

# Class-based Classical Propositional Logic

Matthew Doty

June 17, 2024

## Abstract

We formulate classical propositional logic as an axiom class. Our class represents a Hilbert-style proof system with the axioms  $\vdash \varphi \rightarrow \psi \rightarrow \varphi$ ,  $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$ , and  $\vdash ((\varphi \rightarrow \perp) \rightarrow \perp) \rightarrow \varphi$  along with the rule *modus ponens*  $\vdash \varphi \rightarrow \psi \implies \vdash \varphi \implies \vdash \psi$ . In this axiom class we provide lemmas to obtain *Maximally Consistent Sets* via Zorn's lemma. We define the concrete classical propositional calculus inductively and show it instantiates our axiom class. We formulate the usual semantics for the propositional calculus and show strong soundness and completeness. We provide conventional definitions of the other logical connectives and prove various common identities. Finally, we show that the propositional calculus *embeds* into any logic in our axiom class.

# Contents

<b>1</b>	<b>Implication Logic</b>	<b>3</b>
1.1	Axiomatization . . . . .	3
1.2	Common Rules . . . . .	3
1.3	Lists of Assumptions . . . . .	4
1.3.1	List Implication . . . . .	4
1.3.2	Deduction From a List of Assumptions . . . . .	4
1.3.3	List Deduction as Implication Logic . . . . .	4
1.4	The Deduction Theorem . . . . .	5
1.5	Monotonic Growth in Deductive Power . . . . .	6
1.6	The Deduction Theorem Revisited . . . . .	9
1.7	Reflection . . . . .	9
1.8	The Cut Rule . . . . .	9
1.9	Sets of Assumptions . . . . .	11
1.10	Definition of Deduction . . . . .	11
1.10.1	Interpretation as Implication Logic . . . . .	11
1.11	The Deduction Theorem . . . . .	12
1.12	Monotonic Growth in Deductive Power . . . . .	13
1.13	The Deduction Theorem Revisited . . . . .	13
1.14	Reflection . . . . .	13
1.15	The Cut Rule . . . . .	13
1.16	Maximally Consistent Sets For Implication Logic . . . . .	15
<b>2</b>	<b>Classical Propositional Logic</b>	<b>19</b>
2.1	Axiomatization . . . . .	19
2.2	Common Rules . . . . .	19
2.3	Maximally Consistent Sets For Classical Logic . . . . .	22
<b>3</b>	<b>Classical Soundness and Completeness</b>	<b>26</b>
3.1	Syntax . . . . .	26
3.2	Propositional Calculus . . . . .	27
3.3	Propositional Semantics . . . . .	27
3.4	Soundness and Completeness Proofs . . . . .	28
3.5	Embedding Theorem For the Propositional Calculus . . . . .	31

<b>4</b>	<b>List Utility Theorems</b>	<b>32</b>
4.1	Multisets	32
4.2	List Mapping	37
4.3	Laws for Searching a List	40
4.4	Permutations	40
4.5	List Duplicates	42
4.6	List Subtraction	43
4.7	Tuple Lists	52
4.8	List Intersection	55
<b>5</b>	<b>Classical Logic Connectives</b>	<b>57</b>
5.1	Verum	57
5.2	Conjunction	57
5.3	Biconditional	58
5.4	Negation	59
5.5	Disjunction	60
5.6	Mutual Exclusion	61
5.7	Subtraction	61
5.8	Negated Lists	61
5.9	Common (& Uncommon) Identities	61
5.9.1	Biconditional Equivalence Relation	61
5.9.2	Biconditional Weakening	62
5.9.3	Conjunction Identities	63
5.9.4	Disjunction Identities	68
5.9.5	Monotony of Conjunction and Disjunction	72
5.9.6	Distribution Identities	73
5.9.7	Negation	76
5.9.8	Mutual Exclusion Identities	77
5.9.9	Miscellaneous Disjunctive Normal Form Identities	81

# Chapter 1

## Implication Logic

```
theory Implication-Logic  
  imports Main  
begin
```

This theory presents the pure implicational fragment of intuitionistic logic. That is to say, this is the fragment of intuitionistic logic containing *implication only*, and no other connectives nor *falsum* (i.e.,  $\perp$ ). We shall refer to this logic as *implication logic* in future discussion.

For further reference see [7].

### 1.1 Axiomatization

Implication logic can be given by the a Hilbert-style axiom system, following Troelstra and Schwichtenberg [6, §1.3.9, pg. 33].

```
class implication-logic =  
  fixes deduction :: 'a  $\Rightarrow$  bool ( $\vdash$  - [60] 55)  
  fixes implication :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (infixr  $\rightarrow$  70)  
  assumes axiom-k:  $\vdash \varphi \rightarrow \psi \rightarrow \varphi$   
  assumes axiom-s:  $\vdash (\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$   
  assumes modus-ponens:  $\vdash \varphi \rightarrow \psi \Longrightarrow \vdash \varphi \Longrightarrow \vdash \psi$ 
```

### 1.2 Common Rules

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

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

**lemma** (in *implication-logic*) *hypothetical-syllogism*:

$\vdash (\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \chi$   
**by** (*meson axiom-k axiom-s modus-ponens*)

**lemma** (in *implication-logic*) *flip-hypothetical-syllogism*:

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

**lemma** (in *implication-logic*) *implication-absorption*:

$\vdash (\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \varphi \rightarrow \psi$   
**by** (*meson axiom-k axiom-s modus-ponens*)

## 1.3 Lists of Assumptions

### 1.3.1 List Implication

Implication given a list of assumptions can be expressed recursively

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

*list-implication* :: 'a list  $\Rightarrow$  'a  $\Rightarrow$  'a (**infix**  $\rightarrow$  80) **where**  
 $\square \rightarrow \varphi = \varphi$   
 $| (\psi \# \Psi) \rightarrow \varphi = \psi \rightarrow \Psi \rightarrow \varphi$

### 1.3.2 Deduction From a List of Assumptions

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

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

$\Gamma \vdash \varphi \equiv \vdash \Gamma \rightarrow \varphi$

### 1.3.3 List Deduction as Implication Logic

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

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

**lemma** (in *implication-logic*) *list-implication-axiom-k*:

$\vdash \varphi \rightarrow \Gamma \rightarrow \varphi$   
**by** (*induct*  $\Gamma$ , (*simp, meson axiom-k axiom-s modus-ponens*)+)

**lemma** (in *implication-logic*) *list-implication-axiom-s*:

$\vdash \Gamma \rightarrow (\varphi \rightarrow \psi) \rightarrow \Gamma \rightarrow \varphi \rightarrow \Gamma \rightarrow \psi$   
**by** (*induct*  $\Gamma$ ,  
(*simp, meson axiom-k axiom-s modus-ponens hypothetical-syllogism*)+)

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

```

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

```

The following *weakening* rule can also be derived.

```

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

```

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

```

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

```

```

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

```

## 1.4 The Deduction Theorem

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

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

```

lemma (in implication-logic) list-flip-implication1:
   $\vdash (\varphi \# \Gamma) \rightarrow \chi \rightarrow \Gamma \rightarrow (\varphi \rightarrow \chi)$ 
  by (induct  $\Gamma$ ,
    (simp,
      meson
        axiom-k
        axiom-s

```

*modus-ponens*  
*flip-implication*  
*hypothetical-syllogism*)+)

**lemma** (in *implication-logic*) *list-flip-implication2*:

$\vdash \Gamma \text{ :}\rightarrow (\varphi \rightarrow \chi) \rightarrow (\varphi \# \Gamma) \text{ :}\rightarrow \chi$

**by** (*induct*  $\Gamma$ ,  
*(simp,*  
*meson*  
*axiom-k*  
*axiom-s*  
*modus-ponens*  
*flip-implication*  
*hypothetical-syllogism*)+)

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

**theorem** (in *implication-logic*) *list-deduction-theorem*:

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

**unfolding** *list-deduction-def*

**by** (*metis modus-ponens list-flip-implication1 list-flip-implication2*)

## 1.5 Monotonic Growth in Deductive Power

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

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

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

**lemma** (in *implication-logic*) *list-implication-removeAll*:

$\vdash \Gamma \text{ :}\rightarrow \psi \rightarrow (\text{removeAll } \varphi \Gamma) \text{ :}\rightarrow (\varphi \rightarrow \psi)$

**proof** –

**have**  $\forall \psi. \vdash \Gamma \text{ :}\rightarrow \psi \rightarrow (\text{removeAll } \varphi \Gamma) \text{ :}\rightarrow (\varphi \rightarrow \psi)$

**proof**(*induct*  $\Gamma$ )

**case** *Nil*

**then show** *?case* **by** (*simp, meson axiom-k*)

**next**

**case** (*Cons*  $\chi \Gamma$ )



```

assume
  inductive-hypothesis:  $\forall \psi. \vdash \Gamma \Rightarrow \psi \rightarrow \text{removeAll } \varphi \Gamma \Rightarrow (\varphi \rightarrow \psi)$ 
moreover {
  assume  $\varphi \neq \chi$ 
  with inductive-hypothesis
  have  $\forall \psi. \vdash (\chi \# \Gamma) \Rightarrow \psi \rightarrow \text{removeAll } \varphi (\chi \# \Gamma) \Rightarrow (\varphi \rightarrow \psi)$ 
    by (simp, meson modus-ponens hypothetical-syllogism)
}
moreover {
  fix  $\psi$ 
  assume  $\varphi$ -equals- $\chi$ :  $\varphi = \chi$ 
  moreover with inductive-hypothesis
  have  $\vdash \Gamma \Rightarrow (\chi \rightarrow \psi) \rightarrow \text{removeAll } \varphi (\chi \# \Gamma) \Rightarrow (\varphi \rightarrow \chi \rightarrow \psi)$  by simp
  hence  $\vdash \Gamma \Rightarrow (\chi \rightarrow \psi) \rightarrow \text{removeAll } \varphi (\chi \# \Gamma) \Rightarrow (\varphi \rightarrow \psi)$ 
    by (metis
      calculation
      modus-ponens
      implication-absorption
      list-flip-implication1
      list-flip-implication2
      list-implication.simps(2))
  ultimately have  $\vdash (\chi \# \Gamma) \Rightarrow \psi \rightarrow \text{removeAll } \varphi (\chi \# \Gamma) \Rightarrow (\varphi \rightarrow \psi)$ 
    by (simp,
      metis
      modus-ponens
      hypothetical-syllogism
      list-flip-implication1
      list-implication.simps(2))
}
ultimately show ?case by simp
qed
thus ?thesis by blast
qed

```

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

**theorem** (in *implication-logic*) *list-implication-monotonic*:

*set*  $\Sigma \subseteq \text{set } \Gamma \implies \vdash \Sigma \Rightarrow \varphi \rightarrow \Gamma \Rightarrow \varphi$

**proof** –

**assume** *set*  $\Sigma \subseteq \text{set } \Gamma$

**moreover have**  $\forall \Sigma \varphi. \text{set } \Sigma \subseteq \text{set } \Gamma \longrightarrow \vdash \Sigma \Rightarrow \varphi \rightarrow \Gamma \Rightarrow \varphi$

**proof**(*induct*  $\Gamma$ )

**case** *Nil*

**then show** *?case*

**by** (*metis*

*list-implication.simps(1)*

*list-implication-axiom-k*

*set-empty*

*subset-empty*)

```

next
case (Cons  $\psi$   $\Gamma$ )
assume
  inductive-hypothesis:  $\forall \Sigma \varphi. \text{set } \Sigma \subseteq \text{set } \Gamma \longrightarrow \vdash \Sigma \rightarrow \varphi \rightarrow \Gamma \rightarrow \varphi$ 
{
  fix  $\Sigma$ 
  fix  $\varphi$ 
  assume  $\Sigma$ -subset-relation:  $\text{set } \Sigma \subseteq \text{set } (\psi \# \Gamma)$ 
  have  $\vdash \Sigma \rightarrow \varphi \rightarrow (\psi \# \Gamma) \rightarrow \varphi$ 
  proof -
    {
      assume  $\text{set } \Sigma \subseteq \text{set } \Gamma$ 
      hence ?thesis
      by (metis
          inductive-hypothesis
          axiom-k modus-ponens
          flip-implication
          list-implication.simps(2))
    }
    moreover {
      let ? $\Delta$  = removeAll  $\psi$   $\Sigma$ 
      assume  $\neg (\text{set } \Sigma \subseteq \text{set } \Gamma)$ 
      hence  $\text{set } ?\Delta \subseteq \text{set } \Gamma$ 
      using  $\Sigma$ -subset-relation by auto
      hence  $\vdash ?\Delta \rightarrow (\psi \rightarrow \varphi) \rightarrow \Gamma \rightarrow (\psi \rightarrow \varphi)$ 
      using inductive-hypothesis by auto
      hence  $\vdash ?\Delta \rightarrow (\psi \rightarrow \varphi) \rightarrow (\psi \# \Gamma) \rightarrow \varphi$ 
      by (metis
          modus-ponens
          flip-implication
          list-flip-implication2
          list-implication.simps(2))
      moreover have  $\vdash \Sigma \rightarrow \varphi \rightarrow ?\Delta \rightarrow (\psi \rightarrow \varphi)$ 
      by (simp add: local.list-implication-removeAll)
      ultimately have ?thesis
      using modus-ponens hypothetical-syllogism by blast
    }
    ultimately show ?thesis by blast
  qed
}
thus ?case by simp
qed
ultimately show ?thesis by simp
qed

```

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

**theorem** (in *implication-logic*) *list-deduction-monotonic*:  
 $\text{set } \Sigma \subseteq \text{set } \Gamma \Longrightarrow \Sigma \vdash \varphi \Longrightarrow \Gamma \vdash \varphi$

**unfolding** *list-deduction-def*  
**using** *modus-ponens list-implication-monotonic*  
**by** *blast*

## 1.6 The Deduction Theorem Revisited

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

**theorem** (in *implication-logic*) *alternate-list-deduction-theorem*:

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

**by** (*metis*  
*list-deduction-def*  
*modus-ponens*  
*filter-is-subset*  
*list-deduction-monotonic*  
*list-deduction-theorem*  
*list-implication-removeAll*  
*removeAll.simps(2)*  
*removeAll-filter-not-eq*)

## 1.7 Reflection

In logic the *reflection* principle sometimes refers to when a collection of assumptions can deduce any of its members. It is automatically derivable from  $\llbracket \text{set } \Sigma \subseteq \text{set } \Gamma; \Sigma \vdash \varphi \rrbracket \implies \Gamma \vdash \varphi$  among the other rules provided.

**lemma** (in *implication-logic*) *list-deduction-reflection*:

$\varphi \in \text{set } \Gamma \implies \Gamma \vdash \varphi$

**by** (*metis*  
*list-deduction-def*  
*insert-subset*  
*list.simps(15)*  
*list-deduction-monotonic*  
*list-implication.simps(2)*  
*list-implication-axiom-k*  
*order-refl*)

## 1.8 The Cut Rule

*Cut* is a rule commonly presented in sequent calculi, dating back to Gerhard Gentzen's *Investigations in Logical Deduction* (1935) [4]

The cut rule is not generally necessary in sequent calculi. It can often be shown that the rule can be eliminated without reducing the power of the underlying logic. However, as demonstrated by George Boolos' *Don't*

*Eliminate Cut* (1984) [3], removing the rule can often lead to very inefficient proof systems.

Here the rule is presented just as a meta theorem.

**theorem** (in *implication-logic*) *list-deduction-cut-rule*:

$(\varphi \# \Gamma) \vdash \psi \implies \Delta \vdash \varphi \implies \Gamma @ \Delta \vdash \psi$

**by** (*metis*

*(no-types, lifting)*

*Un-upper1*

*Un-upper2*

*list-deduction-modus-ponens*

*list-deduction-monotonic*

*list-deduction-theorem*

*set-append*)

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

**theorem** (in *implication-logic*) *strong-list-deduction-cut-rule*:

$(\Phi @ \Gamma) \vdash \psi \implies \forall \varphi \in \text{set } \Phi. \Delta \vdash \varphi \implies \Gamma @ \Delta \vdash \psi$

**proof** –

**have**  $\forall \psi. (\Phi @ \Gamma \vdash \psi \longrightarrow (\forall \varphi \in \text{set } \Phi. \Delta \vdash \varphi) \longrightarrow \Gamma @ \Delta \vdash \psi)$

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

**case** *Nil*

**then show** *?case*

**by** (*metis*

*Un-iff*

*append.left-neutral*

*list-deduction-monotonic*

*set-append*

*subsetI*)

**next**

**case** (*Cons*  $\chi$   $\Phi$ ) **assume** *inductive-hypothesis*:

$\forall \psi. \Phi @ \Gamma \vdash \psi \longrightarrow (\forall \varphi \in \text{set } \Phi. \Delta \vdash \varphi) \longrightarrow \Gamma @ \Delta \vdash \psi$

{

**fix**  $\psi \chi$

**assume**  $(\chi \# \Phi) @ \Gamma \vdash \psi$

**hence** *A*:  $\Phi @ \Gamma \vdash \chi \rightarrow \psi$  **using** *list-deduction-theorem* **by** *auto*

**assume**  $\forall \varphi \in \text{set } (\chi \# \Phi). \Delta \vdash \varphi$

**hence** *B*:  $\forall \varphi \in \text{set } \Phi. \Delta \vdash \varphi$

**and** *C*:  $\Delta \vdash \chi$  **by** *auto*

**from** *A B* **have**  $\Gamma @ \Delta \vdash \chi \rightarrow \psi$  **using** *inductive-hypothesis* **by** *blast*

**with** *C* **have**  $\Gamma @ \Delta \vdash \psi$

**by** (*meson*

*list.set-intros(1)*

*list-deduction-cut-rule*

*list-deduction-modus-ponens*

*list-deduction-reflection*)

}

**thus** *?case* **by** *simp*

**qed**

```

moreover assume ( $\Phi @ \Gamma$ )  $:\vdash \psi$ 
moreover assume  $\forall \varphi \in \text{set } \Phi. \Delta :\vdash \varphi$ 
ultimately show ?thesis by blast
qed

```

## 1.9 Sets of Assumptions

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

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

### 1.10 Definition of Deduction

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

**definition** (in *implication-logic*) *set-deduction*  $:: 'a \text{ set} \Rightarrow 'a \Rightarrow \text{bool}$  (**infix**  $\vdash$  60)  
**where**  
 $\Gamma \vdash \varphi \equiv \exists \Psi. \text{set } \Psi \subseteq \Gamma \wedge \Psi :\vdash \varphi$

#### 1.10.1 Interpretation as Implication Logic

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

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

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

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

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

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

**lemma** (in *implication-logic*) *set-deduction-modus-ponens*:  
 $\Gamma \vdash \varphi \rightarrow \psi \Longrightarrow \Gamma \vdash \varphi \Longrightarrow \Gamma \vdash \psi$

**proof** –  
**assume**  $\Gamma \vdash \varphi \rightarrow \psi$   
**then obtain**  $\Phi$  **where**  $A: \text{set } \Phi \subseteq \Gamma$  **and**  $B: \Phi :\vdash \varphi \rightarrow \psi$

```

    using set-deduction-def by blast
  assume  $\Gamma \Vdash \varphi$ 
  then obtain  $\Psi$  where  $C: \text{set } \Psi \subseteq \Gamma$  and  $D: \Psi \vdash \varphi$ 
    using set-deduction-def by blast
  from  $B D$  have  $\Phi @ \Psi \vdash \psi$ 
    using list-deduction-cut-rule list-deduction-theorem by blast
  moreover from  $A C$  have  $\text{set } (\Phi @ \Psi) \subseteq \Gamma$  by simp
  ultimately show ?thesis
    using set-deduction-def by blast
qed

```

```

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

```

## 1.11 The Deduction Theorem

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

```

theorem (in implication-logic) set-deduction-theorem:
  insert  $\varphi \Gamma \Vdash \psi = \Gamma \Vdash \varphi \rightarrow \psi$ 
proof -
  have  $\Gamma \Vdash \varphi \rightarrow \psi \implies \text{insert } \varphi \Gamma \Vdash \psi$ 
    by (metis
      set-deduction-def
      insert-mono
      list.simps(15)
      list-deduction-theorem)
  moreover {
    assume  $\text{insert } \varphi \Gamma \Vdash \psi$ 
    then obtain  $\Phi$  where  $\text{set } \Phi \subseteq \text{insert } \varphi \Gamma$  and  $\Phi \vdash \psi$ 
      using set-deduction-def by auto
    hence  $\text{set } (\text{removeAll } \varphi \Phi) \subseteq \Gamma$  by auto
    moreover from  $\langle \Phi \vdash \psi \rangle$  have  $\text{removeAll } \varphi \Phi \vdash \varphi \rightarrow \psi$ 
      using modus-ponens list-implication-removeAll list-deduction-def
      by blast
  }

```

```

    ultimately have  $\Gamma \Vdash \varphi \rightarrow \psi$ 
      using set-deduction-def by blast
  }
  ultimately show insert  $\varphi$   $\Gamma \Vdash \psi = \Gamma \Vdash \varphi \rightarrow \psi$  by metis
qed

```

## 1.12 Monotonic Growth in Deductive Power

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

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

## 1.13 The Deduction Theorem Revisited

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

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

## 1.14 Reflection

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

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

## 1.15 The Cut Rule

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

First, the weak form of the rule is established.

