 $\text{map} (\text{uncurry } (\rightarrow)) \Psi \ @ \ \sim \Gamma \ominus \text{map snd } \Psi \ \$\vdash \Phi$   
**using** *measure-deduction.simps(2)*  
**by** *metis*  
**from this obtain**  $\Delta$  **where**  $\Delta$ :  
 $\text{mset } \Delta \subseteq\# \text{mset } \Gamma$   
 $\text{map snd } \Psi = \sim \Delta$   
**unfolding** *map-negation-def*  
**using** *mset-sub-map-list-exists* [**where**  $f = \sim$  **and**  $\Gamma = \Gamma$ ]  
**by** *metis*  
**let**  $? \Psi = \text{zip } \Delta (\text{map fst } \Psi)$   
**from**  $\Delta(2)$  **have**  $\text{map fst } ? \Psi = \Delta$   
**unfolding** *map-negation-def*  
**by** (*metis length-map map-fst-zip*)  
**with**  $\Delta(1)$  **have**  $\text{mset} (\text{map fst } ? \Psi) \subseteq\# \text{mset } \Gamma$   
**by** *simp*  
**moreover have**  $\forall \Delta. \text{map snd } \Psi = \sim \Delta \longrightarrow$   
 $\text{map} (\text{uncurry } (\sqcup)) \Psi \preceq \sim (\text{map} (\text{uncurry } (\backslash)) (\text{zip } \Delta (\text{map fst}$

$\Psi)))$

**proof** (*induct*  $\Psi$ )

**case** *Nil*

**then show** *?case* **by** *simp*

**next**

**case** (*Cons*  $\psi \Psi$ )

**let**  $? \psi = \text{fst } \psi$

{

**fix**  $\Delta$

**assume**  $\text{map snd } (\psi \# \Psi) = \sim \Delta$

**from this obtain**  $\gamma$  **where**  $\gamma: \sim \gamma = \text{snd } \psi \ \gamma = \text{hd } \Delta$  **by** *auto*

**from**  $\langle \text{map snd } (\psi \# \Psi) = \sim \Delta \rangle$  **have**  $\text{map snd } \Psi = \sim (\text{tl } \Delta)$  **by** *auto*

**with** *Cons.hyps* **have**

$\text{map} (\text{uncurry } (\sqcup)) \Psi \preceq \sim (\text{map} (\text{uncurry } (\backslash)) (\text{zip} (\text{tl } \Delta) (\text{map fst } \Psi)))$

**by** *auto*

```

moreover
{
  fix  $\psi \ \gamma$ 
  have  $\vdash \sim(\gamma \setminus \psi) \rightarrow (\psi \sqcup \sim \gamma)$ 
    unfolding disjunction-def
      subtraction-def
      conjunction-def
      negation-def
    by (meson modus-ponens
      flip-implication
      hypothetical-syllogism)
} note tautology = this
have  $\text{uncurry } (\sqcup) = (\lambda \ \psi. (\text{fst } \psi) \sqcup (\text{snd } \psi))$ 
  by fastforce
with  $\gamma$  have  $\text{uncurry } (\sqcup) \ \psi = ?\psi \sqcup \sim \gamma$ 
  by simp
with tautology have  $\vdash \sim(\gamma \setminus ?\psi) \rightarrow \text{uncurry } (\sqcup) \ \psi$ 
  by simp
ultimately have  $\text{map } (\text{uncurry } (\sqcup)) (\psi \# \Psi) \preceq$ 
   $\sim (\text{map } (\text{uncurry } (\setminus)) ((\text{zip } ((\text{hd } \Delta) \# (\text{tl } \Delta)) (\text{map } \text{fst } (\psi \#$ 
 $\Psi))))))$ 
  using stronger-theory-left-right-cons  $\gamma(2)$ 
  by simp
hence  $\text{map } (\text{uncurry } (\sqcup)) (\psi \# \Psi) \preceq$ 
   $\sim (\text{map } (\text{uncurry } (\setminus)) (\text{zip } \Delta (\text{map } \text{fst } (\psi \# \Psi))))$ 
  using  $\langle \text{map } \text{snd } (\psi \# \Psi) = \sim \Delta \rangle$  by force
}
thus ?case by blast
qed
with  $\Psi(2) \ \Delta(2)$  have  $\sim (\text{map } (\text{uncurry } (\setminus)) \ ?\Psi) \vdash \varphi$ 
  using stronger-theory-deduction-monotonic by blast
moreover
have  $(\text{map } (\text{uncurry } (\rightarrow)) \ \Psi) @ \sim \Gamma \ominus \text{map } \text{snd } \Psi \preceq$ 
   $\sim (\text{map } (\text{uncurry } (\sqcap)) \ ?\Psi) @ \Gamma \ominus (\text{map } \text{fst } \ ?\Psi)$ 
proof –
  from  $\Delta(1)$  have  $\text{mset } (\sim \Gamma \ominus \sim \Delta) = \text{mset } (\sim (\Gamma \ominus \Delta))$ 
    by (simp add: image-mset-Diff)
  hence  $\text{mset } (\sim \Gamma \ominus \text{map } \text{snd } \Psi) = \text{mset } (\sim (\Gamma \ominus \text{map } \text{fst } \ ?\Psi))$ 
    using  $\Psi(1) \ \Delta(2) \ \langle \text{map } \text{fst } \ ?\Psi = \Delta \rangle$  by simp
  hence  $(\sim \Gamma \ominus \text{map } \text{snd } \Psi) \preceq \sim (\Gamma \ominus \text{map } \text{fst } \ ?\Psi)$ 
    by (simp add: msub-stronger-theory-intro)
  moreover have  $\forall \ \Delta. \text{map } \text{snd } \Psi = \sim \Delta \rightarrow$ 
     $\text{map } (\text{uncurry } (\rightarrow)) \ \Psi \preceq \sim (\text{map } (\text{uncurry } (\sqcap)) (\text{zip } \Delta (\text{map}$ 
 $\text{fst } \Psi)))$ 
  proof (induct  $\Psi$ )
    case Nil
    then show ?case by simp
  next
  case (Cons  $\psi \ \Psi$ )

```

```

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

```

```

    by metis
next
assume  $\exists \Psi. \text{mset } (\text{map fst } \Psi) \subseteq \# \text{mset } \Gamma \wedge$ 
       $\sim (\text{map } (\text{uncurry } (\backslash)) \Psi) \vdash \varphi \wedge$ 
       $\sim (\text{map } (\text{uncurry } (\sqcap)) \Psi) @ \Gamma \ominus \text{map fst } \Psi \text{ \$}\vdash \Phi$ 
from this obtain  $\Psi$  where  $\Psi$ :
   $\text{mset } (\text{map fst } \Psi) \subseteq \# \text{mset } \Gamma$ 
   $\sim (\text{map } (\text{uncurry } (\backslash)) \Psi) \vdash \varphi$ 
   $\sim (\text{map } (\text{uncurry } (\sqcap)) \Psi) @ \Gamma \ominus \text{map fst } \Psi \text{ \$}\vdash \Phi$ 
by auto
let  $?\Psi = \text{zip } (\text{map snd } \Psi) (\sim (\text{map fst } \Psi))$ 
from  $\Psi(1)$  have  $\text{mset } (\text{map snd } ?\Psi) \subseteq \# \text{mset } (\sim \Gamma)$ 
  by (simp, metis image-mset-subseteq-mono multiset.map-comp)
moreover have  $\sim (\text{map } (\text{uncurry } (\backslash)) \Psi) \preceq \text{map } (\text{uncurry } (\sqcup)) ?\Psi$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case by simp
next
case (Cons  $\psi \Psi$ )
let  $? \gamma = \text{fst } \psi$ 
let  $? \psi = \text{snd } \psi$ 
{
  fix  $\psi \gamma$ 
  have  $\vdash (\psi \sqcup \sim \gamma) \rightarrow \sim(\gamma \backslash \psi)$ 
    unfolding disjunction-def
      subtraction-def
      conjunction-def
      negation-def
  by (meson modus-ponens
      flip-implication
      hypothetical-syllogism)
} note tautology = this
have  $\sim \circ \text{uncurry } (\backslash) = (\lambda \psi. \sim ((\text{fst } \psi) \backslash (\text{snd } \psi)))$ 
   $\text{uncurry } (\sqcup) = (\lambda (\psi, \gamma). \psi \sqcup \gamma)$ 
by fastforce+
with tautology have  $\vdash \text{uncurry } (\sqcup) (? \psi, \sim ? \gamma) \rightarrow (\sim \circ \text{uncurry } (\backslash)) \psi$ 
by fastforce
with Cons.hyps have
   $((\sim \circ \text{uncurry } (\backslash)) \psi \# \sim (\text{map } (\text{uncurry } (\backslash)) \Psi)) \preceq$ 
   $(\text{uncurry } (\sqcup) (? \psi, \sim ? \gamma) \# \text{map } (\text{uncurry } (\sqcup)) (\text{zip } (\text{map snd } \Psi) (\sim (\text{map}$ 
fst  $\Psi))))$ 
  using stronger-theory-left-right-cons by blast
thus ?case by simp
qed
with  $\Psi(2)$  have  $\text{map } (\text{uncurry } (\sqcup)) ?\Psi \vdash \varphi$ 
  using stronger-theory-deduction-monotonic by blast
moreover have  $\sim (\text{map } (\text{uncurry } (\sqcap)) \Psi) @ \Gamma \ominus \text{map fst } \Psi \preceq$ 
   $(\text{map } (\text{uncurry } (\rightarrow)) ?\Psi) @ \sim \Gamma \ominus \text{map snd } ?\Psi$ 
proof -

```



```

have  $\sim (map (uncurry (\sqcap)) \Psi) \preceq map (uncurry (\rightarrow)) ?\Psi$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case by simp
next
case (Cons  $\psi \Psi$ )
let ? $\gamma = fst \psi$ 
let ? $\psi = snd \psi$ 
{
  fix  $\psi \gamma$ 
  have  $\vdash (\psi \rightarrow \sim \gamma) \rightarrow \sim(\gamma \sqcap \psi)$ 
  unfolding disjunction-def
    conjunction-def
    negation-def
  by (meson modus-ponens
    flip-implication
    hypothetical-syllogism)
} note tautology = this
have  $\sim \circ uncurry (\sqcap) = (\lambda \psi. \sim ((fst \psi) \sqcap (snd \psi)))$ 
   $uncurry (\rightarrow) = (\lambda (\psi, \gamma). \psi \rightarrow \gamma)$ 
  by fastforce+
with tautology have  $\vdash uncurry (\rightarrow) (?\psi, \sim ?\gamma) \rightarrow (\sim \circ uncurry (\sqcap)) \psi$ 
  by fastforce
with Cons.hyps have
   $((\sim \circ uncurry (\sqcap)) \psi \# \sim (map (uncurry (\sqcap)) \Psi)) \preceq$ 
   $(uncurry (\rightarrow) (?\psi, \sim ?\gamma) \# map (uncurry (\rightarrow)) (zip (map snd \Psi) (\sim (map$ 
fst  $\Psi))))$ 
  using stronger-theory-left-right-cons by blast
then show ?case by simp
qed
moreover have  $mset (\sim (\Gamma \ominus map fst \Psi)) = mset (\sim \Gamma \ominus map snd ?\Psi)$ 
  using  $\Psi(1)$ 
  by (simp add: image-mset-Diff multiset.map-comp)
hence  $\sim (\Gamma \ominus map fst \Psi) \preceq (\sim \Gamma \ominus map snd ?\Psi)$ 
  using
    stronger-theory-reflexive
    stronger-theory-right-permutation
  by blast
ultimately show ?thesis
  using stronger-theory-combine
  by simp
qed
hence  $map (uncurry (\rightarrow)) ?\Psi @ \sim \Gamma \ominus map snd ?\Psi \ \$\vdash \Phi$ 
  using  $\Psi(3)$  measure-stronger-theory-left-monotonic by blast
ultimately show  $\sim \Gamma \ \$\vdash (\varphi \# \Phi)$ 
  using measure-deduction.simps(2) by blast
qed

```

lemma (in probability-logic) measure-deduction-soundness:

```

assumes  $\sim \Gamma \ \$\vdash \sim \Phi$ 
shows  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
proof -
have  $\forall \Gamma. \sim \Gamma \ \$\vdash \sim \Phi \longrightarrow (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
proof (induct  $\Phi$ )
  case Nil
  then show ?case
  by (simp, metis (full-types) ex-map-conv probability-non-negative sum-list-nonneg)
next
case (Cons  $\varphi \Phi$ )
  {
    fix  $\Gamma$ 
    assume  $\sim \Gamma \ \$\vdash \sim (\varphi \# \Phi)$ 
    hence  $\sim \Gamma \ \$\vdash (\sim \varphi \# \sim \Phi)$  by simp
    from this obtain  $\Psi$  where  $\Psi$ :
       $mset (map \text{fst } \Psi) \subseteq \# mset \Gamma$ 
       $\sim (map (uncurry (\)) \Psi) \vdash \sim \varphi$ 
       $\sim (map (uncurry (\square)) \Psi) @ \Gamma \ominus (map \text{fst } \Psi) \ \$\vdash \sim \Phi$ 
    using negated-measure-deduction by blast
    let  $?\Gamma = \Gamma \ominus (map \text{fst } \Psi)$ 
    let  $?\Psi_1 = map (uncurry (\)) \Psi$ 
    let  $?\Psi_2 = map (uncurry (\square)) \Psi$ 
    have  $(\sum \varphi' \leftarrow \Phi. \mathcal{P} \varphi') \leq (\sum \varphi \leftarrow (?\Psi_2 @ ?\Gamma). \mathcal{P} \varphi)$ 
      using Cons  $\Psi(3)$  by blast
    moreover
    have  $\mathcal{P} \varphi \leq (\sum \varphi \leftarrow ?\Psi_1. \mathcal{P} \varphi)$ 
      using  $\Psi(2)$ 
      biconditional-weaken
      list-deduction-def
      map-negation-list-implication
      set-deduction-base-theory
      implication-list-summation-inequality
    by blast
    ultimately have  $(\sum \varphi' \leftarrow (\varphi \# \Phi). \mathcal{P} \varphi') \leq (\sum \gamma \leftarrow (?\Psi_1 @ ?\Psi_2 @ ?\Gamma). \mathcal{P} \gamma)$ 
      by simp
    moreover have  $(\sum \varphi' \leftarrow (?\Psi_1 @ ?\Psi_2). \mathcal{P} \varphi') = (\sum \gamma \leftarrow (map \text{fst } \Psi). \mathcal{P} \gamma)$ 
    proof (induct  $\Psi$ )
      case Nil
      then show ?case by simp
    next
    case (Cons  $\psi \Psi$ )
    let  $?\Psi_1 = map (uncurry (\)) \Psi$ 
    let  $?\Psi_2 = map (uncurry (\square)) \Psi$ 
    let  $? \psi_1 = uncurry (\) \psi$ 
    let  $? \psi_2 = uncurry (\square) \psi$ 
    assume  $(\sum \varphi' \leftarrow (?\Psi_1 @ ?\Psi_2). \mathcal{P} \varphi') = (\sum \gamma \leftarrow (map \text{fst } \Psi). \mathcal{P} \gamma)$ 
    moreover
    {
      let  $? \gamma = \text{fst } \psi$ 

```

```

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

```

## Chapter 3

# MaxSAT

We turn now to showing that counting deduction reduces to MaxSAT, the problem of finding the maximal number of satisfiable clauses in a list of clauses.

### 3.1 Definition of Relative Maximal Clause Collections

Given a list of assumptions  $\Phi$  and formula  $\varphi$ , we can think of those maximal sublists of  $\Phi$  that do not prove  $\varphi$ . While in practice we will care about  $\varphi = \perp$ , we provide a general definition in the more general axiom class *implication-logic*.

**definition** (in *implication-logic*) *relative-maximals* :: 'a list  $\Rightarrow$  'a  $\Rightarrow$  'a list set ( $\mathcal{M}$ )  
where

$$\begin{aligned} \mathcal{M} \Gamma \varphi = & \\ & \{ \Phi. \text{mset } \Phi \subseteq\# \text{mset } \Gamma \\ & \quad \wedge \neg \Phi \vdash \varphi \\ & \quad \wedge (\forall \Psi. \text{mset } \Psi \subseteq\# \text{mset } \Gamma \longrightarrow \neg \Psi \vdash \varphi \longrightarrow \text{length } \Psi \leq \text{length } \Phi) \} \end{aligned}$$

**lemma** (in *implication-logic*) *relative-maximals-finite*: finite ( $\mathcal{M} \Gamma \varphi$ )

**proof** –

```
{
  fix  $\Phi$ 
  assume  $\Phi \in \mathcal{M} \Gamma \varphi$ 
  hence set  $\Phi \subseteq$  set  $\Gamma$ 
    length  $\Phi \leq$  length  $\Gamma$ 
  unfolding relative-maximals-def
  using mset-subset-eqD
    length-sub-mset
    mset-eq-length
  by fastforce+
}
```

```

hence  $\mathcal{M} \Gamma \varphi \subseteq \{xs. \text{set } xs \subseteq \text{set } \Gamma \wedge \text{length } xs \leq \text{length } \Gamma\}$ 
  by auto
moreover
have  $\text{finite } \{xs. \text{set } xs \subseteq \text{set } \Gamma \wedge \text{length } xs \leq \text{length } \Gamma\}$ 
  using finite-lists-length-le by blast
ultimately show ?thesis using rev-finite-subset by auto
qed

```

We know that  $\varphi$  is not a tautology if and only if the set of relative maximal sublists has an element.

**lemma** (in *implication-logic*) *relative-maximals-existence*:

$$(\neg \vdash \varphi) = (\exists \Sigma. \Sigma \in \mathcal{M} \Gamma \varphi)$$

**proof** (*rule iffI*)

**assume**  $\neg \vdash \varphi$

**show**  $\exists \Sigma. \Sigma \in \mathcal{M} \Gamma \varphi$

**proof** (*rule ccontr*)

**assume**  $\# \Sigma. \Sigma \in \mathcal{M} \Gamma \varphi$

**hence**  $\diamond: \forall \Phi. \text{mset } \Phi \subseteq \# \text{mset } \Gamma \longrightarrow$

$$\neg \Phi \vdash \varphi \longrightarrow$$

$$(\exists \Psi. \text{mset } \Psi \subseteq \# \text{mset } \Gamma \wedge \neg \Psi \vdash \varphi \wedge \text{length } \Psi > \text{length } \Phi)$$

**unfolding** *relative-maximals-def*

**by** *fastforce*

{

**fix**  $n$

**have**  $\exists \Psi. \text{mset } \Psi \subseteq \# \text{mset } \Gamma \wedge \neg \Psi \vdash \varphi \wedge \text{length } \Psi > n$

**using**  $\diamond$

**by** (*induct*  $n$ ,

*metis*

$\langle \neg \vdash \varphi \rangle$

*list.size(3)*

*list-deduction-base-theory*

*mset.simps(1)*

*subset-mset.zero-le,*

*metis*

*Nat.lessE*

*Suc-less-eq*)

}

**hence**  $\exists \Psi. \text{mset } \Psi \subseteq \# \text{mset } \Gamma \wedge \text{length } \Psi > \text{length } \Gamma$

**by** *auto*

**thus** *False*

**using** *size-mset-mono* **by** *fastforce*

**qed**

**next**

**assume**  $\exists \Sigma. \Sigma \in \mathcal{M} \Gamma \varphi$

**thus**  $\neg \vdash \varphi$

**unfolding** *relative-maximals-def*

**using** *list-deduction-weaken*

**by** *blast*

**qed**

**lemma** (in *implication-logic*) *relative-maximals-complement-deduction*:  
**assumes**  $\Phi \in \mathcal{M} \Gamma \varphi$   
**and**  $\psi \in \text{set } (\Gamma \ominus \Phi)$   
**shows**  $\Phi \vdash \psi \rightarrow \varphi$   
**proof** (*rule ccontr*)  
**assume**  $\neg \Phi \vdash \psi \rightarrow \varphi$   
**hence**  $\neg (\psi \# \Phi) \vdash \varphi$   
**by** (*simp add: list-deduction-theorem*)  
**moreover**  
**have**  $\text{mset } \Phi \subseteq \# \text{ mset } \Gamma \psi \in \# \text{ mset } (\Gamma \ominus \Phi)$   
**using** *assms*  
**unfolding** *relative-maximals-def*  
**by** (*blast, meson in-multiset-in-set*)  
**hence**  $\text{mset } (\psi \# \Phi) \subseteq \# \text{ mset } \Gamma$   
**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**  $\text{length } (\psi \# \Phi) \leq \text{length } (\Phi)$   
**using** *assms*  
**unfolding** *relative-maximals-def*  
**by** *blast*  
**thus** *False*  
**by** *simp*  
**qed**

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

```

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

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

lemma (in implication-logic) maximals-length-equiv:
  assumes  $\Phi \in \mathcal{M}\ \Gamma\ \varphi$ 
    and  $\Psi \in \mathcal{M}\ \Gamma\ \varphi$ 
  shows  $length\ \Phi = length\ \Psi$ 
  using assms
  by (simp add: dual-order.antisym relative-maximals-def)

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

```

We can think of  $\Gamma \vdash \varphi$  as saying "the relative maximal sublists of  $\Gamma$  are not

the entire list”.

**lemma** (in *implication-logic*) *relative-maximals-max-list-deduction*:

$$\Gamma \vdash \varphi = (\forall \Phi \in \mathcal{M} \Gamma \varphi. 1 \leq \text{length} (\Gamma \ominus \Phi))$$

**proof** *cases*

**assume**  $\vdash \varphi$

**hence**  $\Gamma \vdash \varphi \mathcal{M} \Gamma \varphi = \{\}$

**unfolding** *relative-maximals-def*

**by** (*simp add: list-deduction-weaken*)+

**then show** *?thesis* **by** *blast*

**next**

**assume**  $\neg \vdash \varphi$

**from** *this* **obtain**  $\Omega$  **where**  $\Omega: \Omega \in \mathcal{M} \Gamma \varphi$

**using** *relative-maximals-existence* **by** *blast*

**from** *this* **have**  $\text{mset } \Omega \subseteq\# \text{mset } \Gamma$

**unfolding** *relative-maximals-def* **by** *blast*

**hence**  $\diamond: \text{length} (\Gamma \ominus \Omega) = \text{length } \Gamma - \text{length } \Omega$

**by** (*metis list-subtract-mset-homomorphism*  
*size-Diff-submset*  
*size-mset*)

**show** *?thesis*

**proof** (*cases*  $\Gamma \vdash \varphi$ )

**assume**  $\Gamma \vdash \varphi$

**from**  $\Omega$  **have**  $\text{mset } \Omega \subset\# \text{mset } \Gamma$

**by** (*metis (no-types, lifting)*

*Diff-cancel*

*Diff-eq-empty-iff*

$\langle \Gamma \vdash \varphi \rangle$

*list-deduction-monotonic*

*relative-maximals-def*

*mem-Collect-eq*

*mset-eq-setD*

*subset-mset.dual-order.not-eq-order.implies-strict*)

**hence**  $\text{length } \Omega < \text{length } \Gamma$

**using** *mset-subset-size* **by** *fastforce*

**hence**  $1 \leq \text{length } \Gamma - \text{length } \Omega$

**by** (*simp add: Suc-leI*)

**with**  $\diamond$  **have**  $1 \leq \text{length} (\Gamma \ominus \Omega)$

**by** *simp*

**with**  $\langle \Gamma \vdash \varphi \rangle \Omega$  **show** *?thesis*

**by** (*metis maximals-list-subtract-length-equiv*)

**next**

**assume**  $\neg \Gamma \vdash \varphi$

**moreover** **have**  $\text{mset } \Gamma \subseteq\# \text{mset } \Gamma$

**by** *simp*

**moreover** **have**  $\text{length } \Omega \leq \text{length } \Gamma$

**using**  $\langle \text{mset } \Omega \subseteq\# \text{mset } \Gamma \rangle$  *length-sub-mset mset-eq-length*

**by** *fastforce*

**ultimately** **have**  $\text{length } \Omega = \text{length } \Gamma$

**using**  $\Omega$



```

    unfolding relative-maximals-def
  by (simp add: dual-order.antisym)
hence 1 > length (Γ ⊖ Ω)
  using ◇
  by simp
with ⟨¬ Γ ⊢ φ⟩ Ω show ?thesis
  by fastforce
qed
qed

```

## 3.2 Definition of MaxSAT

We next turn to defining an abstract form of MaxSAT, which is largest the number of simultaneously satisfiable propositions in a list of propositions.

