UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

INTUITIONISTIC LOGIC IN LEAN

Absolvent
Dafina Trufaș

Coordonator științific
Prof. dr. Laurențiu Leuștean

București, iulie 2024

Rezumat

Demonstrarea de teoreme cu ajutorul calculatorului reprezintă un punct crucial în evoluția informaticii, și a științei, în general, din a doua jumătate a secolului trecut, până în prezent. Se disting două direcții, corespunzătoare modalităților în care calculatorul poate constitui un sprijin în elaborarea de demonstrații pentru rezultate specificate formal. Demonstrarea automată nu necesită intervenție din partea utilizatorului, ci utilizează intern tehnici precum rezoluția, pentru a genera o demonstrație. Pe de altă parte, demonstrarea interactivă, utilizată în această lucrare, presupune aportul utilizatorului și se axează pe aspectul de verificare. Sistemele de demonstrare interactivă bazate pe teoria tipurilor folosesc izomorfismul Curry-Howard și verifică dacă tipurile de date ale expresiilor introduse pe parcursul unei demonstrații sunt consistente cu cele așteptate la momentul respectiv, scopul ultim fiind construirea unui termen care să ateste validitatea enunțului vizat. Această lucrare își propune studiul și formalizarea în limbajul Lean a Logicii Intuiționiste Propoziționale. Rezultatul central demonstrat în această teză este teorema de completitudine, pentru care oferim două demonstrații. Formalizarea semanticii algebrice, a teoremei de completitudine corespunzătoare, dar și a echivalenței dintre noțiunile de validitate generate de cele două semantici, reprezintă contribuții originale.

Abstract

Computer-assisted theorem proving is a crucial step in the evolution of informatics and science, in general, since the last half of the 20^{th} century and until nowadays. Two main directions can be distinguished. They correspond to the ways in which a machine can support humans in the process of establishing the truth of a formally-specified claim. Automated theorem proving doesn't require any input from the user - it generates proofs using specific techniques such as resolution. On the other hand, interactive theorem proving requires much more input from the user and focuses on the verification aspect. Interactive theorem-provers based on type theory use the Curry-Howard isomorphism and type-check the expressions along a proof, the final purpose of the human prover being to construct an inhabitant-term of the desired type. This thesis aims to study and formalize Intuitionistic Propositional Logic in the Lean proof assistant. The central result proved throughout this thesis is the completeness theorem, for which we provide two different proofs. The formalization of the algebraic semantics, along with the corresponding completeness theorem and the equivalence between the generated validity notions in the two semantics are original contributions of this thesis.

Contents

Chapter 1

Introduction

The central aim of this thesis is to illustrate how the Lean 4 interactive thoerem prover can be used in order to formalize Intuitionistic Propositional Logic and prove its completeness. In the first part, we provide a detailed theoretical presentation of the formal system, described from a tripartite perspective (syntax, semantics and algebra), culminating with two proofs of the completeness theorem. Our exposition is based on the approaches in [12, 3, 15, 11, 5, 4]. In the second section of the thesis, we proceed to present our formalization of the definitions and results, motivating our main design options and emphasizing the technical difficulties that arose, along with the ways we dealt with them. To the best of our knowledge, the only existent formalization of Intuitionistic Propositional Logic is [6], where the authors give a Henkin-style proof of the completeness theorem, using Kripke semantics. In this thesis, we provide an alternative implementation, adding also an algebraic completeness proof and the connection between the two semantics.

The development of intuitionism in mathematics started at the beginning of the 20^{th} century, during the so-called foundational crisis of mathematics. The father of intuitionism is widely considered to be L.E.J. Brouwer (for a biography of Brouwer we refer to [17]), who, in 1908, was the first to reject the law of excluded middle, one of the main valid principles in classical reasoning. This change of paradigm led to serious consequences, such as the rejection of the law of double negation elimination. The next crucial point in the history of intuitionism and intuitionistic logic is represented by the formalizations given by Arend Heyting (student of Brouwer) and Gerhard Gentzen, in the 1930s, as Hilbert and Natural Deduction systems respectively. In the following decade (1943, to be more precise), Arend Heyting and Andrey Kolmogorov independently proposed an interpretation of intuitionistic logic, inductively specifying what is intended to be a proof of a given formula. This is known as the Brouwer-Heyting-Kolmogorov (BHK) interpretation, honorifically including Brouwer's name first. In this thesis, we focus on two semantics for the Intuitionistic Propositional Logic. The first one, introduced by Saul Kripke in the 1960s, in [9, 10], uses Kripke models. The second one is an algebraic semantics, based on Heyting algebras, also called pseudo-boolean algebras (we refer to [13] for a textbook treatment of these algebras).

The thesis is structured as follows: in Chapter 2, we describe the language of Intuitionistic Propositional Logic, define its syntax and proof-system, within we prove some theorems and derived deduction rules. Also, we prove the deduction theorem and define disjunctive theories, consistent and complete pairs of sets of formulas. These notions and some results regarding them we prove hereby will be used later, in the first proof of the completeness theorem. Chapter 3 is focused on defining the two types of semantics we are interested in: Kripke and algebraic semantics, concluding with the proofs of their completeness and the equivalence modulo validity between them. Finally, Chapter 4 proceeds by briefly sketching an overview of the Lean proof assistant and then providing relevant code-sequences, along with natural language explanations for the main stages of the formalization process.

Chapter 2

Intuitionistic Propositional Logic - language and syntax

In this chapter we define the language and syntax of Intuitionistic Propositional Logic (IPL). The syntax is given using the Hilbert-style proof system $\mathcal{G}$, introduced by Gödel in [7]. Furthermore, we prove some basic but essential results about Γ-theorems.

2.1 Language

Definition 2.1.1. *The* ***language*** *of IPL consists of:*

(i) a countable set $Var = \{v_n \mid n \in \mathbb{N}\}$ of variables;

(ii) the connectives $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implies);

(iii) the propositional constant $\perp$ (false);

The set of **symbols** of the above defined language is $Sym := Var \cup \{\wedge, \vee, \rightarrow, \perp, (,)\}$. The set of all finite sequences of these symbols is the set of IPL-**expressions**, which we denote by $Expr$. Below, we define the **formulas** of our language, that is, the set of well-formed IPL-expressions.