**theorem** (in *implication-logic*) *set-deduction-cut-rule*:

*insert*  $\varphi \Gamma \Vdash \psi \implies \Delta \Vdash \varphi \implies \Gamma \cup \Delta \Vdash \psi$

**proof** –

**assume** *insert*  $\varphi \Gamma \Vdash \psi$

**hence**  $\Gamma \Vdash \varphi \rightarrow \psi$  **using** *set-deduction-theorem* **by** *auto*

**hence**  $\Gamma \cup \Delta \Vdash \varphi \rightarrow \psi$  **using** *set-deduction-def* **by** *auto*

**moreover assume**  $\Delta \Vdash \varphi$

**hence**  $\Gamma \cup \Delta \Vdash \varphi$  **using** *set-deduction-def* **by** *auto*

**ultimately show** *?thesis* **using** *set-deduction-modus-ponens* **by** *metis*

**qed**

Another lemma is shown next in order to establish the strong form of the cut rule. The lemma shows the existence of a *covering list* of assumptions  $\Psi$  in the event some set of assumptions  $\Delta$  proves everything in a finite set of assumptions  $\Phi$ .

**lemma** (in *implication-logic*) *finite-set-deduction-list-deduction*:

**assumes** *finite*  $\Phi$

**and**  $\forall \varphi \in \Phi. \Delta \Vdash \varphi$

**shows**  $\exists \Psi. \text{set } \Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi. \Psi \vdash \varphi)$

**using** *assms*

**proof**(*induct*  $\Phi$  *rule: finite-induct*)

**case empty** **thus** *?case* **by** (*metis all-not-in-conv empty-subsetI set-empty*)

**next**

**case** (*insert*  $\chi \Phi$ )

**assume**  $\forall \varphi \in \Phi. \Delta \Vdash \varphi \implies \exists \Psi. \text{set } \Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi. \Psi \vdash \varphi)$

**and**  $\forall \varphi \in \text{insert } \chi \Phi. \Delta \Vdash \varphi$

**hence**  $\exists \Psi. \text{set } \Psi \subseteq \Delta \wedge (\forall \varphi \in \Phi. \Psi \vdash \varphi)$  **and**  $\Delta \Vdash \chi$  **by** *simp+*

**then obtain**  $\Psi_1 \Psi_2$  **where**

*set*  $(\Psi_1 @ \Psi_2) \subseteq \Delta$

$\forall \varphi \in \Phi. \Psi_1 \vdash \varphi$

$\Psi_2 \vdash \chi$

**using** *set-deduction-def* **by** *auto*

**moreover from** *this* **have**  $\forall \varphi \in (\text{insert } \chi \Phi). \Psi_1 @ \Psi_2 \vdash \varphi$

**by** (*metis*

*insert-iff*

*le-sup-iff*

*list-deduction-monotonic*

*order-refl set-append*)

**ultimately show** *?case* **by** *blast*

**qed**

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

**theorem** (in *implication-logic*) *strong-set-deduction-cut-rule*:

**assumes**  $\Phi \cup \Gamma \Vdash \psi$

**and**  $\forall \varphi \in \Phi. \Delta \Vdash \varphi$

**shows**  $\Gamma \cup \Delta \Vdash \psi$

**proof** –

**obtain**  $\Sigma$  **where**



```

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

```

## 1.16 Maximally Consistent Sets For Implication Logic

*Maximally Consistent Sets* are a common construction for proving completeness of logical calculi. For a classic presentation, see Dirk van Dalen's *Logic and Structure* (2013, §1.5, pgs. 42–45) [8].

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

The models we are centrally concerned are derived from maximally consistent sets. These include probability measures used in completeness theorems of probability logic found in §??, as well as arbitrage protection and trading strategies stipulated by our formulation of the *Dutch Book Theorem* we present in §??.

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

**definition** (in *implication-logic*)

*formula-consistent* :: 'a  $\Rightarrow$  'a set  $\Rightarrow$  bool (*--consistent - [100] 100*)

**where**

[*simp*]:  $\varphi$ -consistent  $\Gamma \equiv \neg (\Gamma \Vdash \varphi)$

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

**definition** (in *implication-logic*)

*formula-maximally-consistent-set-def* :: 'a  $\Rightarrow$  'a set  $\Rightarrow$  bool (–MCS - [100] 100)  
**where**  
 [simp]:  $\varphi$ -MCS  $\Gamma \equiv (\varphi$ -consistent  $\Gamma) \wedge (\forall \psi. \psi \in \Gamma \vee (\psi \rightarrow \varphi) \in \Gamma)$

Every consistent set  $\Gamma$  may be extended to a maximally consistent set.

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

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

**lemma** (in *implication-logic*) *formula-consistent-extension*:

**assumes**  $\varphi$ -consistent  $\Gamma$   
**shows**  $(\varphi$ -consistent (insert  $\psi$   $\Gamma)) \vee (\varphi$ -consistent (insert  $(\psi \rightarrow \varphi)$   $\Gamma))$   
**proof** –  
 {  
   **assume**  $\neg \varphi$ -consistent insert  $\psi$   $\Gamma$   
   **hence**  $\Gamma \Vdash \psi \rightarrow \varphi$   
   **using** *set-deduction-theorem*  
   **unfolding** *formula-consistent-def*  
   **by** *simp*  
   **hence**  $\varphi$ -consistent insert  $(\psi \rightarrow \varphi)$   $\Gamma$   
   **by** (*metis Un-absorb assms formula-consistent-def set-deduction-cut-rule*)  
 }  
**thus** ?thesis **by** *blast*  
**qed**

**theorem** (in *implication-logic*) *formula-maximally-consistent-extension*:

**assumes**  $\varphi$ -consistent  $\Gamma$   
**shows**  $\exists \Omega. (\varphi$ -MCS  $\Omega) \wedge \Gamma \subseteq \Omega$   
**proof** –  
**let** ? $\Gamma$ -extensions =  $\{\Sigma. (\varphi$ -consistent  $\Sigma) \wedge \Gamma \subseteq \Sigma\}$   
**have**  $\exists \Omega \in$  ? $\Gamma$ -extensions.  $\forall \Sigma \in$  ? $\Gamma$ -extensions.  $\Omega \subseteq \Sigma \longrightarrow \Sigma = \Omega$   
**proof** (*rule subset-Zorn*)  
**fix**  $\mathcal{C} ::$  'a set set  
**assume** *subset-chain-C*: *subset.chain* ? $\Gamma$ -extensions  $\mathcal{C}$   
**hence**  $\mathcal{C}$ :  $\forall \Sigma \in \mathcal{C}. \Gamma \subseteq \Sigma \wedge \forall \Sigma \in \mathcal{C}. \varphi$ -consistent  $\Sigma$   
**unfolding** *subset.chain-def*  
**by** *blast+*  
**show**  $\exists \Omega \in$  ? $\Gamma$ -extensions.  $\forall \Sigma \in \mathcal{C}. \Sigma \subseteq \Omega$   
**proof** *cases*  
**assume**  $\mathcal{C} = \{\}$  **thus** ?thesis **using** *assms* **by** *blast*  
**next**  
**let** ? $\Omega = \bigcup \mathcal{C}$   
**assume**  $\mathcal{C} \neq \{\}$   
**hence**  $\Gamma \subseteq$  ? $\Omega$  **by** (*simp add: C(1) less-eq-Sup*)  
**moreover** **have**  $\varphi$ -consistent ? $\Omega$   
**proof** –  
 {

```

assume  $\neg \varphi$ -consistent  $?\Omega$ 
then obtain  $\omega$  where  $\omega$ :
  finite  $\omega$ 
   $\omega \subseteq ?\Omega$ 
   $\neg \varphi$ -consistent  $\omega$ 
  unfolding
    formula-consistent-def
    set-deduction-def
  by auto
from  $\omega(1)$   $\omega(2)$  have  $\exists \Sigma \in \mathcal{C}. \omega \subseteq \Sigma$ 
proof (induct  $\omega$  rule: finite-induct)
  case empty thus  $?\text{case}$  using  $\langle \mathcal{C} \neq \{\} \rangle$  by blast
next
  case (insert  $\psi$   $\omega$ )
  from this obtain  $\Sigma_1$   $\Sigma_2$  where
     $\Sigma_1$ :
       $\omega \subseteq \Sigma_1$ 
       $\Sigma_1 \in \mathcal{C}$ 
    and  $\Sigma_2$ :
       $\psi \in \Sigma_2$ 
       $\Sigma_2 \in \mathcal{C}$ 
    by auto
  hence  $\Sigma_1 \subseteq \Sigma_2 \vee \Sigma_2 \subseteq \Sigma_1$ 
  using subset-chain-C
  unfolding subset.chain-def
  by blast
  hence (insert  $\psi$   $\omega$ )  $\subseteq \Sigma_1 \vee$  (insert  $\psi$   $\omega$ )  $\subseteq \Sigma_2$ 
  using  $\Sigma_1$   $\Sigma_2$  by blast
  thus  $?\text{case}$  using  $\Sigma_1$   $\Sigma_2$  by blast
qed
hence  $\exists \Sigma \in \mathcal{C}. (\varphi$ -consistent  $\Sigma) \wedge \neg (\varphi$ -consistent  $\Sigma)$ 
  using  $\mathcal{C}(2)$   $\omega(3)$ 
  unfolding
    formula-consistent-def
    set-deduction-def
  by auto
  hence False by auto
}
thus  $?\text{thesis}$  by blast
qed
ultimately show  $?\text{thesis}$  by blast
qed
qed
then obtain  $\Omega$  where  $\Omega$ :
   $\Omega \in ?\Gamma$ -extensions
   $\forall \Sigma \in ?\Gamma$ -extensions.  $\Omega \subseteq \Sigma \longrightarrow \Sigma = \Omega$ 
  by auto+
{
  fix  $\psi$ 

```

```

have ( $\varphi$ -consistent insert  $\psi$   $\Omega$ )  $\vee$  ( $\varphi$ -consistent insert ( $\psi \rightarrow \varphi$ )  $\Omega$ )
   $\Gamma \subseteq$  insert  $\psi$   $\Omega$ 
   $\Gamma \subseteq$  insert ( $\psi \rightarrow \varphi$ )  $\Omega$ 
using  $\Omega(1)$  formula-consistent-extension formula-consistent-def
by auto
hence insert  $\psi$   $\Omega \in$   $\mathcal{T}$ -extensions
   $\vee$  insert ( $\psi \rightarrow \varphi$ )  $\Omega \in$   $\mathcal{T}$ -extensions
by blast
hence  $\psi \in \Omega \vee (\psi \rightarrow \varphi) \in \Omega$  using  $\Omega(2)$  by blast
}
thus ?thesis
using  $\Omega(1)$ 
unfolding formula-maximally-consistent-set-def-def
by blast
qed

```

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

**lemma** (in *implication-logic*) *formula-maximally-consistent-set-def-reflection*:

$\varphi$ -MCS  $\Gamma \implies \psi \in \Gamma = \Gamma \Vdash \psi$

**proof** –

**assume**  $\varphi$ -MCS  $\Gamma$

{

**assume**  $\Gamma \Vdash \psi$

**moreover from**  $\langle \varphi$ -MCS  $\Gamma \rangle$  **have**  $\psi \in \Gamma \vee (\psi \rightarrow \varphi) \in \Gamma \neg \Gamma \Vdash \varphi$

**unfolding**

formula-maximally-consistent-set-def-def

formula-consistent-def

**by** auto

**ultimately have**  $\psi \in \Gamma$

**using** set-deduction-reflection set-deduction-modus-ponens

**by** metis

}

**thus**  $\psi \in \Gamma = \Gamma \Vdash \psi$

**using** set-deduction-reflection

**by** metis

**qed**

**theorem** (in *implication-logic*) *formula-maximally-consistent-set-def-implication-elimination*:

**assumes**  $\varphi$ -MCS  $\Omega$

**shows**  $(\psi \rightarrow \chi) \in \Omega \implies \psi \in \Omega \implies \chi \in \Omega$

**using**

assms

formula-maximally-consistent-set-def-reflection

set-deduction-modus-ponens

**by** blast

This concludes our introduction to implication logic.

**end**

## Chapter 2

# Classical Propositional Logic

```
theory Classical-Logic  
  imports Implication-Logic  
begin
```

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

### 2.1 Axiomatization

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

```
class classical-logic = implication-logic +  
  fixes falsum :: 'a ( $\perp$ )  
  assumes double-negation:  $\vdash (((\varphi \rightarrow \perp) \rightarrow \perp) \rightarrow \varphi)$ 
```

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

```
class consistent-classical-logic = classical-logic +  
  assumes consistency:  $\neg \vdash \perp$ 
```

### 2.2 Common Rules

There are many common tautologies in classical logic. Once we have established *completeness* in §3, we will be able to leverage Isabelle/HOL's automation for proving these elementary results.

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

```
lemma (in classical-logic)  
  ex-falso-quodlibet:  $\vdash \perp \rightarrow \varphi$   
  using axiom-k double-negation modus-ponens hypothetical-syllogism
```

by *blast*

**lemma** (in *classical-logic*)

*Contraposition*:  $\vdash ((\varphi \rightarrow \perp) \rightarrow (\psi \rightarrow \perp)) \rightarrow \psi \rightarrow \varphi$

**proof** –

**have**  $[\varphi \rightarrow \perp, \psi, (\varphi \rightarrow \perp) \rightarrow (\psi \rightarrow \perp)] \vdash \perp$

**using** *flip-implication list-deduction-theorem list-implication.simps(1)*

**unfolding** *list-deduction-def*

**by** *presburger*

**hence**  $[\psi, (\varphi \rightarrow \perp) \rightarrow (\psi \rightarrow \perp)] \vdash (\varphi \rightarrow \perp) \rightarrow \perp$

**using** *list-deduction-theorem* **by** *blast*

**hence**  $[\psi, (\varphi \rightarrow \perp) \rightarrow (\psi \rightarrow \perp)] \vdash \varphi$

**using** *double-negation list-deduction-weaken list-deduction-modus-ponens*

**by** *blast*

**thus** *?thesis*

**using** *list-deduction-base-theory list-deduction-theorem* **by** *blast*

**qed**

**lemma** (in *classical-logic*)

*double-negation-converse*:  $\vdash \varphi \rightarrow (\varphi \rightarrow \perp) \rightarrow \perp$

**by** (*meson axiom-k modus-ponens flip-implication*)

The following lemma is sometimes referred to as *The Principle of Pseudo-Scotus*[2].

**lemma** (in *classical-logic*)

*pseudo-scotus*:  $\vdash (\varphi \rightarrow \perp) \rightarrow \varphi \rightarrow \psi$

**using** *ex-falso-quodlibet modus-ponens hypothetical-syllogism* **by** *blast*

Another popular lemma is attributed to Charles Sanders Peirce, and has come to be known as *Peirces Law*[5].

**lemma** (in *classical-logic*) *Peirces-law*:

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

**proof** –

**have**  $[\varphi \rightarrow \perp, (\varphi \rightarrow \psi) \rightarrow \varphi] \vdash \varphi \rightarrow \psi$

**using**

*pseudo-scotus*

*list-deduction-theorem*

*list-deduction-weaken*

**by** *blast*

**hence**  $[\varphi \rightarrow \perp, (\varphi \rightarrow \psi) \rightarrow \varphi] \vdash \varphi$

**by** (*meson*

*list.set-intros(1)*

*list-deduction-reflection*

*list-deduction-modus-ponens*

*set-subset-Cons*

*subsetCE*)

**hence**  $[\varphi \rightarrow \perp, (\varphi \rightarrow \psi) \rightarrow \varphi] \vdash \perp$

**by** (*meson*

*list.set-intros(1)*)

```

      list-deduction-modus-ponens
      list-deduction-reflection)
hence  $[(\varphi \rightarrow \psi) \rightarrow \varphi] : \vdash (\varphi \rightarrow \perp) \rightarrow \perp$ 
  using list-deduction-theorem by blast
hence  $[(\varphi \rightarrow \psi) \rightarrow \varphi] : \vdash \varphi$ 
  using double-negation
      list-deduction-modus-ponens
      list-deduction-weaken
  by blast
thus ?thesis
  using list-deduction-def
  by auto
qed

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

```

by *simp*  
 qed

## 2.3 Maximally Consistent Sets For Classical Logic

*Relativized* maximally consistent sets were introduced in §1.16. Often this is exactly what we want in a proof. A completeness theorem typically starts by assuming  $\varphi$  is not provable, then finding a  $\varphi$ -MCS  $\Gamma$  which gives rise to a model which does not make  $\varphi$  true.

A more conventional presentation says that  $\Gamma$  is maximally consistent if and only if  $\neg \Gamma \Vdash \perp$  and  $\forall \psi. \psi \in \Gamma \vee \psi \rightarrow \varphi \in \Gamma$ . This conventional presentation will come up when formulating MAXSAT in §??. This in turn allows us to formulate MAXSAT completeness for probability inequalities in §??. and reduce checking if a strategy will always lose money or if it will always make money if matched to bounded MAXSAT as part of our proof of the *Dutch Book Theorem* in §?? and §?? respectively.