Unlike conventional MaxSAT, we don't actually work at the *semantic* level, i.e. constructing a model for the Tarski truth relation  $\models$ . Instead, we just count the elements in a maximal, consistent sublist (i.e., a maximal sub list  $\Sigma$  such that  $\neg \Sigma \vdash \perp$ ) of the list of assumptions  $\Gamma$  we have at hand.

Because we do not work at the semantic level, computing if  $MaxSAT \Gamma \leq n$  is not in general CoNP-Complete, as it is classically classified [1]. In the special case that the underlying logic is the *classical propositional calculus*, then the complexity is CoNP-Complete. But we could imagine the underlying logic to be linear temporal logic or even first order logic. In such cases the complexity class would be higher in the complexity hierarchy.

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

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

**definition** (in *implication-logic*) *complement-relative-MaxSAT* :: 'a list  $\Rightarrow$  'a  $\Rightarrow$  nat (|| - || [45])  
**where**  
 $(|| \Gamma ||_{\varphi}) = \text{length } \Gamma - | \Gamma |_{\varphi}$

**lemma** (in *implication-logic*) *relative-MaxSAT-intro*:  
**assumes**  $\Phi \in \mathcal{M} \Gamma \varphi$   
**shows**  $\text{length } \Phi = | \Gamma |_{\varphi}$   
**proof** –  
**have**  $\forall n \in \{ \text{length } \Psi \mid \Psi. \Psi \in \mathcal{M} \Gamma \varphi \}, n \leq \text{length } \Phi$   
 $\text{length } \Phi \in \{ \text{length } \Psi \mid \Psi. \Psi \in \mathcal{M} \Gamma \varphi \}$   
**using** *assms relative-maximals-def*  
**by** *auto*

**moreover**  
**have**  $finite \{ length \Psi \mid \Psi. \Psi \in \mathcal{M} \Gamma \varphi \}$   
**using** *finite-imageI relative-maximals-finite*  
**by** *simp*  
**ultimately have**  $Max \{ length \Psi \mid \Psi. \Psi \in \mathcal{M} \Gamma \varphi \} = length \Phi$   
**using** *Max-eqI*  
**by** *blast*  
**thus** *?thesis*  
**using** *assms relative-MaxSAT-def*  
**by** *auto*  
**qed**

**lemma** (in *implication-logic*) *complement-relative-MaxSAT-intro:*

**assumes**  $\Phi \in \mathcal{M} \Gamma \varphi$   
**shows**  $length (\Gamma \ominus \Phi) = \|\Gamma\|_\varphi$   
**proof** –  
**have**  $mset \Phi \subseteq\# mset \Gamma$   
**using** *assms*  
**unfolding** *relative-maximals-def*  
**by** *auto*  
**moreover from this have**  $length (\Gamma \ominus \Phi) = length \Gamma - length \Phi$   
**by** (*metis list-subtract-mset-homomorphism size-Diff-submset size-mset*)  
**ultimately show** *?thesis*  
**unfolding** *complement-relative-MaxSAT-def*  
**by** (*metis assms relative-MaxSAT-intro*)  
**qed**

**lemma** (in *implication-logic*) *length-MaxSAT-decomposition:*

$length \Gamma = (\|\Gamma\|_\varphi) + \|\Gamma\|_\varphi$   
**proof** (*cases*  $\mathcal{M} \Gamma \varphi = \{\}$ )  
**case** *True*  
**then show** *?thesis*  
**unfolding** *relative-MaxSAT-def*  
*complement-relative-MaxSAT-def*  
**by** *simp*  
**next**  
**case** *False*  
**from this obtain**  $\Phi$  **where**  $\Phi \in \mathcal{M} \Gamma \varphi$   
**by** *fast*  
**moreover from this have**  $mset \Phi \subseteq\# mset \Gamma$   
**unfolding** *relative-maximals-def*  
**by** *auto*  
**moreover from this have**  $length (\Gamma \ominus \Phi) = length \Gamma - length \Phi$   
**by** (*metis list-subtract-mset-homomorphism size-Diff-submset size-mset*)  
**ultimately show** *?thesis*  
**unfolding** *complement-relative-MaxSAT-def*  
**using** *list-subtract-mset-eq relative-MaxSAT-intro*  
**by** *fastforce*  
**qed**

### 3.3 Reducing Counting Deduction to MaxSAT

Here we present a major result: counting deduction may be reduced to MaxSAT.

```

primrec MaxSAT-optimal-pre-witness :: 'a list  $\Rightarrow$  ('a list  $\times$  'a) list ( $\mathfrak{W}$ )
  where
     $\mathfrak{W} [] = []$ 
    |  $\mathfrak{W} (\psi \# \Psi) = (\Psi, \psi) \# \mathfrak{W} \Psi$ 

```

```

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

```

```

lemma MaxSAT-optimal-pre-witness-nonelement:
  assumes length  $\Delta \geq$  length  $\Psi$ 
  shows  $(\Delta, \delta) \notin \text{set } (\mathfrak{W} \Psi)$ 
  using assms

```

```

proof (induct  $\Psi$ )
  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 MaxSAT-optimal-pre-witness-distinct: distinct ( $\mathfrak{W} \Psi$ )
  by (induct  $\Psi$ , simp, simp add: MaxSAT-optimal-pre-witness-nonelement)

```

```

lemma MaxSAT-optimal-pre-witness-length-iff-eq:
   $\forall (\Delta, \delta) \in \text{set } (\mathfrak{W} \Psi). \forall (\Sigma, \sigma) \in \text{set } (\mathfrak{W} \Psi). (\text{length } \Delta = \text{length } \Sigma) = ((\Delta, \delta) = (\Sigma, \sigma))$ 

```

```

proof (induct  $\Psi$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\psi$   $\Psi$ )
  {
    fix  $\Delta$ 
    fix  $\delta$ 
    assume  $(\Delta, \delta) \in \text{set } (\mathfrak{W} (\psi \# \Psi))$ 
    and length  $\Delta =$  length  $\Psi$ 
    hence  $(\Delta, \delta) = (\Psi, \psi)$ 
    by (simp add: MaxSAT-optimal-pre-witness-nonelement)
  }
  hence  $\forall (\Delta, \delta) \in \text{set } (\mathfrak{W} (\psi \# \Psi)). (\text{length } \Delta = \text{length } \Psi) = ((\Delta, \delta) = (\Psi, \psi))$ 
  by blast
  with Cons show ?case
  by auto

```

qed

**lemma** *mset-distinct-msub-down*:

**assumes**  $mset\ A \subseteq\# mset\ B$

**and** *distinct B*

**shows** *distinct A*

**using** *assms*

**by** (*meson distinct-append mset-le-perm-append perm-distinct-iff*)

**lemma** *mset-remdups-set-sub-iff*:

$(mset\ (remdups\ A) \subseteq\# mset\ (remdups\ B)) = (set\ A \subseteq set\ B)$

**proof** –

**have**  $\forall B. (mset\ (remdups\ A) \subseteq\# mset\ (remdups\ B)) = (set\ A \subseteq set\ B)$

**proof** (*induct A*)

**case** *Nil*

**then show** *?case by simp*

**next**

**case** (*Cons a A*)

**then show** *?case*

**proof** (*cases a ∈ set A*)

**case** *True*

**then show** *?thesis using Cons by auto*

**next**

**case** *False*

{

**fix** *B*

**have**  $(mset\ (remdups\ (a\ \#\ A)) \subseteq\# mset\ (remdups\ B)) = (set\ (a\ \#\ A) \subseteq set\ B)$

**proof** (*rule iffI*)

**assume** *assm: mset (remdups (a # A)) ⊆# mset (remdups B)*

**hence**  $mset\ (remdups\ A) \subseteq\# mset\ (remdups\ B) - \{\#a\#$

**using** *False*

**by** (*simp add: insert-subset-eq-iff*)

**hence**  $mset\ (remdups\ A) \subseteq\# mset\ (remdups\ (removeAll\ a\ B))$

**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**  $set\ A \subseteq set\ (removeAll\ a\ B)$

**using** *Cons.hyps by blast*

**moreover from** *assm False have a ∈ set B*

**using** *mset-subset-eq-insertD by fastforce*

**ultimately show**  $set\ (a\ \#\ A) \subseteq set\ B$

**by** *auto*

**next**

**assume** *assm: set (a # A) ⊆ set B*

```

hence  $set\ A \subseteq set\ (removeAll\ a\ B)$  using False
  by auto
hence  $mset\ (remdups\ A) \subseteq\# mset\ (remdups\ B) - \{\#a\}$ 
  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  $a \in set\ B$  by auto
ultimately show  $mset\ (remdups\ (a\ \# A)) \subseteq\# mset\ (remdups\ B)$ 
  by (simp add: False insert-subset-eq-iff)
qed
}
then show ?thesis by simp
qed
qed
thus ?thesis by blast
qed

lemma range-characterization:
  ( $mset\ X = mset\ [0..<length\ X]$ ) = ( $distinct\ X \wedge (\forall x \in set\ X. x < length\ X)$ )
proof (rule iffI)
  assume  $mset\ X = mset\ [0..<length\ X]$ 
  thus  $distinct\ X \wedge (\forall x \in set\ X. x < length\ X)$ 
    by (metis atLeastLessThan-iff count-mset-0-iff distinct-count-atmost-1 distinct-upt set-upt)
next
  assume  $distinct\ X \wedge (\forall x \in set\ X. x < length\ X)$ 
  moreover
  {
    fix  $n$ 
    have  $\forall X. n = length\ X \longrightarrow$ 
       $distinct\ X \wedge (\forall x \in set\ X. x < length\ X) \longrightarrow$ 
       $mset\ X = mset\ [0..<length\ X]$ 
    proof (induct n)
      case  $0$ 
      then show ?case by simp
    next
    case (Suc n)
    {
      fix  $X$ 
      assume  $A: n + 1 = length\ X$ 
      and  $B: distinct\ X$ 
      and  $C: \forall x \in set\ X. x < length\ X$ 
      have  $n \in set\ X$ 
      proof (rule ccontr)
      assume  $n \notin set\ X$ 

```

```

from A have A': n = length (tl X)
  by simp
from B have B': distinct (tl X)
  by (simp add: distinct-tl)
have C':  $\forall x \in \text{set } (tl X). x < \text{length } (tl X)$ 
  by (metis
    A
    A'
    C
    <n  $\notin$  set X>
    Suc-eq-plus1
    Suc-le-eq
    Suc-le-mono
    le-less
    list.set-sel(2)
    list.size(3)
    nat.simps(3))
from A' B' C' Suc have mset (tl X) = mset [0..<n]
  by blast
from A have X = hd X # tl X
  by (metis Suc-eq-plus1 list.exhaust-sel list.size(3) nat.simps(3))
with B <mset (tl X) = mset [0..<n]> have hd X  $\notin$  set [0..<n]
  by (metis distinct.simps(2) mset-eq-setD)
hence hd X  $\geq$  n by simp
with C <n  $\notin$  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  $\in$  set X> diff-add-inverse2 length-remove1)
have B': distinct ?X'
  by (simp add: B)
have C':  $\forall x \in \text{set } ?X'. x < \text{length } ?X'$ 
  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 <n  $\in$  set X> perm-remove)
}

```

```

    then show ?case by fastforce
  qed
}
ultimately show mset X = mset [0..<length X]
  by blast
qed

```

**lemma** *distinct-pigeon-hole*:

```

fixes X :: nat list
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** *MaxSAT-optimal-pre-witness-pigeon-hole*:

```

assumes mset Σ ⊆# mset (ℳ Ψ)
  and Σ ≠ []
shows ∃ (Δ, δ) ∈ set Σ. length Δ + 1 ≥ length Σ
proof -
have distinct Σ
  using assms
    MaxSAT-optimal-pre-witness-distinct
    mset-distinct-msub-down
  by blast
with assms(1) have distinct (map (length ∘ fst) Σ)
proof (induct Σ)
case Nil
  then show ?case by simp
next
case (Cons σ Σ)
hence mset Σ ⊆# mset (ℳ 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**  $\delta \Delta$  **where**  $\sigma = (\Delta, \delta)$   
   **by** *fastforce*  
**hence**  $(\Delta, \delta) \in \text{set } (\mathfrak{V} \Psi)$   
   **using** *Cons.premis mset-subset-eq-insertD*  
   **by** *fastforce*  
**hence**  $\forall (\Sigma, \sigma) \in \text{set } (\mathfrak{V} \Psi). (\text{length } \Delta = \text{length } \Sigma) = ((\Delta, \delta) = (\Sigma, \sigma))$   
   **using** *MaxSAT-optimal-pre-witness-length-iff-eq [where  $\Psi = \Psi$ ]*  
   **by** *fastforce*  
**hence**  $\forall (\Sigma, \sigma) \in \text{set } \Sigma. (\text{length } \Delta = \text{length } \Sigma) = ((\Delta, \delta) = (\Sigma, \sigma))$   
   **using**  $\langle \text{mset } \Sigma \subseteq \# \text{ mset } (\mathfrak{V} \Psi) \rangle$   
   **by** *(metis (no-types, lifting) Un-iff mset-le-perm-append perm-set-eq set-append)*  
**hence**  $\text{length } (\text{fst } \sigma) \notin \text{set } (\text{map } (\lambda a. \text{length } (\text{fst } a)) \Sigma)$   
   **using** *Cons.premis(2)  $\langle \sigma = (\Delta, \delta) \rangle$*   
   **by** *fastforce*  
**ultimately show** *?case by simp*  
**qed**  
**moreover have**  $\text{length } (\text{map } (\text{length } \circ \text{fst}) \Sigma) = \text{length } \Sigma$  **by** *simp*  
**moreover have**  $\text{map } (\text{length } \circ \text{fst}) \Sigma \neq []$  **using** *assms by simp*  
**ultimately show** *?thesis*  
   **using** *distinct-pigeon-hole*  
   **by** *fastforce*  
**qed**

**abbreviation (in classical-logic)**  
*MaxSAT-optimal-witness*  $:: 'a \Rightarrow 'a \text{ list} \Rightarrow ('a \times 'a) \text{ list} (\mathfrak{W})$   
**where**  $\mathfrak{W} \varphi \Xi \equiv \text{map } (\lambda (\Psi, \psi). (\Psi \rightarrow \varphi, \psi)) (\mathfrak{V} \Xi)$

**abbreviation (in classical-logic)**  
*disjunction-MaxSAT-optimal-witness*  $:: 'a \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} (\mathfrak{W}_{\sqcup})$   
**where**  $\mathfrak{W}_{\sqcup} \varphi \Psi \equiv \text{map } (\text{uncurry } (\sqcup)) (\mathfrak{W} \varphi \Psi)$

**abbreviation (in classical-logic)**  
*implication-MaxSAT-optimal-witness*  $:: 'a \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list} (\mathfrak{W}_{\rightarrow})$   
**where**  $\mathfrak{W}_{\rightarrow} \varphi \Psi \equiv \text{map } (\text{uncurry } (\rightarrow)) (\mathfrak{W} \varphi \Psi)$

**lemma (in classical-logic)** *MaxSAT-optimal-witness-conjunction-identity:*

$\vdash \sqcap (\mathfrak{W}_{\sqcup} \varphi \Psi) \leftrightarrow (\varphi \sqcup \sqcap \Psi)$

**proof** *(induct  $\Psi$ )*

**case** *Nil*

**then show** *?case*

**unfolding** *biconditional-def*  
*disjunction-def*

**using** *axiom-k*  
*modus-ponens*  
*verum-tautology*

**by** *(simp, blast)*

**next**

**case** *(Cons  $\psi \Psi$ )*



```

have  $\vdash (\Psi \rightarrow \varphi) \leftrightarrow (\prod \Psi \rightarrow \varphi)$ 
  by (simp add: list-curry-uncurry)
hence  $\vdash \prod (\text{map } (\text{uncurry } (\sqcup)) (\mathfrak{W} \varphi (\psi \# \Psi)))$ 
   $\leftrightarrow ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap \prod (\text{map } (\text{uncurry } (\sqcup)) (\mathfrak{W} \varphi \Psi)))$ 
  unfolding biconditional-def
  using conjunction-monotonic
  disjunction-monotonic
  by simp
moreover have  $\vdash ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap \prod (\text{map } (\text{uncurry } (\sqcup)) (\mathfrak{W} \varphi \Psi)))$ 
   $\leftrightarrow ((\prod \Psi \rightarrow \varphi \sqcup \psi) \sqcap (\varphi \sqcup \prod \Psi))$ 
  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  $\vdash (\downarrow ? \varphi \downarrow)$  using propositional-semantic by blast
    thus ?thesis by simp
  qed
}
ultimately have  $\vdash \prod (\text{map } (\text{uncurry } (\sqcup)) (\mathfrak{W} \varphi (\psi \# \Psi))) \leftrightarrow (\varphi \sqcup (\psi \sqcap \prod \Psi))$ 
  using biconditional-transitivity-rule
  by blast
  then show ?case by simp
qed

lemma (in classical-logic) MaxSAT-optimal-witness-deduction:
 $\vdash \mathfrak{W}_{\sqcup} \varphi \Psi \rightarrow \varphi \leftrightarrow \Psi \rightarrow \varphi$ 
proof –
  have  $\vdash \mathfrak{W}_{\sqcup} \varphi \Psi \rightarrow \varphi \leftrightarrow (\prod (\mathfrak{W}_{\sqcup} \varphi \Psi) \rightarrow \varphi)$ 
  by (simp add: list-curry-uncurry)
  moreover
{
    fix  $\alpha \beta \gamma$ 
    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  $\vdash (\downarrow ? \varphi \downarrow)$  using propositional-semantic by blast
      thus ?thesis by simp
    qed
  }
  ultimately have  $\vdash \mathfrak{W}_{\sqcup} \varphi \Psi \rightarrow \varphi \leftrightarrow ((\varphi \sqcup \prod \Psi) \rightarrow \varphi)$ 
  using modus-ponens
  biconditional-transitivity-rule

```

*MaxSAT-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  $\vdash (\langle ? \varphi \rangle)$  using *propositional-semantic* by *blast*

    thus *?thesis* by *simp*

  qed

}

ultimately have  $\vdash \mathfrak{W}_{\sqcup} \varphi \Psi : \rightarrow \varphi \leftrightarrow (\prod \Psi \rightarrow \varphi)$

  using *biconditional-transitivity-rule* by *blast*

  thus *?thesis*

  using *biconditional-symmetry-rule*

*biconditional-transitivity-rule*

*list-curry-uncurry*

  by *blast*

qed

**lemma (in *classical-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*  $\Xi$ )

  case *Nil*

  have  $\vdash ((\varphi \sqcup \psi) \rightarrow \varphi) \rightarrow ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi$

  proof –

    let  $? \varphi = (((\langle \varphi \rangle \sqcup \langle \psi \rangle) \rightarrow \langle \varphi \rangle) \rightarrow ((\langle \varphi \rangle \rightarrow \langle \psi \rangle) \rightarrow \langle \varphi \rangle) \rightarrow \langle \varphi \rangle)$

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

    hence  $\vdash (\langle ? \varphi \rangle)$  using *propositional-semantic* by *blast*

    thus *?thesis* by *simp*

  qed

  then show *?case* by *simp*

next

  case (*Cons*  $\xi \Xi$ )

  let  $?A = \mathfrak{W}_{\sqcup} \varphi \Xi : \rightarrow \varphi$

  let  $?B = \mathfrak{W}_{\rightarrow} \varphi \Xi : \rightarrow \varphi$

  let  $?X = \Xi : \rightarrow \varphi$

  from *Cons.hyps* have  $\vdash ((?X \sqcup \psi) \rightarrow ?A) \rightarrow ((?X \rightarrow \psi) \rightarrow ?B) \rightarrow ?X$  by *simp*

**moreover**

  have  $\vdash (((?X \sqcup \psi) \rightarrow ?A) \rightarrow ((?X \rightarrow \psi) \rightarrow ?B) \rightarrow ?X)$

$\rightarrow ((\xi \rightarrow ?X \sqcup \psi) \rightarrow (?X \sqcup \xi) \rightarrow ?A) \rightarrow (((\xi \rightarrow ?X) \rightarrow \psi) \rightarrow (?X \rightarrow \xi)$

$\rightarrow ?B) \rightarrow \xi \rightarrow ?X$

**proof** –

  let  $? \varphi = (((\langle ?X \rangle \sqcup \langle \psi \rangle) \rightarrow \langle ?A \rangle) \rightarrow (((\langle ?X \rangle \rightarrow \langle \psi \rangle) \rightarrow \langle ?B \rangle) \rightarrow \langle ?X \rangle) \rightarrow$

$((\langle \xi \rangle \rightarrow \langle ?X \rangle \sqcup \langle \psi \rangle) \rightarrow (\langle ?X \rangle \sqcup \langle \xi \rangle) \rightarrow \langle ?A \rangle) \rightarrow$

$((\langle \xi \rangle \rightarrow \langle ?X \rangle) \rightarrow \langle \psi \rangle) \rightarrow (\langle ?X \rangle \rightarrow \langle \xi \rangle) \rightarrow \langle ?B \rangle) \rightarrow$

$\langle \xi \rangle \rightarrow$

$\langle ?X \rangle$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**ultimately**  
**have**  $\vdash ((\xi \rightarrow ?X \sqcup \psi) \rightarrow (?X \sqcup \xi) \rightarrow ?A) \rightarrow (((\xi \rightarrow ?X) \rightarrow \psi) \rightarrow (?X \rightarrow \xi) \rightarrow ?B) \rightarrow \xi \rightarrow ?X$   
**using** *modus-ponens*  
**by** *blast*  
**thus** *?case* **by** *simp*  
**qed**

**lemma** (*in classical-logic*) *disj-conj-impl-duality*:  
 $\vdash (\varphi \rightarrow \chi \sqcap \psi \rightarrow \chi) \leftrightarrow ((\varphi \sqcup \psi) \rightarrow \chi)$   
**proof** –  
**let**  $? \varphi = ((\langle \varphi \rangle \rightarrow \langle \chi \rangle \sqcap \langle \psi \rangle \rightarrow \langle \chi \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcup \langle \psi \rangle) \rightarrow \langle \chi \rangle))$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**

**lemma** (*in classical-logic*) *weak-disj-of-conj-equiv*:  
 $(\forall \sigma \in set \Sigma. \sigma \vdash \varphi) = \vdash \sqcup (map \sqcap \Sigma) \rightarrow \varphi$   
**proof** (*induct*  $\Sigma$ )  
**case** *Nil*  
**then show** *?case*  
**by** (*simp add: ex-falso-quodlibet*)  
**next**  
**case** (*Cons*  $\sigma \Sigma$ )  
**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**  $\dots = (\vdash \sigma \rightarrow \varphi \wedge \vdash \sqcup (map \sqcap \Sigma) \rightarrow \varphi)$  **using** *Cons.hyps*  
*list-deduction-def* **by** *simp*  
**also have**  $\dots = (\vdash \sqcap \sigma \rightarrow \varphi \wedge \vdash \sqcup (map \sqcap \Sigma) \rightarrow \varphi)$   
**using** *list-curry-uncurry weak-biconditional-weaken* **by** *blast*  
**also have**  $\dots = (\vdash \sqcap \sigma \rightarrow \varphi \sqcap \sqcup (map \sqcap \Sigma) \rightarrow \varphi)$  **by** *simp*  
**also have**  $\dots = (\vdash (\sqcap \sigma \sqcup \sqcup (map \sqcap \Sigma)) \rightarrow \varphi)$   
**using** *disj-conj-impl-duality weak-biconditional-weaken* **by** *blast*  
**finally show** *?case* **by** *simp*  
**qed**