Definition 2.1.2. *The* ***formulas*** *of IPL are the IPL-expressions defined as follows:*

(i) Any variable is a formula.

(ii) $\perp$ is a formula.

(iii) If φ and ψ are formulas, then $(\varphi \wedge \psi)$ is a formula.

(iv) If φ and ψ are formulas, then $(\varphi \vee \psi)$ is a formula.

(v) If φ and ψ are formulas, then $(\varphi \to \psi)$ is a formula.

(vi) Only the expressions obtained by applying one of the above rules are formulas.

Alternatively, the set of IPL-formulas can be defined as the smallest set of IPL-expressions which includes Var and false and is closed under the three connectives.

Definition 2.1.3. *The set of IPL-formulas is the intersection of all sets Γ of expressions that have the following properties:*

(i) $Var \cup \{\perp\} \subseteq \Gamma$.

(ii) Γ *is closed to* $\wedge$, $\vee$, *and* $\to$, *that is:*

$$\text{if } \varphi, \psi \in \Gamma, \text{ then } (\varphi \wedge \psi), (\varphi \vee \psi) \text{ and } (\varphi \to \psi) \in \Gamma.$$

Notation 2.1.4. *We denote the set of IPL-formulas by $Form$.*

As a consequence of the inductive definition of formulas, one gets the induction principle on formulas, a method of proof which will be intensively used throughout the thesis.

Proposition 2.1.5 (induction principle on formulas)**.** *Let Γ be a set of formulas satisfying the following properties:*

(i) Γ *contains all variables.*

(ii) Γ *contains* $\perp$.

(iii) Γ *is closed to* $\wedge$, $\vee$ *and* $\to$.

Then $\Gamma = Form$.

This induction principle is used to prove that all formulas have a property $\mathcal{P}$: we define Γ as the set of all formulas satisfying $\mathcal{P}$ and apply induction on formulas to conclude that $\Gamma = Form$.

Another consequence of the inductive definition of formulas is the recursion principle on formulas, the method used to define functions whose domain is the set of all formulas.

Proposition 2.1.6 (recursion principle on formulas). *Let D be a set and the mappings*

$$\begin{array}{ll} G_0 : Var \to D & G_\wedge : D^2 \times Form^2 \to D \\ G_\perp : \{\perp\} \to D & G_\vee : D^2 \times Form^2 \to D \\ & G_\to : D^2 \times Form^2 \to D \end{array}$$

Then there exists a unique mapping

$$F : Form \to D$$

that satisfies the following properties:

(i) $F(v) = G_0(v)$ *for any variable* v.

(ii) $F(\perp) = G_0(\perp)$.

(iii) $F((\varphi \wedge \psi)) = G_\wedge(F(\varphi), F(\psi), \varphi, \psi)$ *for any formulas* φ, ψ.

(iv) $F((\varphi \vee \psi)) = G_\vee(F(\varphi), F(\psi), \varphi, \psi)$ *for any formulas* φ, ψ.

(v) $F((\varphi \to \psi)) = G_\to(F(\varphi), F(\psi), \varphi, \psi)$ *for any formulas* φ, ψ.

We introduce derived connectives by the following abbreviations:

$$\neg\varphi = (\varphi \to \perp), \quad (\varphi \leftrightarrow \psi) = (\varphi \to \psi) \wedge (\psi \to \varphi),$$

To reduce the use of parentheses, we omit the exterior parentheses, when they are not necessary. Thus, we write $\varphi \to \psi$, instead of $(\varphi \to \psi)$, but we have to write $(\varphi \to \psi) \to \chi$. Additionally, we assume that:

(i) $\neg$ has higher precedence than $\to$, $\wedge$, $\vee$, $\leftrightarrow$;

(ii) $\wedge, \vee$ have higher precedence than $\to$, $\leftrightarrow$.

Applying this rules, the formula $((\neg\varphi) \leftrightarrow (\psi \vee \chi))$ will be written $\neg\varphi \leftrightarrow \psi \vee \chi$.

We define in the sequel **finite conjunctions and disjunctions**. Let $\varphi_1, \ldots, \varphi_n (n \geq 1)$ be formulas. Then $\bigwedge_{i=1}^{n} \varphi_i$ and $\bigvee_{i=1}^{n} \varphi_i$ are defined inductively as follows:

$$\bigwedge_{i=1}^{1} \varphi_i = \varphi_1, \quad \bigwedge_{i=1}^{2} \varphi_i = \varphi_1 \wedge \varphi_2, \quad \bigwedge_{i=1}^{n+1} \varphi_i = \left(\bigwedge_{i=1}^{n} \varphi_i\right) \wedge \varphi_{n+1}, \tag{2.1}$$

$$\bigvee_{i=1}^{1} \varphi_i = \varphi_1, \quad \bigvee_{i=1}^{2} \varphi_i = \varphi_1 \vee \varphi_2, \quad \bigvee_{i=1}^{n+1} \varphi_i = \left(\bigvee_{i=1}^{n} \varphi_i\right) \vee \varphi_{n+1}. \tag{2.2}$$

We also write $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$ instead of $\bigwedge_{i=1}^{n} \varphi_i$ and $\varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_n$ instead of $\bigvee_{i=1}^{n} \varphi_i$.

2.2 Syntax

We consider Gödel's proof system for IPL, denoted by $\mathcal{G}$.
The axioms of $\mathcal{G}$ are:

(CONTRACTION)	$\varphi \vee \varphi \to \varphi$,	$\varphi \to \varphi \wedge \varphi$
(WEAKENING)	$\varphi \to \varphi \vee \psi$,	$\varphi \wedge \psi \to \varphi$
(PERMUTATION)	$\varphi \vee \psi \to \psi \vee \varphi$,	$\varphi \wedge \psi \to \psi \wedge \varphi$
(EX FALSO QUODLIBET)	$\bot \to \varphi$	

The deduction rules of $\mathcal{G}$ are:

(MODUS PONENS) $\dfrac{\varphi \quad \varphi \to \psi}{\psi}$

(SYLLOGISM) $\dfrac{\varphi \to \psi \quad \psi \to \chi}{\varphi \to \chi}$

(EXPORTATION) $\dfrac{\varphi \wedge \psi \to \chi}{\varphi \to (\psi \to \chi)}$

(IMPORTATION) $\dfrac{\varphi \to (\psi \to \chi)}{\varphi \wedge \psi \to \chi}$

(EXPANSION) $\dfrac{\varphi \to \psi}{(\chi \vee \varphi) \to (\chi \vee \psi)}$

There are two types of deduction rules:

(i) $\dfrac{\varphi}{\psi}$: from φ deduce ψ. φ is the **premise** of the rule and ψ is the **conclusion**.