**definition** (in *classical-logic*)

*consistent* :: 'a set  $\Rightarrow$  bool **where**  
 [*simp*]: *consistent*  $\Gamma \equiv \perp$ -consistent  $\Gamma$

**definition** (in *classical-logic*)

*maximally-consistent-set* :: 'a set  $\Rightarrow$  bool (MCS) **where**  
 [*simp*]: *MCS*  $\Gamma \equiv \perp$ -MCS  $\Gamma$

**lemma** (in *classical-logic*)

*formula-maximally-consistent-set-def-negation*:  $\varphi$ -MCS  $\Gamma \Longrightarrow \varphi \rightarrow \perp \in \Gamma$

**proof** –

**assume**  $\varphi$ -MCS  $\Gamma$   
 {  
   **assume**  $\varphi \rightarrow \perp \notin \Gamma$   
   **hence**  $(\varphi \rightarrow \perp) \rightarrow \varphi \in \Gamma$   
     **using**  $\langle \varphi$ -MCS  $\Gamma \rangle$   
     **unfolding** *formula-maximally-consistent-set-def-def*  
     **by** *blast*  
   **hence**  $\Gamma \Vdash (\varphi \rightarrow \perp) \rightarrow \varphi$   
     **using** *set-deduction-reflection*  
     **by** *simp*  
   **hence**  $\Gamma \Vdash \varphi$   
     **using**  
       *Peirces-law*  
       *set-deduction-modus-ponens*  
       *set-deduction-weaken*  
     **by** *metis*  
   **hence** *False*  
     **using**  $\langle \varphi$ -MCS  $\Gamma \rangle$



```

unfolding
  formula-maximally-consistent-set-def-def
  formula-consistent-def
  by simp
}
thus ?thesis by blast
qed

```

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

**lemma** (in *classical-logic*)

*formula-maximal-consistency*:  $(\exists \varphi. \varphi\text{-MCS } \Gamma) = \text{MCS } \Gamma$

**proof** –

```

{
  fix  $\varphi$ 
  have  $\varphi\text{-MCS } \Gamma \implies \text{MCS } \Gamma$ 
  proof –
    assume  $\varphi\text{-MCS } \Gamma$ 
    have consistent  $\Gamma$ 
    using
       $\langle \varphi\text{-MCS } \Gamma \rangle$ 
      ex-falso-quodlibet [where  $\varphi=\varphi$ ]
      set-deduction-weaken [where  $\Gamma=\Gamma$ ]
      set-deduction-modus-ponens
    unfolding
      formula-maximally-consistent-set-def-def
      consistent-def
      formula-consistent-def
    by metis
  moreover {
    fix  $\psi$ 
    have  $\psi \rightarrow \perp \notin \Gamma \implies \psi \in \Gamma$ 
    proof –
      assume  $\psi \rightarrow \perp \notin \Gamma$ 
      hence  $(\psi \rightarrow \perp) \rightarrow \varphi \in \Gamma$ 
      using  $\langle \varphi\text{-MCS } \Gamma \rangle$ 
      unfolding formula-maximally-consistent-set-def-def
      by blast
      hence  $\Gamma \Vdash (\psi \rightarrow \perp) \rightarrow \varphi$ 
      using set-deduction-reflection
      by simp
      also have  $\Gamma \Vdash \varphi \rightarrow \perp$ 
      using  $\langle \varphi\text{-MCS } \Gamma \rangle$ 
        formula-maximally-consistent-set-def-negation
        set-deduction-reflection
      by simp
      hence  $\Gamma \Vdash (\psi \rightarrow \perp) \rightarrow \perp$ 
      using calculation
        hypothetical-syllogism

```

```

      [where  $\varphi=\psi \rightarrow \perp$  and  $\psi=\varphi$  and  $\chi=\perp$ ]
      set-deduction-weaken
      [where  $\Gamma=\Gamma$ ]
      set-deduction-modus-ponens
    by metis
  hence  $\Gamma \vdash \psi$ 
  using double-negation
      [where  $\varphi=\psi$ ]
      set-deduction-weaken
      [where  $\Gamma=\Gamma$ ]
      set-deduction-modus-ponens
  by metis
  thus ?thesis
  using  $\langle \varphi\text{-MCS } \Gamma \rangle$ 
      formula-maximally-consistent-set-def-reflection
  by blast
qed
}
ultimately show ?thesis
  unfolding maximally-consistent-set-def
      formula-maximally-consistent-set-def-def
      formula-consistent-def
      consistent-def
  by blast
qed
}
thus ?thesis
  unfolding maximally-consistent-set-def
  by metis
qed

```

Finally, classical logic allows us to strengthen  $\llbracket \varphi\text{-MCS } \Omega; \psi \rightarrow \chi \in \Omega; \psi \in \Omega \rrbracket \implies \chi \in \Omega$  to a biconditional.

**lemma** (in *classical-logic*)

*formula-maximally-consistent-set-def-implication:*

**assumes**  $\varphi\text{-MCS } \Gamma$

**shows**  $\psi \rightarrow \chi \in \Gamma = (\psi \in \Gamma \longrightarrow \chi \in \Gamma)$

**proof** –

```

{
  assume hypothesis:  $\psi \in \Gamma \longrightarrow \chi \in \Gamma$ 
  {
    assume  $\psi \notin \Gamma$ 
    have  $\forall \psi. \varphi \rightarrow \psi \in \Gamma$ 
      by (meson assms
          formula-maximally-consistent-set-def-negation
          formula-maximally-consistent-set-def-implication-elimination
          formula-maximally-consistent-set-def-reflection
          pseudo-scotus set-deduction-weaken)
    then have  $\forall \chi \psi. \text{insert } \chi \Gamma \vdash \psi \vee \chi \rightarrow \varphi \notin \Gamma$ 

```

```

by (meson assms
      axiom-k
      formula-maximally-consistent-set-def-reflection
      set-deduction-modus-ponens
      set-deduction-theorem
      set-deduction-weaken)
hence  $\psi \rightarrow \chi \in \Gamma$ 
by (meson  $\langle \psi \notin \Gamma \rangle$ 
      assms
      formula-maximally-consistent-set-def-def
      formula-maximally-consistent-set-def-reflection
      set-deduction-theorem)
}
moreover {
  assume  $\chi \in \Gamma$ 
  hence  $\psi \rightarrow \chi \in \Gamma$ 
  by (metis assms
        calculation
        insert-absorb
        formula-maximally-consistent-set-def-reflection
        set-deduction-theorem)
}
ultimately have  $\psi \rightarrow \chi \in \Gamma$  using hypothesis by blast
}
thus ?thesis
using assms
      formula-maximally-consistent-set-def-implication-elimination
by metis
qed
end

```

## Chapter 3

# Classical Soundness and Completeness

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

The following presents soundness completeness of the classical propositional calculus for propositional semantics. The classical propositional calculus is sometimes referred to as the *sentential calculus*. We give a concrete algebraic data type for propositional formulae in §3.1. We inductively define a logical judgement  $\vdash_{prop}$  for these formulae. We also define the Tarski truth relation  $\models_{prop}$  inductively, which we present in §3.3.

The most significant results here are the *embedding theorems*. These theorems show that the propositional calculus can be embedded in any logic extending *classical-logic*. These theorems are proved in §3.5.

### 3.1 Syntax

Here we provide the usual language for formulae in the propositional calculus. It contains *falsum*  $\perp$ , implication ( $\rightarrow$ ), and a way of constructing *atomic* propositions  $\lambda \varphi . \langle \varphi \rangle$ . Defining the language is straight-forward using an algebraic data type.

```
datatype 'a classical-propositional-formula =  
  Falsum ( $\perp$ )  
  | Proposition 'a ( $\langle - \rangle$  [45])  
  | Implication  
    'a classical-propositional-formula  
    'a classical-propositional-formula (infixr  $\rightarrow$  70)
```

## 3.2 Propositional Calculus

In this section we recursively define what a proof is in the classical propositional calculus. We provide the familiar  $K$  and  $S$  axioms, as well as *double negation* and *modus ponens*.

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

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

Our proof system for our propositional calculus is trivially an instance of *classical-logic*. The introduction rules for  $\vdash_{prop}$  naturally reflect the axioms of the classical logic axiom class.

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

## 3.3 Propositional Semantics

Below we give the typical definition of the Tarski truth relation  $\models_{prop}$ .

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

Soundness of our calculus for these semantics is trivial.

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

### 3.4 Soundness and Completeness Proofs

**definition** *strong-classical-propositional-deduction* ::

```
'a classical-propositional-formula set
⇒ 'a classical-propositional-formula ⇒ bool
(infix  $\Vdash_{prop}$  65)
where
[simp]:  $\Gamma \Vdash_{prop} \varphi \equiv \Gamma \Vdash \varphi$ 
```

**definition** *strong-classical-propositional-tarski-truth* ::

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

**definition** *theory-propositions* ::

```
'a classical-propositional-formula set ⇒ 'a set ( $\{\!|$  -  $\!\}$  [50])
where
[simp]:  $\{\!| \Gamma \!\} = \{p \cdot \Gamma \Vdash_{prop} \langle p \rangle\}$ 
```

Below we give the main lemma for completeness: the *truth lemma*. This proof connects the maximally consistent sets developed in §1.16 and §2.3 with the semantics given in §3.3.

All together, the technique we are using essentially follows the approach by Blackburn et al. [1, §4.2, pgs. 196-201].

**lemma** *truth-lemma*:

```
assumes MCS  $\Gamma$ 
shows  $\Gamma \Vdash_{prop} \varphi \equiv \{\!| \Gamma \!\} \models_{prop} \varphi$ 
proof (induct  $\varphi$ )
case Falsum
then show ?case using assms by auto
next
case (Proposition  $x$ )
then show ?case by simp
next
case (Implication  $\psi \chi$ )
thus ?case
unfolding strong-classical-propositional-deduction-def
by (metis
assms
maximally-consistent-set-def
formula-maximally-consistent-set-def-implication
classical-propositional-semantic.simps(2)
implication-classical-propositional-formula-def
set-deduction-modus-ponens
set-deduction-reflection)
qed
```

Here the truth lemma above is combined with  $\varphi$ -consistent  $\Gamma \implies \exists \Omega$ .  $\varphi$ -MCS  $\Omega \wedge \Gamma \subseteq \Omega$  proven in §3.3. These theorems together give rise to strong completeness for the propositional calculus.

**theorem** *classical-propositional-calculus-strong-soundness-and-completeness:*

$\Gamma \Vdash_{prop} \varphi = \Gamma \Vdash_{prop} \varphi$

**proof** –

**have** *soundness*:  $\Gamma \Vdash_{prop} \varphi \implies \Gamma \Vdash_{prop} \varphi$

**proof** –

**assume**  $\Gamma \Vdash_{prop} \varphi$

**from** *this* **obtain**  $\Gamma'$  **where**  $\Gamma'$ : set  $\Gamma' \subseteq \Gamma$   $\Gamma' \vdash \varphi$

**by** (*simp add: set-deduction-def, blast*)

{

**fix**  $\mathfrak{M}$

**assume**  $\forall \gamma \in \Gamma. \mathfrak{M} \models_{prop} \gamma$

**hence**  $\forall \gamma \in \text{set } \Gamma'. \mathfrak{M} \models_{prop} \gamma$  **using**  $\Gamma'(1)$  **by** *auto*

**hence**  $\forall \varphi. \Gamma' \vdash \varphi \longrightarrow \mathfrak{M} \models_{prop} \varphi$

**proof** (*induct*  $\Gamma'$ )

**case** *Nil*

**then show** *?case*

**by** (*simp add:*

*classical-propositional-calculus-soundness*

*list-deduction-def*)

**next**

**case** (*Cons*  $\psi$   $\Gamma'$ )

**thus** *?case* **using** *list-deduction-theorem* **by** *fastforce*

**qed**

**with**  $\Gamma'(2)$  **have**  $\mathfrak{M} \models_{prop} \varphi$  **by** *blast*

}

**thus**  $\Gamma \Vdash_{prop} \varphi$

**using** *strong-classical-propositional-tarski-truth-def* **by** *blast*

**qed**

**have** *completeness*:  $\Gamma \Vdash_{prop} \varphi \implies \Gamma \Vdash_{prop} \varphi$

**proof** (*erule contrapos-pp*)

**assume**  $\neg \Gamma \Vdash_{prop} \varphi$

**hence**  $\exists \mathfrak{M}. (\forall \gamma \in \Gamma. \mathfrak{M} \models_{prop} \gamma) \wedge \neg \mathfrak{M} \models_{prop} \varphi$

**proof** –

**from**  $\langle \neg \Gamma \Vdash_{prop} \varphi \rangle$  **obtain**  $\Omega$  **where**  $\Omega$ :  $\Gamma \subseteq \Omega$   $\varphi$ -MCS  $\Omega$

**by** (*meson*

*formula-consistent-def*

*formula-maximally-consistent-extension*

*strong-classical-propositional-deduction-def*)

**hence**  $(\varphi \rightarrow \perp) \in \Omega$

**using** *formula-maximally-consistent-set-def-negation* **by** *blast*

**hence**  $\neg \{\!\! \{ \Omega \}\!\! \} \models_{prop} \varphi$

**using**  $\Omega$

*formula-consistent-def*

*formula-maximal-consistency*

*formula-maximally-consistent-set-def-def*

*truth-lemma*

```

    unfolding strong-classical-propositional-deduction-def
  by blast
moreover have  $\forall \gamma \in \Gamma. \{ \Omega \} \models_{prop} \gamma$ 
  using
    formula-maximal-consistency
    truth-lemma
     $\Omega$ 
    set-deduction-reflection
  unfolding strong-classical-propositional-deduction-def
  by blast
ultimately show ?thesis by auto
qed
thus  $\neg \Gamma \models_{prop} \varphi$ 
  unfolding strong-classical-propositional-tarski-truth-def
  by simp
qed
from soundness completeness show  $\Gamma \vdash_{prop} \varphi = \Gamma \models_{prop} \varphi$ 
  by linarith
qed

```

For our applications in §sec:propositional-embedding, we will only need a weaker form of soundness and completeness rather than the stronger form proved above.

```

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

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

```



### 3.5 Embedding Theorem For the Propositional Calculus

A recurring technique to prove theorems in logic moving forward is *embed* our theorem into the classical propositional calculus.

Using our embedding, we can leverage completeness to turn our problem into semantics and dispatch to Isabelle/HOL's classical theorem provers.

In future work we may make a tactic for this, but for now we just manually leverage the technique throughout our subsequent proofs.

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

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

The following theorem in particular shows that it suffices to prove theorems using classical semantics to prove theorems about the logic under investigation.

**theorem** (in *classical-logic*) *propositional-semantics*:  
 $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} \varphi \Longrightarrow \vdash \langle \langle \varphi \rangle \rangle$   
**by** (*simp add:*  
*classical-propositional-calculus-soundness-and-completeness*  
*propositional-calculus*)

**end**

# Chapter 4

## List Utility Theorems

```
theory List-Utilities
  imports
    HOL-Combinatorics.List-Permutation
begin
```

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

### 4.1 Multisets

```
lemma length-sub-mset:
  assumes mset  $\Psi \subseteq\#$  mset  $\Gamma$ 
    and length  $\Psi \geq$  length  $\Gamma$ 
  shows mset  $\Psi =$  mset  $\Gamma$ 
using assms
by (metis
  append-Nil2
  append-eq-append-conv
  leD
  linorder-neqE-nat
  mset-le-perm-append
  perm-length
  size-mset
  size-mset-mono)

lemma set-exclusion-mset-simplify:
  assumes  $\neg (\exists \psi \in \text{set } \Psi. \psi \in \text{set } \Sigma)$ 
    and mset  $\Psi \subseteq\#$  mset  $(\Sigma @ \Gamma)$ 
  shows mset  $\Psi \subseteq\#$  mset  $\Gamma$ 
using assms
proof (induct  $\Sigma$ )
  case Nil
  then show ?case by simp
```

```

next
  case (Cons  $\sigma$   $\Sigma$ )
  then show ?case
    by (cases  $\sigma \in \text{set } \Psi$ ,
      fastforce,
      metis
      add.commute
      add-mset-add-single
      diff-single-trivial
      in-multiset-in-set
      mset.simps(2)
      notin-set-remove1
      remove-hd
      subset-eq-diff-conv
      union-code
      append-Cons)

```

**qed**

**lemma** *image-mset-cons-homomorphism*:  
 $\text{image-mset mset (image-mset ((\#) \varphi) \Phi) = image-mset ((+) \{\# \varphi \#\}) (image-mset mset \Phi)}$   
**by** (*induct*  $\Phi$ , *simp+*)

**lemma** *image-mset-append-homomorphism*:  
 $\text{image-mset mset (image-mset ((@) \Delta) \Phi) = image-mset ((+) (mset \Delta)) (image-mset mset \Phi)}$   
**by** (*induct*  $\Phi$ , *simp+*)

**lemma** *image-mset-add-collapse*:  
**fixes**  $A B :: 'a \text{ multiset}$   
**shows**  $\text{image-mset ((+) A) (image-mset ((+) B) X) = image-mset ((+) (A + B)) X$   
**by** (*induct*  $X$ , *simp*, *simp*)

**lemma** *remove1-remdups-removeAll*:  $\text{remove1 } x (\text{remdups } A) = \text{remdups (removeAll } x A)$

**proof** (*induct*  $A$ )  
**case** *Nil*  
**then show** ?*case* **by** *simp*

**next**

**case** (*Cons*  $a$   $A$ )  
**then show** ?*case*  
**by** (*cases*  $a = x$ , (*simp add: Cons*) $+$ )

**qed**

**lemma** *mset-remdups*:  
**assumes**  $\text{mset } A = \text{mset } B$   
**shows**  $\text{mset (remdups } A) = \text{mset (remdups } B)$   
**proof** –

```

have  $\forall B. \text{mset } A = \text{mset } B \longrightarrow \text{mset } (\text{remdups } A) = \text{mset } (\text{remdups } B)$ 
proof (induct A)
  case Nil
  then show ?case by simp
next
  case (Cons a A)
  {
    fix B
    assume  $\text{mset } (a \# A) = \text{mset } B$ 
    hence  $\text{mset } A = \text{mset } (\text{remove1 } a B)$ 
    by (metis add-mset-add-mset-same-iff
      list.set-intros(1)
      mset.simps(2)
      mset-eq-setD
      perm-remove)
    hence  $\text{mset } (\text{remdups } A) = \text{mset } (\text{remdups } (\text{remove1 } a B))$ 
    using Cons.hyps by blast
    hence  $\text{mset } (\text{remdups } (a \# (\text{remdups } A))) = \text{mset } (\text{remdups } (a \# (\text{remdups } (\text{remove1 } a B))))$ 
    by (metis mset-eq-setD set-eq-iff-mset-remdups-eq list.simps(15))
    hence  $\text{mset } (\text{remdups } (a \# (\text{removeAll } a (\text{remdups } A))))$ 
       $= \text{mset } (\text{remdups } (a \# (\text{removeAll } a (\text{remdups } (\text{remove1 } a B))))$ 
    by (metis insert-Diff-single list.set(2) set-eq-iff-mset-remdups-eq set-removeAll)
    hence  $\text{mset } (\text{remdups } (a \# (\text{remdups } (\text{removeAll } a A))))$ 
       $= \text{mset } (\text{remdups } (a \# (\text{remdups } (\text{removeAll } a (\text{remove1 } a B))))$ 
    by (metis distinct-remdups distinct-remove1-removeAll remove1-remdups-removeAll)
    hence  $\text{mset } (\text{remdups } (\text{remdups } (a \# A))) = \text{mset } (\text{remdups } (\text{remdups } (a \# (\text{remove1 } a B))))$ 
    by (metis  $\langle \text{mset } A = \text{mset } (\text{remove1 } a B) \rangle$ 
      list.set(2)
      mset-eq-setD
      set-eq-iff-mset-remdups-eq)
    hence  $\text{mset } (\text{remdups } (a \# A)) = \text{mset } (\text{remdups } (a \# (\text{remove1 } a B)))$ 
    by (metis remdups-remdups)
    hence  $\text{mset } (\text{remdups } (a \# A)) = \text{mset } (\text{remdups } B)$ 
    using  $\langle \text{mset } (a \# A) = \text{mset } B \rangle$  mset-eq-setD set-eq-iff-mset-remdups-eq by
blast
  }
  then show ?case by simp
qed
thus ?thesis using assms by blast
qed

lemma mset-mset-map-snd-remdups:
  assumes  $\text{mset } (\text{map } \text{mset } A) = \text{mset } (\text{map } \text{mset } B)$ 
  shows  $\text{mset } (\text{map } (\text{mset } \circ (\text{map } \text{snd}) \circ \text{remdups}) A) = \text{mset } (\text{map } (\text{mset } \circ (\text{map } \text{snd}) \circ \text{remdups}) B)$ 
proof –
  {

```

```

fix B :: ('a × 'b) list list
fix b :: ('a × 'b) list
assume b ∈ set B
hence mset (map (mset ∘ (map snd) ∘ remdups) (b # (remove1 b B)))
      = mset (map (mset ∘ (map snd) ∘ remdups) B)
proof (induct B)
  case Nil
  then show ?case by simp
next
  case (Cons b' B)
  then show ?case
  by (cases b = b', simp+)
  qed
}
note ◇ = this
have
  ∀ B :: ('a × 'b) list list.
  mset (map mset A) = mset (map mset B)
  → mset (map (mset ∘ (map snd) ∘ remdups) A) = mset (map (mset ∘ (map
snd) ∘ remdups) B)
proof (induct A)
  case Nil
  then show ?case by simp
next
  case (Cons a A)
  {
  fix B
  assume ♠: mset (map mset (a # A)) = mset (map mset B)
  hence mset a ∈# mset (map mset B)
  by (simp,
      metis ♠
      image-set
      list.set-intros(1)
      list.simps(9)
      mset-eq-setD)
  from this obtain b where †:
    b ∈ set B
    mset a = mset b
    by auto
  with ♠ have mset (map mset A) = mset (remove1 (mset b) (map mset B))
  by (simp add: union-single-eq-diff)
  moreover have mset B = mset (b # remove1 b B) using † by simp
  hence mset (map mset B) = mset (map mset (b # (remove1 b B)))
  by (simp,
      metis image-mset-add-mset
      mset.simps(2)
      mset-remove1)
  ultimately have mset (map mset A) = mset (map mset (remove1 b B))
  by simp