**lemma** (*in classical-logic*) *arbitrary-disj-concat-equiv*:  
 $\vdash \sqcup (\Phi @ \Psi) \leftrightarrow (\sqcup \Phi \sqcup \sqcup \Psi)$   
**proof** (*induct*  $\Phi$ )  
**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*  $\varphi$   $\Phi$ )  
**have**  $\vdash \sqcup (\Phi @ \Psi) \leftrightarrow (\sqcup \Phi \sqcup \sqcup \Psi) \rightarrow (\varphi \sqcup \sqcup (\Phi @ \Psi)) \leftrightarrow ((\varphi \sqcup \sqcup \Phi) \sqcup \sqcup \Psi)$   
**proof** –  
**let**  $?\varphi =$   
 $(\langle \sqcup (\Phi @ \Psi) \rangle \leftrightarrow (\langle \sqcup \Phi \rangle \sqcup \langle \sqcup \Psi \rangle)) \rightarrow (\langle \varphi \rangle \sqcup \langle \sqcup (\Phi @ \Psi) \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcup \langle \sqcup \Phi \rangle) \sqcup \langle \sqcup \Psi \rangle)$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus**  $?thesis$  **by** *simp*  
**qed**  
**then show**  $?case$  **using** *Cons modus-ponens* **by** *simp*  
**qed**

**lemma** (*in classical-logic*) *arbitrary-conj-concat-equiv*:

$\vdash \sqcap (\Phi @ \Psi) \leftrightarrow (\sqcap \Phi \sqcap \sqcap \Psi)$

**proof** (*induct*  $\Phi$ )

**case** *Nil*

**then show**  $?case$

**by** (*simp*,

*meson modus-ponens*

*biconditional-introduction*

*conjunction-introduction*

*conjunction-right-elimination*

*verum-tautology*)

**next**

**case** (*Cons*  $\varphi$   $\Phi$ )

**have**  $\vdash \sqcap (\Phi @ \Psi) \leftrightarrow (\sqcap \Phi \sqcap \sqcap \Psi) \rightarrow (\varphi \sqcap \sqcap (\Phi @ \Psi)) \leftrightarrow ((\varphi \sqcap \sqcap \Phi) \sqcap \sqcap \Psi)$

**proof** –

**let**  $?\varphi =$

$(\langle \sqcap (\Phi @ \Psi) \rangle \leftrightarrow (\langle \sqcap \Phi \rangle \sqcap \langle \sqcap \Psi \rangle)) \rightarrow (\langle \varphi \rangle \sqcap \langle \sqcap (\Phi @ \Psi) \rangle) \leftrightarrow ((\langle \varphi \rangle \sqcap \langle \sqcap \Phi \rangle) \sqcap \langle \sqcap \Psi \rangle)$

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

**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*

**thus**  $?thesis$  **by** *simp*

**qed**

**then show**  $?case$  **using** *Cons modus-ponens* **by** *simp*

**qed**

**lemma** (*in classical-logic*) *conj-absorption*:

**assumes**  $\chi \in set \Phi$

**shows**  $\vdash \sqcap \Phi \leftrightarrow (\chi \sqcap \sqcap \Phi)$

**using** *assms*

```

proof (induct  $\Phi$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\varphi$   $\Phi$ )
  then show ?case
  proof (cases  $\varphi = \chi$ )
    case True
    then show ?thesis
    by (simp,
      metis biconditional-def
      implication-distribution
      trivial-implication
      weak-biconditional-weaken
      weak-conjunction-deduction-equivalence)

  next
  case False
  then show ?thesis
  by (metis Cons.prems
    arbitrary-conjunction.simps(2)
    modus-ponens
    arbitrary-conjunction-antitone
    biconditional-introduction
    remdups.simps(2)
    set-remdups
    set-subset-Cons)

qed
qed

lemma (in classical-logic) conj-extract:  $\vdash \sqcup (map ((\sqcap) \varphi) \Psi) \leftrightarrow (\varphi \sqcap \sqcup \Psi)$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case
  by (simp add: ex-falso-quodlibet biconditional-def conjunction-right-elimination)
next
  case (Cons  $\psi$   $\Psi$ )
  have  $\vdash \sqcup (map ((\sqcap) \varphi) \Psi) \leftrightarrow (\varphi \sqcap \sqcup \Psi)$ 
  →  $((\varphi \sqcap \psi) \sqcup \sqcup (map ((\sqcap) \varphi) \Psi)) \leftrightarrow (\varphi \sqcap (\psi \sqcup \sqcup \Psi))$ 
  proof –
  let  $?\varphi = \langle \sqcup (map ((\sqcap) \varphi) \Psi) \rangle \leftrightarrow (\langle \varphi \rangle \sqcap \langle \sqcup \Psi \rangle)$ 
  →  $((\langle \varphi \rangle \sqcap \langle \psi \rangle) \sqcup \langle \sqcup (map ((\sqcap) \varphi) \Psi) \rangle) \leftrightarrow (\langle \varphi \rangle \sqcap (\langle \psi \rangle \sqcup \langle \sqcup \Psi \rangle))$ 
  have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  by fastforce
  hence  $\vdash (\langle ?\varphi \rangle)$  using propositional-semantics by blast
  thus ?thesis by simp
qed
then show ?case using Cons modus-ponens by simp
qed

lemma (in classical-logic) conj-multi-extract:

```

```

  ⊢ ⊔ (map ⊔ (map ((@) Δ) Σ)) ↔ (⊔ Δ ⊔ ⊔ (map ⊔ Σ))
proof (induct Σ)
  case Nil
  then show ?case
    by (simp, metis list.simps(8) arbitrary-disjunction.simps(1) conj-extract)
next
  case (Cons σ Σ)
  moreover have
    ⊢ ⊔ (map ⊔ (map ((@) Δ) Σ)) ↔ (⊔ Δ ⊔ ⊔ (map ⊔ Σ))
      → ⊔ (Δ @ σ) ↔ (⊔ Δ ⊔ ⊔ σ)
      → (⊔ (Δ @ σ) ⊔ ⊔ (map (⊔ ∘ (@) Δ) Σ)) ↔ (⊔ Δ ⊔ (⊔ σ ⊔ ⊔ (map ⊔
Σ)))
proof -
  let ?φ =
    ⟨⊔ (map ⊔ (map ((@) Δ) Σ))⟩ ↔ (⟨⊔ Δ⟩ ⊔ ⟨⊔ (map ⊔ Σ)⟩)
    → ⟨⊔ (Δ @ σ)⟩ ↔ (⟨⊔ Δ⟩ ⊔ ⟨⊔ σ⟩)
    → (⟨⊔ (Δ @ σ)⟩ ⊔ ⟨⊔ (map (⊔ ∘ (@) Δ) Σ)⟩) ↔ (⟨⊔ Δ⟩ ⊔ (⟨⊔ σ⟩ ⊔ ⟨⊔
(map ⊔ Σ)⟩))
  have ∀ M. M ⊨prop ?φ by fastforce
  hence ⊢ ( ?φ ) using propositional-semantic by blast
  thus ?thesis by simp
qed
hence
  ⊢ (⊔ (Δ @ σ) ⊔ ⊔ (map (⊔ ∘ (@) Δ) Σ)) ↔ (⊔ Δ ⊔ (⊔ σ ⊔ ⊔ (map ⊔
Σ)))
  using Cons.hyps arbitrary-conj-concat-equiv modus-ponens by blast
  then show ?case by simp
qed

lemma (in classical-logic) extract-inner-concat:
  ⊢ ⊔ (map (⊔ ∘ (map snd ∘ (@) Δ)) Ψ) ↔ (⊔ (map snd Δ) ⊔ ⊔ (map (⊔ ∘
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 ?Δ' = map snd Δ
  let ?χ' = snd χ
  let ?Π = λφ. ⊔ (map snd φ)
  let ?ΠΔ = λφ. ⊔ (?Δ' @ map snd φ)
  from Cons have
    ⊢ ⊔ (map ?ΠΔ Ψ) ↔ (⊔ ?Δ' ⊔ ⊔ (map ?Π Ψ))

```

by *auto*  
**moreover have**  $\star$ :  $\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) = \text{map } ((\sqcap) ?\chi') \circ \text{map } ?\Pi\Delta$   
 by *fastforce*  
**have**  $\sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) = \sqcup$   $(\text{map } ((\sqcap) ?\chi') (\text{map } ?\Pi\Delta \Psi))$   
 by (*simp add:  $\star$* )  
**hence**  
 $\vdash \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) \leftrightarrow (?\chi' \sqcap \sqcup$   $(\text{map } (\lambda\varphi. ?\Pi\Delta \varphi) \Psi))$   
**using** *conj-extract by presburger*  
**moreover have**  
 $\vdash \sqcup$   $(\text{map } ?\Pi\Delta \Psi) \leftrightarrow (\sqcap ?\Delta' \sqcap \sqcup$   $(\text{map } ?\Pi \Psi))$   
 $\rightarrow \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) \leftrightarrow (?\chi' \sqcap \sqcup$   $(\text{map } ?\Pi\Delta \Psi))$   
 $\rightarrow \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) \leftrightarrow ((?\chi' \sqcap \sqcap ?\Delta') \sqcap \sqcup$   $(\text{map } ?\Pi \Psi))$   
**proof** –  
**let**  $?\varphi = \langle \sqcup$   $(\text{map } ?\Pi\Delta \Psi) \rangle \leftrightarrow (\langle \sqcap ?\Delta' \rangle \sqcap \langle \sqcup$   $(\text{map } ?\Pi \Psi) \rangle)$   
 $\rightarrow \langle \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) \rangle \leftrightarrow (\langle ?\chi' \rangle \sqcap \langle \sqcup$   $(\text{map } ?\Pi\Delta \Psi) \rangle)$   
 $\rightarrow \langle \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap ?\Pi\Delta \varphi) \Psi) \rangle \leftrightarrow ((\langle ?\chi' \rangle \sqcap \langle \sqcap ?\Delta' \rangle) \sqcap \langle \sqcup$   $(\text{map } ?\Pi \Psi) \rangle)$   
 ( $\text{map } ?\Pi \Psi$ )  
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  by *fastforce*  
**hence**  $\vdash$   $( ?\varphi )$  **using** *propositional-semantic by blast*  
**thus** *?thesis by simp*  
**qed**  
**ultimately have**  $\vdash \sqcup$   $(\text{map } (\lambda\varphi. ?\chi' \sqcap \sqcap$   $(?\Delta' @ \text{map snd } \varphi)) \Psi)$   
 $\leftrightarrow ((?\chi' \sqcap \sqcap ?\Delta') \sqcap \sqcup$   $(\text{map } (\lambda\varphi. \sqcap$   $(\text{map snd } \varphi)) \Psi))$   
**using** *modus-ponens by blast*  
**thus** *?case by simp*  
**qed**

**lemma (in classical-logic) extract-inner-concat-remdups:**  
 $\vdash \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi) \leftrightarrow$   
 $(\sqcap$   $(\text{map snd } \Delta) \sqcap \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$   
**proof** –  
**have**  $\forall \Psi. \vdash \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi) \leftrightarrow$   
 $(\sqcap$   $(\text{map snd } \Delta) \sqcap \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$   
**proof (induct  $\Delta$ )**  
**case Nil**  
**then show** *?case*  
**by** (*simp,*  
*meson modus-ponens*  
*biconditional-introduction*  
*conjunction-introduction*  
*conjunction-right-elimination*  
*verum-tautology*)

**next**  
**case (Cons  $\delta \Delta$ )**  
**{**  
**fix  $\Psi$**   
**have**  $\vdash \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi)$   
 $\leftrightarrow (\sqcap$   $(\text{map snd } (\delta \# \Delta)) \sqcap \sqcup$   $(\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$   
**proof (cases  $\delta \in \text{set } \Delta$ )**

**assume**  $\delta \in \text{set } \Delta$   
**have**  
 $\vdash \sqcap (\text{map snd } \Delta) \leftrightarrow (\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta))$   
 $\rightarrow \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi)$   
 $\leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$   
 $\rightarrow \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi)$   
 $\leftrightarrow ((\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta)) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})))$   
 $\Psi))$   
**proof** –  
**let**  $? \varphi = \langle \sqcap (\text{map snd } \Delta) \rangle \leftrightarrow (\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map snd } \Delta) \rangle)$   
 $\rightarrow \langle \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi) \rangle$   
 $\leftrightarrow (\langle \sqcap (\text{map snd } \Delta) \rangle \sqcap \langle \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})))$   
 $\Psi \rangle)$   
 $\rightarrow \langle \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi) \rangle$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map snd } \Delta) \rangle) \sqcap \langle \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi) \rangle)$   
 $\text{remdups} \rangle \Psi))$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ? \varphi$  **by** *fastforce*  
**hence**  $\vdash (\ ? \varphi )$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**moreover have**  $\vdash \sqcap (\text{map snd } \Delta) \leftrightarrow (\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta))$   
**by** (*simp add: <math>\delta \in \text{set } \Delta</math>* *conj-absorption*)  
**ultimately have**  
 $\vdash \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi)$   
 $\leftrightarrow ((\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta)) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})))$   
 $\Psi))$   
**using** *Cons.hyps modus-ponens* **by** *blast*  
**moreover have**  $\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta) = \text{map snd} \circ \text{remdups}$   
 $\circ (@) \Delta$   
**using**  $\langle \delta \in \text{set } \Delta \rangle$  **by** *fastforce*  
**ultimately show** *?thesis* **using** *Cons* **by** *simp*  
**next**  
**assume**  $\delta \notin \text{set } \Delta$   
**hence**  $\dagger$ :  
 $\sqcap \circ (\text{map snd} \circ \text{remdups}) = (\lambda \psi. \sqcap (\text{map snd } (\text{remdups } \psi)))$   
 $(\lambda \psi. \sqcap (\text{map snd } (\text{if } \delta \in \text{set } \psi \text{ then } \text{remdups } (\Delta @ \psi) \text{ else } \delta \# \text{remdups}$   
 $(\Delta @ \psi))))$   
 $= \sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))$   
**by** *fastforce+*  
**show** *?thesis*  
**proof** (*induct*  $\Psi$ )  
**case** *Nil*  
**then show** *?case*  
**by** (*simp,metis list.simps(8) arbitrary-disjunction.simps(1) conj-extract*)  
**next**  
**case** (*Cons*  $\psi \Psi$ )  
**have**  $\vdash \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) [\psi])$   
 $\leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) [\psi]))$   
**using**  $\langle \forall \Psi. \vdash \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) \Delta)) \Psi)$



$\leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$

**by blast**

**hence**

$\vdash (\sqcap (\text{map snd } (\text{remdups } (\Delta @ \psi))) \sqcup \perp)$

$\leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcap (\text{map snd } (\text{remdups } \psi)) \sqcup \perp)$

**by simp**

**hence  $\star$ :**

$\vdash \sqcap (\text{map snd } (\text{remdups } (\Delta @ \psi))) \leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcap (\text{map snd } (\text{remdups } \psi)))$

**by (metis**

*(no-types, opaque-lifting)*

*biconditional-conjunction-weaken-rule*

*biconditional-symmetry-rule*

*biconditional-transitivity-rule*

*disjunction-def*

*double-negation-biconditional*

*negation-def*

**have**  $\vdash \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi)$

$\leftrightarrow (\sqcap (\text{map snd } (\delta \# \Delta)) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$

**using Cons by blast**

**hence  $\diamond$ :**  $\vdash \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi)$

$\leftrightarrow ((\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta)) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$

**by simp**

**show ?case**

**proof (cases  $\delta \in \text{set } \psi$ )**

**assume  $\delta \in \text{set } \psi$**

**have  $\text{snd } \delta \in \text{set } (\text{map snd } (\text{remdups } \psi))$**

**using  $\langle \delta \in \text{set } \psi \rangle$  by auto**

**hence  $\spadesuit$ :**  $\vdash \sqcap (\text{map snd } (\text{remdups } \psi)) \leftrightarrow (\text{snd } \delta \sqcap \sqcap (\text{map snd } (\text{remdups } \psi)))$

**using conj-absorption by blast**

**have**

$\vdash (\sqcap (\text{map snd } (\text{remdups } \psi)) \leftrightarrow (\text{snd } \delta \sqcap \sqcap (\text{map snd } (\text{remdups } \psi))))$

$\rightarrow (\sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi)$

$\leftrightarrow ((\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta)) \sqcap \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$

$\rightarrow (\sqcap (\text{map snd } (\text{remdups } (\Delta @ \psi))) \leftrightarrow (\sqcap (\text{map snd } \Delta) \sqcap \sqcap (\text{map snd } (\text{remdups } \psi))))$

$\rightarrow (\sqcap (\text{map snd } (\text{remdups } (\Delta @ \psi)))$

$\sqcup \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi)$

$\leftrightarrow ((\text{snd } \delta \sqcap \sqcap (\text{map snd } \Delta))$

$\sqcap (\sqcap (\text{map snd } (\text{remdups } \psi)) \sqcup \sqcup (\text{map } (\sqcap \circ (\text{map snd} \circ \text{remdups})) \Psi))$

**proof –**

**let  $?\varphi =$**

$(\langle \sqcap (\text{map snd } (\text{remdups } \psi)) \rangle \leftrightarrow (\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map snd } (\text{remdups } \psi)) \rangle))$

$\psi))))$   
 $\rightarrow (\langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle) \sqcap \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
 $\rightarrow (\langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle$   
 $\leftrightarrow (\langle \sqcap (\text{map } \text{snd } \Delta) \rangle \sqcap \langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle))$   
 $\rightarrow (\langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle$   
 $\sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle)$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle)$   
 $\sqcap (\langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle \sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**hence**  
 $\vdash (\langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi)))$   
 $\sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle)$   
 $\sqcap (\langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle \sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
**using**  $\star \diamond \spadesuit$  *modus-ponens* **by** *blast*  
**thus** *?thesis* **using**  $\langle \delta \notin \text{set } \Delta \rangle \langle \delta \in \text{set } \psi \rangle$   
**by** (*simp add: †*)  
**next**  
**assume**  $\delta \notin \text{set } \psi$   
**have**  
 $\vdash (\langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle) \sqcap \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
 $\rightarrow (\langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle \leftrightarrow (\langle \sqcap (\text{map } \text{snd } \Delta) \rangle \sqcap \langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle))$   
 $\rightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle)$   
 $\sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle)$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle)$   
 $\sqcap (\langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle \sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
**proof** –  
**let**  $?\varphi =$   
 $(\langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle) \sqcap \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$   
 $\rightarrow (\langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle$   
 $\leftrightarrow (\langle \sqcap (\text{map } \text{snd } \Delta) \rangle \sqcap \langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle))$   
 $\rightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } (\text{remdups } (\Delta @ \psi))) \rangle)$   
 $\sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\delta \# \Delta))) \Psi) \rangle)$   
 $\leftrightarrow ((\langle \text{snd } \delta \rangle \sqcap \langle \sqcap (\text{map } \text{snd } \Delta) \rangle)$   
 $\sqcap (\langle \sqcap (\text{map } \text{snd } (\text{remdups } \psi)) \rangle \sqcup \langle \sqcup (\text{map } (\sqcap \circ (\text{map } \text{snd} \circ$   
 $\text{remdups}) \Psi) \rangle))$

```

      have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  by fastforce
      hence  $\vdash (\ ?\varphi \ )$  using propositional-semantic by blast
      thus ?thesis by simp
    qed
  hence
     $\vdash ((snd \ \delta \ \sqcap \ \sqcap \ (map \ snd \ (remdups \ (\Delta \ @ \ \psi))))$ 
       $\sqcup \ \sqcup \ (map \ (\sqcap \ \circ \ (map \ snd \ \circ \ remdups \ \circ \ (@) \ (\delta \ \# \ \Delta))) \ \Psi))$ 
       $\leftrightarrow ((snd \ \delta \ \sqcap \ \sqcap \ (map \ snd \ \Delta))$ 
         $\sqcap \ (\sqcap \ (map \ snd \ (remdups \ \psi)) \sqcup \ \sqcup \ (map \ (\sqcap \ \circ \ (map \ snd \ \circ$ 
remdups)) \ \Psi)))
      using  $\star \ \diamond$  modus-ponens by blast
      then show ?thesis using  $\langle \delta \notin set \ \psi \rangle \ \langle \delta \notin set \ \Delta \rangle$  by (simp add:  $\dagger$ )
    qed
  qed
}
then show ?case by fastforce
qed
thus ?thesis by blast
qed

```

**lemma (in classical-logic) optimal-witness-list-intersect-biconditional:**

```

  assumes  $mset \ \Xi \subseteq\# \ mset \ \Gamma$ 
    and  $mset \ \Phi \subseteq\# \ mset \ (\Gamma \ominus \ \Xi)$ 
    and  $mset \ \Psi \subseteq\# \ mset \ (\mathfrak{M} \rightarrow \varphi \ \Xi)$ 
  shows  $\exists \ \Sigma. \vdash ((\Phi \ @ \ \Psi) \rightarrow \varphi) \leftrightarrow (\sqcup \ (map \ \sqcap \ \Sigma) \rightarrow \varphi)$ 
     $\wedge (\forall \ \sigma \in set \ \Sigma. mset \ \sigma \subseteq\# \ mset \ \Gamma \wedge length \ \sigma + 1 \geq length \ (\Phi \ @ \ \Psi))$ 

```

**proof** –

```

  have  $\exists \ \Sigma. \vdash (\Psi \rightarrow \varphi) \leftrightarrow (\sqcup \ (map \ \sqcap \ \Sigma) \rightarrow \varphi)$ 
     $\wedge (\forall \ \sigma \in set \ \Sigma. mset \ \sigma \subseteq\# \ mset \ \Xi \wedge length \ \sigma + 1 \geq length \ \Psi)$ 

```

**proof** –

```

  from assms( $\beta$ ) obtain  $\Psi_0 :: ('a \ list \times 'a) \ list$  where  $\Psi_0$ :
     $mset \ \Psi_0 \subseteq\# \ mset \ (\mathfrak{M} \ \Xi)$ 
     $map \ (\lambda(\Psi, \psi). (\Psi \rightarrow \varphi \rightarrow \psi)) \ \Psi_0 = \Psi$ 
  using mset-sub-map-list-exists by fastforce
  let  $? \Pi_C = \lambda \ (\Delta, \delta) \ \Sigma. (map \ ((\#) \ (\Delta, \delta)) \ \Sigma) \ @ \ (map \ ((@) \ (\mathfrak{M} \ \Delta)) \ \Sigma)$ 
  let  $?T_\Sigma = \lambda \ \Psi. foldr \ ? \Pi_C \ \Psi \ []$ 
  let  $? \Sigma = map \ (map \ snd \ \circ \ remdups) \ (?T_\Sigma \ \Psi_0)$ 
  have  $I: \vdash (\Psi \rightarrow \varphi) \leftrightarrow (\sqcup \ (map \ \sqcap \ \Sigma) \rightarrow \varphi)$ 
  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  $\Psi :: ('a \ list \times 'a) \ list$ 
      let  $? \Sigma_\alpha = map \ (map \ snd) \ (?T_\Sigma \ \Psi)$ 
      let  $? \Sigma = map \ (map \ snd \ \circ \ remdups) \ (?T_\Sigma \ \Psi)$ 
      have  $\vdash (\sqcup \ (map \ \sqcap \ \Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\sqcup \ (map \ \sqcap \ \Sigma) \rightarrow \varphi)$ 
      proof (induct  $\Psi$ )
        case Nil