(ii) $\dfrac{\varphi \quad \psi}{\chi}$: from φ and ψ deduce χ. φ and ψ are the **premises** of the rule and χ is the **conclusion**.

Let Γ be a set of formulas. In the sequel, we define the notion of Γ-theorem, followed by the corresponding induction principle.

Definition 2.2.1. *The set of* Γ***-theorems*** *is the intersection of all sets* Δ *of formulas that have the following properties:*

(i) Δ *contains all the axioms.*

(ii) Δ *contains all the formulas from* Γ*, that is* $\Gamma \subseteq \Delta$*.*

(iii) For every inference rule, the following holds: If Δ *contains its premise(s), then its conclusion is also in* Δ*.*

The set of Γ-theorems is denoted by $Thm(\Gamma)$. If φ is a Γ-theorem, then we also say that φ is deduced from the hypotheses Γ. As an immediate consequence of Definition 2.2.1, we get the induction principle on Γ-theorems.

Proposition 2.2.2. *[Induction principle on* Γ*-theorems]*
Let Δ *be a set of formulas satisfying the following properties:*

(i) Δ *contains all the axioms.*

(ii) $\Gamma \subseteq \Delta$*.*

(iii) For every inference rule, the following holds: If Δ *contains its premise(s), then its conclusion is also in* Δ*.*

Then $Thm(\Gamma) \subseteq \Delta$*.*

Notation 2.2.3. *Let* Γ, Δ *be sets of formulas and* φ *be a formula. We use the following notations:*

$$\begin{array}{lcl} \Gamma \vdash \varphi & = & \varphi \textit{ is a } \Gamma\textit{-theorem.} \\ \vdash \varphi & = & \emptyset \vdash \varphi, \\ \Gamma \vdash \Delta & \Leftrightarrow & \Gamma \vdash \varphi \textit{ for any } \varphi \in \Delta. \end{array}$$

Definition 2.2.4. *A formula* φ *is called a* IPL***-theorem*** *if* $\vdash \varphi$*.*

The following proposition contains some useful results about theorems.

Proposition 2.2.5. *Let* Γ, Δ *be sets of formulas.*

(i) Assume that $\Delta \subseteq \Gamma$*. Then,* $Thm(\Delta) \subseteq Thm(\Gamma)$*, that is, for every formula* φ*,*

$$\Delta \vdash \varphi \textit{ implies } \Gamma \vdash \varphi.$$

(ii) *For every formula* φ, $Thm(\emptyset) \subseteq Thm(\Gamma)$, *that is*

$$\vdash \varphi \textit{ implies } \Gamma \vdash \varphi.$$

(iii) *Assume that* $\Gamma \vdash \Delta$. *Then* $Thm(\Delta) \subseteq Thm(\Gamma)$, *that is, for every formula* φ,

$$\Delta \vdash \varphi \textit{ implies } \Gamma \vdash \varphi.$$

(iv) *Assume that* $\Gamma \vdash \Delta$ *and that* $\Delta \vdash \Gamma$. *Then* $Thm(\Delta) = Thm(\Gamma)$, *that is, for every formula* φ,

$$\Delta \vdash \varphi \textit{ iff } \Gamma \vdash \varphi.$$

(v) *For every formula* φ, $Thm(Thm(\Gamma)) = Thm(\Gamma)$, *that is*

$$Thm(\Gamma) \vdash \varphi \quad \textit{iff } \Gamma \vdash \varphi.$$

Proof. (i) As $\Delta \subseteq \Gamma$, one proves immediately by induction on Δ-theorems that $Thm(\Delta) \subseteq Thm(\Gamma)$.

(ii) Apply (i) with $\Delta = \emptyset$.

(iii) As, by hypothesis, $\Delta \subseteq Thm(\Gamma)$, one proves immediately by induction on Δ-theorems that $Thm(\Delta) \subseteq Thm(\Gamma)$.

(iv) Apply (iii) twice.

(v) $\Leftarrow$ As, by (ii), $\Gamma \subseteq Thm(\Gamma)$, we can apply (i) to get that $Thm(\Gamma) \subseteq Thm(Thm(\Gamma))$.

$\Rightarrow$ We have that $\Gamma \vdash Thm(\Gamma)$, so we can apply (iii) with $\Delta = Thm(\Gamma)$ to get that $Thm(Thm(\Gamma)) \subseteq Thm(\Gamma)$.

□

Definition 2.2.6. *A* Γ***-proof*** *(or* ***proof from the hypotheses*** Γ*) is a sequence of formulas* $\theta_1, \dots, \theta_n$ *such that for all* $i \in \{1, \dots, n\}$, *one of the following holds:*

(i) θ_i *is an axiom.*

(ii) $\theta_i \in \Gamma$.

(iii) θ_i *is the conclusion of an inference rule whose premise(s) are previous formula(e).*

An $\emptyset$*-proof is called simply a* ***proof****.*

Definition 2.2.7. *Let φ be a formula. A* Γ***-proof of*** φ *or a* ***proof of*** φ ***from the hypotheses*** Γ *is a Γ-proof $\theta_1, \ldots, \theta_n$ such that $\theta_n = \varphi$.*

Proposition 2.2.8. *For any formula φ,*

$$\Gamma \vdash \varphi \quad \textit{iff} \quad \textit{there exists a } \Gamma\textit{-proof of } \varphi.$$

Proof. Let us denote $\Theta = \{\varphi \in Form \mid \text{there exists a } \Gamma\text{-proof of } \varphi\}$.

"$\Rightarrow$" We prove by induction on Γ-theorems that $Thm(\Gamma) \subseteq \Theta$:

(i) φ is an axiom or a member of Γ. Then φ is a Γ-proof of φ. Hence, $\varphi \in \Theta$.