```

**hence**  $mset (map (mset \circ (map snd) \circ remdups) A)$   
 $= mset (map (mset \circ (map snd) \circ remdups) (remove1 b B))$   
**using** *Cons.hyps* **by** *blast*  
**moreover have**  $(mset \circ (map snd) \circ remdups) a = (mset \circ (map snd) \circ$   
 $remdups) b$   
**using**  $\dagger(2)$  *mset-remdups* **by** *fastforce*  
**ultimately have**  
 $mset (map (mset \circ (map snd) \circ remdups) (a \# A))$   
 $= mset (map (mset \circ (map snd) \circ remdups) (b \# (remove1 b B)))$   
**by** *simp*  
**moreover have**  
 $mset (map (mset \circ (map snd) \circ remdups) (b \# (remove1 b B)))$   
 $= mset (map (mset \circ (map snd) \circ remdups) B)$   
**using**  $\dagger(1)$   $\diamond$  **by** *blast*  
**ultimately have**  
 $mset (map (mset \circ (map snd) \circ remdups) (a \# A))$   
 $= mset (map (mset \circ (map snd) \circ remdups) B)$   
**by** *simp*  
**}**  
**then show** *?case* **by** *blast*  
**qed**  
**thus** *?thesis* **using** *assms* **by** *blast*  
**qed**

**lemma** *mset-remdups-append-msub*:

$mset (remdups A) \subseteq\# mset (remdups (B @ A))$

**proof** –

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

**proof** (*induct A*)

**case** *Nil*

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

**next**

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

**{**

**fix** *B*

**have**  $\dagger: mset (remdups (B @ (a \# A))) = mset (remdups (a \# (B @ A)))$

**by** (*induct B, simp+*)

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

**proof** (*cases a \in set B \wedge a \notin set A*)

**case** *True*

**hence**  $\dagger: mset (remove1 a (remdups (B @ A))) = mset (remdups ((removeAll$   
 $a B) @ A))$

**by** (*simp add: remove1-remdups-removeAll*)

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

$= (mset (remdups A) \subseteq\# mset (remdups ((removeAll a B) @ A)))$

**using** *True*

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

**then show** *?thesis*

**by** (*metis \dagger Cons True*)

```

      Un-insert-right
      list.set(2)
      mset.simps(2)
      mset-subset-eq-insertD
      remdups.simps(2)
      set-append
      set-eq-iff-mset-remdups-eq
      set-mset-mset set-remdups)
next
  case False
  then show ?thesis using † Cons by simp
qed
}
thus ?case by blast
qed
thus ?thesis by blast
qed

```

## 4.2 List Mapping

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

**notation**  $perm$  (**infix**  $\rightleftharpoons$  50)

**lemma** *map-monotonic*:

```

  assumes mset A  $\subseteq\#$  mset B
  shows mset (map f A)  $\subseteq\#$  mset (map f B)
  by (simp add: assms image-mset-subseteq-mono)

```

**lemma** *perm-map-perm-list-exists*:

```

  assumes A  $\rightleftharpoons$  map f B
  shows  $\exists B'. A = map f B' \wedge B' \rightleftharpoons B$ 
proof –
  have  $\forall B. A \rightleftharpoons map f B \longrightarrow (\exists B'. A = map f B' \wedge B' \rightleftharpoons B)$ 
proof (induct A)
  case Nil
  then show ?case by simp
next
  case (Cons a A)
  {
  fix B
  assume a  $\#$  A  $\rightleftharpoons$  map f B
  from this obtain b where b:
    b  $\in$  set B
    f b = a
  by (metis
      (full-types)
      imageE)
  }

```

```

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

lemma mset-sub-map-list-exists:
assumes  $\text{mset } \Phi \subseteq\# \ \text{mset } (\text{map } f \ \Gamma)$ 
shows  $\exists \Phi'. \text{mset } \Phi' \subseteq\# \ \text{mset } \Gamma \wedge \Phi = (\text{map } f \ \Phi')$ 
proof –
have  $\forall \Phi. \text{mset } \Phi \subseteq\# \ \text{mset } (\text{map } f \ \Gamma)$ 
       $\longrightarrow (\exists \Phi'. \text{mset } \Phi' \subseteq\# \ \text{mset } \Gamma \wedge \Phi = (\text{map } f \ \Phi'))$ 
proof (induct  $\Gamma$ )
case Nil
then show ?case by simp
next
case (Cons  $\gamma \ \Gamma$ )
{
fix  $\Phi$ 
assume  $\text{mset } \Phi \subseteq\# \ \text{mset } (\text{map } f \ (\gamma \ \# \ \Gamma))$ 
have  $\exists \Phi'. \text{mset } \Phi' \subseteq\# \ \text{mset } (\gamma \ \# \ \Gamma) \wedge \Phi = \text{map } f \ \Phi'$ 
proof cases
assume  $f \ \gamma \in \text{set } \Phi$ 

```



```

hence  $f \gamma \# (\text{remove1 } (f \gamma) \Phi) \equiv \Phi$ 
  by force
with  $\langle \text{mset } \Phi \subseteq\# \text{mset } (\text{map } f (\gamma \# \Gamma)) \rangle$ 
have  $\text{mset } (\text{remove1 } (f \gamma) \Phi) \subseteq\# \text{mset } (\text{map } f \Gamma)$ 
  by (metis
    insert-subset-eq-iff
    list.simps(9)
    mset.simps(2)
    mset-remove1
    remove-hd)
from this Cons obtain  $\Phi'$  where  $\Phi'$ :
   $\text{mset } \Phi' \subseteq\# \text{mset } \Gamma$ 
   $\text{remove1 } (f \gamma) \Phi = \text{map } f \Phi'$ 
  by blast
hence  $\text{mset } (\gamma \# \Phi') \subseteq\# \text{mset } (\gamma \# \Gamma)$ 
  and  $f \gamma \# (\text{remove1 } (f \gamma) \Phi) = \text{map } f (\gamma \# \Phi')$ 
  by simp+
hence  $\Phi \equiv \text{map } f (\gamma \# \Phi')$ 
  using  $\langle f \gamma \in \text{set } \Phi \rangle$  perm-remove
  by metis
from this obtain  $\Phi''$  where  $\Phi''$ :
   $\Phi = \text{map } f \Phi''$ 
   $\Phi'' \equiv \gamma \# \Phi'$ 
  using perm-map-perm-list-exists
  by blast
hence  $\text{mset } \Phi'' \subseteq\# \text{mset } (\gamma \# \Gamma)$ 
  by (metis  $\langle \text{mset } (\gamma \# \Phi') \subseteq\# \text{mset } (\gamma \# \Gamma) \rangle$ )
thus ?thesis using  $\Phi''$  by blast
next
assume  $f \gamma \notin \text{set } \Phi$ 
have  $\text{mset } \Phi - \{\#f \gamma\# \} = \text{mset } \Phi$ 
  by (metis (no-types)
     $\langle f \gamma \notin \text{set } \Phi \rangle$ 
    diff-single-trivial
    set-mset-mset)
moreover
have  $\text{mset } (\text{map } f (\gamma \# \Gamma))$ 
   $= \text{add-mset } (f \gamma) (\text{image-mset } f (\text{mset } \Gamma))$ 
  by simp
ultimately have  $\text{mset } \Phi \subseteq\# \text{mset } (\text{map } f \Gamma)$ 
  by (metis (no-types)
    Diff-eq-empty-iff-mset
     $\langle \text{mset } \Phi \subseteq\# \text{mset } (\text{map } f (\gamma \# \Gamma)) \rangle$ 
    add-mset-add-single
    cancel-ab-semigroup-add-class.diff-right-commute
    diff-diff-add mset-map)
with Cons show ?thesis
  by (metis
    mset-le-perm-append)

```

```

      perm-append-single
      subset-mset.order-trans)
    qed
  }
  thus ?case using Cons by blast
qed
thus ?thesis using assms by blast
qed

```

### 4.3 Laws for Searching a List

```

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

```

```

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

```

### 4.4 Permutations

```

lemma perm-count-list:
  assumes Φ = Ψ
  shows count-list Φ φ = count-list Ψ φ
  using assms
proof (induct Φ arbitrary: Ψ)
  case Nil
  then show ?case
    by blast
next
  case (Cons χ Φ Ψ)
  hence ◇: count-list Φ φ = count-list (remove1 χ Ψ) φ

```

by (*metis mset-remove1 remove-hd*)  
**show** *?case*  
**proof** *cases*  
 assume  $\chi = \varphi$   
 hence  $\text{count-list } (\chi \# \Phi) \varphi = \text{count-list } \Phi \varphi + 1$  **by** *simp*  
 with  $\diamond$  **have**  $\text{count-list } (\chi \# \Phi) \varphi = \text{count-list } (\text{remove1 } \chi \Psi) \varphi + 1$   
 by *simp*  
**moreover**  
 have  $\chi \in \text{set } \Psi$   
 by (*metis Cons.prem1 list.set-intros(1) set-mset-mset*)  
 hence  $\text{count-list } (\text{remove1 } \chi \Psi) \varphi + 1 = \text{count-list } \Psi \varphi$   
 using  $\langle \chi = \varphi \rangle$   
 by (*induct*  $\Psi$ , *simp*, *auto*)  
**ultimately show** *?thesis* **by** *simp*  
**next**  
 assume  $\chi \neq \varphi$   
 with  $\diamond$  **have**  $\text{count-list } (\chi \# \Phi) \varphi = \text{count-list } (\text{remove1 } \chi \Psi) \varphi$   
 by *simp*  
**moreover have**  $\text{count-list } (\text{remove1 } \chi \Psi) \varphi = \text{count-list } \Psi \varphi$   
 using  $\langle \chi \neq \varphi \rangle$   
 by (*induct*  $\Psi$ , *simp+*)  
**ultimately show** *?thesis* **by** *simp*  
**qed**  
**qed**

**lemma** *count-list-append*:  
 $\text{count-list } (A @ B) a = \text{count-list } A a + \text{count-list } B a$   
**by** (*induct*  $A$ , *simp*, *simp*)

**lemma** *concat-remove1*:  
 assumes  $\Psi \in \text{set } \mathcal{L}$   
 shows  $\text{concat } \mathcal{L} \Rightarrow \Psi @ \text{concat } (\text{remove1 } \Psi \mathcal{L})$   
 using *assms*  
**by** (*induct*  $\mathcal{L}$ , *simp*, *simp*, *metis*)

**lemma** *concat-set-membership-mset-containment*:  
 assumes  $\text{concat } \Gamma \Rightarrow \Lambda$   
**and**  $\Phi \in \text{set } \Gamma$   
 shows  $\text{mset } \Phi \subseteq \# \text{mset } \Lambda$   
 using *assms*  
**by** (*induct*  $\Gamma$ , *simp*, *simp*, *metis concat-remove1 mset-le-perm-append*)

**lemma** (*in comm-monoid-add*) *perm-list-summation*:  
 assumes  $\Psi \Rightarrow \Phi$   
 shows  $(\sum \psi' \leftarrow \Psi. f \psi') = (\sum \varphi' \leftarrow \Phi. f \varphi')$   
 using *assms*  
**proof** (*induct*  $\Psi$  *arbitrary*:  $\Phi$ )  
 case *Nil*  
**then show** *?case* **by** *auto*

```

next
case (Cons  $\psi$   $\Psi$   $\Phi$ )
hence  $(\sum \psi' \leftarrow \Psi. f \psi') = (\sum \varphi' \leftarrow (\text{remove1 } \psi \ \Phi). f \varphi')$ 
  by (metis mset-remove1 remove-hd)
moreover have  $\psi \in \text{set } \Phi$ 
  by (metis Cons.prem1 list.set-intros(1) set-mset-mset)
hence  $(\sum \varphi' \leftarrow (\psi \# (\text{remove1 } \psi \ \Phi)). f \varphi') = (\sum \varphi' \leftarrow \Phi. f \varphi')$ 
proof (induct  $\Phi$ )
  case Nil
  then show ?case by auto
next
case (Cons  $\varphi$   $\Phi$ )
show ?case
proof cases
  assume  $\varphi = \psi$ 
  then show ?thesis by simp
next
  assume  $\varphi \neq \psi$ 
  hence  $\psi \in \text{set } \Phi$ 
  using Cons.prem1 by auto
  hence  $(\sum \varphi' \leftarrow (\psi \# (\text{remove1 } \psi \ \Phi)). f \varphi') = (\sum \varphi' \leftarrow \Phi. f \varphi')$ 
  using Cons.hyps by blast
  hence  $(\sum \varphi' \leftarrow (\varphi \# \Phi). f \varphi')$ 
    =  $(\sum \varphi' \leftarrow (\psi \# \varphi \# (\text{remove1 } \psi \ \Phi)). f \varphi')$ 
  by (simp add: add.left-commute)
  moreover
  have  $(\psi \# (\varphi \# (\text{remove1 } \psi \ \Phi))) = (\psi \# (\text{remove1 } \psi (\varphi \# \Phi)))$ 
  using  $\langle \varphi \neq \psi \rangle$  by simp
  ultimately show ?thesis
  by simp
qed
qed
ultimately show ?case
  by simp
qed

```

## 4.5 List Duplicates

primrec *duplicates* :: 'a list  $\Rightarrow$  'a set

where

```

  duplicates [] = {}
| duplicates (x # xs) =
  (if (x  $\in$  set xs)
   then insert x (duplicates xs)
   else duplicates xs)

```

lemma *duplicates-subset*:

```

duplicates  $\Phi \subseteq \text{set } \Phi$ 
by (induct  $\Phi$ , simp, auto)

```

**lemma** *duplicates-alt-def*:  
 $duplicates\ xs = \{x. count\text{-}list\ xs\ x \geq 2\}$   
**proof** (*induct xs*)  
  **case** *Nil*  
  **then show** *?case by simp*  
**next**  
  **case** (*Cons x xs*)  
  **assume** *inductive-hypothesis: duplicates xs = {x. 2 ≤ count-list xs x}*  
  **then show** *?case*  
  **proof** *cases*  
    **assume**  $x \in set\ xs$   
    **hence**  $count\text{-}list\ (x \# xs)\ x \geq 2$   
    **by** (*simp, induct xs, simp, simp, blast*)  
    **hence**  $\{y. 2 \leq count\text{-}list\ (x \# xs)\ y\}$   
       $= insert\ x\ \{y. 2 \leq count\text{-}list\ xs\ y\}$   
    **by** (*simp, blast*)  
    **thus** *?thesis using inductive-hypothesis ⟨x ∈ set xs⟩*  
    **by** *simp*  
  **next**  
  **assume**  $x \notin set\ xs$   
  **hence**  $\{y. 2 \leq count\text{-}list\ (x \# xs)\ y\} = \{y. 2 \leq count\text{-}list\ xs\ y\}$   
  **by** (*simp, auto*)  
  **thus** *?thesis using inductive-hypothesis ⟨x ∉ set xs⟩*  
  **by** *simp*  
**qed**  
**qed**

## 4.6 List Subtraction

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

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

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

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

**lemma** *list-subtract-cons*:  
**assumes**  $\varphi \notin set\ \Lambda$

shows  $(\varphi \# \Phi) \ominus \Lambda = \varphi \# (\Phi \ominus \Lambda)$   
 using *assms*  
 by (*induct*  $\Lambda$ , *simp*, *simp*, *blast*)

**lemma** *list-subtract-cons-absorb*:  
 assumes *count-list*  $\Phi \varphi \geq \text{count-list } \Lambda \varphi$   
 shows  $\varphi \# (\Phi \ominus \Lambda) \equiv (\varphi \# \Phi) \ominus \Lambda$   
 using *assms*  
**proof** (*induct*  $\Lambda$  *arbitrary*:  $\Phi$ )  
 case *Nil*  
 then show ?case using *list-subtract-cons* by *fastforce*  
**next**  
 case (*Cons*  $\psi \Lambda \Phi$ )  
 then show ?case  
**proof** *cases*  
 assume  $\varphi = \psi$   
 hence  $\varphi \in \text{set } \Phi$   
 using *Cons.prem*s *count-notin* by *force*  
 hence  $\Phi \equiv \varphi \# (\text{remove1 } \psi \Phi)$   
 unfolding  $\langle \varphi = \psi \rangle$   
 by *force*  
 thus ?thesis using *perm-count-list*  
 by (*metis*  
   (*no-types*, *lifting*)  
   *Cons.hyps*  
   *Cons.prem*s  
    $\langle \varphi = \psi \rangle$   
   *add-le-cancel-right*  
   *add-mset-diff-bothsides*  
   *count-list.simps*(2)  
   *list-subtract-mset-homomorphism*  
   *mset.simps*(2))  
**next**  
 assume  $\varphi \neq \psi$   
 hence *count-list*  $(\psi \# \Lambda) \varphi = \text{count-list } \Lambda \varphi$   
 by *simp*  
 moreover have *count-list*  $\Phi \varphi = \text{count-list } (\text{remove1 } \psi \Phi) \varphi$   
**proof** (*induct*  $\Phi$ )  
 case *Nil*  
 then show ?case by *simp*  
**next**  
 case (*Cons*  $\varphi' \Phi$ )  
 show ?case  
**proof** *cases*  
 assume  $\varphi' = \varphi$   
 with  $\langle \varphi \neq \psi \rangle$   
 have *count-list*  $(\varphi' \# \Phi) \varphi = 1 + \text{count-list } \Phi \varphi$   
   *count-list*  $(\text{remove1 } \psi (\varphi' \# \Phi)) \varphi$   
   =  $1 + \text{count-list } (\text{remove1 } \psi \Phi) \varphi$

```

    by simp+
  with Cons show ?thesis by linarith
next
  assume  $\varphi' \neq \varphi$ 
  with Cons show ?thesis by (cases  $\varphi' = \psi$ , simp+)
qed
qed
ultimately show ?thesis
  using ⟨count-list ( $\psi \# \Lambda$ )  $\varphi \leq$  count-list  $\Phi \varphi$ ⟩
  by (metis
      Cons.hyps
      ⟨ $\varphi \neq \psi$ ⟩
      list-subtract-remove1-cons-perm
      mset.simps(2)
      remove1.simps(2))
qed
qed

lemma list-subtract-cons-remove1-perm:
  assumes  $\varphi \in \text{set } \Lambda$ 
  shows  $(\varphi \# \Phi) \ominus \Lambda \Rightarrow \Phi \ominus (\text{remove1 } \varphi \Lambda)$ 
  using assms
  by (metis
      list-subtract-mset-homomorphism
      list-subtract-remove1-cons-perm
      perm-remove
      remove-hd)

lemma list-subtract-removeAll-perm:
  assumes count-list  $\Phi \varphi \leq$  count-list  $\Lambda \varphi$ 
  shows  $\Phi \ominus \Lambda \Rightarrow (\text{removeAll } \varphi \Phi) \ominus (\text{removeAll } \varphi \Lambda)$ 
  using assms
proof (induct  $\Phi$  arbitrary:  $\Lambda$ )
  case Nil
  then show ?case by auto
next
  case (Cons  $\xi \Phi \Lambda$ )
  hence  $\Phi \ominus \Lambda \Rightarrow (\text{removeAll } \varphi \Phi) \ominus (\text{removeAll } \varphi \Lambda)$ 
  by (metis add-leE count-list.simps(2))
  show ?case
proof cases
  assume  $\xi = \varphi$ 
  hence count-list  $\Phi \varphi <$  count-list  $\Lambda \varphi$ 
  using ⟨count-list ( $\xi \# \Phi$ )  $\varphi \leq$  count-list  $\Lambda \varphi$ ⟩
  by auto
  hence count-list  $\Phi \varphi \leq$  count-list ( $\text{remove1 } \varphi \Lambda$ )  $\varphi$ 
  by (induct  $\Lambda$ , simp, auto)
  hence  $\Phi \ominus (\text{remove1 } \varphi \Lambda)$ 
   $\Rightarrow \text{removeAll } \varphi \Phi \ominus \text{removeAll } \varphi (\text{remove1 } \varphi \Lambda)$ 