```

```

then show ?case by (simp add: biconditional-reflection)
next
  case (Cons Δδ Ψ)
  let ?Δ = fst Δδ
  let ?δ = snd Δδ
  let ?Σα = map (map snd) (?TΣ Ψ)
  let ?Σ = map (map snd ∘ remdups) (?TΣ Ψ)
  let ?Σ'α = map (map snd) (?TΣ ((?Δ, ?δ) # Ψ))
  let ?Σ' = map (map snd ∘ remdups) (?TΣ ((?Δ, ?δ) # Ψ))
  {
    fix Δ :: 'a list
    fix δ :: 'a
    let ?Σ'α' = map (map snd) (?TΣ ((Δ, δ) # Ψ))
    let ?Σ' = map (map snd ∘ remdups) (?TΣ ((Δ, δ) # Ψ))
    let ?Φ = map (map snd ∘ (@) [(Δ, δ)]) (?TΣ Ψ)
    let ?Ψ = map (map snd ∘ (@) (ℳ Δ)) (?TΣ Ψ)
    let ?Δ = map (map snd ∘ remdups ∘ (@) [(Δ, δ)]) (?TΣ Ψ)
    let ?Ω = map (map snd ∘ remdups ∘ (@) (ℳ Δ)) (?TΣ Ψ)
    have ⊢ (⊔ (map ⊔ ?Φ @ map ⊔ ?Ψ) ↔ (⊔ (map ⊔ ?Φ) ⊔ ⊔ (map
⊔ ?Ψ))) →
      (⊔ (map ⊔ ?Δ @ map ⊔ ?Ω) ↔ (⊔ (map ⊔ ?Δ) ⊔ ⊔ (map ⊔
?Ω))) →
        (⊔ (map ⊔ ?Φ) ↔ (⊔ [δ] ⊔ ⊔ (map ⊔ ?Σα))) →
        (⊔ (map ⊔ ?Ψ) ↔ (⊔ Δ ⊔ ⊔ (map ⊔ ?Σα))) →
        (⊔ (map ⊔ ?Δ) ↔ (⊔ [δ] ⊔ ⊔ (map ⊔ ?Σ))) →
        (⊔ (map ⊔ ?Ω) ↔ (⊔ Δ ⊔ ⊔ (map ⊔ ?Σ))) →
        ((⊔ (map ⊔ ?Σα) → φ) ↔ (⊔ (map ⊔ ?Σ) → φ)) →
        ((⊔ (map ⊔ ?Φ @ map ⊔ ?Ψ) → φ) ↔ (⊔ (map ⊔ ?Δ @ map
⊔ ?Ω) → φ))
    proof –
      let ?φ =
        ((⊔ (map ⊔ ?Φ @ map ⊔ ?Ψ)) ↔ ((⊔ (map ⊔ ?Φ)) ⊔ ⊔ (map
⊔ ?Ψ))) →
        ((⊔ (map ⊔ ?Δ @ map ⊔ ?Ω)) ↔ ((⊔ (map ⊔ ?Δ)) ⊔ ⊔ (map
⊔ ?Ω))) →
        ((⊔ (map ⊔ ?Φ)) ↔ ((⊔ [δ]) ⊔ ⊔ (map ⊔ ?Σα))) →
        ((⊔ (map ⊔ ?Ψ)) ↔ ((⊔ Δ) ⊔ ⊔ (map ⊔ ?Σα))) →
        ((⊔ (map ⊔ ?Δ)) ↔ ((⊔ [δ]) ⊔ ⊔ (map ⊔ ?Σ))) →
        ((⊔ (map ⊔ ?Ω)) ↔ ((⊔ Δ) ⊔ ⊔ (map ⊔ ?Σ))) →
        (((⊔ (map ⊔ ?Σα) → φ) ↔ ((⊔ (map ⊔ ?Σ) → φ))) →
        ((⊔ (map ⊔ ?Φ @ map ⊔ ?Ψ) → φ) ↔ ((⊔ (map ⊔ ?Δ @
map ⊔ ?Ω) → φ)))
      have ∀ℳ. ℳ ⊧prop ?φ by fastforce
      hence ⊢ (⊔ ?φ) using propositional-semantic by blast
      thus ?thesis by simp
    qed
  }
moreover
  have map snd (ℳ Δ) = Δ by (induct Δ, auto)
  hence ⊢ ⊔ (map ⊔ ?Φ @ map ⊔ ?Ψ) ↔ (⊔ (map ⊔ ?Φ) ⊔ ⊔ (map

```

$\sqcap ?\Psi$ )  
 $\Psi\Omega$ )  
 $\vdash \sqcup (map \sqcap ?\Delta @ map \sqcap ?\Omega) \leftrightarrow (\sqcup (map \sqcap ?\Delta) \sqcup \sqcup (map \sqcap ?\Omega))$   
 $\vdash \sqcup (map \sqcap ?\Phi) \leftrightarrow (\sqcap [\delta] \sqcap \sqcup (map \sqcap ?\Sigma_\alpha))$   
 $\vdash \sqcup (map \sqcap ?\Psi) \leftrightarrow (\sqcap \Delta \sqcap \sqcup (map \sqcap ?\Sigma_\alpha))$   
 $\vdash \sqcup (map \sqcap ?\Delta) \leftrightarrow (\sqcap [\delta] \sqcap \sqcup (map \sqcap ?\Sigma))$   
 $\vdash \sqcup (map \sqcap ?\Omega) \leftrightarrow (\sqcap \Delta \sqcap \sqcup (map \sqcap ?\Sigma))$   
**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  $\Psi = ?T_\Sigma \Psi$ ]  
 $\Psi$ ]  
*extract-inner-concat-remdups* [where  $\Delta = \mathfrak{V} \Delta$  and  $\Psi = ?T_\Sigma \Psi$ ]  
**by** *auto*  
**ultimately have**  
 $\vdash ((\sqcup (map \sqcap ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma) \rightarrow \varphi)) \rightarrow$   
 $(\sqcup (map \sqcap ?\Phi @ map \sqcap ?\Psi) \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Delta @ map \sqcap ?\Omega) \rightarrow \varphi)$   
**using** *modus-ponens* **by** *blast*  
**moreover have**  $(\#) (\Delta, \delta) = (@) [(\Delta, \delta)]$  **by** *fastforce*  
**ultimately have**  
 $\vdash ((\sqcup (map \sqcap ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma) \rightarrow \varphi)) \rightarrow$   
 $((\sqcup (map \sqcap ?\Sigma_\alpha') \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma') \rightarrow \varphi))$   
**by** *auto*  
**}**  
**hence**  
 $\vdash ((\sqcup (map \sqcap ?\Sigma_\alpha') \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma') \rightarrow \varphi))$   
**using** *Cons modus-ponens* **by** *blast*  
**moreover have**  $\Delta\delta = (?\Delta, ?\delta)$  **by** *fastforce*  
**ultimately show** *?case* **by** *metis*  
**qed**  
**}**  
**hence**  $\vdash (\sqcup (map \sqcap ?\Sigma_\alpha) \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma) \rightarrow \varphi)$  **by** *blast*  
**moreover have**  $\vdash (?\Psi' : \rightarrow \varphi) \leftrightarrow (\sqcup (map \sqcap ?\Sigma_\alpha) \rightarrow \varphi)$   
**proof** (*induct*  $\Psi_0$ )  
**case** *Nil*  
**have**  $\vdash \varphi \leftrightarrow ((\top \sqcup \perp) \rightarrow \varphi)$   
**proof** –  
**let**  $? \varphi = \langle \varphi \rangle \leftrightarrow ((\top \sqcup \perp) \rightarrow \langle \varphi \rangle)$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ? \varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ? \varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**thus** *?case* **by** *simp*  
**next**  
**case** (*Cons*  $\psi_0 \Psi_0$ )  
**let**  $? \Xi = fst \psi_0$   
**let**  $? \delta = snd \psi_0$   
**let**  $? \Psi' = map (\lambda(\Psi, \psi). (\Psi : \rightarrow \varphi \rightarrow \psi)) \Psi_0$

```

let ?Σα = map (map snd) (?TΣ Ψ0)
{
  fix Ξ :: 'a list
  have map snd (⋈ Ξ) = Ξ by (induct Ξ, auto)
  hence map snd ∘ (@) (⋈ Ξ) = (@) Ξ ∘ map snd by fastforce
}
moreover have (map snd ∘ (#) (?Ξ, ?δ)) = (@) [?δ] ∘ map snd by fastforce
ultimately have †:
  map (map snd) (?TΣ (ψ0 # Ψ0)) = map ((#) ?δ) ?Σα @ map ((@) ?Ξ)
?Σα
  map (λ(Ψ,ψ). (Ψ :→ φ → ψ)) (ψ0 # Ψ0) = ?Ξ :→ φ → ?δ # ?Ψ'
  by (simp add: case-prod-beta')+
have A: † (?Ψ' :→ φ) ↔ (⋈ (map † ?Σα) → φ) using Cons.hyps by auto
have B: † (?Ξ :→ φ) ↔ (∏ ?Ξ → φ)
  by (simp add: list-curry-uncurry)
have C: † (⋈ (map † (map ((#) ?δ) ?Σα) @ map † (map ((@) ?Ξ)
?Σα))
  ↔ (⋈ (map † (map ((#) ?δ) ?Σα)) ∪ ⋈ (map † (map ((@) ?Ξ)
?Σα)))
  using arbitrary-disj-concat-equiv by blast
have map † (map ((#) ?δ) ?Σα) = (map ((∏) ?δ) (map † ?Σα)) by auto
hence D: † (⋈ (map † (map ((#) ?δ) ?Σα)) ↔ (?δ ∏ ⋈ (map † ?Σα))
  using conj-extract by presburger
have E: † (⋈ (map † (map ((@) ?Ξ) ?Σα)) ↔ (∏ ?Ξ ∏ ⋈ (map † ?Σα))
  using conj-multi-extract by blast
have
  †
  ( ?Ψ' :→ φ) ↔ (⋈ (map † ?Σα) → φ)
  → (?Ξ :→ φ) ↔ (∏ ?Ξ → φ)
  → ⋈ (map † (map ((#) ?δ) ?Σα) @ map † (map ((@) ?Ξ) ?Σα))
  ↔ (⋈ (map † (map ((#) ?δ) ?Σα)) ∪ ⋈ (map † (map ((@) ?Ξ)
?Σα)))
  → ⋈ (map † (map ((#) ?δ) ?Σα)) ↔ (?δ ∏ ⋈ (map † ?Σα))
  → ⋈ (map † (map ((@) ?Ξ) ?Σα)) ↔ (∏ ?Ξ ∏ ⋈ (map † ?Σα))
  → ((?Ξ :→ φ → ?δ) → ?Ψ' :→ φ)
  ↔ (⋈ (map † (map ((#) ?δ) ?Σα) @ map † (map ((@) ?Ξ) ?Σα))
→ φ)
proof –
  let ?φ =
    ⟨ ?Ψ' :→ φ ⟩ ↔ (⟨ ⋈ (map † ?Σα) ⟩ → ⟨ φ ⟩)
  → ⟨ (?Ξ :→ φ) ⟩ ↔ (⟨ ∏ ?Ξ ⟩ → ⟨ φ ⟩)
  → ⟨ ⋈ (map † (map ((#) ?δ) ?Σα) @ map † (map ((@) ?Ξ)
?Σα)) ⟩
  ↔ (⟨ ⋈ (map † (map ((#) ?δ) ?Σα)) ⟩ ∪ ⟨ ⋈ (map † (map ((@)
?Ξ) ?Σα)) ⟩)
  → ⟨ ⋈ (map † (map ((#) ?δ) ?Σα)) ⟩ ↔ (⟨ ?δ ⟩ ∏ ⟨ ⋈ (map †
?Σα)) ⟩)
  → ⟨ ⋈ (map † (map ((@) ?Ξ) ?Σα)) ⟩ ↔ (⟨ ∏ ?Ξ ⟩ ∏ ⟨ ⋈ (map †
?Σα)) ⟩)
  → (⟨ (?Ξ :→ φ) → ⟨ ?δ ⟩ ⟩ → ⟨ ?Ψ' :→ φ ⟩)

```

```

      ↔ (⟨⊔ (map ⊔ (map ((#) ?δ) ?Σα) @ map ⊔ (map ((@) ?Ξ)
?Σα))⟩ → ⟨φ⟩)
    have ∀ M. M ⊨prop ?φ by fastforce
    hence ⊢ ( ?φ ) using propositional-semantic by blast
    thus ?thesis by simp
  qed
  hence
    ⊢ ((?Ξ :→ φ → ?δ) → ?Ψ' :→ φ)
      ↔ (⊔ (map ⊔ (map ((#) ?δ) ?Σα) @ map ⊔ (map ((@) ?Ξ) ?Σα)) →
φ)
    using A B C D E modus-ponens by blast
    thus ?case using † by simp
  qed
  ultimately show ?thesis using biconditional-transitivity-rule Ψ0 by blast
  qed
  have II: ∀ σ ∈ set ?Σ. length σ + 1 ≥ length Ψ
  proof -
    let ?F = length ∘ fst
    let ?S = sort-key (- ?F)
    let ?Σ' = map (map snd ∘ remdups) (?TΣ (?S Ψ0))
    have mset Ψ0 = mset (?S Ψ0) by simp

    have ∀ Φ. mset Ψ0 = mset Φ → mset (map mset (?TΣ Ψ0)) = mset (map
mset (?TΣ Φ))
    proof (induct Ψ0)
      case Nil
      then show ?case by simp
    next
      case (Cons ψ Ψ0)
      obtain Δ δ where ψ = (Δ,δ) by fastforce
      {
        fix Φ
        assume mset (ψ # Ψ0) = mset Φ
        hence mset Ψ0 = mset (remove1 ψ Φ)
          by (simp add: union-single-eq-diff)
        have ψ ∈ set Φ using ⟨mset (ψ # Ψ0) = mset Φ⟩
          by (metis list.set-intros(1) set-mset-mset)
        hence mset (map mset (?TΣ Φ)) = mset (map mset (?TΣ (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' = ?TΣ (ψ # (remove1 ψ Φ))
have †: mset (map mset ?Σ') = mset (map mset (?TΣ Φ))
  using Cons False by simp
obtain Δ' δ'
  where φ = (Δ', δ')
  by fastforce
let ?Σ = ?TΣ (remove1 ψ Φ)
let ?m = image-mset mset
have
  mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
    mset (map mset (?ΠC ψ (?ΠC φ ?Σ)))
  using False by simp
hence mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
  (?m ∘ image-mset ((#) ψ) ∘ image-mset ((#) φ)) (mset ?Σ) +
  (?m ∘ image-mset ((#) ψ) ∘ image-mset ((@) (ℳ Δ')))) (mset
?Σ) +
  (?m ∘ image-mset ((@) (ℳ Δ)) ∘ image-mset ((#) φ)) (mset
?Σ) +
  (?m ∘ image-mset ((@) (ℳ Δ)) ∘ image-mset ((@) (ℳ Δ'))))
(mset ?Σ)
  using ⟨ψ = (Δ, δ)⟩ ⟨φ = (Δ', δ')⟩
  by (simp add: multiset.map-comp)
hence mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
  (?m ∘ image-mset ((#) φ) ∘ image-mset ((#) ψ)) (mset ?Σ) +
  (?m ∘ image-mset ((@) (ℳ Δ')) ∘ image-mset ((#) ψ)) (mset
?Σ) +
  (?m ∘ image-mset ((#) φ) ∘ image-mset ((@) (ℳ Δ))) (mset
?Σ) +
  (?m ∘ image-mset ((@) (ℳ Δ')) ∘ image-mset ((@) (ℳ Δ)))
(mset ?Σ)
  by (simp add: image-mset-cons-homomorphism
        image-mset-append-homomorphism
        image-mset-add-collapse
        add-mset-commute
        add.commute)
hence mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
  (?m ∘ image-mset ((#) φ)) (mset ?Σ') +
  (?m ∘ image-mset ((@) (ℳ Δ')))) (mset ?Σ')
  using ⟨ψ = (Δ, δ)⟩
  by (simp add: multiset.map-comp)
hence mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
  image-mset ((+) {#φ#}) (mset (map mset ?Σ')) +
  image-mset ((+) (mset (ℳ Δ')))) (mset (map mset ?Σ'))
  by (simp add: image-mset-cons-homomorphism
        image-mset-append-homomorphism)
hence mset (map mset (?TΣ (ψ # remove1 ψ (φ # Φ)))) =
  image-mset ((+) {#φ#}) (mset (map mset (?TΣ Φ))) +
  image-mset ((+) (mset (ℳ Δ')))) (mset (map mset (?TΣ Φ)))
  using † by auto

```



```

hence mset (map mset (?T $\Sigma$  ( $\psi$  # remove1  $\psi$  ( $\varphi$  #  $\Phi$ )))) =
  (?m  $\circ$  (image-mset ((#)  $\varphi$ )) (mset (?T $\Sigma$   $\Phi$ )) +
  (?m  $\circ$  (image-mset ((@) ( $\mathfrak{V}$   $\Delta'$ )))) (mset (?T $\Sigma$   $\Phi$ ))
  by (simp add: image-mset-cons-homomorphism
    image-mset-append-homomorphism)
  thus ?thesis using  $\langle \varphi = (\Delta', \delta') \rangle$  by (simp add: multiset.map-comp)
qed
qed
hence image-mset mset (image-mset ((#)  $\psi$ ) (mset (?T $\Sigma$  (remove1  $\psi$ 
 $\Phi$ )))) +
  image-mset mset (image-mset ((@) ( $\mathfrak{V}$   $\Delta$ )) (mset (?T $\Sigma$  (remove1
 $\psi$   $\Phi$ ))))
  = image-mset mset (mset (?T $\Sigma$   $\Phi$ ))
  by (simp add:  $\langle \psi = (\Delta, \delta) \rangle$  multiset.map-comp)
hence
  image-mset ((+) {#  $\psi$  #}) (image-mset mset (mset (?T $\Sigma$  (remove1  $\psi$ 
 $\Phi$ )))) +
  image-mset ((+) (mset ( $\mathfrak{V}$   $\Delta$ ))) (image-mset mset (mset (?T $\Sigma$  (remove1
 $\psi$   $\Phi$ ))))
  = image-mset mset (mset (?T $\Sigma$   $\Phi$ ))
by (simp add: image-mset-cons-homomorphism image-mset-append-homomorphism)
hence
  image-mset ((+) {#  $\psi$  #}) (image-mset mset (mset (?T $\Sigma$   $\Psi_0$ ))) +
  image-mset ((+) (mset ( $\mathfrak{V}$   $\Delta$ ))) (image-mset mset (mset (?T $\Sigma$   $\Psi_0$ )))
  = image-mset mset (mset (?T $\Sigma$   $\Phi$ ))
  using Cons  $\langle$  mset  $\Psi_0 =$  mset (remove1  $\psi$   $\Phi$ )  $\rangle$ 
  by fastforce
hence
  image-mset mset (image-mset ((#)  $\psi$ ) (mset (?T $\Sigma$   $\Psi_0$ ))) +
  image-mset mset (image-mset ((@) ( $\mathfrak{V}$   $\Delta$ )) (mset (?T $\Sigma$   $\Psi_0$ )))
  = image-mset mset (mset (?T $\Sigma$   $\Phi$ ))
by (simp add: image-mset-cons-homomorphism image-mset-append-homomorphism)
  hence mset (map mset (?T $\Sigma$  ( $\psi$  #  $\Psi_0$ ))) = mset (map mset (?T $\Sigma$   $\Phi$ ))
  by (simp add:  $\langle \psi = (\Delta, \delta) \rangle$  multiset.map-comp)
}
then show ?case by blast
qed
hence mset (map mset (?T $\Sigma$   $\Psi_0$ )) = mset (map mset (?T $\Sigma$  (? $\mathcal{S}$   $\Psi_0$ )))
  using  $\langle$  mset  $\Psi_0 =$  mset (? $\mathcal{S}$   $\Psi_0$ )  $\rangle$  by blast
hence mset (map (mset  $\circ$  (map snd)  $\circ$  remdups) (?T $\Sigma$   $\Psi_0$ ))
  = mset (map (mset  $\circ$  (map snd)  $\circ$  remdups) (?T $\Sigma$  (? $\mathcal{S}$   $\Psi_0$ )))
  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  $\sigma =$  mset  $\sigma'$ 
  by fastforce
hence  $\forall \sigma \in$  set ? $\Sigma$ .  $\exists \sigma' \in$  set ? $\Sigma'$ . length  $\sigma =$  length  $\sigma'$ 

```

```

using mset-eq-length by blast
have  $mset (?S \Psi_0) \subseteq\# mset (\mathfrak{V} \Xi)$ 
by (simp add: \Psi_0(1))
{
  fix  $n$ 
  have  $\forall \Psi. mset \Psi \subseteq\# mset (\mathfrak{V} \Xi) \longrightarrow$ 
     $sorted (map (- ?F) \Psi) \longrightarrow$ 
     $length \Psi = n \longrightarrow$ 
     $(\forall \sigma' \in set (map (map snd \circ remdups) (?T_\Sigma \Psi)). length \sigma' + 1$ 
 $\geq n)$ 
  proof (induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  {
    fix  $\Psi :: ('a list \times 'a) list$ 
    assume  $A: mset \Psi \subseteq\# mset (\mathfrak{V} \Xi)$ 
      and  $B: sorted (map (- ?F) \Psi)$ 
      and  $C: length \Psi = n + 1$ 
    obtain  $\Delta \delta$  where  $(\Delta, \delta) = hd \Psi$ 
      using prod.collapse by blast
    let  $?\Psi' = tl \Psi$ 
    have  $mset ?\Psi' \subseteq\# mset (\mathfrak{V} \Xi)$  using  $A$ 
    by (induct \Psi, simp, simp, meson mset-subset-eq-insertD subset-mset-def)
    moreover
    have  $sorted (map (- ?F) (tl \Psi))$ 
      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  $A \langle\Psi = (\Delta, \delta) \# ?\Psi'\rangle$ 
      MaxSAT-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*

$\Delta'$

**hence**  $\forall (\Delta', \delta') \in \text{set } ?\Psi'. (\Delta, \delta) \neq (\Delta', \delta')$   
 $\forall (\Delta', \delta') \in \text{set } (\mathfrak{A} \Xi). ((\Delta, \delta) \neq (\Delta', \delta')) \longrightarrow (\text{length } \Delta \neq \text{length } \Delta')$

$\text{set } ?\Psi' \subseteq \text{set } (\mathfrak{A} \Xi)$   
**using** *MaxSAT-optimal-pre-witness-length-iff-eq* [where  $\Psi = \Xi$ ]  
 $\langle \text{distinct } ((\Delta, \delta) \# ?\Psi') \rangle$   
**by** *auto*

**hence**  $\forall (\Delta', \delta') \in \text{set } ?\Psi'. \text{length } \Delta \neq \text{length } \Delta'$   
 $\text{sorted } (\text{map } (- ?\mathcal{F}) ((\Delta, \delta) \# ?\Psi'))$   
**using**  $B \langle \Psi = (\Delta, \delta) \# ?\Psi' \rangle$   
**by** (*fastforce, auto*)

**hence**  $\forall (\Delta', \delta') \in \text{set } ?\Psi'. \text{length } \Delta > \text{length } \Delta'$   
**by** *fastforce*

{

**fix**  $\sigma' :: 'a \text{ list}$   
**assume**  $\sigma' \in \text{set } (\text{map } (\text{map } \text{snd} \circ \text{remdups}) (?T_{\Sigma} \Psi))$   
**hence**  $\sigma' \in \text{set } (\text{map } (\text{map } \text{snd} \circ \text{remdups}) (?T_{\Sigma} ((\Delta, \delta) \# ?\Psi')))$   
**using**  $\langle \Psi = (\Delta, \delta) \# ?\Psi' \rangle$   
**by** *simp*

**from this obtain**  $\psi$  **where**  $\psi$ :  
 $\psi \in \text{set } (?T_{\Sigma} ?\Psi')$   
 $\sigma' = (\text{map } \text{snd} \circ \text{remdups} \circ (\#) (\Delta, \delta)) \psi \vee$   
 $\sigma' = (\text{map } \text{snd} \circ \text{remdups} \circ (@) (\mathfrak{A} \Delta)) \psi$   
**by** *fastforce*

**hence**  $\text{length } \sigma' \geq n$   
**proof** (*cases*  $\sigma' = (\text{map } \text{snd} \circ \text{remdups} \circ (\#) (\Delta, \delta)) \psi$ )  
**case** *True*

{

**fix**  $\Psi :: ('a \text{ list} \times 'a) \text{ list}$   
**fix**  $n :: \text{nat}$   
**assume**  $\forall (\Delta, \delta) \in \text{set } \Psi. n > \text{length } \Delta$   
**hence**  $\forall \sigma \in \text{set } (?T_{\Sigma} \Psi). \forall (\Delta, \delta) \in \text{set } \sigma. n > \text{length } \Delta$   
**proof** (*induct*  $\Psi$ )  
**case** *Nil*  
**then show** *?case* **by** *simp*

**next**  
**case** (*Cons*  $\psi \Psi$ )  
**obtain**  $\Delta \delta$  **where**  $\psi = (\Delta, \delta)$   
**by** *fastforce*  
**hence**  $n > \text{length } \Delta$  **using** *Cons.prem*s **by** *fastforce*  
**have**  $0: \forall \sigma \in \text{set } (?T_{\Sigma} \Psi). \forall (\Delta', \delta') \in \text{set } \sigma. n > \text{length } \Delta'$   
**using** *Cons* **by** *simp*

{

**fix**  $\sigma :: ('a \text{ list} \times 'a) \text{ list}$   
**fix**  $\psi' :: 'a \text{ list} \times 'a$