(ii) Let $\dfrac{\psi}{\varphi}$ be an inference rule such that $\psi \in \Theta$. Then there exists a Γ-proof $\theta_1, \ldots, \theta_n = \psi$ of ψ. It follows that $\theta_1, \ldots, \theta_n = \psi, \theta_{n+1} = \varphi$ is a Γ-proof of φ. Thus, $\varphi \in \Theta$.

(iii) Let $\dfrac{\psi \quad \chi}{\varphi}$ be an inference rule such that $\psi, \chi \in \Theta$. Then there exists a Γ-proof $\theta_1, \ldots, \theta_n = \psi$ of ψ and a Γ-proof $\delta_1, \ldots, \delta_p = \chi$ of χ. It follows that $\theta_1, \ldots, \theta_n = \psi, \theta_{n+1} = \delta_1, \ldots, \theta_{n+p} = \delta_p = \chi, \theta_{n+p+1} = \varphi$ is a Γ-proof of φ. Thus, $\varphi \in \Theta$.

"$\Leftarrow$" Assume that φ has a Γ-proof $\theta_1, \ldots, \theta_n = \varphi$. We prove by induction on i that for all $i = 1, \ldots, n$, $\Gamma \vdash \theta_i$. As a consequence, $\Gamma \vdash \theta_n = \varphi$.

(i) $i = 1$. Then θ_1 must be an axiom or a member of Γ. Then obviously $\Gamma \vdash \theta_1$.

(ii) Assume that the induction hypothesis is true for all $j = 1, \ldots, i$. If θ_{i+1} is an axiom or a member of Γ, then obviously $\Gamma \vdash \theta_{i+1}$. Assume that θ_{i+1} is the conclusion of an inference rule whose premise(s) are previous formula(s). If the inference rule is of type $\dfrac{\theta_j}{\theta_{i+1}}$ with $j \leq i$, then by the induction hypothesis we have that $\Gamma \vdash \theta_j$. By the definition of Γ-theorems, it follows that $\Gamma \vdash \theta_{i+1}$. If the inference rule is of type $\dfrac{\theta_j \quad \theta_k}{\theta_{i+1}}$ with $j, k \leq i$, then by the induction hypothesis we have that $\Gamma \vdash \theta_j$ and $\Gamma \vdash \theta_k$. By the definition of Γ-theorems, it follows that $\Gamma \vdash \theta_{i+1}$.

□

Lemma 2.2.9. *For any set of formulas Γ and formula φ, we have that:*

$$\Gamma \vdash \varphi \textit{ implies that there exists a finite subset } \Delta \textit{ of } \Gamma\textit{, such that } \Delta \vdash \varphi$$

Proof. By Proposition 2.2, we have that there exists a Γ-proof $\theta_1, \ldots, \theta_n = \varphi$.
Let $\Delta = \Gamma \cap \{\theta_1, \ldots, \theta_n\}$. Then, Δ is finite, $\Delta \subseteq \Gamma$ and $\theta_1, \ldots, \theta_n = \varphi$ is a Δ-proof of φ, so $\Delta \vdash \varphi$. □

2.3 Theorems and derived deduction rules

Throughout this section, we aim to prove useful theorems and deduction rules of IPL. Kleene's textbook [8] is a main source of inspiration for obtaining these proofs.
In the sequel, Γ is a set of formulas and $\varphi, \psi, \chi, \gamma, \delta$ are formulas.

Lemma 2.3.1.

$$\Gamma \vdash \psi \to \varphi \vee \psi \tag{2.3}$$
$$\Gamma \vdash \varphi \wedge \psi \to \psi \tag{2.4}$$
$$\Gamma \vdash \varphi \to (\psi \to \varphi) \tag{2.5}$$
$$\Gamma \vdash \varphi \wedge \psi \to \varphi \vee \gamma \tag{2.6}$$
$$\Gamma \vdash \varphi \to \varphi \tag{2.7}$$
$$\Gamma \vdash \varphi \to (\psi \to \varphi \wedge \psi) \tag{2.8}$$
$$\Gamma \vdash (\varphi \to \psi) \wedge \varphi \to \psi \tag{2.9}$$
$$\Gamma \vdash \varphi \wedge (\varphi \to \psi) \to \psi \tag{2.10}$$
$$\Gamma \vdash \varphi \to ((\varphi \to \psi) \to \psi) \tag{2.11}$$
$$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \tag{2.12}$$
$$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \psi \tag{2.13}$$
$$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi \tag{2.14}$$
$$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \chi \tag{2.15}$$
$$\Gamma \vdash \psi \to \varphi \vee (\psi \vee \chi) \tag{2.16}$$
$$\Gamma \vdash \chi \to \varphi \vee (\psi \vee \chi) \tag{2.17}$$
$$\Gamma \vdash \varphi \to (\varphi \vee \psi) \vee \chi \tag{2.18}$$
$$\Gamma \vdash \psi \to (\varphi \vee \psi) \vee \chi \tag{2.19}$$

Proof. (2.3):

(1)	$\Gamma \vdash \psi \to \psi \vee \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \psi \vee \varphi \to \varphi \vee \psi$	(PERMUTATION)
(3)	$\Gamma \vdash \psi \to \varphi \vee \psi$	(SYLLOGISM): (1), (2)

(2.4):

(1)	$\Gamma \vdash \varphi \wedge \psi \to \psi \wedge \varphi$	(PERMUTATION)
(2)	$\Gamma \vdash \psi \wedge \varphi \to \psi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \wedge \psi \to \psi$	(SYLLOGISM): (1), (2)

(2.5):

(1)	$\Gamma \vdash \varphi \wedge \psi \to \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \to (\psi \to \varphi)$	(EXPORTATION): (1)

(2.6):

(1)	$\Gamma \vdash \varphi \wedge \psi \to \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \to \varphi \vee \gamma$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \wedge \psi \to \varphi \vee \gamma$	(SYLLOGISM): (1), (2)

(2.7):

(1)	$\Gamma \vdash \varphi \to \varphi \wedge \varphi$	(CONTRACTION)
(2)	$\Gamma \vdash \varphi \wedge \varphi \to \varphi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \to \varphi$	(SYLLOGISM): (1), (2)

(2.8):

(1)	$\Gamma \vdash \varphi \wedge \psi \to \varphi \wedge \psi$	(2.7)
(2)	$\Gamma \vdash \varphi \to (\psi \to \varphi \wedge \psi)$	(EXPORTATION): (1)

(2.9):