```

```

using Cons.hyps by blast
hence  $\Phi \ominus (\text{remove1 } \varphi \ \Lambda) \equiv \text{removeAll } \varphi \ \Phi \ominus \text{removeAll } \varphi \ \Lambda$ 
by (simp add: filter-remove1 removeAll-filter-not-eq)
moreover have  $\varphi \in \text{set } \Lambda$  and  $\varphi \in \text{set } (\varphi \# \Phi)$ 
using  $\langle \xi = \varphi \rangle$ 
 $\langle \text{count-list } (\xi \# \Phi) \ \varphi \leq \text{count-list } \Lambda \ \varphi \rangle$ 
gr-implies-not0
by fastforce+
hence  $(\varphi \# \Phi) \ominus \Lambda \equiv (\text{remove1 } \varphi \ (\varphi \# \Phi)) \ominus (\text{remove1 } \varphi \ \Lambda)$ 
by (metis list-subtract-cons-remove1-perm remove-hd)

hence  $(\varphi \# \Phi) \ominus \Lambda \equiv \Phi \ominus (\text{remove1 } \varphi \ \Lambda)$  by simp
ultimately show ?thesis using  $\langle \xi = \varphi \rangle$  by auto
next
assume  $\xi \neq \varphi$ 
show ?thesis
proof cases
assume  $\xi \in \text{set } \Lambda$ 
hence  $(\xi \# \Phi) \ominus \Lambda \equiv \Phi \ominus \text{remove1 } \xi \ \Lambda$ 
by (meson list-subtract-cons-remove1-perm)
moreover have  $\text{count-list } \Lambda \ \varphi = \text{count-list } (\text{remove1 } \xi \ \Lambda) \ \varphi$ 
by (metis
 $\text{count-list.simps}(2)$ 
 $\langle \xi \neq \varphi \rangle$ 
 $\langle \xi \in \text{set } \Lambda \rangle$ 
perm-count-list
perm-remove)
hence  $\text{count-list } \Phi \ \varphi \leq \text{count-list } (\text{remove1 } \xi \ \Lambda) \ \varphi$ 
using  $\langle \xi \neq \varphi \rangle$   $\langle \text{count-list } (\xi \# \Phi) \ \varphi \leq \text{count-list } \Lambda \ \varphi \rangle$  by auto
hence  $\Phi \ominus \text{remove1 } \xi \ \Lambda$ 
 $\equiv (\text{removeAll } \varphi \ \Phi) \ominus (\text{removeAll } \varphi \ (\text{remove1 } \xi \ \Lambda))$ 
using Cons.hyps by blast
moreover
have  $(\text{removeAll } \varphi \ \Phi) \ominus (\text{removeAll } \varphi \ (\text{remove1 } \xi \ \Lambda)) \equiv$ 
 $(\text{removeAll } \varphi \ \Phi) \ominus (\text{remove1 } \xi \ (\text{removeAll } \varphi \ \Lambda))$ 
by (induct  $\Lambda$ ,
simp,
metis
 $\langle \xi \neq \varphi \rangle$ 
list-subtract.simps(2)
mset-remove1
remove1.simps(2)
removeAll.simps(2))
hence  $(\text{removeAll } \varphi \ \Phi) \ominus (\text{removeAll } \varphi \ (\text{remove1 } \xi \ \Lambda)) \equiv$ 
 $(\text{removeAll } \varphi \ (\xi \# \Phi)) \ominus (\text{removeAll } \varphi \ \Lambda)$ 
by (metis
 $\langle \xi \in \text{set } \Lambda \rangle$ 
 $\langle \xi \neq \varphi \rangle$ 
list-subtract-cons-remove1-perm)

```



```

      member-remove removeAll.simps(2)
      remove-code(1)
    ultimately show ?thesis
      by presburger
  next
  assume  $\xi \notin \text{set } \Lambda$ 
  hence  $(\xi \# \Phi) \ominus \Lambda \Rightarrow \xi \# (\Phi \ominus \Lambda)$ 
    by fastforce
  hence  $(\xi \# \Phi) \ominus \Lambda \Rightarrow \xi \# ((\text{removeAll } \varphi \Phi) \ominus (\text{removeAll } \varphi \Lambda))$ 
    using  $\langle \Phi \ominus \Lambda \Rightarrow \text{removeAll } \varphi \Phi \ominus \text{removeAll } \varphi \Lambda \rangle$ 
    by simp
  hence  $(\xi \# \Phi) \ominus \Lambda \Rightarrow (\xi \# (\text{removeAll } \varphi \Phi)) \ominus (\text{removeAll } \varphi \Lambda)$ 
    by (simp add:  $\langle \xi \notin \text{set } \Lambda \rangle$  list-subtract-cons)
  thus ?thesis using  $\langle \xi \neq \varphi \rangle$  by auto
qed
qed
qed

lemma list-subtract-permute:
  assumes  $\Phi \Rightarrow \Psi$ 
  shows  $\Phi \ominus \Lambda \Rightarrow \Psi \ominus \Lambda$ 
  using assms
  by simp

lemma append-perm-list-subtract-intro:
  assumes  $A \Rightarrow B @ C$ 
  shows  $A \ominus C \Rightarrow B$ 
proof -
  from  $\langle A \Rightarrow B @ C \rangle$  have  $\text{mset } (A \ominus C) = \text{mset } B$ 
    by simp
  thus ?thesis by blast
qed

lemma list-subtract-concat:
  assumes  $\Psi \in \text{set } \mathcal{L}$ 
  shows  $\text{concat } (\mathcal{L} \ominus [\Psi]) \Rightarrow (\text{concat } \mathcal{L}) \ominus \Psi$ 
  using assms
  by (simp add: concat-remove1)

lemma (in comm-monoid-add) listSubtract-multisubset-list-summation:
  assumes  $\text{mset } \Psi \subseteq\# \text{mset } \Phi$ 
  shows  $(\sum \psi \leftarrow \Psi. f \psi) + (\sum \varphi' \leftarrow (\Phi \ominus \Psi). f \varphi') = (\sum \varphi' \leftarrow \Phi. f \varphi')$ 
proof -
  have  $\forall \Phi. \text{mset } \Psi \subseteq\# \text{mset } \Phi$ 
     $\longrightarrow (\sum \psi' \leftarrow \Psi. f \psi') + (\sum \varphi' \leftarrow (\Phi \ominus \Psi). f \varphi') = (\sum \varphi' \leftarrow \Phi. f \varphi')$ 
  proof(induct  $\Psi$ )
    case Nil
    then show ?case
      by simp
  end

```

```

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

```

**lemma** *list-subtract-set-difference-lower-bound:*

*set*  $\Gamma - set \Phi \subseteq set (\Gamma \ominus \Phi)$

using *subset-Diff-insert*

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

**lemma** *list-subtract-set-trivial-upper-bound:*

```

set ( $\Gamma \ominus \Phi$ )  $\subseteq$  set  $\Gamma$ 
  by (induct  $\Phi$ ,
      simp,
      simp,
      meson
      dual-order.trans
      set-remove1-subset)

```

**lemma** *list-subtract-mset-eq:*

```

assumes mset  $\Phi \subseteq\#$  mset  $\Gamma$ 
  and length ( $\Gamma \ominus \Phi$ ) =  $m$ 
  shows length  $\Gamma$  =  $m +$  length  $\Phi$ 
using assms
proof -
  have  $\forall \Gamma. \text{mset } \Phi \subseteq\# \text{mset } \Gamma$ 
     $\longrightarrow$  length ( $\Gamma \ominus \Phi$ ) =  $m \dashrightarrow$  length  $\Gamma$  =  $m +$  length  $\Phi$ 
  proof (induct  $\Phi$ )
    case Nil
    then show ?case by simp
  next
    case (Cons  $\varphi \Phi$ )
    {
      fix  $\Gamma :: 'a \text{ list}$ 
      assume mset ( $\varphi \# \Phi$ )  $\subseteq\#$  mset  $\Gamma$ 
        length ( $\Gamma \ominus (\varphi \# \Phi)$ ) =  $m$ 
      moreover from this have
        mset  $\Phi \subseteq\#$  mset (remove1  $\varphi \Gamma$ )
        mset ( $\Gamma \ominus (\varphi \# \Phi)$ ) = mset ((remove1  $\varphi \Gamma$ )  $\ominus \Phi$ )
      by (metis
          append-Cons
          mset-le-perm-append
          perm-remove-perm
          remove-hd,
          simp)
      ultimately have length (remove1  $\varphi \Gamma$ ) =  $m +$  length  $\Phi$ 
        using Cons.hyps
        by (metis mset-eq-length)
      hence length ( $\varphi \#$  (remove1  $\varphi \Gamma$ )) =  $m +$  length ( $\varphi \# \Phi$ )
        by simp
      moreover have  $\varphi \in$  set  $\Gamma$ 
        by (metis
             $\langle \text{mset } (\Gamma \ominus (\varphi \# \Phi)) = \text{mset } (\text{remove1 } \varphi \Gamma \ominus \Phi) \rangle$ 
             $\langle \text{mset } (\varphi \# \Phi) \subseteq\# \text{mset } \Gamma \rangle$ 
             $\langle \text{mset } \Phi \subseteq\# \text{mset } (\text{remove1 } \varphi \Gamma) \rangle$ 
            add-diff-cancel-left'
            add-right-cancel
            eq-iff)
    }
  end

```

*impossible-Cons*  
*list-subtract-mset-homomorphism*  
*mset-subset-eq-exists-conv*  
*remove1-idem size-mset*  
**hence**  $\text{length } (\varphi \# (\text{remove1 } \varphi \Gamma)) = \text{length } \Gamma$   
**by** (*metis*  
*One-nat-def*  
*Suc-pred*  
*length-Cons*  
*length-pos-if-in-set*  
*length-remove1*)  
**ultimately have**  $\text{length } \Gamma = m + \text{length } (\varphi \# \Phi)$  **by** *simp*  
**}**  
**thus** *?case* **by** *blast*  
**qed**  
**thus** *?thesis* **using** *assms* **by** *blast*  
**qed**

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

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

**lemma** *map-list-subtract-mset-containment*:  
 $\text{mset } ((\text{map } f A) \ominus (\text{map } f B)) \subseteq\# \text{mset } (\text{map } f (A \ominus B))$   
**by** (*induct*  $B$ , *simp*, *simp*,  
*metis*  
*diff-subset-eq-self*)

*diff-zero*  
*image-mset-add-mset*  
*image-mset-subseteq-mono*  
*image-mset-union*  
*subset-eq-diff-conv*  
*subset-eq-diff-conv*)

**lemma** *map-list-subtract-mset-equivalence:*

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

**lemma** *mset-list-subtract-elem-cons-mset:*

**assumes**  $mset\ \Xi \subseteq\# mset\ \Gamma$   
**and**  $\psi \in set\ (\Gamma \ominus \Xi)$   
**shows**  $mset\ (\psi \# \Xi) \subseteq\# mset\ \Gamma$

**proof** –

**have**  $\forall \Gamma. mset\ \Xi \subseteq\# mset\ \Gamma$   
 $\longrightarrow \psi \in set\ (\Gamma \ominus \Xi) \longrightarrow mset\ (\psi \# \Xi) \subseteq\# mset\ \Gamma$

**proof**(*induct*  $\Xi$ )

**case** *Nil*

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

**next**

**case** (*Cons*  $\xi\ \Xi$ )

{

**fix**  $\Gamma$

**assume**

$mset\ (\xi \# \Xi) \subseteq\# mset\ \Gamma$

$\psi \in set\ (\Gamma \ominus (\xi \# \Xi))$

**hence**

$\xi \in set\ \Gamma$

$mset\ \Xi \subseteq\# mset\ (remove1\ \xi\ \Gamma)$

$\psi \in set\ ((remove1\ \xi\ \Gamma) \ominus \Xi)$

**by** (*simp,*

*metis*

*ex-mset*

*list.set-intros(1)*

*mset.simps(2)*

*mset-eq-setD*

*subset-mset.le-iff-add*

*union-mset-add-mset-left,*

*metis*

*list-subtract.simps(1)*

*list-subtract.simps(2)*

*list-subtract-monotonic*

*remove-hd,*

*simp,*

*metis*

```

      list-subtract-remove1-cons-perm
      perm-set-eq)
with Cons.hyps have
  mset  $\Gamma = \text{mset } (\xi \# (\text{remove1 } \xi \Gamma))$ 
  mset  $(\psi \# \Xi) \subseteq\# \text{mset } (\text{remove1 } \xi \Gamma)$ 
  by (simp, blast)
hence mset  $(\psi \# \xi \# \Xi) \subseteq\# \text{mset } \Gamma$ 
  by (simp,
      metis
      add-mset-commute
      mset-subset-eq-add-mset-cancel)
}
then show ?case by auto
qed
thus ?thesis using assms by blast
qed

```

## 4.7 Tuple Lists

```

lemma remove1-pairs-list-projections-fst:
  assumes  $(\gamma, \sigma) \in\# \text{mset } \Phi$ 
  shows  $\text{mset } (\text{map fst } (\text{remove1 } (\gamma, \sigma) \Phi)) = \text{mset } (\text{map fst } \Phi) - \{\#\gamma \#\}$ 
using assms
proof (induct  $\Phi$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\varphi \Phi$ )
  assume  $(\gamma, \sigma) \in\# \text{mset } (\varphi \# \Phi)$ 
  show ?case
  proof (cases  $\varphi = (\gamma, \sigma)$ )
    assume  $\varphi = (\gamma, \sigma)$ 
    then show ?thesis by simp
  next
    assume  $\varphi \neq (\gamma, \sigma)$ 
    then have  $\text{add-mset } \varphi (\text{mset } \Phi - \{\#(\gamma, \sigma)\#})$ 
      =  $\text{add-mset } \varphi (\text{mset } \Phi) - \{\#(\gamma, \sigma)\#}$ 
      by force
    then have  $\text{add-mset } (\text{fst } \varphi) (\text{image-mset fst } (\text{mset } \Phi - \{\#(\gamma, \sigma)\#}))$ 
      =  $\text{add-mset } (\text{fst } \varphi) (\text{image-mset fst } (\text{mset } \Phi)) - \{\#\gamma\#\}$ 
      by (metis (no-types) Cons.prem1
          add-mset-remove-trivial
          fst-conv
          image-mset-add-mset
          insert-DiffM mset.simps(2))
    with  $\langle \varphi \neq (\gamma, \sigma) \rangle$  show ?thesis
      by simp
  qed
qed
qed

```

```

lemma remove1-pairs-list-projections-snd:
  assumes  $(\gamma, \sigma) \in \# \text{mset } \Phi$ 
  shows  $\text{mset } (\text{map } \text{snd } (\text{remove1 } (\gamma, \sigma) \Phi)) = \text{mset } (\text{map } \text{snd } \Phi) - \{\# \sigma \#\}$ 
using assms
proof (induct  $\Phi$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\varphi$   $\Phi$ )
  assume  $(\gamma, \sigma) \in \# \text{mset } (\varphi \# \Phi)$ 
  show ?case
  proof (cases  $\varphi = (\gamma, \sigma)$ )
    assume  $\varphi = (\gamma, \sigma)$ 
    then show ?thesis by simp
  next
  assume  $\varphi \neq (\gamma, \sigma)$ 
  then have  $\text{add-mset } (\text{snd } \varphi) (\text{image-mset } \text{snd } (\text{mset } \Phi - \{\#(\gamma, \sigma)\#\}))$ 
     $= \text{image-mset } \text{snd } (\text{mset } (\varphi \# \Phi) - \{\#(\gamma, \sigma)\#\})$ 
    by auto
  moreover have  $\text{add-mset } (\text{snd } \varphi) (\text{image-mset } \text{snd } (\text{mset } \Phi))$ 
     $= \text{add-mset } \sigma (\text{image-mset } \text{snd } (\text{mset } (\varphi \# \Phi) - \{\#(\gamma, \sigma)\#\}))$ 
    by (metis (no-types)
      Cons.prem
      image-mset-add-mset
      insert-DiffM
      mset.simps(2)
      snd-conv)
  ultimately
  have  $\text{add-mset } (\text{snd } \varphi) (\text{image-mset } \text{snd } (\text{mset } \Phi - \{\#(\gamma, \sigma)\#\}))$ 
     $= \text{add-mset } (\text{snd } \varphi) (\text{image-mset } \text{snd } (\text{mset } \Phi)) - \{\#\sigma\#\}$ 
    by simp
  with  $\langle \varphi \neq (\gamma, \sigma) \rangle$  show ?thesis
    by simp
qed
qed

```

```

lemma triple-list-exists:
  assumes  $\text{mset } (\text{map } \text{snd } \Psi) \subseteq \# \text{mset } \Sigma$ 
  and  $\text{mset } \Sigma \subseteq \# \text{mset } (\text{map } \text{snd } \Delta)$ 
  shows  $\exists \Omega. \text{map } (\lambda (\psi, \sigma, -). (\psi, \sigma)) \Omega = \Psi \wedge$ 
     $\text{mset } (\text{map } (\lambda (-, \sigma, \gamma). (\gamma, \sigma)) \Omega) \subseteq \# \text{mset } \Delta$ 
using assms(1)
proof (induct  $\Psi$ )
  case Nil
  then show ?case by fastforce
next
  case (Cons  $\psi$   $\Psi$ )
  from Cons obtain  $\Omega$  where  $\Omega$ :

```

```

map (λ (ψ, σ, -). (ψ, σ)) Ω = Ψ
mset (map (λ (-, σ, γ). (γ, σ)) Ω) ⊆# mset Δ
by (metis
  (no-types, lifting)
  diff-subset-eq-self
  list.set-intros(1)
  remove1-pairs-list-projections-snd
  remove-hd
  set-mset-mset
  subset-mset.dual-order.trans
  surjective-pairing)
let ?ΔΩ = map (λ (-, σ, γ). (γ, σ)) Ω
let ?ψ = fst ψ
let ?σ = snd ψ
from Cons.premis have add-mset ?σ (image-mset snd (mset Ψ)) ⊆# mset Σ
  by simp
then have mset Σ - {#?σ#} - image-mset snd (mset Ψ)
  ≠ mset Σ - image-mset snd (mset Ψ)
  by (metis
    (no-types)
    insert-subset-eq-iff
    mset-subset-eq-insertD
    multi-drop-mem-not-eq
    subset-mset.diff-add
    subset-mset-def)
hence ?σ ∈# mset Σ - mset (map snd Ψ)
  using diff-single-trivial by fastforce
have mset (map snd (ψ # Ψ)) ⊆# mset (map snd Δ)
  by (meson
    Cons.premis
    ⟨mset Σ ⊆# mset (map snd Δ)⟩
    subset-mset.dual-order.trans)
then have
  mset (map snd Δ) - mset (map snd (ψ # Ψ)) + ({#} + {#snd ψ#})
  = mset (map snd Δ) + ({#} + {#snd ψ#})
  - add-mset (snd ψ) (mset (map snd Ψ))
  by (metis
    (no-types)
    list.simps(9)
    mset.simps(2)
    mset-subset-eq-multiset-union-diff-commute)
then have
  mset (map snd Δ) - mset (map snd (ψ # Ψ)) + ({#} + {#snd ψ#})
  = mset (map snd Δ) - mset (map snd Ψ)
  by auto
hence ?σ ∈# mset (map snd Δ) - mset (map snd Ψ)
  using add-mset-remove-trivial-eq by fastforce
moreover have snd ∘ (λ (ψ, σ, -). (ψ, σ)) = snd ∘ (λ (-, σ, γ). (γ, σ))
  by auto

```



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

## 4.8 List Intersection

**primrec** *list-intersect*  $:: 'a \text{ list} \Rightarrow 'a \text{ list} \Rightarrow 'a \text{ list}$  (*infixl*  $\cap$  60)

```

where
  -  $\cap [] = []$ 
  |  $xs \cap (y \# ys) =$ 
    (if ( $y \in \text{set } xs$ )
      then ( $y \# (\text{remove1 } y \text{ } xs \cap ys)$ )
      else ( $xs \cap ys$ ))

lemma list-intersect-mset-homomorphism [simp]:
   $mset (\Phi \cap \Psi) = mset \Phi \cap \# mset \Psi$ 
proof -
  have  $\forall \Phi. mset (\Phi \cap \Psi) = mset \Phi \cap \# mset \Psi$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case by simp
next
  case (Cons  $\psi \Psi$ )
  {
  fix  $\Phi$ 
  have  $mset (\Phi \cap \psi \# \Psi) = mset \Phi \cap \# mset (\psi \# \Psi)$ 
  using Cons.hyps
  by (cases  $\psi \in \text{set } \Phi$ ,
    simp add: inter-add-right2,
    simp add: inter-add-right1)
  }
  then show ?case by blast
qed
thus ?thesis by simp
qed

lemma list-intersect-left-empty [simp]:  $[] \cap \Phi = []$  by (induct  $\Phi$ , simp+)

lemma list-diff-intersect-comp:
   $mset \Phi = mset (\Phi \ominus \Psi) + mset (\Phi \cap \Psi)$ 
by (metis
  diff-intersect-left-idem
  list-intersect-mset-homomorphism
  list-subtract-mset-homomorphism
  subset-mset.inf-le1
  subset-mset.le-imp-diff-is-add)

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

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

end

```

## Chapter 5

# Classical Logic Connectives

```
theory Classical-Connectives
imports
  Classical-Logic-Completeness
  List-Utilities
begin
```

Here we define the usual connectives for classical logic.