```

assume 1:  $\sigma \in \text{set } (?T_\Sigma (\psi \# \Psi))$ 
  and 2:  $\psi' \in \text{set } \sigma$ 
obtain  $\Delta' \delta'$  where  $\psi' = (\Delta', \delta')$ 
  by fastforce
have 3:  $\sigma \in (\#) (\Delta, \delta) \text{ ' set } (?T_\Sigma \Psi) \vee \sigma \in (@) (\mathfrak{A} \Delta) \text{ ' set}$ 
(?TΣ Ψ)

  using 1  $\langle \psi = (\Delta, \delta) \rangle$  by simp
have  $n > \text{length } \Delta'$ 
proof (cases  $\sigma \in (\#) (\Delta, \delta) \text{ ' set } (?T_\Sigma \Psi)$ )
  case True
    from this obtain  $\sigma'$  where
      set  $\sigma = \text{insert } (\Delta, \delta) (\text{set } \sigma')$ 
       $\sigma' \in \text{set } (?T_\Sigma \Psi)$ 
      by auto
    then show ?thesis
      using 0  $\langle \psi' \in \text{set } \sigma \rangle \langle \psi' = (\Delta', \delta') \rangle \langle n > \text{length } \Delta \rangle$ 
      by auto
  next
    case False
    from this and 3 obtain  $\sigma'$  where
      set  $\sigma = \text{set } (\mathfrak{A} \Delta) \cup (\text{set } \sigma')$ 
       $\sigma' \in \text{set } (?T_\Sigma \Psi)$ 
      by auto
    have  $\forall (\Delta', \delta') \in \text{set } (\mathfrak{A} \Delta). \text{length } \Delta > \text{length } \Delta'$ 
    by (metis (mono-tags, lifting)
        case-prodI2
        MaxSAT-optimal-pre-witness-nonelement
        not-le)
    hence  $\forall (\Delta', \delta') \in \text{set } (\mathfrak{A} \Delta). n > \text{length } \Delta'$ 
    using  $\langle n > \text{length } \Delta \rangle$  by auto
    then show ?thesis using 0  $\sigma' \langle \psi' \in \text{set } \sigma \rangle \langle \psi' = (\Delta', \delta') \rangle$  by
fastforce

  qed
  hence  $n > \text{length } (\text{fst } \psi')$  using  $\langle \psi' = (\Delta', \delta') \rangle$  by fastforce
}
then show ?case by fastforce
qed
}
hence  $\forall \sigma \in \text{set } (?T_\Sigma ?\Psi'). \forall (\Delta', \delta') \in \text{set } \sigma. \text{length } \Delta > \text{length } \Delta'$ 
  using  $\langle \forall (\Delta', \delta') \in \text{set } ?\Psi'. \text{length } \Delta > \text{length } \Delta' \rangle$ 
  by blast
then show ?thesis using True  $\star \psi(1)$  by fastforce
next
case False
have  $\forall (\Delta', \delta') \in \text{set } ?\Psi'. \text{length } \Delta \geq \text{length } \Delta'$ 
  using  $\langle \forall (\Delta', \delta') \in \text{set } ?\Psi'. \text{length } \Delta > \text{length } \Delta' \rangle$ 
  by auto
hence  $\forall (\Delta', \delta') \in \text{set } \Psi. \text{length } \Delta \geq \text{length } \Delta'$ 
  using  $\langle \Psi = (\Delta, \delta) \# ?\Psi' \rangle$ 

```



```

hence  $\forall \sigma \in \text{set } (?T_\Sigma \Psi_0). \text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi)$ 
proof (induct  $\Psi_0$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\psi \Psi_0$ )
  hence  $\forall \sigma \in \text{set } (?T_\Sigma \Psi_0). \text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi)$  by auto
  obtain  $\Delta \delta$  where  $\psi = (\Delta, \delta)$  by fastforce
  hence  $(\Delta, \delta) \in \text{set } (\mathfrak{V} \Xi)$  using Cons by simp
  {
    fix  $\sigma :: ('a \text{ list} \times 'a) \text{ list}$ 
    assume  $\star: \sigma \in (\#) (\Delta, \delta) \text{ ' set } (?T_\Sigma \Psi_0) \cup (@) (\mathfrak{V} \Delta) \text{ ' set } (?T_\Sigma \Psi_0)$ 
    have  $\text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi)$ 
    proof (cases  $\sigma \in (\#) (\Delta, \delta) \text{ ' set } (?T_\Sigma \Psi_0)$ )
      case True
      then show ?thesis
        using  $\langle \forall \sigma \in \text{set } (?T_\Sigma \Psi_0). \text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi) \rangle \langle (\Delta, \delta) \in \text{set } (\mathfrak{V} \Xi) \rangle$ 
        by fastforce
      next
      case False
      hence  $\sigma \in (@) (\mathfrak{V} \Delta) \text{ ' set } (?T_\Sigma \Psi_0)$  using  $\star$  by simp
      moreover have  $\text{set } (\mathfrak{V} \Delta) \subseteq \text{set } (\mathfrak{V} \Xi)$ 
      using MaxSAT-optimal-pre-witness-element-inclusion  $\langle (\Delta, \delta) \in \text{set } (\mathfrak{V} \Xi) \rangle$ 
      by fastforce
      ultimately show ?thesis
        using  $\langle \forall \sigma \in \text{set } (?T_\Sigma \Psi_0). \text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi) \rangle$ 
        by force
    }
    qed
  }
  hence  $\forall \sigma \in (\#) (\Delta, \delta) \text{ ' set } (?T_\Sigma \Psi_0) \cup (@) (\mathfrak{V} \Delta) \text{ ' set } (?T_\Sigma \Psi_0). \text{set } \sigma \subseteq \text{set } (\mathfrak{V} \Xi)$ 
  by auto
  thus ?case using  $\langle \psi = (\Delta, \delta) \rangle$  by simp
qed
hence  $\forall \sigma \in \text{set } (?T_\Sigma \Psi_0). \text{mset } (\text{remdups } \sigma) \subseteq\# \text{mset } (\text{remdups } (\mathfrak{V} \Xi))$ 
using mset-remdups-set-sub-iff by blast
hence  $\forall \sigma \in \text{set } ?\Sigma. \text{mset } \sigma \subseteq\# \text{mset } (\text{map } \text{snd } (\mathfrak{V} \Xi))$ 
using map-monotonic  $\langle \text{remdups } (\mathfrak{V} \Xi) = \mathfrak{V} \Xi \rangle$ 
by auto
moreover have  $\text{map } \text{snd } (\mathfrak{V} \Xi) = \Xi$  by (induct  $\Xi$ , simp+)
ultimately show ?thesis by simp
qed
show ?thesis using I II III by fastforce
qed
from this obtain  $\Sigma_0$  where  $\Sigma_0$ :
 $\vdash (\Psi \rightarrow \varphi) \leftrightarrow (\bigsqcup (\text{map } \sqcap \Sigma_0) \rightarrow \varphi)$ 
 $\forall \sigma \in \text{set } \Sigma_0. \text{mset } \sigma \subseteq\# \text{mset } \Xi \wedge \text{length } \sigma + 1 \geq \text{length } \Psi$ 
by blast

```

**moreover**  
**have**  $(\Phi @ \Psi) \rightarrow \varphi = \Phi \rightarrow (\Psi \rightarrow \varphi)$  **by** *(induct  $\Phi$ , simp+)*  
**hence**  $\vdash ((\Phi @ \Psi) \rightarrow \varphi) \leftrightarrow (\prod \Phi \rightarrow (\Psi \rightarrow \varphi))$   
**by** *(simp add: list-curry-uncurry)*  
**moreover have**  $\vdash (\Psi \rightarrow \varphi) \leftrightarrow (\bigsqcup (\text{map } \prod \Sigma_0) \rightarrow \varphi)$   
 $\rightarrow (\Phi @ \Psi) \rightarrow \varphi \leftrightarrow (\prod \Phi \rightarrow \Psi \rightarrow \varphi)$   
 $\rightarrow (\Phi @ \Psi) \rightarrow \varphi \leftrightarrow ((\prod \Phi \sqcap \bigsqcup (\text{map } \prod \Sigma_0)) \rightarrow \varphi)$   
**proof –**  
**let**  $?\varphi = \langle \Psi \rightarrow \varphi \rangle \leftrightarrow (\langle \bigsqcup (\text{map } \prod \Sigma_0) \rangle \rightarrow \langle \varphi \rangle)$   
 $\rightarrow \langle (\Phi @ \Psi) \rightarrow \varphi \rangle \leftrightarrow (\langle \prod \Phi \rangle \rightarrow \langle \Psi \rightarrow \varphi \rangle)$   
 $\rightarrow \langle (\Phi @ \Psi) \rightarrow \varphi \rangle \leftrightarrow ((\langle \prod \Phi \rangle \sqcap \langle \bigsqcup (\text{map } \prod \Sigma_0) \rangle) \rightarrow \langle \varphi \rangle)$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ?\varphi$  **by** *fastforce*  
**hence**  $\vdash (\langle ?\varphi \rangle)$  **using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**moreover**  
**let**  $?\Sigma = \text{map } ((@) \Phi) \Sigma_0$   
**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**  $\vdash ((\prod \Phi \sqcap \bigsqcup (\text{map } \prod \Sigma_0)) \rightarrow \varphi) \leftrightarrow (\bigsqcup (\text{map } \prod ?\Sigma) \rightarrow \varphi)$   
**using** *append-dnf-distribute biconditional-def* **by** *fastforce*  
**ultimately have**  $\vdash (\Phi @ \Psi) \rightarrow \varphi \leftrightarrow (\bigsqcup (\text{map } \prod ?\Sigma) \rightarrow \varphi)$   
**using** *modus-ponens biconditional-transitivity-rule*  
**by** *blast*  
**moreover**  
**{**  
**fix**  $\sigma$   
**assume**  $\sigma \in \text{set } ?\Sigma$   
**from** *this* **obtain**  $\sigma_0$  **where**  $\sigma_0: \sigma = \Phi @ \sigma_0$   $\sigma_0 \in \text{set } \Sigma_0$  **by** *(simp, blast)*  
**hence**  $\text{mset } \sigma_0 \subseteq\# \text{mset } \Xi$  **using**  $\Sigma_0(2)$  **by** *blast*  
**hence**  $\text{mset } \sigma \subseteq\# \text{mset } (\Phi @ \Xi)$  **using**  $\sigma_0(1)$  **by** *simp*  
**hence**  $\text{mset } \sigma \subseteq\# \text{mset } \Gamma$  **using** *assms(1) assms(2)*  
**by** *(simp, meson subset-mset.dual-order.trans subset-mset.le-diff-conv2)*  
**moreover**  
**have**  $\text{length } \sigma + 1 \geq \text{length } (\Phi @ \Psi)$  **using**  $\Sigma_0(2)$   $\sigma_0$  **by** *simp*  
**ultimately have**  $\text{mset } \sigma \subseteq\# \text{mset } \Gamma$   $\text{length } \sigma + 1 \geq \text{length } (\Phi @ \Psi)$  **by** *auto*  
**}**  
**ultimately**  
**show** *?thesis* **by** *blast*  
**qed**

**lemma (in classical-logic) relative-maximals-optimal-witness:**  
**assumes**  $\neg \vdash \varphi$   
**shows**  $0 < (\| \Gamma \|_\varphi)$   
 $= (\exists \Sigma. \text{mset } (\text{map } \text{snd } \Sigma) \subseteq\# \text{mset } \Gamma \wedge$   
 $\text{map } (\text{uncurry } (\sqcup)) \Sigma \vdash \varphi \wedge$   
 $1 + (\| \text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus \text{map } \text{snd } \Sigma \|_\varphi) = \| \Gamma \|_\varphi)$   
**proof (rule iffI)**  
**assume**  $0 < \| \Gamma \|_\varphi$

```

from this obtain  $\Xi$  where  $\Xi: \Xi \in \mathcal{M} \Gamma \varphi$  length  $\Xi < \text{length } \Gamma$ 
  using  $\langle \neg \vdash \varphi \rangle$ 
    complement-relative-MaxSAT-def
    relative-MaxSAT-intro
    relative-maximals-existence
  by fastforce
from this obtain  $\psi$  where  $\psi: \psi \in \text{set } (\Gamma \ominus \Xi)$ 
  by (metis  $\langle 0 < \|\Gamma\|_{\varphi} \rangle$ )
    less-not-refl
    list.exhaust
    list.set-intros(1)
    list.size(3)
    complement-relative-MaxSAT-intro
let  $? \Sigma = \mathfrak{W}_{\varphi} (\psi \# \Xi)$ 
let  $? \Sigma_A = \mathfrak{W}_{\sqcup} \varphi (\psi \# \Xi)$ 
let  $? \Sigma_B = \mathfrak{W}_{\rightarrow} \varphi (\psi \# \Xi)$ 
have  $\diamond: \text{mset } (\psi \# \Xi) \subseteq \# \text{mset } \Gamma$ 
   $\psi \# \Xi \vdash \varphi$ 
  using  $\Xi(1) \psi$ 
    relative-maximals-def
    list-deduction-theorem
    relative-maximals-complement-deduction
    mset-list-subtract-elem-cons-mset [where  $\Xi = \Xi$ ]
  by blast+
moreover have map snd  $? \Sigma = \psi \# \Xi$  by (induct  $\Xi$ , simp+)
ultimately have  $? \Sigma_A \vdash \varphi$ 
  mset (map snd ?Σ) ⊆# mset Γ
  using MaxSAT-optimal-witness-deduction
    list-deduction-def weak-biconditional-weaken
  by (metis+)
moreover
{
  let  $? \Gamma' = ? \Sigma_B @ \Gamma \ominus \text{map snd } ? \Sigma$ 
  have  $A: \text{length } ? \Sigma_B = 1 + \text{length } \Xi$ 
    by (induct  $\Xi$ , simp+)
  have  $B: ? \Sigma_B \in \mathcal{M} ? \Gamma' \varphi$ 
  proof –
    have  $\neg ? \Sigma_B \vdash \varphi$ 
      by (metis (no-types, lifting))
         $\Xi(1) \langle ? \Sigma_A \vdash \varphi \rangle$ 
        modus-ponens list-deduction-def
        optimal-witness-split-identity
        relative-maximals-def
        mem-Collect-eq
    moreover have mset  $? \Sigma_B \subseteq \# \text{mset } ? \Gamma'$ 
      by simp
    hence  $\forall \Psi. \text{mset } \Psi \subseteq \# \text{mset } ? \Gamma' \longrightarrow \neg \Psi \vdash \varphi \longrightarrow \text{length } \Psi \leq \text{length } ? \Sigma_B$ 
    proof –
      have  $\forall \Psi \in \mathcal{M} ? \Gamma' \varphi. \text{length } \Psi = \text{length } ? \Sigma_B$ 

```



```

proof (rule ccontr)
  assume  $\neg (\forall \Psi \in \mathcal{M} \ ?\Gamma' \ \varphi. \text{length } \Psi = \text{length } ?\Sigma_B)$ 
  from this obtain  $\Psi$  where
     $\Psi: \Psi \in \mathcal{M} \ ?\Gamma' \ \varphi$ 
     $\text{length } \Psi \neq \text{length } ?\Sigma_B$ 
  by blast
  have  $\text{length } \Psi \geq \text{length } ?\Sigma_B$ 
  using  $\Psi(1)$ 
     $\langle \neg ?\Sigma_B \text{ :+ } \varphi \rangle$ 
     $\langle \text{mset } ?\Sigma_B \subseteq\# \text{mset } ?\Gamma' \rangle$ 
  unfolding relative-maximals-def
  by blast
  hence  $\text{length } \Psi > \text{length } ?\Sigma_B$ 
  using  $\Psi(2)$ 
  by linarith
  have  $\text{length } \Psi = \text{length } (\Psi \ominus ?\Sigma_B) + \text{length } (\Psi \cap ?\Sigma_B)$ 
  (is  $\text{length } \Psi = \text{length } ?A + \text{length } ?B$ )
  by (metis (no-types, lifting)
    length-append
    list-diff-intersect-comp
    mset-append
    mset-eq-length)
  {
    fix  $\sigma$ 
    assume  $\text{mset } \sigma \subseteq\# \text{mset } \Gamma$ 
     $\text{length } \sigma + 1 \geq \text{length } (?A @ ?B)$ 
    hence  $\text{length } \sigma + 1 \geq \text{length } \Psi$ 
    using  $\langle \text{length } \Psi = \text{length } ?A + \text{length } ?B \rangle$ 
    by simp
    hence  $\text{length } \sigma + 1 > \text{length } ?\Sigma_B$ 
    using  $\langle \text{length } \Psi > \text{length } ?\Sigma_B \rangle$  by linarith
    hence  $\text{length } \sigma + 1 > \text{length } \Xi + 1$ 
    using  $A$  by simp
    hence  $\text{length } \sigma > \text{length } \Xi$  by linarith
    have  $\sigma \text{ :+ } \varphi$ 
    proof (rule ccontr)
      assume  $\neg \sigma \text{ :+ } \varphi$ 
      hence  $\text{length } \sigma \leq \text{length } \Xi$ 
      using  $\langle \text{mset } \sigma \subseteq\# \text{mset } \Gamma \rangle \Xi(1)$ 
      unfolding relative-maximals-def
      by blast
      thus False using  $\langle \text{length } \sigma > \text{length } \Xi \rangle$  by linarith
    qed
  }
  moreover
  have  $\text{mset } \Psi \subseteq\# \text{mset } ?\Gamma'$ 
     $\neg \Psi \text{ :+ } \varphi$ 
     $\forall \Phi. \text{mset } \Phi \subseteq\# \text{mset } ?\Gamma' \wedge \neg \Phi \text{ :+ } \varphi \longrightarrow \text{length } \Phi \leq \text{length } \Psi$ 
  using  $\Psi(1)$  relative-maximals-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**  $\langle map \ snd \ ?\Sigma = \psi \# \Xi \rangle$  **by** *metis*  
**moreover**  
**have**  $mset \ ?B \subseteq\# mset (\mathfrak{W} \rightarrow \varphi (\psi \# \Xi))$   
**using** *list-intersect-right-project* **by** *blast*  
**ultimately obtain**  $\Sigma$  **where**  $\Sigma: \vdash ((?A \ @ \ ?B) : \rightarrow \varphi) \leftrightarrow (\bigsqcup (map \ \sqcap \ \Sigma)$   
 $\rightarrow \varphi)$   
 $\forall \sigma \in set \ \Sigma. \ \sigma : \vdash \varphi$   
**using**  $\diamond$  *optimal-witness-list-intersect-biconditional*  
**by** *metis*  
**hence**  $\vdash \bigsqcup (map \ \sqcap \ \Sigma) \rightarrow \varphi$   
**using** *weak-disj-of-conj-equiv* **by** *blast*  
**hence**  $?A \ @ \ ?B : \vdash \varphi$   
**using**  $\Sigma(1)$  *modus-ponens list-deduction-def weak-biconditional-weaken*  
**by** *blast*  
**moreover have**  $set \ (?A \ @ \ ?B) = set \ \Psi$   
**using** *list-diff-intersect-comp union-code set-mset-mset* **by** *metis*  
**hence**  $?A \ @ \ ?B : \vdash \varphi = \Psi : \vdash \varphi$   
**using** *list-deduction-monotonic* **by** *blast*  
**ultimately have**  $\Psi : \vdash \varphi$  **by** *metis*  
**thus** *False* **using**  $\Psi(1)$  *unfolding relative-maximals-def* **by** *blast*  
**qed**  
**moreover have**  $\exists \ \Psi. \ \Psi \in \mathcal{M} \ \Gamma' \ \varphi$   
**using** *assms relative-maximals-existence* **by** *blast*  
**ultimately show** *?thesis*  
**using** *relative-maximals-def*  
**by** *fastforce*  
**qed**  
**ultimately show** *?thesis*  
**unfolding** *relative-maximals-def*  
**by** *fastforce*  
**qed**  
**have**  $C: \forall \ \Xi \ \Gamma \ \varphi. \ \Xi \in \mathcal{M} \ \Gamma \ \varphi \longrightarrow length \ \Xi = | \ \Gamma \ |_{\varphi}$   
**using** *relative-MaxSAT-intro* **by** *blast*  
**then have**  $D: length \ \Xi = | \ \Gamma \ |_{\varphi}$   
**using**  $\langle \Xi \in \mathcal{M} \ \Gamma \ \varphi \rangle$  **by** *blast*  
**have**  
 $\forall (\Sigma :: 'a \ list) \ \Gamma \ n. (\neg mset \ \Sigma \subseteq\# mset \ \Gamma \vee length \ (\Gamma \ominus \Sigma) \neq n) \vee length \ \Gamma$   
 $= n + length \ \Sigma$   
**using** *list-subtract-msub-eq* **by** *blast*  
**then have**  $E: length \ \Gamma = length \ (\Gamma \ominus map \ snd \ (\mathfrak{W} \ \varphi \ (\psi \# \Xi))) + length \ (\psi$   
 $\# \Xi)$   
**using**  $\langle map \ snd \ (\mathfrak{W} \ \varphi \ (\psi \# \Xi)) = \psi \# \Xi \rangle$   $\langle mset \ (\psi \# \Xi) \subseteq\# mset \ \Gamma \rangle$  **by**  
*presburger*  
**have**  $1 + length \ \Xi = | \ \mathfrak{W} \rightarrow \varphi \ (\psi \# \Xi) \ @ \ \Gamma \ominus map \ snd \ (\mathfrak{W} \ \varphi \ (\psi \# \Xi)) \ |_{\varphi}$   
**using** *C B A* **by** *presburger*  
**hence**  $1 + (|| map \ (uncurry \ (\rightarrow)) \ ?\Sigma \ @ \ \Gamma \ominus map \ snd \ ?\Sigma ||_{\varphi}) = || \ \Gamma \ ||_{\varphi}$

```

    using D E ⟨map snd (ℳ φ (ψ # Ξ)) = ψ # Ξ⟩ complement-relative-MaxSAT-def
  by force
}
ultimately
show ∃ Σ. mset (map snd Σ) ⊆# mset Γ ∧
    map (uncurry (⊔)) Σ ⊢ φ ∧
    1 + (|| map (uncurry (→)) Σ @ Γ ⊖ map snd Σ ||φ) = || Γ ||φ
  by metis
next
assume ∃ Σ. mset (map snd Σ) ⊆# mset Γ ∧
    map (uncurry (⊔)) Σ ⊢ φ ∧
    1 + (|| map (uncurry (→)) Σ @ Γ ⊖ map snd Σ ||φ) = || Γ ||φ
thus 0 < || Γ ||φ
  by auto
qed

```

```

primrec (in implication-logic)
MaxSAT-witness :: ('a × 'a) list ⇒ 'a list ⇒ ('a × 'a) list (ℳ)
where
  ℳ - [] = []
  | ℳ Σ (ξ # Ξ) = (case find (λ σ. ξ = snd σ) Σ of
    None ⇒ ℳ Σ Ξ
  | Some σ ⇒ σ # (ℳ (remove1 σ Σ) Ξ))

```

**lemma** (in implication-logic) *MaxSAT-witness-right-msub*:

```

mset (map snd (ℳ Σ Ξ)) ⊆# mset Ξ
proof -
  have ∀ Σ. mset (map snd (ℳ Σ Ξ)) ⊆# mset Ξ
  proof (induct Ξ)
    case Nil
    then show ?case by simp
  next
    case (Cons ξ Ξ)
    {
      fix Σ
      have mset (map snd (ℳ Σ (ξ # Ξ))) ⊆# mset (ξ # Ξ)
      proof (cases find (λ σ. ξ = snd σ) Σ)
        case None
        then show ?thesis
          by (simp, metis Cons.hyps
            add-mset-add-single
            mset-map mset-subset-eq-add-left subset-mset.order-trans)
      next
        case (Some σ)
        note σ = this
        hence ξ = snd σ
          by (meson find-Some-predicate)
        moreover