(1)	$\Gamma \vdash (\varphi \to \psi) \to (\varphi \to \psi)$	(2.7)
(2)	$\Gamma \vdash (\varphi \to \psi) \wedge \varphi \to \psi$	(IMPORTATION): (1)

(2.10):

(1)	$\Gamma \vdash (\varphi \to \psi) \wedge \varphi \to \psi$	(2.9)
(2)	$\Gamma \vdash \varphi \wedge (\varphi \to \psi) \to (\varphi \to \psi) \wedge \varphi$	(PERMUTATION)
(3)	$\Gamma \vdash \varphi \wedge (\varphi \to \psi) \to \psi$	(SYLLOGISM): (1), (2)

(2.11):

(1)	$\Gamma \vdash \varphi \wedge (\varphi \to \psi) \to \psi$	(2.10)
(2)	$\Gamma \vdash \varphi \to ((\varphi \to \psi) \to \psi)$	(EXPORTATION): (1)

(2.12):

(1)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \wedge \psi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \wedge \psi \to \varphi$	(WEAKENING)
(3)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi$	(SYLLOGISM): (1), (2)

(2.13):

(1)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \wedge \psi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \wedge \psi \to \psi$	(2.4)
(3)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \psi$	(SYLLOGISM): (1), (2)

(2.14):

(1)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi \wedge \chi$	(2.4)
(2)	$\Gamma \vdash \psi \wedge \chi \to \psi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi$	(SYLLOGISM): (1), (2)

(2.15):

(1)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi \wedge \chi$	(2.4)
(2)	$\Gamma \vdash \psi \wedge \chi \to \chi$	(2.4)
(3)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi$	(SYLLOGISM): (1), (2)

(2.16):

(1)	$\Gamma \vdash \psi \to \psi \vee \chi$	(WEAKENING)
(2)	$\Gamma \vdash \psi \vee \chi \to \varphi \vee (\psi \vee \chi)$	(2.3)
(3)	$\Gamma \vdash \psi \to \varphi \vee (\psi \vee \chi)$	(SYLLOGISM): (1), (2)

(2.17):

(1)	$\Gamma \vdash \chi \to \psi \vee \chi$	(2.3)
(2)	$\Gamma \vdash \psi \vee \chi \to \varphi \vee (\psi \vee \chi)$	(2.3)
(3)	$\Gamma \vdash \chi \to \varphi \vee (\psi \vee \chi)$	(SYLLOGISM): (1), (2)

(2.18):

(1)	$\Gamma \vdash \varphi \to \varphi \vee \psi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \vee \psi \to (\varphi \vee \psi) \vee \chi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \to (\varphi \vee \psi) \vee \chi$	(SYLLOGISM): (1), (2)

(2.19):

(1)	$\Gamma \vdash \psi \to \varphi \vee \psi$	(2.3)
(2)	$\Gamma \vdash \varphi \vee \psi \to (\varphi \vee \psi) \vee \chi$	(WEAKENING)
(3)	$\Gamma \vdash \psi \to (\varphi \vee \psi) \vee \chi$	(SYLLOGISM): (1), (2)

□

Lemma 2.3.2.

$$\Gamma \vdash \varphi \textit{ and } \Gamma \vdash \psi \quad \textit{iff} \quad \Gamma \vdash \varphi \wedge \psi \tag{2.20}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \varphi \to \chi \quad \textit{implies} \quad \Gamma \vdash \varphi \to \psi \wedge \chi \tag{2.21}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \psi \to \varphi \quad \textit{implies} \quad \Gamma \vdash \varphi \leftrightarrow \psi \tag{2.22}$$

$$\Gamma \vdash \varphi \quad \textit{implies} \quad \Gamma \vdash \psi \to \varphi \tag{2.23}$$

Proof. (2.20):

"$\Rightarrow$"

(1)	$\Gamma \vdash \varphi$	(Assumption)
(2)	$\Gamma \vdash \psi$	(Assumption)
(3)	$\Gamma \vdash \varphi \to (\psi \to \varphi \wedge \psi)$	(2.8)
(4)	$\Gamma \vdash \psi \to \varphi \wedge \psi$	(MODUS PONENS): (1), (3)
(5)	$\Gamma \vdash \varphi \wedge \psi$	(MODUS PONENS): (2), (4)

"$\Leftarrow$"

(1)	$\Gamma \vdash \varphi \wedge \psi$	(Assumption)
(2)	$\Gamma \vdash \varphi \wedge \psi \to \varphi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \wedge \psi \to \psi$	(2.4)
(4)	$\Gamma \vdash \varphi$	(MODUS-PONENS): (1), (2)
(5)	$\Gamma \vdash \psi$	(MODUS-PONENS): (1), (3)

(2.21):

(1)	$\Gamma \vdash \psi \to (\chi \to \psi \wedge \chi)$	(2.8)
(2)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(3)	$\Gamma \vdash \varphi \to (\chi \to \psi \wedge \chi)$	(SYLLOGISM): (2), (1)
(4)	$\Gamma \vdash \varphi \wedge \chi \to \psi \wedge \chi$	(IMPORTATION): (3)
(5)	$\Gamma \vdash \chi \wedge \varphi \to \varphi \wedge \chi$	(PERMUTATION)
(6)	$\Gamma \vdash \chi \wedge \varphi \to \psi \wedge \chi$	(SYLLOGISM): (5), (4)
(7)	$\Gamma \vdash \varphi \to \chi$	(Assumption)
(8)	$\Gamma \vdash \chi \to (\varphi \to \psi \wedge \chi)$	(EXPORTATION): (6)
(9)	$\Gamma \vdash \varphi \to (\varphi \to \psi \wedge \chi)$	(SYLLOGISM): (7), (8)
(10)	$\Gamma \vdash \varphi \to \varphi \wedge \varphi$	(CONTRACTION)
(11)	$\Gamma \vdash \varphi \wedge \varphi \to \psi \wedge \chi$	(IMPORTATION): (9)
(12)	$\Gamma \vdash \varphi \to \psi \wedge \chi$	(SYLLOGISM): (10), (11)

(2.22): By (2.20) and the definition of $\leftrightarrow$.