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

### 5.1 Verum

```
definition (in classical-logic) verum :: 'a  $\top$ 
where
   $\top = \perp \rightarrow \perp$ 
```

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

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

```
lemma (in classical-logic) verum-embedding [simp]:
   $\langle \top \rangle = \top$ 
by (simp add: classical-logic-class.verum-def verum-def)
```

### 5.2 Conjunction

```
definition (in classical-logic)
  conjunction :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (infixr  $\sqcap$  67)
where
   $\varphi \sqcap \psi = (\varphi \rightarrow \psi \rightarrow \perp) \rightarrow \perp$ 
```

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

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

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

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

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

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

## 5.3 Biconditional

**definition** (in *classical-logic*) *biconditional* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (**infixr**  $\leftrightarrow$  75)  
**where**

$$\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \sqcap (\psi \rightarrow \varphi)$$

**lemma** (in *classical-logic*) *biconditional-introduction*:

$$\vdash (\varphi \rightarrow \psi) \rightarrow (\psi \rightarrow \varphi) \rightarrow (\varphi \leftrightarrow \psi)$$

**by** (*simp add: biconditional-def conjunction-introduction*)

**lemma** (in *classical-logic*) *biconditional-left-elimination*:

$$\vdash (\varphi \leftrightarrow \psi) \rightarrow \varphi \rightarrow \psi$$

**by** (*simp add: biconditional-def conjunction-left-elimination*)

**lemma** (in *classical-logic*) *biconditional-right-elimination*:

$$\vdash (\varphi \leftrightarrow \psi) \rightarrow \psi \rightarrow \varphi$$

**by** (*simp add: biconditional-def conjunction-right-elimination*)

**lemma** (in *classical-logic*) *biconditional-embedding [simp]*:

$$\langle \! \langle \varphi \leftrightarrow \psi \rangle \! \rangle = \langle \! \langle \varphi \rangle \! \rangle \leftrightarrow \langle \! \langle \psi \rangle \! \rangle$$

**unfolding** *biconditional-def classical-logic-class.biconditional-def*

**by** *simp*

**lemma** *biconditional-semantic [simp]*:

$$\mathfrak{M} \models_{prop} \varphi \leftrightarrow \psi = (\mathfrak{M} \models_{prop} \varphi \longleftrightarrow \mathfrak{M} \models_{prop} \psi)$$

**unfolding** *biconditional-def*

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

## 5.4 Negation

**definition** (in *classical-logic*) *negation* :: 'a  $\Rightarrow$  'a ( $\sim$ )

**where**

$$\sim \varphi = \varphi \rightarrow \perp$$

**lemma** (in *classical-logic*) *negation-introduction*:

$$\vdash (\varphi \rightarrow \perp) \rightarrow \sim \varphi$$

**unfolding** *negation-def*

**by** (*metis axiom-k modus-ponens implication-absorption*)

**lemma** (in *classical-logic*) *negation-elimination*:

$$\vdash \sim \varphi \rightarrow (\varphi \rightarrow \perp)$$

**unfolding** *negation-def*

**by** (*metis axiom-k modus-ponens implication-absorption*)

**lemma** (in *classical-logic*) *negation-embedding [simp]*:

$$\langle \! \langle \sim \varphi \rangle \! \rangle = \sim \langle \! \langle \varphi \rangle \! \rangle$$

**by** (*simp add:*

*classical-logic-class.negation-def*

*negation-def*)

**lemma** *negation-semantic [simp]*:

$$\mathfrak{M} \models_{prop} \sim \varphi = (\neg \mathfrak{M} \models_{prop} \varphi)$$

**unfolding** *negation-def*

by *simp*

## 5.5 Disjunction

**definition** (in *classical-logic*) *disjunction* :: 'a  $\Rightarrow$  'a  $\Rightarrow$  'a (infixr  $\sqcup$  67)

where

$$\varphi \sqcup \psi = (\varphi \rightarrow \perp) \rightarrow \psi$$

**primrec** (in *classical-logic*) *arbitrary-disjunction* :: 'a list  $\Rightarrow$  'a ( $\sqcup$ )

where

$$\sqcup [] = \perp$$

$$| \sqcup (\varphi \# \Phi) = \varphi \sqcup \sqcup \Phi$$

**lemma** (in *classical-logic*) *disjunction-elimination*:

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

**proof** –

let  $?T = [\varphi \rightarrow \chi, \psi \rightarrow \chi, \varphi \sqcup \psi]$

have  $?T \vdash (\varphi \rightarrow \perp) \rightarrow \chi$

unfolding *disjunction-def*

by (*metis hypothetical-syllogism*

*list-deduction-def*

*list-implication.simps(1)*

*list-implication.simps(2)*

*set-deduction-base-theory*

*set-deduction-theorem*

*set-deduction-weaken*)

hence  $?T \vdash \chi$

using *excluded-middle-elimination*

*list-deduction-modus-ponens*

*list-deduction-theorem*

*list-deduction-weaken*

by *blast*

thus *?thesis*

unfolding *list-deduction-def*

by *simp*

qed

**lemma** (in *classical-logic*) *disjunction-left-introduction*:

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

unfolding *disjunction-def*

by (*metis modus-ponens*

*pseudo-scotus*

*flip-implication*)

**lemma** (in *classical-logic*) *disjunction-right-introduction*:

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

unfolding *disjunction-def*

using *axiom-k*

by *simp*

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

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

## 5.6 Mutual Exclusion

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

## 5.7 Subtraction

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

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

## 5.8 Negated Lists

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

## 5.9 Common (& Uncommon) Identities

### 5.9.1 Biconditional Equivalence Relation

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

**lemma** (in *classical-logic*) *biconditional-symmetry*:  
 $\vdash (\varphi \leftrightarrow \psi) \leftrightarrow (\psi \leftrightarrow \varphi)$

by (*metis* (*full-types*) *modus-ponens*  
*biconditional-def*  
*conjunction-def*  
*flip-hypothetical-syllogism*  
*flip-implication*)

**lemma** (in *classical-logic*) *biconditional-symmetry-rule*:

$\vdash \varphi \leftrightarrow \psi \implies \vdash \psi \leftrightarrow \varphi$

by (*meson* *modus-ponens*  
*biconditional-introduction*  
*biconditional-left-elimination*  
*biconditional-right-elimination*)

**lemma** (in *classical-logic*) *biconditional-transitivity*:

$\vdash (\varphi \leftrightarrow \psi) \rightarrow (\psi \leftrightarrow \chi) \rightarrow (\varphi \leftrightarrow \chi)$

**proof** –

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

by *simp*

hence  $\vdash \langle (\langle \varphi \rangle \leftrightarrow \langle \psi \rangle) \rightarrow (\langle \psi \rangle \leftrightarrow \langle \chi \rangle) \rightarrow (\langle \varphi \rangle \leftrightarrow \langle \chi \rangle) \rangle$

using *propositional-semantic* by *blast*

thus *?thesis* by *simp*

qed

**lemma** (in *classical-logic*) *biconditional-transitivity-rule*:

$\vdash \varphi \leftrightarrow \psi \implies \vdash \psi \leftrightarrow \chi \implies \vdash \varphi \leftrightarrow \chi$

using *modus-ponens* *biconditional-transitivity* by *blast*

## 5.9.2 Biconditional Weakening

**lemma** (in *classical-logic*) *biconditional-weaken*:

assumes  $\Gamma \Vdash \varphi \leftrightarrow \psi$

shows  $\Gamma \Vdash \varphi = \Gamma \Vdash \psi$

by (*metis* *assms*  
*biconditional-left-elimination*  
*biconditional-right-elimination*  
*set-deduction-modus-ponens*  
*set-deduction-weaken*)

**lemma** (in *classical-logic*) *list-biconditional-weaken*:

assumes  $\Gamma : \vdash \varphi \leftrightarrow \psi$

shows  $\Gamma : \vdash \varphi = \Gamma : \vdash \psi$

by (*metis* *assms*  
*biconditional-left-elimination*  
*biconditional-right-elimination*  
*list-deduction-modus-ponens*  
*list-deduction-weaken*)

**lemma** (in *classical-logic*) *weak-biconditional-weaken*:

assumes  $\vdash \varphi \leftrightarrow \psi$



**shows**  $\vdash \varphi = \vdash \psi$   
**by** (*metis assms*  
*biconditional-left-elimination*  
*biconditional-right-elimination*  
*modus-ponens*)

### 5.9.3 Conjunction Identities

**lemma** (*in classical-logic*) *conjunction-negation-identity*:

$\vdash \sim (\varphi \sqcap \psi) \leftrightarrow (\varphi \rightarrow \psi \rightarrow \perp)$

**by** (*metis Contraposition*  
*double-negation-converse*  
*modus-ponens*  
*biconditional-introduction*  
*conjunction-def*  
*negation-def*)

**lemma** (*in classical-logic*) *conjunction-set-deduction-equivalence* [*simp*]:

$\Gamma \Vdash \varphi \sqcap \psi = (\Gamma \Vdash \varphi \wedge \Gamma \Vdash \psi)$

**by** (*metis set-deduction-weaken* [**where**  $\Gamma=\Gamma$ ]  
*set-deduction-modus-ponens* [**where**  $\Gamma=\Gamma$ ]  
*conjunction-introduction*  
*conjunction-left-elimination*  
*conjunction-right-elimination*)

**lemma** (*in classical-logic*) *conjunction-list-deduction-equivalence* [*simp*]:

$\Gamma \text{:}\vdash \varphi \sqcap \psi = (\Gamma \text{:}\vdash \varphi \wedge \Gamma \text{:}\vdash \psi)$

**by** (*metis list-deduction-weaken* [**where**  $\Gamma=\Gamma$ ]  
*list-deduction-modus-ponens* [**where**  $\Gamma=\Gamma$ ]  
*conjunction-introduction*  
*conjunction-left-elimination*  
*conjunction-right-elimination*)

**lemma** (*in classical-logic*) *weak-conjunction-deduction-equivalence* [*simp*]:

$\vdash \varphi \sqcap \psi = (\vdash \varphi \wedge \vdash \psi)$

**by** (*metis conjunction-set-deduction-equivalence set-deduction-base-theory*)

**lemma** (*in classical-logic*) *conjunction-set-deduction-arbitrary-equivalence* [*simp*]:

$\Gamma \Vdash \sqcap \Phi = (\forall \varphi \in \text{set } \Phi. \Gamma \Vdash \varphi)$

**by** (*induct*  $\Phi$ , *simp add: set-deduction-weaken, simp*)

**lemma** (*in classical-logic*) *conjunction-list-deduction-arbitrary-equivalence* [*simp*]:

$\Gamma \text{:}\vdash \sqcap \Phi = (\forall \varphi \in \text{set } \Phi. \Gamma \text{:}\vdash \varphi)$

**by** (*induct*  $\Phi$ , *simp add: list-deduction-weaken, simp*)

**lemma** (*in classical-logic*) *weak-conjunction-deduction-arbitrary-equivalence* [*simp*]:

$\vdash \sqcap \Phi = (\forall \varphi \in \text{set } \Phi. \vdash \varphi)$

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

**lemma** (in *classical-logic*) *conjunction-commutativity*:

$\vdash (\psi \sqcap \varphi) \leftrightarrow (\varphi \sqcap \psi)$

**by** (*metis*  
*(full-types)*  
*modus-ponens*  
*biconditional-introduction*  
*conjunction-def*  
*flip-hypothetical-syllogism*  
*flip-implication*)

**lemma** (in *classical-logic*) *conjunction-associativity*:

$\vdash ((\varphi \sqcap \psi) \sqcap \chi) \leftrightarrow (\varphi \sqcap (\psi \sqcap \chi))$

**proof** –

**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ((\langle \varphi \rangle \sqcap \langle \psi \rangle) \sqcap \langle \chi \rangle) \leftrightarrow (\langle \varphi \rangle \sqcap (\langle \psi \rangle \sqcap \langle \chi \rangle))$

**by** *simp*

**hence**  $\vdash \langle ((\langle \varphi \rangle \sqcap \langle \psi \rangle) \sqcap \langle \chi \rangle) \leftrightarrow (\langle \varphi \rangle \sqcap (\langle \psi \rangle \sqcap \langle \chi \rangle)) \rangle$

**using** *propositional-semantic* **by** *blast*

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

**qed**

**lemma** (in *classical-logic*) *arbitrary-conjunction-antitone*:

$set \Phi \subseteq set \Psi \implies \vdash \sqcap \Psi \rightarrow \sqcap \Phi$

**proof** –

**have**  $\forall \Phi. set \Phi \subseteq set \Psi \longrightarrow \vdash \sqcap \Psi \rightarrow \sqcap \Phi$

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

**case** *Nil*

**then show** *?case*

**by** (*simp add: pseudo-scotus verum-def*)

**next**

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

{

**fix**  $\Phi$

**assume**  $set \Phi \subseteq set (\psi \# \Psi)$

**have**  $\vdash \sqcap (\psi \# \Psi) \rightarrow \sqcap \Phi$

**proof** (*cases*  $\psi \in set \Phi$ )

**assume**  $\psi \in set \Phi$

**have**  $\forall \varphi \in set \Phi. \vdash \sqcap \Phi \leftrightarrow (\varphi \sqcap \sqcap (removeAll \varphi \Phi))$

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

**case** *Nil*

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

**next**

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

{

**fix**  $\varphi$

**assume**  $\varphi \in set (\chi \# \Phi)$

**have**  $\vdash \sqcap (\chi \# \Phi) \leftrightarrow (\varphi \sqcap \sqcap (removeAll \varphi (\chi \# \Phi)))$

**proof** *cases*

**assume**  $\varphi \in set \Phi$

**hence**  $\vdash \sqcap \Phi \leftrightarrow (\varphi \sqcap \sqcap (removeAll \varphi \Phi))$

```

using Cons.hyps ⟨φ ∈ set Φ⟩
by auto
moreover
have ⊢ (⊔ Φ ↔ (φ ⊔ ⊔ (removeAll φ Φ))) →
      (χ ⊔ ⊔ Φ) ↔ (φ ⊔ χ ⊔ ⊔ (removeAll φ Φ))
proof -
  have ∀ M. M ⊨prop
    ((⊔ Φ) ↔ (⟨φ⟩ ⊔ ⟨⊔ (removeAll φ Φ)⟩))
    → (⟨χ⟩ ⊔ ⟨⊔ Φ⟩)
    ↔ (⟨φ⟩ ⊔ ⟨χ⟩ ⊔ ⟨⊔ (removeAll φ Φ)⟩)
  by auto
  hence ⊢ (⟨⊔ Φ⟩ ↔ (⟨φ⟩ ⊔ ⟨⊔ (removeAll φ Φ)⟩))
    → (⟨χ⟩ ⊔ ⟨⊔ Φ⟩)
    ↔ (⟨φ⟩ ⊔ ⟨χ⟩ ⊔ ⟨⊔ (removeAll φ Φ)⟩)
  using propositional-semantic by blast
  thus ?thesis by simp
qed
ultimately have ⊢ ⊔ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⊔ (removeAll φ Φ))
  using modus-ponens by auto
show ?thesis
proof cases
  assume φ = χ
  moreover
  {
    fix φ
    have ⊢ (χ ⊔ φ) → (χ ⊔ χ ⊔ φ)
      unfolding conjunction-def
      by (meson
          axiom-s
          double-negation
          modus-ponens
          flip-hypothetical-syllogism
          flip-implication)
  } note tautology = this
  from ⊢ ⊔ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⊔ (removeAll φ Φ))
    ⟨φ = χ⟩
  have ⊢ (χ ⊔ ⊔ (removeAll χ Φ)) → (χ ⊔ ⊔ Φ)
    unfolding biconditional-def
    by (simp, metis tautology hypothetical-syllogism modus-ponens)
  moreover
  from ⊢ ⊔ (χ # Φ) ↔ (φ ⊔ χ ⊔ ⊔ (removeAll φ Φ))
    ⟨φ = χ⟩
  have ⊢ (χ ⊔ ⊔ Φ) → (χ ⊔ ⊔ (removeAll χ Φ))
    unfolding biconditional-def
    by (simp,
        metis conjunction-right-elimination
          hypothetical-syllogism
          modus-ponens)
  ultimately show ?thesis

```

```

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

```

*hypothetical-syllogism*)

**thus** *?thesis*  
**by** *simp*  
**qed**  
**}**  
**thus** *?case by blast*  
**qed**  
**thus** *set  $\Phi \subseteq \text{set } \Psi \implies \vdash \bigwedge \Psi \rightarrow \bigwedge \Phi$  by blast*  
**qed**

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

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

**lemma** (in *classical-logic*) *list-curry-uncurry*:  
 $\vdash (\Phi \text{ :}\rightarrow \chi) \leftrightarrow (\bigwedge \Phi \rightarrow \chi)$

**proof** (*induct*  $\Phi$ )  
**case** *Nil*  
**have**  $\vdash \chi \leftrightarrow (\top \rightarrow \chi)$   
**unfolding** *biconditional-def*  
*conjunction-def*  
*verum-def*  
**using**  
*axiom-k*  
*ex-falso-quodlibet*  
*modus-ponens*  
*conjunction-def*  
*excluded-middle-elimination*  
*set-deduction-base-theory*  
*conjunction-set-deduction-equivalence*  
**by** *metis*  
**with** *Nil show ?case*  
**by** *simp*  
**next**  
**case** (*Cons*  $\varphi \Phi$ )  
**have**  $\vdash ((\varphi \# \Phi) \text{ :}\rightarrow \chi) \leftrightarrow (\varphi \rightarrow (\Phi \text{ :}\rightarrow \chi))$   
**by** (*simp add: biconditional-reflection*)  
**with** *Cons have*  $\vdash ((\varphi \# \Phi) \text{ :}\rightarrow \chi) \leftrightarrow (\varphi \rightarrow \bigwedge \Phi \rightarrow \chi)$   
**by** (*metis modus-ponens*)

*biconditional-def*  
*hypothetical-syllogism*  
*list-implication.simps(2)*  
*weak-conjunction-deduction-equivalence*  
**with** *curry-uncurry* [**where**  $?\varphi=\varphi$  **and**  $?\psi=\sqcap$   $\Phi$  **and**  $?X=X$ ]  
**show**  $?case$   
**unfolding** *biconditional-def*  
**by** (*simp, metis modus-ponens hypothetical-syllogism*)  
**qed**

#### 5.9.4 Disjunction Identities

**lemma** (*in classical-logic*) *bivalence*:

$\vdash \sim \varphi \sqcup \varphi$   
**by** (*simp add: double-negation disjunction-def negation-def*)

**lemma** (*in classical-logic*) *implication-equivalence*:

$\vdash (\sim \varphi \sqcup \psi) \leftrightarrow (\varphi \rightarrow \psi)$   
**by** (*metis double-negation-converse*  
*modus-ponens*  
*biconditional-introduction*  
*bivalence*  
*disjunction-def*  
*flip-hypothetical-syllogism*  
*negation-def*)

**lemma** (*in classical-logic*) *disjunction-commutativity*:

$\vdash (\psi \sqcup \varphi) \leftrightarrow (\varphi \sqcup \psi)$   
**by** (*meson modus-ponens*  
*biconditional-introduction*  
*disjunction-elimination*  
*disjunction-left-introduction*  
*disjunction-right-introduction*)

**lemma** (*in classical-logic*) *disjunction-associativity*:

$\vdash ((\varphi \sqcup \psi) \sqcup \chi) \leftrightarrow (\varphi \sqcup (\psi \sqcup \chi))$