```

```

    have  $\sigma \in \text{set } \Sigma$ 
    using  $\sigma$ 
    proof (induct  $\Sigma$ )
      case Nil
      then show ?case by simp
    next
      case (Cons  $\sigma' \Sigma$ )
      then show ?case
        by (cases  $\xi = \text{snd } \sigma'$ , simp+)
    qed
    ultimately show ?thesis using  $\sigma$  Cons.hyps by simp
  qed
}
then show ?case by simp
qed
thus ?thesis by simp
qed

lemma (in implication-logic) MaxSAT-witness-left-msub:
  mset ( $\mathcal{U} \Sigma \Xi$ )  $\subseteq\#$  mset  $\Sigma$ 
proof -
  have  $\forall \Sigma. \text{mset } (\mathcal{U} \Sigma \Xi) \subseteq\# \text{mset } \Sigma$ 
  proof (induct  $\Xi$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\xi \Xi$ )
    {
      fix  $\Sigma$ 
      have  $\text{mset } (\mathcal{U} \Sigma (\xi \# \Xi)) \subseteq\# \text{mset } \Sigma$ 
      proof (cases find ( $\lambda \sigma. \xi = \text{snd } \sigma$ )  $\Sigma$ )
        case None
        then show ?thesis using Cons.hyps by simp
      next
        case (Some  $\sigma$ )
        note  $\sigma = \text{this}$ 
        hence  $\sigma \in \text{set } \Sigma$ 
        proof (induct  $\Sigma$ )
          case Nil
          then show ?case by simp
        next
          case (Cons  $\sigma' \Sigma$ )
          then show ?case
            by (cases  $\xi = \text{snd } \sigma'$ , simp+)
        qed
      moreover from Cons.hyps have  $\text{mset } (\mathcal{U} (\text{remove1 } \sigma \Sigma) \Xi) \subseteq\# \text{mset } (\text{remove1 } \sigma \Sigma)$ 
      by blast
      hence  $\text{mset } (\mathcal{U} \Sigma (\xi \# \Xi)) \subseteq\# \text{mset } (\sigma \# \text{remove1 } \sigma \Sigma)$  using  $\sigma$  by simp
    }
  qed

```

```

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

lemma (in implication-logic) MaxSAT-witness-right-projection:
  mset (map snd (⋈ Σ Ξ)) = mset ((map snd Σ) ∩ Ξ)
proof -
  have ∀ Σ. mset (map snd (⋈ Σ Ξ)) = mset ((map snd Σ) ∩ Ξ)
  proof (induct Ξ)
    case Nil
    then show ?case by simp
  next
    case (Cons ξ Ξ)
    {
      fix Σ
      have mset (map snd (⋈ Σ (ξ # Ξ))) = mset (map snd Σ ∩ ξ # Ξ)
      proof (cases find (λ σ. ξ = snd σ) Σ)
        case None
        hence ξ ∉ set (map snd Σ)
        proof (induct Σ)
          case Nil
          then show ?case by simp
        next
          case (Cons σ Σ)
          have find (λσ. ξ = snd σ) Σ = None
            ξ ≠ snd σ
          using Cons.premis
          by (auto, metis Cons.premis 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 σ ∈ set Σ ξ = snd σ
    by (meson find-Some-predicate find-Some-set-membership)+
    moreover
    from ⟨σ ∈ set Σ⟩ have mset Σ = mset (σ # (remove1 σ Σ))
    by simp
    hence mset (map snd Σ) = mset ((snd σ) # (remove1 (snd σ) (map snd
Σ)))
      mset (map snd Σ) = mset (map snd (σ # (remove1 σ Σ)))
    by (simp add: ⟨σ ∈ set Σ⟩, metis map-monotonic subset-mset.eq-iff)
    hence mset (map snd (remove1 σ Σ)) = mset (remove1 (snd σ) (map snd
Σ))
    by simp

```

```

    ultimately show ?thesis using Some Cons.hyps by simp
  qed
}
then show ?case by simp
qed
thus ?thesis by simp
qed

```

**lemma** (in *classical-logic*) *witness-list-implication-rule*:

$\vdash (\text{map } (\text{uncurry } (\sqcup)) \Sigma \text{ :} \rightarrow \varphi) \rightarrow \prod (\text{map } (\lambda (\chi, \xi). (\chi \rightarrow \xi) \rightarrow \varphi) \Sigma) \rightarrow \varphi$

**proof** (*induct*  $\Sigma$ )

case *Nil*

then show ?case using *axiom-k* by simp

next

case (*Cons*  $\sigma$   $\Sigma$ )

let  $? \chi = \text{fst } \sigma$

let  $? \xi = \text{snd } \sigma$

let  $? \Sigma_A = \text{map } (\text{uncurry } (\sqcup)) \Sigma$

let  $? \Sigma_B = \text{map } (\lambda (\chi, \xi). (\chi \rightarrow \xi) \rightarrow \varphi) \Sigma$

assume  $\vdash ? \Sigma_A \text{ :} \rightarrow \varphi \rightarrow \prod ? \Sigma_B \rightarrow \varphi$

moreover have

$\vdash (? \Sigma_A \text{ :} \rightarrow \varphi \rightarrow \prod ? \Sigma_B \rightarrow \varphi)$

$\rightarrow ((? \chi \sqcup ? \xi) \rightarrow ? \Sigma_A \text{ :} \rightarrow \varphi) \rightarrow (((? \chi \rightarrow ? \xi) \rightarrow \varphi) \sqcap \prod ? \Sigma_B) \rightarrow \varphi$

**proof** –

let  $? \varphi = (\langle ? \Sigma_A \text{ :} \rightarrow \varphi \rangle \rightarrow \langle \prod ? \Sigma_B \rangle \rightarrow \langle \varphi \rangle)$

$\rightarrow (((\langle ? \chi \rangle \sqcup \langle ? \xi \rangle) \rightarrow \langle ? \Sigma_A \text{ :} \rightarrow \varphi \rangle) \rightarrow (((\langle ? \chi \rangle \rightarrow \langle ? \xi \rangle) \rightarrow \langle \varphi \rangle) \sqcap \langle \prod$

$? \Sigma_B \rangle) \rightarrow \langle \varphi \rangle)$

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

hence  $\vdash (\langle ? \varphi \rangle)$  using *propositional-semantic* by *blast*

thus ?thesis by *simp*

qed

ultimately have  $\vdash ((? \chi \sqcup ? \xi) \rightarrow ? \Sigma_A \text{ :} \rightarrow \varphi) \rightarrow (((? \chi \rightarrow ? \xi) \rightarrow \varphi) \sqcap \prod ? \Sigma_B) \rightarrow \varphi$

using *modus-ponens* by *blast*

moreover

have  $(\lambda \sigma. (\text{fst } \sigma \rightarrow \text{snd } \sigma) \rightarrow \varphi) = (\lambda (\chi, \xi). (\chi \rightarrow \xi) \rightarrow \varphi)$

$\text{uncurry } (\sqcup) = (\lambda \sigma. \text{fst } \sigma \sqcup \text{snd } \sigma)$

by *fastforce+*

hence  $(\lambda (\chi, \xi). (\chi \rightarrow \xi) \rightarrow \varphi) \sigma = (? \chi \rightarrow ? \xi) \rightarrow \varphi$

$\text{uncurry } (\sqcup) \sigma = ? \chi \sqcup ? \xi$

by *metis+*

ultimately show ?case by *simp*

qed

**lemma** (in *classical-logic*) *witness-relative-MaxSAT-increase*:

assumes  $\neg \vdash \varphi$

and  $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } \Gamma$

and  $\text{map } (\text{uncurry } (\sqcup)) \Sigma \text{ :} \vdash \varphi$

shows  $(|\Gamma|_{\varphi}) < (|\text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus \text{map } \text{snd } \Sigma|_{\varphi})$

**proof** –

**from**  $\langle \neg \vdash \varphi \rangle$  **obtain**  $\Xi$  **where**  $\Xi: \Xi \in \mathcal{M} \Gamma \varphi$   
**using** *relative-maximals-existence* **by** *blast*  
**let**  $? \Sigma' = \Sigma \ominus \mathcal{U} \Sigma \Xi$   
**let**  $? \Sigma \Xi' = \text{map} (\text{uncurry} (\sqcup)) (\mathcal{U} \Sigma \Xi) @ \text{map} (\text{uncurry} (\rightarrow)) (\mathcal{U} \Sigma \Xi)$   
**have**  $\text{mset} \Sigma = \text{mset} (\mathcal{U} \Sigma \Xi @ ? \Sigma')$  **by** (*simp add: MaxSAT-witness-left-msub*)  
**hence**  $\text{set} (\text{map} (\text{uncurry} (\sqcup)) \Sigma) = \text{set} (\text{map} (\text{uncurry} (\sqcup)) ((\mathcal{U} \Sigma \Xi) @ ? \Sigma'))$   
**by** (*metis mset-map mset-eq-setD*)  
**hence**  $\text{map} (\text{uncurry} (\sqcup)) ((\mathcal{U} \Sigma \Xi) @ ? \Sigma') \vdash \varphi$   
**using** *list-deduction-monotonic assms(β)*  
**by** *blast*  
**hence**  $\text{map} (\text{uncurry} (\sqcup)) (\mathcal{U} \Sigma \Xi) @ \text{map} (\text{uncurry} (\sqcup)) ? \Sigma' \vdash \varphi$  **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**  $\text{map} (\text{uncurry} (\sqcup)) (\mathcal{U} \Sigma \Xi) \vdash \text{map} (\text{uncurry} (\sqcup)) ? \Sigma' \rightarrow \varphi$   
**by** *simp*  
**moreover have**  $\text{set} (\text{map} (\text{uncurry} (\sqcup)) (\mathcal{U} \Sigma \Xi)) \subseteq \text{set} ? \Sigma \Xi'$   
**by** *simp*  
**ultimately have**  $? \Sigma \Xi' \vdash \text{map} (\text{uncurry} (\sqcup)) ? \Sigma' \rightarrow \varphi$   
**using** *list-deduction-monotonic by blast*  
**hence**  $? \Sigma \Xi' \vdash \prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi$   
**using** *list-deduction-modus-ponens*  
*list-deduction-weaken*  
*witness-list-implication-rule*  
**by** *blast*  
**hence**  $? \Sigma \Xi' \$\vdash [\prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi]$   
**using** *measure-deduction-one-collapse by metis*  
**hence**  
 $? \Sigma \Xi' @ (\text{map} \text{snd} (\mathcal{U} \Sigma \Xi)) \ominus (\text{map} \text{snd} (\mathcal{U} \Sigma \Xi))$   
 $\$ \vdash [\prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi]$   
**by** *simp*  
**hence**  $\text{map} \text{snd} (\mathcal{U} \Sigma \Xi) \$ \vdash [\prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi]$   
**using** *measure-witness-left-split [where Γ=map snd (U Σ Ξ)*  
**and**  $\Sigma = \mathcal{U} \Sigma \Xi]$   
**by** *fastforce*  
**hence**  $\text{map} \text{snd} (\mathcal{U} \Sigma \Xi) \$ \vdash [\prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi]$   
**using** *MaxSAT-witness-right-projection by auto*  
**hence**  $\text{map} \text{snd} (\mathcal{U} \Sigma \Xi) \vdash \prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi) ? \Sigma') \rightarrow \varphi$   
**using** *measure-deduction-one-collapse by blast*  
**hence**  $\star:$   
 $\text{map} \text{snd} (\mathcal{U} \Sigma \Xi) @ \Xi \ominus (\text{map} \text{snd} \Sigma) \vdash \prod (\text{map} (\lambda (\chi, \gamma). (\chi \rightarrow \gamma) \rightarrow \varphi)$   
 $? \Sigma') \rightarrow \varphi$

```

(is ?E0 :| -)
using list-deduction-monotonic
by (metis (no-types, lifting) append-Nil2
      measure-cancel
      measure-deduction.simps(1)
      measure-list-deduction-antitonic)
have mset E = mset (E ⊖ (map snd Σ)) + mset (E ∩ (map snd Σ))
  using list-diff-intersect-comp by blast
hence mset E = mset ((map snd Σ) ∩ E) + mset (E ⊖ (map snd Σ))
  by (metis subset-mset.inf-commute list-intersect-mset-homomorphism union-commute)
hence mset E = mset (map snd (⋈ Σ E)) + mset (E ⊖ (map snd Σ))
  using MaxSAT-witness-right-projection by simp
hence mset E = mset ?E0
  by simp
hence set E = set ?E0
  by (metis mset-eq-setD)
have ¬ ?E0 :| ⌈ (map (λ (χ, γ). (χ → γ) → φ) ?Σ')
proof (rule notI)
  assume ?E0 :| ⌈ (map (λ (χ, γ). (χ → γ) → φ) ?Σ')
  hence ?E0 :| φ
    using * list-deduction-modus-ponens by blast
  hence E :| φ
    using list-deduction-monotonic ⟨set E = set ?E0⟩ by blast
  thus False
    using E relative-maximals-def by blast
qed
moreover
have mset (map snd (⋈ Σ E)) ⊆# mset ?E0
  mset (map (uncurry (→)) (⋈ Σ E) @ ?E0 ⊖ map snd (⋈ Σ E))
  = mset (map (uncurry (→)) (⋈ Σ E) @ E ⊖ (map snd Σ))
  (is - = mset ?E1)
  by auto
hence ?E1 ⪯ ?E0
  by (metis add.commute
      witness-stronger-theory
      add-diff-cancel-right'
      list-subtract.simps(1)
      list-subtract-mset-homomorphism
      list-diff-intersect-comp
      list-intersect-right-project
      msub-stronger-theory-intro
      stronger-theory-combine
      stronger-theory-empty-list-intro
      self-append-conv)
ultimately have
  ¬ ?E1 :| ⌈ (map (λ (χ, γ). (χ → γ) → φ) ?Σ')
  using stronger-theory-deduction-monotonic by blast
from this obtain χ γ where
  (χ, γ) ∈ set ?Σ'

```



```

  ¬ (χ → γ) # ?Ξ₁ :⊢ φ
  using list-deduction-theorem
  by fastforce
  have mset (χ → γ # ?Ξ₁) ⊆# mset (map (uncurry (→)) Σ @ Γ ⊖ map snd Σ)
  proof -
    let ?A = map (uncurry (→)) Σ
    let ?B = map (uncurry (→)) (⋈ Σ Ξ)
    have (χ, γ) ∈ (set Σ - set (⋈ Σ Ξ))
    proof -
      from ⟨(χ, γ) ∈ set ?Σ'⟩ have γ ∈# mset (map snd (Σ ⊖ ⋈ Σ Ξ))
      by (metis set-mset-mset image-eqI set-map snd-conv)
      hence γ ∈# mset (map snd Σ ⊖ map snd (⋈ Σ Ξ))
      by (metis MaxSAT-witness-left-msub map-list-subtract-mset-equivalence)
      hence γ ∈# mset (map snd Σ ⊖ (map snd Σ ∩ Ξ))
      by (metis MaxSAT-witness-right-projection list-subtract-mset-homomorphism)
      hence γ ∈# mset (map snd Σ ⊖ Ξ)
      by (metis add-diff-cancel-right'
          list-subtract-mset-homomorphism
          list-diff-intersect-comp)
      moreover from assms(2) have mset (map snd Σ ⊖ Ξ) ⊆# mset (Γ ⊖ Ξ)
      by (simp, metis list-subtract-monotonic list-subtract-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 (⋈ Σ Ξ)
    proof (induct Ξ)
      case Nil
      then show ?case by simp
    next
      case (Cons ξ Ξ)
      {
        fix Σ
        have (χ, γ) ∉ set (⋈ 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.premis have σ ≠ (χ, γ) by fastforce
          ultimately show ?thesis using Cons by simp
        qed
      }
  qed

```

```

    }
    then show ?case by blast
qed
moreover from ⟨(χ, γ) ∈ set ?Σ'⟩ have (χ, γ) ∈ set Σ
  by (meson list-subtract-set-trivial-upper-bound subsetCE)
ultimately show ?thesis by fastforce
qed
with ⟨(χ, γ) ∈ set ?Σ'⟩ have mset ((χ, γ) #  $\mathfrak{A}$  Σ  $\Xi$ )  $\subseteq$ # mset Σ
  by (meson MaxSAT-witness-left-msub msub-list-subtract-elem-cons-msub)
hence mset (χ → γ # ?B)  $\subseteq$ # mset (map (uncurry (→)) Σ)
  by (metis (no-types, lifting)
      ⟨(χ, γ) ∈ set ?Σ'⟩
      MaxSAT-witness-left-msub
      map-list-subtract-mset-equivalence
      map-monotonic
      mset-eq-setD msub-list-subtract-elem-cons-msub
      pair-imageI
      set-map
      uncurry-def)
moreover
have mset  $\Xi$   $\subseteq$ # mset Γ
  using  $\Xi$  relative-maximals-def
  by blast
hence mset ( $\Xi$   $\ominus$  (map snd Σ))  $\subseteq$ # mset (Γ  $\ominus$  (map snd Σ))
  using list-subtract-monotonic by blast
ultimately show ?thesis
  using subset-mset.add-mono by fastforce
qed
moreover have length ? $\Xi_1$  = length ? $\Xi_0$ 
  by simp
hence length ? $\Xi_1$  = length  $\Xi$ 
  using ⟨mset  $\Xi$  = mset ? $\Xi_0$ ⟩ mset-eq-length
  by metis
hence length ((χ → γ) # ? $\Xi_1$ ) = length  $\Xi$  + 1
  by simp
hence length ((χ → γ) # ? $\Xi_1$ ) = (| Γ |φ) + 1
  using  $\Xi$ 
  by (simp add: relative-MaxSAT-intro)
moreover from ⟨ $\neg \vdash \varphi$ ⟩ obtain Ω where Ω: Ω ∈  $\mathcal{M}$  (map (uncurry (→)) Σ) @
Γ  $\ominus$  map snd Σ) φ
  using relative-maximals-existence by blast
ultimately have length Ω ≥ (| Γ |φ) + 1
  using relative-maximals-def
  by (metis (no-types, lifting) ⟨ $\neg \chi \rightarrow \gamma$  # ? $\Xi_1$  :+ φ⟩ mem-Collect-eq)
thus ?thesis
  using Ω relative-MaxSAT-intro by auto
qed