(2.23): Immediate from (2.5) by (MODUS-PONENS). $\square$

Lemma 2.3.3.

$$\Gamma \vdash \varphi \leftrightarrow \varphi \wedge \varphi \tag{2.24}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi \vee \varphi \tag{2.25}$$

Proof. (2.24):

(1)	$\Gamma \vdash \varphi \to \varphi \wedge \varphi$	(CONTRACTION)
(2)	$\Gamma \vdash \varphi \wedge \varphi \to \varphi$	(WEAKENING)
(3)	$\Gamma \vdash \varphi \leftrightarrow \varphi \wedge \varphi$	(2.20): (1), (2)

(2.25):

(1)	$\Gamma \vdash \varphi \to \varphi \vee \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \vee \varphi \to \varphi$	(CONTRACTION)
(3)	$\Gamma \vdash \varphi \leftrightarrow \varphi \vee \varphi$	(2.20): (1), (2)

□

Lemma 2.3.4.

$$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \wedge (\psi \wedge \chi) \tag{2.26}$$

$$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to (\varphi \wedge \psi) \wedge \chi \tag{2.27}$$

$$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \leftrightarrow (\varphi \wedge \psi) \wedge \chi \tag{2.28}$$

Proof. (2.26):

(1)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi$	(2.12)
(2)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \psi$	(2.13)
(3)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \chi$	(2.4)
(4)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \psi \wedge \chi$	(2.21): (2), (3)
(5)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \wedge (\psi \wedge \chi)$	(2.21): (1), (4)

(2.27):

(1)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \psi$	(2.14)
(3)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \chi$	(2.15)
(4)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \varphi \wedge \psi$	(2.21): (1), (2)
(5)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to (\varphi \wedge \psi) \wedge \chi$	(2.21): (3), (4)

(2.28): Apply (2.26), (2.27), (2.20), and the definition of $\leftrightarrow$. □

Lemma 2.3.5.

$$\Gamma \vdash \varphi \to \psi \quad \textit{implies} \quad \Gamma \vdash \varphi \wedge \chi \to \psi \textit{ and } \Gamma \vdash \chi \wedge \varphi \to \psi \tag{2.29}$$

$$\Gamma \vdash \varphi \to \psi \wedge \chi \quad \textit{implies} \quad \Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \varphi \to \chi \tag{2.30}$$

$$\Gamma \vdash \varphi \wedge \psi \to \chi \quad \textit{implies} \quad \Gamma \vdash \psi \wedge \varphi \to \chi \tag{2.31}$$

$$\Gamma \vdash \varphi \to (\psi \to \chi) \quad \textit{implies} \quad \Gamma \vdash \psi \wedge \varphi \to \chi \tag{2.32}$$

$$\Gamma \vdash \varphi \to \psi \quad \textit{implies} \quad \Gamma \vdash \varphi \to \varphi \wedge \psi \textit{ and } \Gamma \vdash \varphi \to \psi \wedge \varphi \tag{2.33}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \chi \to \gamma \quad \textit{implies} \quad \Gamma \vdash \varphi \wedge \chi \to \psi \wedge \gamma \tag{2.34}$$

Proof. (2.29):

(1) $\Gamma \vdash \varphi \to \psi$ (Assumption)

(2) $\Gamma \vdash \varphi \wedge \chi \to \varphi$ (WEAKENING)

(3) $\Gamma \vdash \varphi \wedge \chi \to \psi$ (SYLLOGISM): (2), (1)

and

(1) $\Gamma \vdash \varphi \to \psi$ (Assumption)

(2) $\Gamma \vdash \chi \wedge \varphi \to \varphi$ (2.4)

(3) $\Gamma \vdash \chi \wedge \varphi \to \psi$ (SYLLOGISM): (2), (1)

(2.30):

(1) $\Gamma \vdash \varphi \to \psi \wedge \chi$ (Assumption)

(2) $\Gamma \vdash \psi \wedge \chi \to \psi$ (WEAKENING)

(3) $\Gamma \vdash \varphi \to \psi$ (SYLLOGISM): (1), (2)

and

(1) $\Gamma \vdash \varphi \to \psi \wedge \chi$ (Assumption)

(2) $\Gamma \vdash \psi \wedge \chi \to \chi$ (2.4)

(3) $\Gamma \vdash \varphi \to \chi$ (SYLLOGISM): (1), (2)

(2.31):

(1) $\Gamma \vdash \varphi \wedge \psi \to \chi$ (Assumption)

(2) $\Gamma \vdash \psi \wedge \varphi \to \varphi \wedge \psi$ (PERMUTATION)

(3) $\Gamma \vdash \psi \wedge \varphi \to \chi$ (SYLLOGISM): (1), (2)

(2.32):

(1) $\Gamma \vdash \varphi \to (\psi \to \chi)$ (Assumption)

(2) $\Gamma \vdash \varphi \wedge \psi \to \chi$ (IMPORTATION): (1)

(3) $\Gamma \vdash \psi \wedge \varphi \to \chi$ (2.31): (2)

(2.33):

(1) $\Gamma \vdash \varphi \to \varphi$ (2.7)

(2) $\Gamma \vdash \varphi \to \psi,$ (Assumption)

(3) $\Gamma \vdash \varphi \to \varphi \wedge \psi$ (2.21): (1), (2)

and

(1) $\Gamma \vdash \varphi \to \psi,$ (Assumption)

(2) $\Gamma \vdash \varphi \to \varphi$ (2.7)

(3) $\Gamma \vdash \varphi \to \psi \wedge \varphi$ (2.21): (1), (2)

(2.34):

(1) $\Gamma \vdash \varphi \to \psi$ (Assumption)

(2) $\Gamma \vdash \chi \to \gamma$ (Assumption)

(3) $\Gamma \vdash \varphi \wedge \chi \to \psi$ (2.29): (1) □

(4) $\Gamma \vdash \varphi \wedge \chi \to \gamma$ (2.29): (2)

(5) $\Gamma \vdash \varphi \wedge \chi \to \psi \wedge \gamma$ (2.20): (3), (4)

Lemma 2.3.6.

$$\Gamma \vdash \varphi \wedge (\varphi \vee \psi) \to \varphi \tag{2.35}$$

$$\Gamma \vdash \varphi \to \varphi \wedge (\varphi \vee \psi) \tag{2.36}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi \wedge (\varphi \vee \psi) \tag{2.37}$$

$$\Gamma \vdash \varphi \vee (\varphi \wedge \psi) \to \varphi \tag{2.38}$$

$$\Gamma \vdash \varphi \to \varphi \vee (\varphi \wedge \psi) \tag{2.39}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi \vee (\varphi \wedge \psi) \tag{2.40}$$

Proof. (2.35): It is obvious, by (WEAKENING).