**proof** –

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

**qed**

**lemma** (*in classical-logic*) *arbitrary-disjunction-monotone*:

$set \Phi \subseteq set \Psi \implies \vdash \sqcup \Phi \rightarrow \sqcup \Psi$

**proof** –

**have**  $\forall \Phi. set \Phi \subseteq set \Psi \implies \vdash \sqcup \Phi \rightarrow \sqcup \Psi$   
**proof** (*induct*  $\Psi$ )

```

case Nil
then show ?case using verum-def verum-tautology by auto
next
case (Cons  $\psi$   $\Psi$ )
{
  fix  $\Phi$ 
  assume set  $\Phi \subseteq$  set ( $\psi \# \Psi$ )
  have  $\vdash \sqcup \Phi \rightarrow \sqcup (\psi \# \Psi)$ 
  proof cases
    assume  $\psi \in$  set  $\Phi$ 
    have  $\forall \varphi \in$  set  $\Phi. \vdash \sqcup \Phi \leftrightarrow (\varphi \sqcup \sqcup (\text{removeAll } \varphi \Phi))$ 
    proof (induct  $\Phi$ )
      case Nil
      then show ?case by simp
    next
    case (Cons  $\chi$   $\Phi$ )
    {
      fix  $\varphi$ 
      assume  $\varphi \in$  set ( $\chi \# \Phi$ )
      have  $\vdash \sqcup (\chi \# \Phi) \leftrightarrow (\varphi \sqcup \sqcup (\text{removeAll } \varphi (\chi \# \Phi)))$ 
      proof cases
        assume  $\varphi \in$  set  $\Phi$ 
        hence  $\vdash \sqcup \Phi \leftrightarrow (\varphi \sqcup \sqcup (\text{removeAll } \varphi \Phi))$ 
          using Cons.hyps  $\langle \varphi \in$  set  $\Phi \rangle$ 
          by auto
        moreover
        have  $\vdash (\sqcup \Phi \leftrightarrow (\varphi \sqcup \sqcup (\text{removeAll } \varphi \Phi))) \rightarrow$ 
           $(\chi \sqcup \sqcup \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \sqcup (\text{removeAll } \varphi \Phi))$ 
        proof –
          have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop}$ 
             $(\langle \sqcup \Phi \rangle \leftrightarrow (\langle \varphi \rangle \sqcup \langle \sqcup (\text{removeAll } \varphi \Phi) \rangle))$ 
             $\rightarrow (\langle \chi \rangle \sqcup \langle \sqcup \Phi \rangle)$ 
             $\leftrightarrow (\langle \varphi \rangle \sqcup \langle \chi \rangle \sqcup \langle \sqcup (\text{removeAll } \varphi \Phi) \rangle)$ 
          by auto
          hence  $\vdash (\langle \sqcup \Phi \rangle \leftrightarrow (\langle \varphi \rangle \sqcup \langle \sqcup (\text{removeAll } \varphi \Phi) \rangle))$ 
             $\rightarrow (\langle \chi \rangle \sqcup \langle \sqcup \Phi \rangle)$ 
             $\leftrightarrow (\langle \varphi \rangle \sqcup \langle \chi \rangle \sqcup \langle \sqcup (\text{removeAll } \varphi \Phi) \rangle)$ 
          using propositional-semantic by blast
          thus ?thesis by simp
        qed
      ultimately have  $\vdash \sqcup (\chi \# \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \sqcup (\text{removeAll } \varphi \Phi))$ 
        using modus-ponens by auto
      show ?thesis
      proof cases
        assume  $\varphi = \chi$ 
        then show ?thesis
          using  $\vdash \sqcup (\chi \# \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \sqcup (\text{removeAll } \varphi \Phi))$ 
          unfolding biconditional-def
          by (simp add: disjunction-def,

```

```

      meson
      axiom-k
      modus-ponens
      flip-hypothetical-syllogism
      implication-absorption)
next
  assume  $\varphi \neq \chi$ 
  then show ?thesis
    using  $\langle \vdash \sqcup (\chi \# \Phi) \leftrightarrow (\varphi \sqcup \chi \sqcup \sqcup (\text{removeAll } \varphi \Phi)) \rangle$ 
    by simp
  qed
next
  assume  $\varphi \notin \text{set } \Phi$ 
  hence  $\varphi = \chi \ \chi \notin \text{set } \Phi$ 
  using  $\langle \varphi \in \text{set } (\chi \# \Phi) \rangle$  by auto
  then show ?thesis
    using biconditional-reflection
    by simp
  qed
}
thus ?case by blast
qed
hence  $\vdash \sqcup \Phi \rightarrow (\psi \sqcup \sqcup (\text{removeAll } \psi \Phi))$ 
  using modus-ponens biconditional-left-elimination  $\langle \psi \in \text{set } \Phi \rangle$ 
  by blast
moreover
from  $\langle \psi \in \text{set } \Phi \rangle \ \langle \text{set } \Phi \subseteq \text{set } (\psi \# \Psi) \rangle$  Cons.hyps
have  $\vdash \sqcup (\text{removeAll } \psi \Phi) \rightarrow \sqcup \Psi$ 
  by (simp add: subset-insert-iff insert-absorb)
hence  $\vdash (\psi \sqcup \sqcup (\text{removeAll } \psi \Phi)) \rightarrow \sqcup (\psi \# \Psi)$ 
  using
    modus-ponens
    disjunction-def
    hypothetical-syllogism
  by fastforce
ultimately show ?thesis
  by (simp, metis modus-ponens hypothetical-syllogism)
next
assume  $\psi \notin \text{set } \Phi$ 
hence  $\vdash \sqcup \Phi \rightarrow \sqcup \Psi$ 
  using Cons.hyps  $\langle \text{set } \Phi \subseteq \text{set } (\psi \# \Psi) \rangle$ 
  by auto
then show ?thesis
  by (metis
      arbitrary-disjunction.simps(2)
      disjunction-def
      list-deduction-def
      list-deduction-theorem
      list-deduction-weaken

```



```

      list-implication.simps(1)
      list-implication.simps(2))
    qed
  }
  then show ?case by blast
qed
thus set  $\Phi \subseteq$  set  $\Psi \implies \vdash \bigsqcup \Phi \rightarrow \bigsqcup \Psi$  by blast
qed

lemma (in classical-logic) arbitrary-disjunction-remdups:
   $\vdash (\bigsqcup \Phi) \leftrightarrow \bigsqcup (\text{remdups } \Phi)$ 
  by (simp add: arbitrary-disjunction-monotone biconditional-def)

lemma (in classical-logic) arbitrary-disjunction-exclusion-MCS:
  assumes MCS  $\Omega$ 
  shows  $\bigsqcup \Psi \notin \Omega \equiv \forall \psi \in \text{set } \Psi. \psi \notin \Omega$ 
proof (induct  $\Psi$ )
  case Nil
  then show ?case
  using
    assms
    formula-consistent-def
    formula-maximally-consistent-set-def-def
    maximally-consistent-set-def
    set-deduction-reflection
  by (simp, blast)
next
  case (Cons  $\psi \Psi$ )
  have  $\bigsqcup (\psi \# \Psi) \notin \Omega = (\psi \notin \Omega \wedge \bigsqcup \Psi \notin \Omega)$ 
  by (simp add: disjunction-def,
      meson
      assms
      formula-consistent-def
      formula-maximally-consistent-set-def-def
      formula-maximally-consistent-set-def-implication
      maximally-consistent-set-def
      set-deduction-reflection)
  thus ?case using Cons.hyps by simp
qed

lemma (in classical-logic) contra-list-curry-uncurry:
   $\vdash (\Phi \text{ :} \rightarrow \chi) \leftrightarrow (\sim \chi \rightarrow \bigsqcup (\sim \Phi))$ 
proof (induct  $\Phi$ )
  case Nil
  then show ?case
  by (simp,
      metis
      biconditional-introduction
      bivalence)

```

*disjunction-def*  
*double-negation-converse*  
*modus-ponens*  
*negation-def*

**next**  
**case** (*Cons*  $\varphi$   $\Phi$ )  
**hence**  $\vdash (\sqcap \Phi \rightarrow \chi) \leftrightarrow (\sim \chi \rightarrow \sqcup (\sim \Phi))$   
**by** (*metis*  
*biconditional-symmetry-rule*  
*biconditional-transitivity-rule*  
*list-curry-uncurry*)  
**have**  $\vdash (\sqcap (\varphi \# \Phi) \rightarrow \chi) \leftrightarrow (\sim \chi \rightarrow \sqcup (\sim (\varphi \# \Phi)))$   
**proof** –  
**have**  $\vdash (\sqcap \Phi \rightarrow \chi) \leftrightarrow (\sim \chi \rightarrow \sqcup (\sim \Phi))$   
 $\rightarrow ((\langle \varphi \rangle \sqcap \langle \sqcap \Phi \rangle) \rightarrow \langle \chi \rangle) \leftrightarrow (\sim \langle \chi \rangle \rightarrow (\sim \langle \varphi \rangle \sqcup \langle \sqcup (\sim \Phi) \rangle))$   
**proof** –  
**have**  
 $\forall \mathfrak{M}. \mathfrak{M} \models_{prop}$   
 $(\langle \sqcap \Phi \rangle \rightarrow \langle \chi \rangle) \leftrightarrow (\sim \langle \chi \rangle \rightarrow \langle \sqcup (\sim \Phi) \rangle)$   
 $\rightarrow ((\langle \varphi \rangle \sqcap \langle \sqcap \Phi \rangle) \rightarrow \langle \chi \rangle) \leftrightarrow (\sim \langle \chi \rangle \rightarrow (\sim \langle \varphi \rangle \sqcup \langle \sqcup (\sim \Phi) \rangle))$   
**by** *auto*  
**hence**  
 $\vdash ((\langle \sqcap \Phi \rangle \rightarrow \langle \chi \rangle) \leftrightarrow (\sim \langle \chi \rangle \rightarrow \langle \sqcup (\sim \Phi) \rangle)$   
 $\rightarrow ((\langle \varphi \rangle \sqcap \langle \sqcap \Phi \rangle) \rightarrow \langle \chi \rangle) \leftrightarrow (\sim \langle \chi \rangle \rightarrow (\sim \langle \varphi \rangle \sqcup \langle \sqcup (\sim \Phi) \rangle))) \triangleright$   
**using** *propositional-semantic* **by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**  
**thus** *?thesis*  
**using**  $\vdash (\sqcap \Phi \rightarrow \chi) \leftrightarrow (\sim \chi \rightarrow \sqcup (\sim \Phi))$  *modus-ponens* **by** *auto*  
**qed**  
**then show** *?case*  
**using** *biconditional-transitivity-rule list-curry-uncurry* **by** *blast*  
**qed**

### 5.9.5 Monotony of Conjunction and Disjunction

**lemma** (in *classical-logic*) *conjunction-monotonic-identity*:

$\vdash (\varphi \rightarrow \psi) \rightarrow (\varphi \sqcap \chi) \rightarrow (\psi \sqcap \chi)$

**unfolding** *conjunction-def*

**using** *modus-ponens*

*flip-hypothetical-syllogism*

**by** *blast*

**lemma** (in *classical-logic*) *conjunction-monotonic*:

**assumes**  $\vdash \varphi \rightarrow \psi$

**shows**  $\vdash (\varphi \sqcap \chi) \rightarrow (\psi \sqcap \chi)$

**using** *assms*

*modus-ponens*

*conjunction-monotonic-identity*

by *blast*

**lemma** (in *classical-logic*) *disjunction-monotonic-identity*:

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

**unfolding** *disjunction-def*

**using** *modus-ponens*

*flip-hypothetical-syllogism*

by *blast*

**lemma** (in *classical-logic*) *disjunction-monotonic*:

**assumes**  $\vdash \varphi \rightarrow \psi$

**shows**  $\vdash (\varphi \sqcup \chi) \rightarrow (\psi \sqcup \chi)$

**using** *assms*

*modus-ponens*

*disjunction-monotonic-identity*

by *blast*

### 5.9.6 Distribution Identities

**lemma** (in *classical-logic*) *conjunction-distribution*:

$\vdash ((\psi \sqcup \chi) \sqcap \varphi) \leftrightarrow ((\psi \sqcap \varphi) \sqcup (\chi \sqcap \varphi))$

**proof** –

**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ((\psi \sqcup \chi) \sqcap \langle \varphi \rangle) \leftrightarrow ((\psi \sqcap \langle \varphi \rangle) \sqcup ((\chi \sqcap \langle \varphi \rangle))$

by *auto*

**hence**  $\vdash \Downarrow ((\psi \sqcup \chi) \sqcap \langle \varphi \rangle) \leftrightarrow ((\psi \sqcap \langle \varphi \rangle) \sqcup ((\chi \sqcap \langle \varphi \rangle)) \Downarrow$

**using** *propositional-semantic* by *blast*

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

qed

**lemma** (in *classical-logic*) *subtraction-distribution*:

$\vdash ((\psi \sqcup \chi) \setminus \varphi) \leftrightarrow ((\psi \setminus \varphi) \sqcup (\chi \setminus \varphi))$

by (*simp add: conjunction-distribution subtraction-def*)

**lemma** (in *classical-logic*) *conjunction-arbitrary-distribution*:

$\vdash (\bigsqcup \Psi \sqcap \varphi) \leftrightarrow \bigsqcup [\psi \sqcap \varphi. \psi \leftarrow \Psi]$

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

**case** *Nil*

**then show** *?case*

by (*simp add: ex-falso-quodlibet*

*biconditional-def*

*conjunction-left-elimination*)

**next**

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

**have**  $\vdash (\bigsqcup (\psi \# \Psi) \sqcap \varphi) \leftrightarrow ((\psi \sqcap \varphi) \sqcup ((\bigsqcup \Psi) \sqcap \varphi))$

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

**moreover**

**from** *Cons* **have**

$\vdash ((\psi \sqcap \varphi) \sqcup ((\bigsqcup \Psi) \sqcap \varphi)) \leftrightarrow ((\psi \sqcap \varphi) \sqcup (\bigsqcup [\psi \sqcap \varphi. \psi \leftarrow \Psi]))$

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

by (*simp*, *meson modus-ponens hypothetical-syllogism*)  
 ultimately show ?case  
 by (*simp*, *metis biconditional-transitivity-rule*)  
 qed

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

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

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

**lemma** (in *classical-logic*) *list-implication-distribution*:  
 $\vdash (\Phi \rightarrow (\psi \sqcap \chi)) \leftrightarrow ((\Phi \rightarrow \psi) \sqcap (\Phi \rightarrow \chi))$   
**proof** (*induct*  $\Phi$ )  
 case *Nil*  
 then show ?case  
 by (*simp add: biconditional-reflection*)  
 next  
 case (*Cons*  $\varphi \Phi$ )  
 hence  $\vdash (\varphi \# \Phi) \rightarrow (\psi \sqcap \chi) \leftrightarrow (\varphi \rightarrow (\Phi \rightarrow \psi \sqcap \Phi \rightarrow \chi))$   
 by (*metis*  
   *modus-ponens*  
   *biconditional-def*  
   *hypothetical-syllogism*  
   *list-implication.simps(2)*  
   *weak-conjunction-deduction-equivalence*)  
 moreover  
 have  $\vdash (\varphi \rightarrow (\Phi \rightarrow \psi \sqcap \Phi \rightarrow \chi)) \leftrightarrow ((\varphi \# \Phi) \rightarrow \psi) \sqcap ((\varphi \# \Phi) \rightarrow \chi)$   
 using *implication-distribution* by *auto*  
 ultimately show ?case  
 by (*simp*, *metis biconditional-transitivity-rule*)

qed

**lemma** (in *classical-logic*) *biconditional-conjunction-weaken*:

$\vdash (\alpha \leftrightarrow \beta) \rightarrow ((\gamma \sqcap \alpha) \leftrightarrow (\gamma \sqcap \beta))$

**proof** –

**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} (\langle \alpha \rangle \leftrightarrow \langle \beta \rangle) \rightarrow ((\langle \gamma \rangle \sqcap \langle \alpha \rangle) \leftrightarrow (\langle \gamma \rangle \sqcap \langle \beta \rangle))$

**by** *auto*

**hence**  $\vdash (\langle \langle \alpha \rangle \leftrightarrow \langle \beta \rangle \rangle \rightarrow (\langle \langle \gamma \rangle \sqcap \langle \alpha \rangle \rangle \leftrightarrow \langle \langle \gamma \rangle \sqcap \langle \beta \rangle \rangle))$   $\square$

**using** *propositional-semantic* **by** *blast*

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

qed

**lemma** (in *classical-logic*) *biconditional-conjunction-weaken-rule*:

$\vdash (\alpha \leftrightarrow \beta) \implies \vdash (\gamma \sqcap \alpha) \leftrightarrow (\gamma \sqcap \beta)$

**using** *modus-ponens biconditional-conjunction-weaken* **by** *blast*

**lemma** (in *classical-logic*) *disjunction-arbitrary-distribution*:

$\vdash (\varphi \sqcup \sqcap \Psi) \leftrightarrow \sqcap [\varphi \sqcup \psi. \psi \leftarrow \Psi]$

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

**case** *Nil*

**then show** *?case*

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

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

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

**next**

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

**have**  $\vdash (\varphi \sqcup \sqcap (\psi \# \Psi)) \leftrightarrow ((\varphi \sqcup \psi) \sqcap (\varphi \sqcup \sqcap \Psi))$

**by** (*simp add: disjunction-distribution*)

**moreover**

**from** *biconditional-conjunction-weaken-rule*

*Cons*

**have**  $\vdash ((\varphi \sqcup \psi) \sqcap \varphi \sqcup \sqcap \Psi) \leftrightarrow \sqcap (\text{map } (\lambda \chi. \varphi \sqcup \chi) (\psi \# \Psi))$

**by** *simp*

**ultimately show** *?case*

**by** (*metis biconditional-transitivity-rule*)

qed

**lemma** (in *classical-logic*) *list-implication-arbitrary-distribution*:

$\vdash (\Phi \rightarrow \sqcap \Psi) \leftrightarrow \sqcap [\Phi \rightarrow \psi. \psi \leftarrow \Psi]$

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

**case** *Nil*

**then show** *?case*

**by** (*simp add: biconditional-def,*

*meson*

*axiom-k*

*modus-ponens*

*list-implication-axiom-k*

*verum-tautology*)

**next**

**case** (*Cons*  $\psi$   $\Psi$ )  
**have**  $\vdash \Phi \rightarrow \prod (\psi \# \Psi) \leftrightarrow (\Phi \rightarrow \psi \wedge \Phi \rightarrow \prod \Psi)$   
**using** *list-implication-distribution*  
**by** *fastforce*  
**moreover**  
**from** *biconditional-conjunction-weaken-rule*  
*Cons*  
**have**  $\vdash (\Phi \rightarrow \psi \wedge \Phi \rightarrow \prod \Psi) \leftrightarrow \prod [\Phi \rightarrow \psi. \psi \leftarrow (\psi \# \Psi)]$   
**by** *simp*  
**ultimately show** *?case*  
**by** (*metis biconditional-transitivity-rule*)  
**qed**

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

### 5.9.7 Negation

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

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

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

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

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

### 5.9.8 Mutual Exclusion Identities

**lemma** (in *classical-logic*) *exclusion-contrapositive-equivalence*:

$$\vdash (\varphi \rightarrow \gamma) \leftrightarrow \sim (\varphi \sqcap \sim \gamma)$$

**proof** –

$$\text{have } \forall \mathfrak{M}. \mathfrak{M} \models_{prop} (\langle \varphi \rangle \rightarrow \langle \gamma \rangle) \leftrightarrow \sim (\langle \varphi \rangle \sqcap \sim \langle \gamma \rangle)$$

by *auto*

$$\text{hence } \vdash \langle (\langle \varphi \rangle \rightarrow \langle \gamma \rangle) \leftrightarrow \sim (\langle \varphi \rangle \sqcap \sim \langle \gamma \rangle) \rangle$$

using *propositional-semantic* by *blast*

thus *?thesis* by *simp*

qed

**lemma** (in *classical-logic*) *disjunction-exclusion-equivalence*:

$$\Gamma \Vdash \sim (\psi \sqcap \sqcup \Phi) \equiv \forall \varphi \in \text{set } \Phi. \Gamma \Vdash \sim (\psi \sqcap \varphi)$$