```

lemma (in classical-logic) relative-maximals-counting-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  $\Gamma$ 
    assume  $\Gamma \# \vdash (\text{Suc } n) \varphi$ 
    from this obtain  $\Sigma$  where  $\Sigma$ :
       $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } \Gamma$ 
       $\text{map } (\text{uncurry } (\sqcup)) \Sigma \vdash \varphi$ 
       $\text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus (\text{map } \text{snd } \Sigma) \# \vdash n \varphi$ 
    by fastforce
    let  $? \Gamma' = \text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus (\text{map } \text{snd } \Sigma)$ 
    have  $\text{length } \Gamma = \text{length } ? \Gamma'$ 
    using  $\Sigma(1)$  list-subtract-msub-eq by fastforce
    hence  $(\|\Gamma\|_\varphi) > (\|? \Gamma'\|_\varphi)$ 
    by (metis  $\Sigma(1)$   $\Sigma(2)$   $\langle \neg \vdash \varphi \rangle$ 
      witness-relative-MaxSAT-increase
      length-MaxSAT-decomposition
      add-less-cancel-right
      nat-add-left-cancel-less)
    with  $\Sigma(3)$  Suc.hyps have  $\text{Suc } n \leq \|\Gamma\|_\varphi$ 
    by auto
  }
moreover
  {
    fix  $\Gamma$ 
    assume  $\text{Suc } n \leq \|\Gamma\|_\varphi$ 
    from this obtain  $\Sigma$  where  $\Sigma$ :
       $\text{mset } (\text{map } \text{snd } \Sigma) \subseteq \# \text{mset } \Gamma$ 
       $\text{map } (\text{uncurry } (\sqcup)) \Sigma \vdash \varphi$ 
       $1 + (\|\text{map } (\text{uncurry } (\rightarrow)) \Sigma @ \Gamma \ominus \text{map } \text{snd } \Sigma\|_\varphi) = \|\Gamma\|_\varphi$ 
      (is  $1 + (\|? \Gamma'\|_\varphi) = \|\Gamma\|_\varphi$ )
    by (metis Suc-le-D assms relative-maximals-optimal-witness zero-less-Suc)
    have  $n \leq \|? \Gamma'\|_\varphi$ 
    using  $\Sigma(3)$   $\langle \text{Suc } n \leq \|\Gamma\|_\varphi \rangle$  by linarith
    hence  $? \Gamma' \# \vdash n \varphi$  using Suc by blast
    hence  $\Gamma \# \vdash (\text{Suc } n) \varphi$  using  $\Sigma(1)$   $\Sigma(2)$  by fastforce
  }
ultimately show ?case by metis
qed
thus ?thesis by auto
qed

```

As a brief aside, we may observe that  $\varphi$  is a tautology if and only if count-

ing deduction can prove it for any given number of times. This follows immediately from  $\neg \vdash \varphi \implies \Gamma \# \vdash n \varphi = (n \leq \|\Gamma\|_\varphi)$ .

**lemma** (in *classical-logic*) *counting-deduction-tautology-equiv*:

$(\forall n. \Gamma \# \vdash n \varphi) = \vdash \varphi$

**proof** (*cases*  $\vdash \varphi$ )

**case** *True*

**then show** *?thesis*

**by** (*simp add: counting-deduction-tautology-weaken*)

**next**

**case** *False*

**have**  $\neg \Gamma \# \vdash (1 + \text{length } \Gamma) \varphi$

**proof** (*rule notI*)

**assume**  $\Gamma \# \vdash (1 + \text{length } \Gamma) \varphi$

**hence**  $1 + \text{length } \Gamma \leq \|\Gamma\|_\varphi$

**using**  $\langle \neg \vdash \varphi \rangle$  *relative-maximals-counting-deduction-lower-bound* **by** *blast*

**hence**  $1 + \text{length } \Gamma \leq \text{length } \Gamma$

**using** *complement-relative-MaxSAT-def* **by** *fastforce*

**thus** *False* **by** *linarith*

**qed**

**then show** *?thesis*

**using**  $\langle \neg \vdash \varphi \rangle$  **by** *blast*

**qed**

**theorem** (in *classical-logic*) *relative-maximals-max-counting-deduction*:

$\Gamma \# \vdash n \varphi = (\forall \Phi \in \mathcal{M} \Gamma \varphi. n \leq \text{length } (\Gamma \ominus \Phi))$

**proof** (*cases*  $\vdash \varphi$ )

**case** *True*

**from**  $\langle \vdash \varphi \rangle$  **have**  $\Gamma \# \vdash n \varphi$

**using** *counting-deduction-tautology-weaken*

**by** *blast*

**moreover from**  $\langle \vdash \varphi \rangle$  **have**  $\mathcal{M} \Gamma \varphi = \{\}$

**using** *relative-maximals-existence* **by** *auto*

**hence**  $\forall \Phi \in \mathcal{M} \Gamma \varphi. n \leq \text{length } (\Gamma \ominus \Phi)$  **by** *blast*

**ultimately show** *?thesis* **by** *meson*

**next**

**case** *False*

**from**  $\langle \neg \vdash \varphi \rangle$  **have**  $(\Gamma \# \vdash n \varphi) = (n \leq \|\Gamma\|_\varphi)$

**by** (*simp add: relative-maximals-counting-deduction-lower-bound*)

**moreover have**  $(n \leq \|\Gamma\|_\varphi) = (\forall \Phi \in \mathcal{M} \Gamma \varphi. n \leq \text{length } (\Gamma \ominus \Phi))$

**proof** (*rule iffI*)

**assume**  $n \leq \|\Gamma\|_\varphi$

{

**fix**  $\Phi$

**assume**  $\Phi \in \mathcal{M} \Gamma \varphi$

**hence**  $n \leq \text{length } (\Gamma \ominus \Phi)$

**using**  $\langle n \leq \|\Gamma\|_\varphi \rangle$  *complement-relative-MaxSAT-intro* **by** *auto*

}

**thus**  $\forall \Phi \in \mathcal{M} \Gamma \varphi. n \leq \text{length } (\Gamma \ominus \Phi)$  **by** *blast*

**next**

**assume**  $\forall \Phi \in \mathcal{M} \Gamma \varphi. n \leq \text{length} (\Gamma \ominus \Phi)$   
**with**  $\langle \neg \vdash \varphi \rangle$  **obtain**  $\Phi$  **where**  
 $\Phi \in \mathcal{M} \Gamma \varphi$   
 $n \leq \text{length} (\Gamma \ominus \Phi)$   
**using** *relative-maximals-existence*  
**by** *blast*  
**thus**  $n \leq \|\Gamma\|_\varphi$   
**by** (*simp add: complement-relative-MaxSAT-intro*)  
**qed**  
**ultimately show** *?thesis* **by** *metis*  
**qed**

**lemma** (*in consistent-classical-logic*) *counting-deduction-to-maxsat*:  
 $(\Gamma \# \vdash n \perp) = (\text{MaxSAT } \Gamma + n \leq \text{length } \Gamma)$   
**by** (*metis*  
*add.commute*  
*consistency*  
*length-MaxSAT-decomposition*  
*relative-maximals-counting-deduction-lower-bound*  
*nat-add-left-cancel-le*)

## Chapter 4

# Inequality Completeness For Probability Logic

### 4.1 Limited Counting Deduction Completeness

The reduction of counting deduction to MaxSAT allows us to first prove completeness for counting deduction, as maximal consistent sublists allow us to recover maximally consistent sets, which give rise to Dirac measures.

The completeness result first presented here, where all of the propositions on the left hand side are the same, will be extended later.

**lemma** (in *probability-logic*) *list-probability-upper-bound*:

$$(\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma) \leq \text{real } (\text{length } \Gamma)$$

**proof** (*induct*  $\Gamma$ )

case *Nil*

then show *?case* by *simp*

next

case (*Cons*  $\gamma$   $\Gamma$ )

moreover have  $\mathcal{P} \gamma \leq 1$  using *unity-upper-bound* by *blast*

ultimately have  $\mathcal{P} \gamma + (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma) \leq 1 + \text{real } (\text{length } \Gamma)$  by *linarith*

then show *?case* by *simp*

qed

**theorem** (in *classical-logic*) *dirac-limited-counting-deduction-completeness*:

$$(\forall \mathcal{P} \in \text{dirac-measures. } \text{real } n * \mathcal{P} \varphi \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) = \sim \Gamma \# \vdash n (\sim \varphi)$$

**proof** –

{

fix  $\mathcal{P} :: 'a \Rightarrow \text{real}$

assume  $\mathcal{P} \in \text{dirac-measures}$

from *this* interpret *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$

unfolding *dirac-measures-def*

by *auto*

assume  $\sim \Gamma \# \vdash n (\sim \varphi)$

moreover have *replicate*  $n (\sim \varphi) = \sim (\text{replicate } n \varphi)$

by (*induct n, auto*)  
**ultimately have**  $\sim \Gamma \S \vdash \sim (\text{replicate } n \ \varphi)$   
 using *counting-deduction-to-measure-deduction by metis*  
**hence**  $(\sum \varphi \leftarrow (\text{replicate } n \ \varphi). \mathcal{P} \ \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \ \gamma)$   
 using *measure-deduction-soundness*  
 by *blast*  
**moreover have**  $(\sum \varphi \leftarrow (\text{replicate } n \ \varphi). \mathcal{P} \ \varphi) = \text{real } n * \mathcal{P} \ \varphi$   
 by (*induct n, simp, simp add: semiring-normalization-rules(3)*)  
**ultimately have**  $\text{real } n * \mathcal{P} \ \varphi \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \ \gamma)$   
 by *simp*  
 }  
**moreover**  
 {  
 assume  $\neg \sim \Gamma \# \vdash n (\sim \varphi)$   
**have**  $\exists \mathcal{P} \in \text{dirac-measures. real } n * \mathcal{P} \ \varphi > (\sum \gamma \leftarrow \Gamma. \mathcal{P} \ \gamma)$   
**proof** –  
**have**  $\exists \Phi. \Phi \in \mathcal{M} (\sim \Gamma) (\sim \varphi)$   
 using  
    $\langle \neg \sim \Gamma \# \vdash n (\sim \varphi) \rangle$   
   *relative-maximals-existence*  
   *counting-deduction-tautology-weaken*  
 by *blast*  
**from this obtain**  $\Phi$  **where**  $\Phi$ :  
    $(\sim \Phi) \in \mathcal{M} (\sim \Gamma) (\sim \varphi)$   
    $\text{mset } \Phi \subseteq \# \text{ mset } \Gamma$   
**unfolding** *map-negation-def*  
 by (*metis*  
   (*mono-tags, lifting*)  
   *relative-maximals-def*  
   *mem-Collect-eq*  
   *mset-sub-map-list-exists*)  
**hence**  $\neg \vdash \varphi \rightarrow \sqcup \Phi$   
 using  
   *biconditional-weaken*  
   *list-deduction-def*  
   *map-negation-list-implication*  
   *set-deduction-base-theory*  
   *relative-maximals-def*  
 by *blast*  
**from this obtain**  $\Omega$  **where**  $\Omega$ :  $\text{MCS } \Omega \ \varphi \in \Omega \sqcup \Phi \notin \Omega$   
 by (*meson*  
   *insert-subset*  
   *formula-consistent-def*  
   *formula-maximal-consistency*  
   *formula-maximally-consistent-extension*  
   *formula-maximally-consistent-set-def-def*  
   *set-deduction-base-theory*  
   *set-deduction-reflection*  
   *set-deduction-theorem*)

```

let ?P = λ χ. if χ ∈ Ω then (1 :: real) else 0
from Ω have ?P ∈ dirac-measures
  using MCS-dirac-measure by blast
moreover
from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ ?P
  unfolding dirac-measures-def
  by auto
have ∀ φ ∈ set Φ. ?P φ = 0
  using Φ(1) Ω(1) Ω(3) arbitrary-disjunction-exclusion-MCS by auto
with Φ(2) have (∑ γ ← Γ. ?P γ) = (∑ γ ← (Γ ⊖ Φ). ?P 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: ¬ mset Φ ⊆# mset Γ
      ∨ ω ∈ set Φ ∧ ω ∈ Ω
      ∨ (∑ γ ← Γ. ?P γ) = (∑ γ ← Γ ⊖ Φ. ?P γ)
  using Cons.hyps by fastforce
  have A:
    ∀ (f :: 'a ⇒ real) (Γ :: 'a list) Φ.
      ¬ mset Φ ⊆# mset Γ
      ∨ sum-list ((∑ φ ← Φ. f φ) # map f (Γ ⊖ Φ)) = (∑ γ ← Γ. f γ)
  using listSubtract-multisubset-list-summation by auto
  have B: ∀ rs. sum-list ((0::real) # rs) = sum-list rs
  by auto
  have C: ∀ r rs. (0::real) = r ∨ sum-list (r # rs) ≠ sum-list rs
  by simp
  have D: ∀ f. sum-list (sum-list (map f (φ # Φ)) # map f (Γ ⊖ (φ # Φ)))
    = (sum-list (map f Γ)::real)
  using A Cons.prem(1) by blast
  have E: mset Φ ⊆# mset Γ
  using Cons.prem(1) subset-mset.dual-order.trans by force
  then have F: ∀ f. (0::real) = sum-list (map f Φ)
    ∨ sum-list (map f Γ) ≠ sum-list (map f (Γ ⊖ Φ))
  using C A by (metis (no-types))
  then have G: (∑ φ' ← (φ # Φ). ?P φ') = 0 ∨ ω ∈ Ω
  using E ω Cons.prem(2) by auto
  have H: ∀ Γ r::real. r = (∑ γ ← Γ. ?P γ)
    ∨ ω ∈ set Φ
    ∨ r ≠ (∑ γ ← (φ # Γ). ?P γ)
  using Cons.prem(2) by auto
  have (1::real) ≠ 0 by linarith
moreover
{ assume ω ∉ set Φ
  then have ω ∉ Ω ∨ (∑ γ ← Γ. ?P γ) = (∑ γ ← Γ ⊖ (φ # Φ). ?P γ)

```



```

      using H F E D B ω by (metis (no-types) sum-list.Cons) }
    ultimately have ?thesis
      using G D B by (metis Cons.premis(2) list.set-intros(2))
    then show ?thesis
      by linarith
  qed
qed
hence (∑ γ←Γ. ?P γ) ≤ real (length (Γ ⊖ Φ))
  using list-probability-upper-bound
  by auto
  moreover
  have length (∼ Γ ⊖ ∼ Φ) < n
    by (metis not-le Φ(1) <¬ (∼ Γ) #⊢ n (∼ φ),
        relative-maximals-max-counting-deduction
        maximals-list-subtract-length-equiv)
  hence real (length (∼ Γ ⊖ ∼ Φ)) < real n
    by simp
  with Ω(2) have real (length (∼ Γ ⊖ ∼ Φ)) < real n * ?P φ
    by simp
  moreover
  have (∼ (Γ ⊖ Φ)) ≡ (∼ Γ ⊖ ∼ Φ)
    unfolding map-negation-def
    by (metis Φ(2) map-list-subtract-mset-equivalence)
  with perm-length have length (Γ ⊖ Φ) = length (∼ Γ ⊖ ∼ Φ)
    by (metis length-map local.map-negation-def)
  hence real (length (Γ ⊖ Φ)) = real (length (∼ Γ ⊖ ∼ Φ))
    by simp
  ultimately show ?thesis
    by force
  qed
}
ultimately show ?thesis by fastforce
qed

```

## 4.2 Measure Deduction Completeness

Since measure deduction may be reduced to counting deduction, we have measure deduction is complete.

**lemma** (in *classical-logic*) *dirac-measure-deduction-completeness*:

$$(\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) = \sim \Gamma \ \$\vdash \sim \Phi$$

**proof** –

```

{
  fix P :: 'a ⇒ real
  assume P ∈ dirac-measures
  from this interpret probability-logic (λ φ. ⊢ φ) (→) ⊥ P
  unfolding dirac-measures-def
  by auto
  assume ∼ Γ \$\vdash ∼ Φ

```

hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
 using *measure-deduction-soundness*  
 by *blast*  
 }  
 moreover  
 {  
 assume  $\neg \sim \Gamma \ \$\vdash \sim \Phi$   
 have  $\exists \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) > (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
 proof –  
 from  $\langle \neg \sim \Gamma \ \$\vdash \sim \Phi \rangle$  have  $\neg \sim (\sim \Phi) @ \sim \Gamma \ \#\vdash (\text{length } (\sim \Phi)) \perp$   
 using *measure-deduction-to-counting-deduction* by *blast*  
 moreover  
 have  $\sim (\sim \Phi) @ \sim \Gamma \ \#\vdash (\text{length } (\sim \Phi)) \perp = \sim (\sim \Phi) @ \sim \Gamma \ \#\vdash (\text{length } (\sim \Phi)) \perp$   
 }  
 by (*induct*  $\Phi$ , *auto*)  
 moreover have  $\vdash \sim \top \rightarrow \perp$   
 by (*simp add: negation-def*)  
 ultimately have  $\neg \sim (\sim \Phi @ \Gamma) \ \#\vdash (\text{length } \Phi) (\sim \top)$   
 using *counting-deduction-implication* by *fastforce*  
 from *this* obtain  $\mathcal{P}$  where  $\mathcal{P}$ :  
 $\mathcal{P} \in \text{dirac-measures}$   
 $\text{real } (\text{length } \Phi) * \mathcal{P} \top > (\sum \gamma \leftarrow (\sim \Phi @ \Gamma). \mathcal{P} \gamma)$   
 using *dirac-limited-counting-deduction-completeness*  
 by *fastforce*  
 from *this* interpret *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$   
 unfolding *dirac-measures-def*  
 by *auto*  
 from  $\mathcal{P}(2)$  have  $\text{real } (\text{length } \Phi) > (\sum \gamma \leftarrow \sim \Phi. \mathcal{P} \gamma) + (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
 by (*simp add: probability-unity*)  
 moreover have  $(\sum \gamma \leftarrow \sim \Phi. \mathcal{P} \gamma) = \text{real } (\text{length } \Phi) - (\sum \gamma \leftarrow \Phi. \mathcal{P} \gamma)$   
 using *complementation*  
 by (*induct*  $\Phi$ , *auto*)  
 ultimately show *?thesis*  
 using  $\mathcal{P}(1)$  by *auto*  
 qed  
 }  
 ultimately show *?thesis* by *fastforce*  
 qed

**theorem** (*in classical-logic*) *measure-deduction-completeness*:  
 $(\forall \mathcal{P} \in \text{probabilities}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) = \sim \Gamma \ \$\vdash \sim \Phi$   
 proof –  
 {  
 fix  $\mathcal{P} :: 'a \Rightarrow \text{real}$   
 assume  $\mathcal{P} \in \text{probabilities}$   
 from *this* interpret *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$   
 unfolding *probabilities-def*  
 by *auto*  
 assume  $\sim \Gamma \ \$\vdash \sim \Phi$

hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
 using *measure-deduction-soundness*  
 by *blast*  
 }  
 thus *?thesis*  
 using *dirac-measures-subset dirac-measure-deduction-completeness*  
 by *fastforce*  
 qed

### 4.3 Counting Deduction Completeness

Leveraging our measure deduction completeness result, we may extend our limited counting deduction completeness theorem to full completeness.

**lemma** (in *classical-logic*) *measure-left-commute*:

$$(\Phi @ \Psi) \$\vdash \Xi = (\Psi @ \Phi) \$\vdash \Xi$$

**proof** –

$$\text{have } (\Phi @ \Psi) \preceq (\Psi @ \Phi) \text{ } (\Psi @ \Phi) \preceq (\Phi @ \Psi)$$

using *stronger-theory-reflexive stronger-theory-right-permutation perm-append-swap*

by *blast+*

thus *?thesis*

using *measure-stronger-theory-left-monotonic*

by *blast*

qed

**lemma** (in *classical-logic*) *stronger-theory-double-negation-right*:

$$\Phi \preceq \sim (\sim \Phi)$$

by (*induct*  $\Phi$ , *simp*, *simp add: double-negation negation-def stronger-theory-left-right-cons*)

**lemma** (in *classical-logic*) *stronger-theory-double-negation-left*:

$$\sim (\sim \Phi) \preceq \Phi$$

by (*induct*  $\Phi$ ,

*simp*,

*simp add: double-negation-converse negation-def stronger-theory-left-right-cons*)

**lemma** (in *classical-logic*) *counting-deduction-completeness*:

$$(\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) = (\sim \Gamma @ \Phi) \#\vdash (\text{length } \Phi) \perp$$

**proof** –

$$\text{have } (\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$$

$$= \sim (\sim \Phi) @ \sim \Gamma \#\vdash (\text{length } (\sim \Phi)) \perp$$

using *dirac-measure-deduction-completeness measure-deduction-to-counting-deduction*

by *blast*

**also have** ... =  $\sim (\sim \Phi) @ \sim \Gamma \#\vdash (\text{length } \Phi) \perp$  by (*induct*  $\Phi$ , *auto*)

**also have** ... =  $\sim \Gamma @ \sim (\sim \Phi) \#\vdash (\text{length } \Phi) \perp$

by (*simp add: measure-left-commute counting-deduction-to-measure-deduction*)

**also have** ... =  $\sim \Gamma @ \Phi \#\vdash (\text{length } \Phi) \perp$

by (*meson measure-cancel*

*stronger-theory-to-measure-deduction*)

```

    measure-transitive
    counting-deduction-to-measure-deduction
    stronger-theory-double-negation-left
    stronger-theory-double-negation-right)
  finally show ?thesis by blast
qed

```

## 4.4 Collapse Theorem For Probability Logic

We now turn to proving the collapse theorem for probability logic. This states that any inequality holds for all finitely additive probability measures if and only if it holds for all Dirac measures.

**theorem** (in *classical-logic*) *weakly-additive-completeness-collapse*:  
 $(\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= (\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 by (*simp add: dirac-measure-deduction-completeness*  
*measure-deduction-completeness*)

The collapse theorem may be strengthened to include an arbitrary constant term  $c$ . This will be key to characterizing MaxSAT completeness in §4.5.

**lemma** (in *classical-logic*) *nat-dirac-probability*:  
 $\forall \mathcal{P} \in \text{dirac-measures. } \exists n :: \text{nat. real } n = (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi)$   
**proof** (*induct*  $\Phi$ )  
 case *Nil*  
 then show ?case by *simp*  
**next**  
 case (*Cons*  $\varphi \Phi$ )  
 {  
 fix  $\mathcal{P} :: 'a \Rightarrow \text{real}$   
 assume  $\mathcal{P} \in \text{dirac-measures}$   
 from *Cons this* obtain  $n$  where  $\text{real } n = (\sum \varphi' \leftarrow \Phi. \mathcal{P} \varphi')$  by *fastforce*  
 hence  $\star: (\sum \varphi' \leftarrow \Phi. \mathcal{P} \varphi') = \text{real } n$  by *simp*  
 have  $\exists n. \text{real } n = (\sum \varphi' \leftarrow (\varphi \# \Phi). \mathcal{P} \varphi')$   
 proof (*cases*  $\mathcal{P} \varphi = 1$ )  
 case *True*  
 then show ?thesis  
 by (*simp add:  $\star$ , metis of-nat-Suc*)  
 next  
 case *False*  
 hence  $\mathcal{P} \varphi = 0$  using  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$  *dirac-measures-def* by *auto*  
 then show ?thesis using  $\star$   
 by *simp*  
 qed  
 }  
 thus ?case by *blast*  
**qed**

**lemma** (in *classical-logic*) *dirac-ceiling*:  
 $\forall \mathcal{P} \in \text{dirac-measures.}$   
 $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$

**proof** –  
{  
  **fix**  $\mathcal{P}$   
  **assume**  $\mathcal{P} \in \text{dirac-measures}$   
  **have**  $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
   $= ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
  **proof** (*rule iffI*)  
  **assume** *assm*:  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **show**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **proof** (*rule ccontr*)  
  **assume**  $\neg (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **moreover**  
  **obtain**  $x :: \text{int}$   
  **and**  $y :: \text{int}$   
  **and**  $z :: \text{int}$   
  **where** *xyz*:  $x = (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi)$   
   $y = \lceil c \rceil$   
   $z = (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **using** *nat-dirac-probability*  
  **by** (*metis*  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$  *of-int-of-nat-eq*)  
  **ultimately have**  $x + y - 1 \geq z$  **by** *linarith*  
  **hence**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c > (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$  **using** *xyz* **by** *linarith*  
  **thus** *False* **using** *assm* **by** *simp*  
  **qed**  
  **next**  
  **assume**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **thus**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
  **by** *linarith*  
  **qed**  
}
**thus** *?thesis* **by** *blast*  
**qed**

**lemma** (in *probability-logic*) *probability-replicate-verum*:  
**fixes**  $n :: \text{nat}$   
**shows**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + n = (\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi)$   
**using** *probability-unity*  
**by** (*induct n, auto*)

**lemma** (in *classical-logic*) *dirac-collapse*:  
 $(\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= (\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$

**proof**  
**assume**  $\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**hence**  $\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$

```

    using dirac-measures-subset by fastforce
  thus  $\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
    using dirac-ceiling by blast
next
assume assm:  $\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
show  $\forall \mathcal{P} \in \text{probabilities}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
proof (cases  $c \geq 0$ )
  case True
    from this obtain  $n :: \text{nat}$  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  $\mathcal{P}$ 
      assume  $\mathcal{P} \in \text{dirac-measures}$ 
      from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
        unfolding dirac-measures-def
        by auto
      have  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
        using assm  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$  by blast
      hence  $(\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
        using  $\langle \text{real } n = \lceil c \rceil \rangle$ 
          probability-replicate-verum [where  $\Phi = \Phi$  and  $n = n$ ]
        by metis
    }
  }
  hence  $\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
    by blast
  hence  $\dagger: \forall \mathcal{P} \in \text{probabilities}. (\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
    using weakly-additive-completeness-collapse by blast
  {
    fix  $\mathcal{P}$ 
    assume  $\mathcal{P} \in \text{probabilities}$ 
    from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
      unfolding probabilities-def
      by auto
    have  $(\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
      using  $\dagger$   $\langle \mathcal{P} \in \text{probabilities} \rangle$  by blast
    hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
      using  $\langle \text{real } n = \lceil c \rceil \rangle$ 
        probability-replicate-verum [where  $\Phi = \Phi$  and  $n = n$ ]
      by linarith
  }
}
then show ?thesis by blast

```

```

next
  case False
  hence  $\lceil c \rceil \leq 0$  by auto
  from this obtain  $n :: \text{nat}$  where  $\text{real } n = - \lceil c \rceil$ 
  by (metis neg-0-le-iff-le of-nat-nat)
  {
  fix  $\mathcal{P}$ 
  assume  $\mathcal{P} \in \text{dirac-measures}$ 
  from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
  unfolding dirac-measures-def
  by auto
  have  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
  using assm  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$  by blast
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  using  $\langle \text{real } n = - \lceil c \rceil \rangle$ 
  probability-replicate-verum [where  $\Phi = \Gamma$  and  $n = n$ ]
  by linarith
  }
  hence  $\forall \mathcal{P} \in \text{dirac-measures}.$ 
   $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  by blast
  hence  $\ddagger: \forall \mathcal{P} \in \text{probabilities}.$ 
   $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  using weakly-additive-completeness-collapse by blast
  {
  fix  $\mathcal{P}$ 
  assume  $\mathcal{P} \in \text{probabilities}$ 
  from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
  unfolding probabilities-def
  by auto
  have  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  using  $\ddagger \langle \mathcal{P} \in \text{probabilities} \rangle$  by blast
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
  using  $\langle \text{real } n = - \lceil c \rceil \rangle$ 
  probability-replicate-verum [where  $\Phi = \Gamma$  and  $n = n$ ]
  by linarith
  }
  }
  then show ?thesis by blast
qed
qed

```

**lemma** (*in classical-logic*) *dirac-strict-floor*:

```

 $\forall \mathcal{P} \in \text{dirac-measures}.$ 
 $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$ 
 $= ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$ 

```

**proof**

```

fix  $\mathcal{P} :: 'a \Rightarrow \text{real}$ 
let  $?P' = (\lambda \varphi. \lfloor \mathcal{P} \varphi \rfloor) :: 'a \Rightarrow \text{int}$ 
assume  $\mathcal{P} \in \text{dirac-measures}$ 

```

**hence**  $\forall \varphi. \mathcal{P} \varphi = ?\mathcal{P}' \varphi$   
**unfolding** *dirac-measures-def*  
**by** (*metis (mono-tags, lifting)*  
*mem-Collect-eq*  
*of-int-0*  
*of-int-1*  
*of-int-floor-cancel*)  
**hence**  $A: (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) = (\sum \varphi \leftarrow \Phi. ?\mathcal{P}' \varphi)$   
**by** (*induct*  $\Phi$ , *auto*)  
**have**  $B: (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma) = (\sum \gamma \leftarrow \Gamma. ?\mathcal{P}' \gamma)$   
**using**  $\langle \forall \varphi. \mathcal{P} \varphi = ?\mathcal{P}' \varphi \rangle$  **by** (*induct*  $\Gamma$ , *auto*)  
**have**  $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= ((\sum \varphi \leftarrow \Phi. ?\mathcal{P}' \varphi) + c < (\sum \gamma \leftarrow \Gamma. ?\mathcal{P}' \gamma))$   
**unfolding**  $A B$  **by** *auto*  
**also have**  $\dots = ((\sum \varphi \leftarrow \Phi. ?\mathcal{P}' \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. ?\mathcal{P}' \gamma))$   
**by** *linarith*  
**finally show**  $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) =$   
 $((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
**using**  $A B$  **by** *linarith*  
**qed**

**lemma** (in *classical-logic*) *strict-dirac-collapse*:

$(\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= (\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$

**proof**

**assume**  $\forall \mathcal{P} \in \text{probabilities. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**hence**  $\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**using** *dirac-measures-subset* **by** *blast*  
**thus**  $\forall \mathcal{P} \in \text{dirac-measures. } ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
**using** *dirac-strict-floor* **by** *blast*

**next**

**assume**  $\forall \mathcal{P} \in \text{dirac-measures. } ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
**moreover have**  $\lfloor c \rfloor + 1 = \lceil (\lfloor c \rfloor + 1) \rceil :: \text{real}$

**by** *simp*

**ultimately have**  $\star$ :

$\forall \mathcal{P} \in \text{probabilities. } ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
**using** *dirac-collapse [of*  $\Phi$   $\lfloor c \rfloor + 1$   $\Gamma]$

**by** *auto*

**show**  $\forall \mathcal{P} \in \text{probabilities. } ((\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$

**proof**

**fix**  $\mathcal{P} :: 'a \Rightarrow \text{real}$

**assume**  $\mathcal{P} \in \text{probabilities}$

**hence**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lfloor c \rfloor + 1 \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$

**using**  $\star$  **by** *auto*

**thus**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c < (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$

**by** *linarith*

**qed**

**qed**



## 4.5 MaxSAT Completeness For Probability Logic

It follows from the collapse theorem that any probability inequality tautology, include those with *constant terms*, may be reduced to a bounded MaxSAT problem. This is not only a key computational complexity result, but suggests a straightforward algorithm for *computing* probability identities.

**lemma** (in *classical-logic*) *relative-maximals-verum-extract*:

```

assumes  $\neg \vdash \varphi$ 
shows  $(| \text{replicate } n \top @ \Phi |_\varphi) = n + (| \Phi |_\varphi)$ 
proof (induct  $n$ )
  case 0
  then show ?case by simp
next
  case (Suc  $n$ )
  {
    fix  $\Phi$ 
    obtain  $\Sigma$  where  $\Sigma \in \mathcal{M} (\top \# \Phi) \varphi$ 
    using assms relative-maximals-existence by fastforce
    hence  $\top \in \text{set } \Sigma$ 
    by (metis (no-types, lifting)
      list.set-intros(1)
      list-deduction-modus-ponens
      list-deduction-weaken
      relative-maximals-complement-equiv
      relative-maximals-def
      verum-tautology
      mem-Collect-eq)
    hence  $\neg (\text{remove1 } \top \Sigma \vdash \varphi)$ 
    by (meson  $\langle \Sigma \in \mathcal{M} (\top \# \Phi) \varphi \rangle$ 
      list.set-intros(1)
      axiom-k
      list-deduction-modus-ponens
      list-deduction-monotonic
      list-deduction-weaken
      relative-maximals-complement-equiv
      set-remove1-subset)

    moreover
    have  $\text{mset } \Sigma \subseteq\# \text{mset } (\top \# \Phi)$ 
    using  $\langle \Sigma \in \mathcal{M} (\top \# \Phi) \varphi \rangle$  relative-maximals-def by blast
    hence  $\text{mset } (\text{remove1 } \top \Sigma) \subseteq\# \text{mset } \Phi$ 
    using subset-eq-diff-conv by fastforce
    ultimately have  $(| \Phi |_\varphi) \geq \text{length } (\text{remove1 } \top \Sigma)$ 
    by (metis (no-types, lifting)
      relative-MaxSAT-intro
      list-deduction-weaken
      relative-maximals-def
      relative-maximals-existence)
  }

```

$mem\text{-}Collect\text{-}eq$   
**hence**  $(|\Phi|_\varphi) + 1 \geq length\ \Sigma$   
**by**  $(simp\ add:\ \langle \top \in set\ \Sigma \rangle\ length\text{-}remove1)$   
**moreover have**  $(|\Phi|_\varphi) < length\ \Sigma$   
**proof**  $(rule\ ccontr)$   
**assume**  $\neg (|\Phi|_\varphi) < length\ \Sigma$   
**hence**  $(|\Phi|_\varphi) \geq length\ \Sigma$  **by**  $linarith$   
**from this obtain  $\Delta$  where**  $\Delta \in \mathcal{M}\ \Phi\ \varphi\ length\ \Delta \geq length\ \Sigma$   
**using**  $assms\ relative\text{-}MaxSAT\text{-}intro\ relative\text{-}maximals\text{-}existence$  **by**  $fastforce$   
**hence**  $\neg (\top \# \Delta) \vdash \varphi$   
**using**  $list\text{-}deduction\text{-}modus\text{-}ponens$   
 $list\text{-}deduction\text{-}theorem$   
 $list\text{-}deduction\text{-}weaken$   
 $relative\text{-}maximals\text{-}def$   
 $verum\text{-}tautology$   
**by**  $blast$   
**moreover have**  $mset\ (\top \# \Delta) \subseteq\# mset\ (\top \# \Phi)$   
**using**  $\langle \Delta \in \mathcal{M}\ \Phi\ \varphi \rangle\ relative\text{-}maximals\text{-}def$  **by**  $auto$   
**ultimately have**  $length\ \Sigma \geq length\ (\top \# \Delta)$   
**using**  $\langle \Sigma \in \mathcal{M}\ (\top \# \Phi)\ \varphi \rangle\ relative\text{-}maximals\text{-}def$  **by**  $blast$   
**hence**  $length\ \Delta \geq length\ (\top \# \Delta)$   
**using**  $\langle length\ \Sigma \leq length\ \Delta \rangle\ dual\text{-}order.trans$  **by**  $blast$   
**thus**  $False$  **by**  $simp$   
**qed**  
**ultimately have**  $(|\top \# \Phi|_\varphi) = (1 + |\Phi|_\varphi)$   
**by**  $(metis\ Suc\text{-}eq\text{-}plus1\ Suc\text{-}le\text{-}eq\ \langle \Sigma \in \mathcal{M}\ (\top \# \Phi)\ \varphi \rangle\ add.commute\ le\text{-}antisym\ relative\text{-}MaxSAT\text{-}intro)$   
**}**  
**thus**  $?case$  **using**  $Suc$  **by**  $simp$   
**qed**

**lemma** **(in**  $classical\text{-}logic$  $)$   $complement\text{-}MaxSAT\text{-}completeness$ :  
 $(\forall \mathcal{P} \in dirac\text{-}measures. (\sum \varphi \leftarrow \Phi. \mathcal{P}\ \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P}\ \gamma)) = (length\ \Phi \leq \|\sim \Gamma @ \Phi\|_\perp)$   
**proof**  $(cases\ \vdash\ \perp)$   
**case**  $True$   
**hence**  $\mathcal{M}\ (\sim \Gamma @ \Phi)\ \perp = \{\}$   
**using**  $relative\text{-}maximals\text{-}existence$  **by**  $auto$   
**hence**  $length\ (\sim \Gamma @ \Phi) = \|\sim \Gamma @ \Phi\|_\perp$   
**unfolding**  $complement\text{-}relative\text{-}MaxSAT\text{-}def\ relative\text{-}MaxSAT\text{-}def$  **by**  $presburger$   
**then show**  $?thesis$   
**using**  $True\ counting\text{-}deduction\text{-}completeness\ counting\text{-}deduction\text{-}tautology\text{-}weaken$   
**by**  $auto$

**next**  
**case**  $False$   
**then show**  $?thesis$   
**using**  $counting\text{-}deduction\text{-}completeness\ relative\text{-}maximals\text{-}counting\text{-}deduction\text{-}lower\text{-}bound$   
**by**  $blast$   
**qed**

```

lemma (in classical-logic) relative-maximals-neg-verum-elim:
  (| replicate n ( $\sim \top$ ) @  $\Phi$  | $\varphi$ ) = (|  $\Phi$  | $\varphi$ )
proof (induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  {
    fix  $\Phi$ 
    have (| ( $\sim \top$ ) #  $\Phi$  | $\varphi$ ) = (|  $\Phi$  | $\varphi$ )
    proof (cases  $\vdash \varphi$ )
      case True
      then show ?thesis
        unfolding relative-MaxSAT-def relative-maximals-def
        by (simp add: list-deduction-weaken)
    next
      case False
      from this obtain  $\Sigma$  where  $\Sigma \in \mathcal{M} ((\sim \top) \# \Phi) \varphi$ 
      using relative-maximals-existence by fastforce
      have [( $\sim \top$ )] : $\vdash \varphi$ 
      by (metis modus-ponens
        Peirces-law
        pseudo-scotus
        list-deduction-theorem
        list-deduction-weaken
        negation-def
        verum-def)
      hence  $\sim \top \notin \text{set } \Sigma$ 
      by (meson  $\langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$ 
        list.set-intros(1)
        list-deduction-base-theory
        list-deduction-theorem
        list-deduction-weaken
        relative-maximals-complement-equiv)
      hence remove1 ( $\sim \top$ )  $\Sigma = \Sigma$ 
      by (simp add: remove1-idem)
      moreover have  $\text{mset } \Sigma \subseteq \# \text{mset } ((\sim \top) \# \Phi)$ 
      using  $\langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$  relative-maximals-def by blast
      ultimately have  $\text{mset } \Sigma \subseteq \# \text{mset } \Phi$ 
      by (metis add-mset-add-single mset.simps(2) mset-remove1 subset-eq-diff-conv)
      moreover have  $\neg (\Sigma : \vdash \varphi)$ 
      using  $\langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$  relative-maximals-def by blast
      ultimately have (|  $\Phi$  | $\varphi$ )  $\geq \text{length } \Sigma$ 
      by (metis (no-types, lifting)
        relative-MaxSAT-intro
        list-deduction-weaken
        relative-maximals-def
        relative-maximals-existence)
  }

```

$mem\text{-}Collect\text{-}eq$   
**hence**  $(\| \Phi \|_\varphi) \geq (\| (\sim \top) \# \Phi \|_\varphi)$   
**using**  $\langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$  *relative- $MaxSAT$ -intro* **by** *auto*  
**moreover**  
**have**  $(\| \Phi \|_\varphi) \leq (\| (\sim \top) \# \Phi \|_\varphi)$   
**proof** –  
**obtain**  $\Delta$  **where**  $\Delta \in \mathcal{M} \Phi \varphi$   
**using** *False relative-maximals-existence* **by** *blast*  
**hence**  
 $\neg \Delta \vdash \varphi$   
 $mset \Delta \subseteq \# mset ((\sim \top) \# \Phi)$   
**unfolding** *relative-maximals-def*  
**by** (*simp*,  
 $metis$  (*mono-tags*, *lifting*)  
 $Diff\text{-}eq\text{-}empty\text{-}iff\text{-}mset$   
 $list\text{-}subtract.simps(2)$   
 $list\text{-}subtract\text{-}mset\text{-}homomorphism$   
 $relative\text{-}maximals\text{-}def$   
 $mem\text{-}Collect\text{-}eq$   
 $mset\text{-}zero\text{-}iff$   
 $remove1.simps(1)$ )  
**hence**  $length \Delta \leq length \Sigma$   
**using**  $\langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$  *relative-maximals-def* **by** *blast*  
**thus** *?thesis*  
**using**  $\langle \Delta \in \mathcal{M} \Phi \varphi \rangle \langle \Sigma \in \mathcal{M} (\sim \top \# \Phi) \varphi \rangle$  *relative- $MaxSAT$ -intro* **by**  
*auto*  
**qed**  
**ultimately show** *?thesis*  
**using** *le-antisym* **by** *blast*  
**qed**  
**}**  
**thus** *?case* **using** *Suc* **by** *simp*  
**qed**

**lemma** (in *classical-logic*) *dirac- $MaxSAT$ -partial-completeness*:

$(\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)) = (MaxSAT (\sim \Gamma @ \Phi) \leq length \Gamma)$

**proof** –

$\{$   
**fix**  $\mathcal{P} :: 'a \Rightarrow real$   
**obtain**  $\varrho :: 'a \text{ list} \Rightarrow 'a \Rightarrow real$  **where**  
 $(\forall \Phi \Gamma. \varrho \Phi \Gamma \in \text{dirac-measures} \wedge \neg (\sum \varphi \leftarrow \Phi. (\varrho \Phi \Gamma) \varphi) \leq (\sum \gamma \leftarrow \Gamma. (\varrho \Phi \Gamma) \gamma))$

$\vee length \Phi \leq \| \sim \Gamma @ \Phi \|_\perp$ )

$\wedge (\forall \Phi \Gamma. length \Phi \leq (\| \sim \Gamma @ \Phi \|_\perp)$

$\longrightarrow (\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)))$

**using** *complement- $MaxSAT$ -completeness* **by** *moura*

**moreover have**  $\forall \Gamma \varphi n. length \Gamma - n \leq (\| \Gamma \|_\varphi) \vee (\| \Gamma \|_\varphi) - n \neq 0$

**by** (*metis add-diff-cancel-right'*)

*cancel-ab-semigroup-add-class.diff-right-commute*  
*diff-is-0-eq length-MaxSAT-decomposition)*

**moreover have**  $\forall \Gamma \Phi n. \text{length } (\Gamma @ \Phi) - n \leq \text{length } \Gamma \vee \text{length } \Phi - n \neq 0$   
**by force**

**ultimately have**  
 $(\mathcal{P} \in \text{dirac-measures} \longrightarrow (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $\wedge (|\sim \Gamma @ \Phi |_{\perp}) \leq \text{length } (\sim \Gamma)$   
 $\vee \neg (|\sim \Gamma @ \Phi |_{\perp}) \leq \text{length } (\sim \Gamma)$   
 $\wedge (\exists \mathcal{P}. \mathcal{P} \in \text{dirac-measures} \wedge \neg (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
**by** (*metis (no-types) add-diff-cancel-left'*  
*add-diff-cancel-right'*  
*diff-is-0-eq length-append*  
*length-MaxSAT-decomposition*)

}

**then show** *?thesis by auto*

**qed**

**lemma** (*in consistent-classical-logic*) *dirac-inequality-elim*:  
**fixes**  $c :: \text{real}$   
**assumes**  $\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**shows**  $(\text{MaxSAT } (\sim \Gamma @ \Phi) + c \leq \text{length } \Gamma)$

**proof** (*cases c ≥ 0*)  
**case True**  
**from this obtain**  $n :: \text{nat}$  **where** *real n = ⌈c⌉*  
**by** (*metis ceiling-mono ceiling-zero of-nat-nat*)  
{  
**fix**  $\mathcal{P}$   
**assume**  $\mathcal{P} \in \text{dirac-measures}$   
**from this interpret** *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$   
**unfolding** *dirac-measures-def*  
**by auto**  
**have**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + n \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**by** (*metis assms ⟨P ∈ dirac-measures⟩ ⟨real n = ⌈c⌉⟩ dirac-ceiling*)  
**hence**  $(\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**using** *probability-replicate-verum [where Φ=Φ and n=n]*  
**by metis**  
}

**hence**  $(|\sim \Gamma @ \text{replicate } n \top @ \Phi |_{\perp}) \leq \text{length } \Gamma$   
**using** *dirac-MaxSAT-partial-completeness by blast*

**moreover have**  $\text{mset } (\sim \Gamma @ \text{replicate } n \top @ \Phi) = \text{mset } (\text{replicate } n \top @ \sim \Gamma @ \Phi)$   
**by simp**

**ultimately have**  $(|\text{replicate } n \top @ \sim \Gamma @ \Phi |_{\perp}) \leq \text{length } \Gamma$   
**unfolding** *relative-MaxSAT-def relative-maximals-def*  
**by metis**

**hence**  $(|\sim \Gamma @ \Phi |_{\perp}) + \lceil c \rceil \leq \text{length } \Gamma$   
**using** *⟨real n = ⌈c⌉⟩ consistency relative-maximals-verum-extract*  
**by auto**

**then show** *?thesis by linarith*

```

next
  case False
  hence  $\lceil c \rceil \leq 0$  by auto
  from this obtain  $n :: \text{nat}$  where  $\text{real } n = - \lceil c \rceil$ 
  by (metis neg-0-le-iff-le of-nat-nat)
  {
  fix  $\mathcal{P}$ 
  assume  $\mathcal{P} \in \text{dirac-measures}$ 
  from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
  unfolding dirac-measures-def
  by auto
  have  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
  using assms  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$  dirac-ceiling
  by blast
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma) + n$ 
  using  $\langle \text{real } n = - \lceil c \rceil \rangle$  by linarith
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  using probability-replicate-verum [where  $\Phi = \Gamma$  and  $n = n$ ]
  by metis
  }
  hence  $(| \sim (\text{replicate } n \top @ \Gamma) @ \Phi |_{\perp}) \leq \text{length } (\text{replicate } n \top @ \Gamma)$ 
  using dirac-MaxSAT-partial-completeness [where  $\Phi = \Phi$  and  $\Gamma = \text{replicate } n \top$ 
  @  $\Gamma$ ]
  by metis
  hence  $(| \sim \Gamma @ \Phi |_{\perp}) \leq n + \text{length } \Gamma$ 
  by (simp add: relative-maximals-neg-verum-elim)
  then show ?thesis using  $\langle \text{real } n = - \lceil c \rceil \rangle$  by linarith
qed

lemma (in classical-logic) dirac-inequality-intro:
  fixes  $c :: \text{real}$ 
  assumes MaxSAT  $(\sim \Gamma @ \Phi) + c \leq \text{length } \Gamma$ 
  shows  $\forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
proof (cases  $\vdash \perp$ )
  assume  $\vdash \perp$ 
  {
  fix  $\mathcal{P}$ 
  assume  $\mathcal{P} \in \text{dirac-measures}$ 
  from this interpret probability-logic  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$ 
  unfolding dirac-measures-def
  by auto
  have False
  using  $\langle \vdash \perp \rangle$  consistency by blast
  }
  then show ?thesis by blast
next
  assume  $\neg \vdash \perp$ 
  then show ?thesis
proof (cases  $c \geq 0$ )

```

**assume**  $c \geq 0$   
**from this obtain**  $n :: \text{nat}$  **where**  $\text{real } n = \lceil c \rceil$   
**by** (*metis ceiling-mono ceiling-zero of-nat-nat*)  
**hence**  $n + (|\sim \Gamma @ \Phi |_{\perp}) \leq \text{length } \Gamma$   
**using** *assms by linarith*  
**hence**  $(|\text{replicate } n \top @ \sim \Gamma @ \Phi |_{\perp}) \leq \text{length } \Gamma$   
**by** (*simp add: <math>\langle \top \vdash \perp \rangle</math> relative-maximals-verum-extract*)  
**moreover have**  $\text{mset } (\text{replicate } n \top @ \sim \Gamma @ \Phi) = \text{mset } (\sim \Gamma @ \text{replicate } n \top @ \Phi)$   
**by** *simp*  
**ultimately have**  $(|\sim \Gamma @ \text{replicate } n \top @ \Phi |_{\perp}) \leq \text{length } \Gamma$   
**unfolding** *relative-MaxSAT-def relative-maximals-def*  
**by** *metis*  
**hence**  $\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**using** *dirac-MaxSAT-partial-completeness by blast*  
{  
**fix**  $\mathcal{P}$   
**assume**  $\mathcal{P} \in \text{dirac-measures}$   
**from this interpret** *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$   
**unfolding** *dirac-measures-def*  
**by** *auto*  
**have**  $(\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**using**  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$   
 $\langle \forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow (\text{replicate } n \top) @ \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma) \rangle$   
**by** *blast*  
**hence**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + n \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**by** (*simp add: probability-replicate-verum*)  
**hence**  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$   
**using**  $\langle \text{real } n = \text{real-of-int } \lceil c \rceil \rangle$  **by** *linarith*  
}  
**then show** *?thesis* **by** *blast*  
**next**  
**assume**  $\neg (c \geq 0)$   
**hence**  $\lceil c \rceil \leq 0$  **by** *auto*  
**from this obtain**  $n :: \text{nat}$  **where**  $\text{real } n = - \lceil c \rceil$   
**by** (*metis neg-0-le-iff-le of-nat-nat*)  
**hence**  $(|\sim \Gamma @ \Phi |_{\perp}) \leq n + \text{length } \Gamma$   
**using** *assms by linarith*  
**hence**  $(|\sim (\text{replicate } n \top @ \Gamma) @ \Phi |_{\perp}) \leq \text{length } (\text{replicate } n \top @ \Gamma)$   
**by** (*simp add: relative-maximals-neg-verum-elim*)  
**hence**  $\forall \mathcal{P} \in \text{dirac-measures. } (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$   
**using** *dirac-MaxSAT-partial-completeness by blast*  
{  
**fix**  $\mathcal{P}$   
**assume**  $\mathcal{P} \in \text{dirac-measures}$   
**from this interpret** *probability-logic*  $(\lambda \varphi. \vdash \varphi) (\rightarrow) \perp \mathcal{P}$

```

    unfolding dirac-measures-def
  by auto
  have  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma)$ 
  using  $\langle \mathcal{P} \in \text{dirac-measures} \rangle$ 
     $\langle \forall \mathcal{P} \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) \leq (\sum \gamma \leftarrow (\text{replicate } n \top) @ \Gamma. \mathcal{P} \gamma) \rangle$ 
  by blast
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + \lceil c \rceil \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
  using  $\langle \text{real } n = - \lceil c \rceil \rangle$  probability-replicate-verum by auto
  hence  $(\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma)$ 
  by linarith
}
then show ?thesis by blast
qed
qed

```

**lemma** (in *consistent-classical-logic*) *dirac-inequality-equiv*:  
 $(\forall \delta \in \text{dirac-measures}. (\sum \varphi \leftarrow \Phi. \delta \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \delta \gamma))$   
 $= (\text{MaxSAT } (\sim \Gamma @ \Phi) + (c :: \text{real}) \leq \text{length } \Gamma)$   
 using *dirac-inequality-elim* *dirac-inequality-intro* *consistency* by *auto*

**theorem** (in *consistent-classical-logic*) *probability-inequality-equiv*:  
 $(\forall \mathcal{P} \in \text{probabilities}. (\sum \varphi \leftarrow \Phi. \mathcal{P} \varphi) + c \leq (\sum \gamma \leftarrow \Gamma. \mathcal{P} \gamma))$   
 $= (\text{MaxSAT } (\sim \Gamma @ \Phi) + (c :: \text{real}) \leq \text{length } \Gamma)$   
 unfolding *dirac-collapse*  
 using *dirac-inequality-equiv* *dirac-ceiling* by *blast*

**no-notation** *first-component* ( $\mathfrak{A}$ )  
**no-notation** *second-component* ( $\mathfrak{B}$ )  
**no-notation** *merge-witness* ( $\mathfrak{J}$ )  
**no-notation** *X-witness* ( $\mathfrak{X}$ )  
**no-notation** *X-component* ( $\mathfrak{X}_\bullet$ )  
**no-notation** *Y-witness* ( $\mathfrak{Y}$ )  
**no-notation** *Y-component* ( $\mathfrak{Y}_\bullet$ )  
**no-notation** *submerge-witness* ( $\mathfrak{E}$ )  
**no-notation** *recover-witness-A* ( $\mathfrak{P}$ )  
**no-notation** *recover-complement-A* ( $\mathfrak{P}^C$ )  
**no-notation** *recover-witness-B* ( $\mathfrak{Q}$ )  
**no-notation** *relative-maximals* ( $\mathcal{M}$ )  
**no-notation** *relative-MaxSAT* ( $| - |$  [45])  
**no-notation** *complement-relative-MaxSAT* ( $\| - \|$  [45])  
**no-notation** *MaxSAT-optimal-pre-witness* ( $\mathfrak{W}$ )  
**no-notation** *MaxSAT-optimal-witness* ( $\mathfrak{W}$ )  
**no-notation** *disjunction-MaxSAT-optimal-witness* ( $\mathfrak{W}_\sqcup$ )  
**no-notation** *implication-MaxSAT-optimal-witness* ( $\mathfrak{W}_\rightarrow$ )  
**no-notation** *MaxSAT-witness* ( $\mathfrak{W}$ )

**notation** *FuncSet.funcset* (**infixr**  $\rightarrow$  60)



**end**

# Bibliography

- [1] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, Feb. 1976.