(2.36):

(1) $\Gamma \vdash \varphi \to \varphi$ (2.7)

(2) $\Gamma \vdash \varphi \to \varphi \vee \psi$ (WEAKENING)

(3) $\Gamma \vdash \varphi \to \varphi \wedge (\varphi \vee \psi)$ (2.21): (1), (2)

(2.37): By (2.35), (2.36), (2.20), and the definition of $\leftrightarrow$.

(2.38):

(1)	$\Gamma \vdash \varphi \wedge \psi \to \varphi$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \vee (\varphi \wedge \psi) \to \varphi \vee \varphi$	(EXPANSION): (1)
(3)	$\Gamma \vdash \varphi \vee \varphi \to \varphi$	(WEAKENING)
(4)	$\Gamma \vdash \varphi \vee (\varphi \wedge \psi) \to \varphi$	(SYLLOGISM): (2), (3)

(2.39): It is obvious, by (WEAKENING).

(2.40): By (2.38), (2.39), (2.20), and the definition of $\leftrightarrow$. $\square$

Lemma 2.3.7.

$$\Gamma \vdash \varphi \to (\psi \to \chi) \quad \textit{implies} \quad \Gamma \vdash \psi \to (\varphi \to \chi) \tag{2.41}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \varphi \to (\psi \to \chi) \quad \textit{implies} \quad \Gamma \vdash \varphi \to \chi \tag{2.42}$$

$$\Gamma \vdash \varphi \to \psi \quad \textit{implies} \quad \Gamma \vdash (\psi \to \chi) \to (\varphi \to \chi) \tag{2.43}$$

$$\Gamma \vdash \varphi \to (\psi \to \chi) \textit{ and } \Gamma \vdash \chi \to \gamma \quad \textit{implies} \quad \Gamma \vdash \varphi \to (\psi \to \gamma) \tag{2.44}$$

Proof. (2.41):

(1)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	(Assumption)
(2)	$\Gamma \vdash \psi \wedge \varphi \to \chi$	(2.32): (1)
(3)	$\Gamma \vdash \psi \to (\varphi \to \chi)$	(EXPORTATION): (2)

(2.42):

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \varphi \to \varphi \wedge \psi$	(2.33): (1)
(3)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	(Assumption)
(4)	$\Gamma \vdash \varphi \wedge \psi \to \chi$	(IMPORTATION): (3)
(5)	$\Gamma \vdash \varphi \to \chi$	(SYLLOGISM): (2), (4)

(2.43):

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash (\psi \to \chi) \to (\psi \to \chi)$	(2.7)
(3)	$\Gamma \vdash \varphi \wedge (\psi \to \chi) \to \psi \wedge (\psi \to \chi)$	(2.34): (1), (2)
(4)	$\Gamma \vdash \psi \wedge (\psi \to \chi) \to \chi$	(2.10)
(5)	$\Gamma \vdash \varphi \wedge (\psi \to \chi) \to \chi$	(SYLLOGISM): (3), (4)
(6)	$\Gamma \vdash (\psi \to \chi) \wedge \varphi \to \chi$	(2.31)
(7)	$\Gamma \vdash (\psi \to \chi) \to (\varphi \to \chi)$	(EXPORTATION): (6)

(2.44):

(1)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	(Assumption)
(2)	$\Gamma \vdash \chi \to \gamma$	(Assumption)
(3)	$\Gamma \vdash \varphi \wedge \psi \to \chi$	(IMPORTATION): (1)
(4)	$\Gamma \vdash \varphi \wedge \psi \to \gamma$	(SYLLOGISM): (3), (2)
(5)	$\Gamma \vdash \varphi \to (\psi \to \gamma)$	(EXPORTATION): (4)

□

Lemma 2.3.8.

$$\Gamma \vdash \varphi \to \psi \quad \textit{implies} \quad \Gamma \vdash \varphi \vee \chi \to \psi \vee \chi \tag{2.45}$$

$$\Gamma \vdash \varphi \to \psi \quad \textit{implies} \quad \Gamma \vdash \varphi \vee \psi \to \psi \textit{ and } \Gamma \vdash \psi \vee \varphi \to \psi \tag{2.46}$$

$$\Gamma \vdash \varphi \to \chi \textit{ and } \Gamma \vdash \psi \to \chi \quad \textit{implies} \quad \Gamma \vdash \varphi \vee \psi \to \chi \tag{2.47}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \chi \to \gamma \quad \textit{implies} \quad \Gamma \vdash \varphi \vee \chi \to \psi \vee \gamma \tag{2.48}$$

Proof. (2.45):

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \chi \vee \varphi \to \chi \vee \psi$	(EXPANSION): (1)
(3)	$\Gamma \vdash \varphi \vee \chi \to \chi \vee \varphi$	(PERMUTATION)
(4)	$\Gamma \vdash \varphi \vee \chi \to \chi \vee \psi$	(SYLLOGISM): (3), (2)
(5)	$\Gamma \vdash \chi \vee \psi \to \psi \vee \chi$	(PERMUTATION)
(6)	$\Gamma \vdash \varphi \vee \chi \to \psi \vee \chi$	(SYLLOGISM): (4), (5)

(2.46):

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \varphi \vee \psi \to \psi \vee \psi$	(2.45): (1)
(3)	$\Gamma \vdash \psi \vee \psi \to \psi$	(CONTRACTION)
(4)	$\Gamma \vdash \varphi \vee \psi \to \psi$	(SYLLOGISM): (2), (3)

and

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \psi \vee \varphi \to \psi \vee \psi$	(EXPANSION): (1)
(3)	$\Gamma \vdash \psi \vee \psi \to \psi$	(CONTRACTION)
(4)	$\Gamma \vdash \psi \vee \varphi \to \psi$	(SYLLOGISM): (2), (3)

(2.47):