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

case *Nil*

then show *?case*

by (*simp add*:

*conjunction-right-elimination*

*negation-def*

*set-deduction-weaken*)

next

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

$$\text{have } \vdash \sim (\psi \sqcap \sqcup (\varphi \# \Phi)) \leftrightarrow \sim (\psi \sqcap (\varphi \sqcup \sqcup \Phi))$$

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

$$\text{moreover have } \vdash \sim (\psi \sqcap (\varphi \sqcup \sqcup \Phi)) \leftrightarrow (\sim (\psi \sqcap \varphi) \sqcap \sim (\psi \sqcap \sqcup \Phi))$$

**proof** –

$$\begin{aligned} \text{have } \forall \mathfrak{M}. \mathfrak{M} \models_{prop} \sim (\langle \psi \rangle \sqcap (\langle \varphi \rangle \sqcup \langle \sqcup \Phi \rangle)) \\ \leftrightarrow (\sim (\langle \psi \rangle \sqcap \langle \varphi \rangle) \sqcap \sim (\langle \psi \rangle \sqcap \langle \sqcup \Phi \rangle)) \end{aligned}$$

by *auto*

$$\text{hence } \vdash \langle \sim (\langle \psi \rangle \sqcap (\langle \varphi \rangle \sqcup \langle \sqcup \Phi \rangle)) \rangle$$

$$\leftrightarrow \langle (\sim (\langle \psi \rangle \sqcap \langle \varphi \rangle) \sqcap \sim (\langle \psi \rangle \sqcap \langle \sqcup \Phi \rangle)) \rangle$$

using *propositional-semantic* by *blast*

thus *?thesis* by *simp*

qed

ultimately

$$\text{have } \vdash \sim (\psi \sqcap \sqcup (\varphi \# \Phi)) \leftrightarrow (\sim (\psi \sqcap \varphi) \sqcap \sim (\psi \sqcap \sqcup \Phi))$$

by *simp*

$$\text{hence } \Gamma \Vdash \sim (\psi \sqcap \sqcup (\varphi \# \Phi)) = (\Gamma \Vdash \sim (\psi \sqcap \varphi)$$

$$\wedge (\forall \varphi \in \text{set } \Phi. \Gamma \Vdash \sim (\psi \sqcap \varphi)))$$

using *set-deduction-weaken* [**where**  $\Gamma=\Gamma$ ]

*conjunction-set-deduction-equivalence* [**where**  $\Gamma=\Gamma$ ]

*Cons.hyps*

*biconditional-def*

*set-deduction-modus-ponens*

by *metis*

$$\text{thus } \Gamma \Vdash \sim (\psi \sqcap \sqcup (\varphi \# \Phi)) = (\forall \varphi \in \text{set } (\varphi \# \Phi). \Gamma \Vdash \sim (\psi \sqcap \varphi))$$

by *simp*

qed

**lemma** (in *classical-logic*) *exclusive-elimination1*:  
**assumes**  $\Gamma \Vdash \coprod \Phi$   
**shows**  $\forall \varphi \in \text{set } \Phi. \forall \psi \in \text{set } \Phi. (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$   
**using** *assms*  
**proof** (*induct*  $\Phi$ )  
  **case** *Nil*  
  **thus** *?case* **by** *auto*  
**next**  
  **case** (*Cons*  $\chi \Phi$ )  
  **assume**  $\Gamma \Vdash \coprod (\chi \# \Phi)$   
  **hence**  $\Gamma \Vdash \coprod \Phi$  **by** *simp*  
  **hence**  $\forall \varphi \in \text{set } \Phi. \forall \psi \in \text{set } \Phi. \varphi \neq \psi \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$   
  **using** *Cons.hyps* **by** *blast*  
  **moreover** **have**  $\Gamma \Vdash \sim (\chi \sqcap \sqcup \Phi)$   
  **using**  $\langle \Gamma \Vdash \coprod (\chi \# \Phi) \rangle$  *conjunction-set-deduction-equivalence* **by** *auto*  
  **hence**  $\forall \varphi \in \text{set } \Phi. \Gamma \Vdash \sim (\chi \sqcap \varphi)$   
  **using** *disjunction-exclusion-equivalence* **by** *auto*  
  **moreover** {  
    **fix**  $\varphi$   
    **have**  $\vdash \sim (\chi \sqcap \varphi) \rightarrow \sim (\varphi \sqcap \chi)$   
    **unfolding** *negation-def*  
      *conjunction-def*  
    **using** *modus-ponens flip-hypothetical-syllogism flip-implication* **by** *blast*  
  }  
  **with**  $\langle \forall \varphi \in \text{set } \Phi. \Gamma \Vdash \sim (\chi \sqcap \varphi) \rangle$  **have**  $\forall \varphi \in \text{set } \Phi. \Gamma \Vdash \sim (\varphi \sqcap \chi)$   
  **using** *set-deduction-weaken* [**where**  $\Gamma=\Gamma$ ]  
  *set-deduction-modus-ponens* [**where**  $\Gamma=\Gamma$ ]  
  **by** *blast*  
  **ultimately**  
  **show**  $\forall \varphi \in \text{set } (\chi \# \Phi). \forall \psi \in \text{set } (\chi \# \Phi). \varphi \neq \psi \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$   
  **by** *simp*  
**qed**

**lemma** (in *classical-logic*) *exclusive-elimination2*:  
**assumes**  $\Gamma \Vdash \coprod \Phi$   
**shows**  $\forall \varphi \in \text{duplicates } \Phi. \Gamma \Vdash \sim \varphi$   
**using** *assms*  
**proof** (*induct*  $\Phi$ )  
  **case** *Nil*  
  **then show** *?case* **by** *simp*  
**next**  
  **case** (*Cons*  $\varphi \Phi$ )  
  **assume**  $\Gamma \Vdash \coprod (\varphi \# \Phi)$   
  **hence**  $\Gamma \Vdash \coprod \Phi$  **by** *simp*  
  **hence**  $\forall \varphi \in \text{duplicates } \Phi. \Gamma \Vdash \sim \varphi$  **using** *Cons.hyps* **by** *auto*  
  **show** *?case*  
**proof** *cases*  
  **assume**  $\varphi \in \text{set } \Phi$   
  **moreover** {



```

fix  $\varphi \ \psi \ \chi$ 
have  $\vdash \sim (\varphi \sqcap (\psi \sqcup \chi)) \leftrightarrow (\sim (\varphi \sqcap \psi) \sqcap \sim (\varphi \sqcap \chi))$ 
proof –
  have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} \sim (\langle \varphi \rangle \sqcap (\langle \psi \rangle \sqcup \langle \chi \rangle))$ 
     $\leftrightarrow (\sim (\langle \varphi \rangle \sqcap \langle \psi \rangle) \sqcap \sim (\langle \varphi \rangle \sqcap \langle \chi \rangle))$ 
    by auto
  hence  $\vdash (\sqcup \sim (\langle \varphi \rangle \sqcap (\langle \psi \rangle \sqcup \langle \chi \rangle)) \leftrightarrow (\sim (\langle \varphi \rangle \sqcap \langle \psi \rangle) \sqcap \sim (\langle \varphi \rangle \sqcap \langle \chi \rangle)) \sqsupset)$ 
    using propositional-semantic by blast
  thus ?thesis by simp
qed
hence  $\Gamma \Vdash \sim (\varphi \sqcap (\psi \sqcup \chi)) \equiv \Gamma \Vdash \sim (\varphi \sqcap \psi) \sqcap \sim (\varphi \sqcap \chi)$ 
  using set-deduction-weaken
    biconditional-weaken by presburger
}
moreover
have  $\vdash \sim (\varphi \sqcap \varphi) \leftrightarrow \sim \varphi$ 
proof –
  have  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} \sim (\langle \varphi \rangle \sqcap \langle \varphi \rangle) \leftrightarrow \sim \langle \varphi \rangle$ 
    by auto
  hence  $\vdash (\sqcup \sim (\langle \varphi \rangle \sqcap \langle \varphi \rangle) \leftrightarrow \sim \langle \varphi \rangle \sqsupset)$ 
    using propositional-semantic by blast
  thus ?thesis by simp
qed
hence  $\Gamma \Vdash \sim (\varphi \sqcap \varphi) \equiv \Gamma \Vdash \sim \varphi$ 
  using set-deduction-weaken
    biconditional-weaken by presburger
moreover have  $\Gamma \Vdash \sim (\varphi \sqcap \sqcup \Phi) \text{ using } \langle \Gamma \Vdash \sqcup (\varphi \# \Phi) \rangle \text{ by } \textit{simp}$ 
ultimately have  $\Gamma \Vdash \sim \varphi \text{ by } (\textit{induct } \Phi, \textit{simp}, \textit{simp}, \textit{blast})$ 
thus ?thesis using  $\langle \varphi \in \textit{set } \Phi \rangle \langle \forall \varphi \in \textit{duplicates } \Phi. \Gamma \Vdash \sim \varphi \rangle \text{ by } \textit{simp}$ 
next
assume  $\varphi \notin \textit{set } \Phi$ 
hence duplicates  $(\varphi \# \Phi) = \textit{duplicates } \Phi \text{ by } \textit{simp}$ 
then show ?thesis using  $\langle \forall \varphi \in \textit{duplicates } \Phi. \Gamma \Vdash \sim \varphi \rangle$ 
  by auto
qed
qed

lemma (in classical-logic) exclusive-equivalence:
 $\Gamma \Vdash \sqcup \Phi =$ 
   $(\forall \varphi \in \textit{duplicates } \Phi. \Gamma \Vdash \sim \varphi) \wedge$ 
   $(\forall \varphi \in \textit{set } \Phi. \forall \psi \in \textit{set } \Phi. (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi))$ 
proof –
{
  assume  $\forall \varphi \in \textit{duplicates } \Phi. \Gamma \Vdash \sim \varphi$ 
     $\forall \varphi \in \textit{set } \Phi. \forall \psi \in \textit{set } \Phi. (\varphi \neq \psi) \longrightarrow \Gamma \Vdash \sim (\varphi \sqcap \psi)$ 
  hence  $\Gamma \Vdash \sqcup \Phi$ 
proof (induct  $\Phi$ )
  case Nil
  then show ?case

```



```

    }
    ultimately show ?thesis by auto
  qed
qed
with ⟨Γ ⊢ ∼ φ⟩ have Γ ⊢ ∼(φ ⊓ ⊔ Φ)
  using biconditional-weaken set-deduction-weaken by blast
with ⟨Γ ⊢ ⊔ Φ⟩ show ?thesis by simp
next
assume φ ∉ duplicates (φ # Φ)
hence φ ∉ set Φ by auto
with B have ∀ψ∈set Φ. Γ ⊢ ∼(φ ⊓ ψ) by (simp, metis)
hence Γ ⊢ ∼(φ ⊓ ⊔ Φ)
  by (simp add: disjunction-exclusion-equivalence)
with ⟨Γ ⊢ ⊔ Φ⟩ show ?thesis by simp
qed
qed
}
thus ?thesis
  by (metis exclusive-elimination1 exclusive-elimination2)
qed

```

### 5.9.9 Miscellaneous Disjunctive Normal Form Identities

**lemma** (in *classical-logic*) *map-negation-list-implication*:

$\vdash ((\sim \Phi) \rightarrow (\sim \varphi)) \leftrightarrow (\varphi \rightarrow \sqcup \Phi)$

**proof** (*induct* Φ)

*case Nil*

**then show** ?case

**unfolding**

*biconditional-def*

*map-negation-def*

*negation-def*

**using**

*conjunction-introduction*

*modus-ponens*

*trivial-implication*

**by** *simp*

**next**

*case (Cons ψ Φ)*

**have**  $\vdash (\sim \Phi \rightarrow \sim \varphi \leftrightarrow (\varphi \rightarrow \sqcup \Phi))$

$\rightarrow (\sim \psi \rightarrow \sim \Phi \rightarrow \sim \varphi) \leftrightarrow (\varphi \rightarrow (\psi \sqcup \sqcup \Phi))$

**proof** –

**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ((\sim \Phi \rightarrow \sim \varphi) \leftrightarrow (\langle \varphi \rangle \rightarrow \langle \sqcup \Phi \rangle)) \rightarrow$   
 $(\sim \langle \psi \rangle \rightarrow \langle \sim \Phi \rightarrow \sim \varphi \rangle) \leftrightarrow (\langle \varphi \rangle \rightarrow (\langle \psi \rangle \sqcup \langle \sqcup \Phi \rangle))$

**by** *fastforce*

**hence**  $\vdash (\langle \sim \Phi \rightarrow \sim \varphi \rangle \leftrightarrow (\langle \varphi \rangle \rightarrow \langle \sqcup \Phi \rangle)) \rightarrow$

$(\sim \langle \psi \rangle \rightarrow \langle \sim \Phi \rightarrow \sim \varphi \rangle) \leftrightarrow (\langle \varphi \rangle \rightarrow (\langle \psi \rangle \sqcup \langle \sqcup \Phi \rangle)) \Downarrow$

**using** *propositional-semantic* **by** *blast*

**thus** ?thesis

by *simp*  
**qed**  
**with** *Cons* **show** *?case*  
 by (*metis*  
   *map-negation-def*  
   *list.simps(9)*  
   *arbitrary-disjunction.simps(2)*  
   *modus-ponens*  
   *list-implication.simps(2)*)

**qed**

**lemma** (in *classical-logic*) *conj-dnf-distribute*:

$\vdash \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) \Lambda) \leftrightarrow (\varphi \sqcap \sqcup (map \sqcap \Lambda))$

**proof**(*induct*  $\Lambda$ )

case *Nil*

**have**  $\vdash \perp \leftrightarrow (\varphi \sqcap \perp)$

**proof** –

let  $? \varphi = \perp \leftrightarrow (\langle \varphi \rangle \sqcap \perp)$

**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*  $\Psi \Lambda$ )

**assume**  $\vdash \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) \Lambda) \leftrightarrow (\varphi \sqcap \sqcup (map \sqcap \Lambda))$

(**is**  $\vdash ?A \leftrightarrow (\varphi \sqcap ?B)$ )

**moreover**

**have**  $\vdash (?A \leftrightarrow (\varphi \sqcap ?B)) \rightarrow (((\varphi \sqcap \sqcap \Psi) \sqcup ?A) \leftrightarrow (\varphi \sqcap \sqcap \Psi \sqcup ?B))$

**proof** –

let  $? \varphi = (\langle ?A \rangle \leftrightarrow (\langle \varphi \rangle \sqcap \langle ?B \rangle)) \rightarrow$

$((\langle \varphi \rangle \sqcap \langle \sqcap \Psi \rangle) \sqcup \langle ?A \rangle) \leftrightarrow (\langle \varphi \rangle \sqcap \langle \sqcap \Psi \rangle \sqcup \langle ?B \rangle)$

**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**  $\vdash ((\varphi \sqcap \sqcap \Psi) \sqcup ?A) \leftrightarrow (\varphi \sqcap \sqcap \Psi \sqcup ?B)$

**using** *modus-ponens*

**by** *blast*

**moreover**

**have**  $map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) \Lambda = map (\lambda \Psi. \varphi \sqcap \sqcap \Psi) \Lambda$  **by** *simp*

**ultimately show** *?case* **by** *simp*

**qed**

**lemma** (in *classical-logic*) *append-dnf-distribute*:

$\vdash \sqcup (map (\sqcap \circ (\lambda \Psi. \Phi @ \Psi)) \Lambda) \leftrightarrow (\sqcap \Phi \sqcap \sqcup (map \sqcap \Lambda))$

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

**case Nil**  
**have**  $\vdash \sqcup (map \sqcap \Lambda) \leftrightarrow (\top \sqcap \sqcup (map \sqcap \Lambda))$   
(is  $\vdash ?A \leftrightarrow (\top \sqcap ?A)$ )  
**proof** –  
**let**  $? \varphi = \langle ?A \rangle \leftrightarrow ((\perp \rightarrow \perp) \sqcap \langle ?A \rangle)$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ? \varphi$  **by simp**  
**hence**  $\vdash (\langle ? \varphi \rangle)$  **using propositional-semantic by blast**  
**thus**  $?thesis$   
**unfolding** *verum-def*  
**by simp**  
**qed**  
**then show**  $?case$  **by simp**  
**next**  
**case (Cons  $\varphi \Phi$ )**  
**have**  $\vdash \sqcup (map (\sqcap \circ (@) \Phi) \Lambda) \leftrightarrow (\sqcap \Phi \sqcap \sqcup (map \sqcap \Lambda))$   
 $= \vdash \sqcup (map \sqcap (map ((@) \Phi) \Lambda)) \leftrightarrow (\sqcap \Phi \sqcap \sqcup (map \sqcap \Lambda))$   
**by simp**  
**with Cons have**  
 $\vdash \sqcup (map \sqcap (map (\lambda \Psi. \Phi @ \Psi) \Lambda)) \leftrightarrow (\sqcap \Phi \sqcap \sqcup (map \sqcap \Lambda))$   
(is  $\vdash \sqcup (map \sqcap ?A) \leftrightarrow (?B \sqcap ?C)$ )  
**by meson**  
**moreover have**  $\vdash \sqcup (map \sqcap ?A) \leftrightarrow (?B \sqcap ?C)$   
 $\rightarrow (\sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) ?A) \leftrightarrow (\varphi \sqcap \sqcup (map \sqcap ?A)))$   
 $\rightarrow \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) ?A) \leftrightarrow ((\varphi \sqcap ?B) \sqcap ?C)$   
**proof** –  
**let**  $? \varphi = \langle \sqcup (map \sqcap ?A) \rangle \leftrightarrow (\langle ?B \rangle \sqcap \langle ?C \rangle)$   
 $\rightarrow (\langle \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) ?A) \rangle \leftrightarrow (\langle \varphi \rangle \sqcap \langle \sqcup (map \sqcap ?A) \rangle))$   
 $\rightarrow \langle \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) ?A) \rangle \leftrightarrow ((\langle \varphi \rangle \sqcap \langle ?B \rangle) \sqcap \langle ?C \rangle)$   
**have**  $\forall \mathfrak{M}. \mathfrak{M} \models_{prop} ? \varphi$  **by simp**  
**hence**  $\vdash (\langle ? \varphi \rangle)$  **using propositional-semantic by blast**  
**thus**  $?thesis$   
**by simp**  
**qed**  
**ultimately have**  $\vdash \sqcup (map (\sqcap \circ (\lambda \varphi s. \varphi \# \varphi s)) ?A) \leftrightarrow ((\varphi \sqcap ?B) \sqcap ?C)$   
**using** *modus-ponens conj-dnf-distribute*  
**by blast**  
**moreover**  
**have**  $\sqcap \circ (@) (\varphi \# \Phi) = \sqcap \circ (\#) \varphi \circ (@) \Phi$  **by auto**  
**hence**  
 $\vdash \sqcup (map (\sqcap \circ (@) (\varphi \# \Phi)) \Lambda) \leftrightarrow (\sqcap (\varphi \# \Phi) \sqcap ?C)$   
 $= \vdash \sqcup (map (\sqcap \circ (\#) \varphi) ?A) \leftrightarrow ((\varphi \sqcap ?B) \sqcap ?C)$   
**by simp**  
**ultimately show**  $?case$  **by meson**  
**qed**  
  
**notation** *FuncSet.funcset* (**infixr**  $\rightarrow 60$ )  
  
**end**

# Bibliography

- [1] P. Blackburn, M. de Rijke, and Y. Venema. Section 4.2 Canonical Models. In *Modal Logic*, pages 196–201.
- [2] A. Bobenrieth. The Origins of the Use of the Argument of Trivialization in the Twentieth Century. 31(2):111–121.
- [3] G. Boolos. Don't Eliminate Cut. 13(4):373–378.
- [4] G. Gentzen. Untersuchungen über das logische schließen. i. 39(1):176–210.
- [5] C. S. Peirce. On the Algebra of Logic: A Contribution to the Philosophy of Notation. 7(2):180–196.
- [6] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2nd ed edition.
- [7] A. Urquhart. Implicational Formulas in Intuitionistic Logic. 39(4):661–664.
- [8] D. van Dalen. *Logic and Structure*. Universitext. Springer-Verlag, 5 edition.