(1)	$\Gamma \vdash \varphi \to \chi$	(Assumption)
(2)	$\Gamma \vdash \varphi \vee \chi \to \chi$	(2.46): (1)
(3)	$\Gamma \vdash \psi \to \chi$	(Assumption)
(4)	$\Gamma \vdash \varphi \vee \psi \to \varphi \vee \chi$	(EXPANSION): (3)
(5)	$\Gamma \vdash \varphi \vee \psi \to \chi$	(SYLLOGISM): (4), (2)

(2.48):

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \chi \to \gamma$	(Assumption)
(3)	$\Gamma \vdash \psi \to \psi \vee \gamma$	(WEAKENING)
(4)	$\Gamma \vdash \varphi \to \psi \vee \gamma$	(SYLLOGISM): (1), (3)
(5)	$\Gamma \vdash \gamma \to \psi \vee \gamma$	(2.3)
(6)	$\Gamma \vdash \chi \to \psi \vee \gamma$	(SYLLOGISM): (2), (5)
(7)	$\Gamma \vdash \varphi \vee \chi \to \psi \vee \gamma$	(2.47): (4), (6)

□

Lemma 2.3.9.

$$\Gamma \vdash (\varphi \vee \psi) \vee \chi \to \varphi \vee (\psi \vee \chi) \tag{2.49}$$
$$\Gamma \vdash \varphi \vee (\psi \vee \chi) \to (\varphi \vee \psi) \vee \chi \tag{2.50}$$
$$\Gamma \vdash \varphi \vee (\psi \vee \chi) \leftrightarrow (\varphi \vee \psi) \vee \chi \tag{2.51}$$

Proof. (2.49):

(1)	$\Gamma \vdash \varphi \to \varphi \vee (\psi \vee \chi)$	(WEAKENING)
(2)	$\Gamma \vdash \psi \to \varphi \vee (\psi \vee \chi)$	(2.16)
(3)	$\Gamma \vdash \chi \to \varphi \vee (\psi \vee \chi)$	(2.17)
(4)	$\Gamma \vdash \varphi \vee \psi \to \varphi \vee (\psi \vee \chi)$	(2.47): (1), (2)
(5)	$\Gamma \vdash (\varphi \vee \psi) \vee \chi \to \varphi \vee (\psi \vee \chi)$	(2.47): (4), (3)

(2.50):

(1)	$\Gamma \vdash \chi \to (\varphi \vee \psi) \vee \chi$	(2.3)
(2)	$\Gamma \vdash \varphi \to (\varphi \vee \psi) \vee \chi$	(2.18)
(3)	$\Gamma \vdash \psi \to (\varphi \vee \psi) \vee \chi$	(2.19)
(4)	$\Gamma \vdash \psi \vee \chi \to (\varphi \vee \psi) \vee \chi$	(2.47): (3), (1)
(5)	$\Gamma \vdash \varphi \vee (\psi \vee \chi) \to (\varphi \vee \psi) \vee \chi$	(2.47): (2), (4)

(2.51): Apply (2.49), (2.50), (2.20), and the definition of $\leftrightarrow$. □

Lemma 2.3.10.

$$\Gamma \vdash (\varphi \to \psi) \to ((\psi \to \chi) \to (\varphi \to \chi)) \tag{2.52}$$
$$\Gamma \vdash (\varphi \wedge \psi \to \chi) \to (\varphi \to (\psi \to \chi)) \tag{2.53}$$
$$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to (\varphi \wedge \psi \to \chi) \tag{2.54}$$
$$\Gamma \vdash (\varphi \to (\psi \to \chi)) \leftrightarrow (\varphi \wedge \psi \to \chi) \tag{2.55}$$
$$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to (\psi \to (\varphi \to \chi)) \tag{2.56}$$

Proof. (2.52): Let $\gamma := ((\varphi \to \psi) \wedge (\psi \to \chi)) \wedge \varphi$.

(1)	$\Gamma \vdash \gamma \to (\varphi \to \psi)$	(2.12)
(2)	$\Gamma \vdash \gamma \to \varphi$	(2.4)
(3)	$\Gamma \vdash \gamma \to \psi$	(2.42): (2), (1)
(4)	$\Gamma \vdash \gamma \to (\psi \to \chi)$	(2.13)
(5)	$\Gamma \vdash \gamma \to \chi$	(2.42): (3), (4)
(6)	$\Gamma \vdash ((\varphi \to \psi) \wedge (\psi \to \chi)) \to (\varphi \to \chi)$	(EXPORTATION): (5)
(7)	$\Gamma \vdash (\varphi \to \psi) \to ((\psi \to \chi) \to (\varphi \to \chi))$	(EXPORTATION): (6)

(2.53): Let $\gamma := ((\varphi \wedge \psi \to \chi) \wedge \varphi) \wedge \psi$.

(1)	$\Gamma \vdash \gamma \to (\varphi \wedge \psi \to \chi)$	(2.12)
(2)	$\Gamma \vdash \gamma \to \varphi$	(2.13)
(3)	$\Gamma \vdash \gamma \to \psi$	(2.4)
(4)	$\Gamma \vdash \gamma \to \varphi \wedge \psi$	(2.20): (2), (3)
(5)	$\Gamma \vdash \gamma \to \chi$	(2.42): (4), (1)
(6)	$\Gamma \vdash ((\varphi \wedge \psi \to \chi) \wedge \varphi) \to (\psi \to \chi)$	(EXPORTATION): (5)
(7)	$\Gamma \vdash (\varphi \wedge \psi \to \chi) \to (\varphi \to (\psi \to \chi))$	(EXPORTATION): (6)

(2.54): Let $\gamma := (\varphi \to (\psi \to \chi)) \wedge (\varphi \wedge \psi)$.

(1)	$\Gamma \vdash \gamma \to (\varphi \to (\psi \to \chi))$	(WEAKENING)
(2)	$\Gamma \vdash \gamma \to \varphi$	(2.14)
(3)	$\Gamma \vdash \gamma \to (\psi \to \chi)$	(2.42): (2), (3)
(4)	$\Gamma \vdash \gamma \to \psi$	(2.15)
(5)	$\Gamma \vdash \gamma \to \chi$	(2.42): (3), (4)
(6)	$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to (\varphi \wedge \psi \to \chi)$	(EXPORTATION): (5)

(2.55): Apply (2.53), (2.54), (2.20) and the definition of $\leftrightarrow$.

(2.56): Let $\gamma := ((\varphi \to (\psi \to \chi)) \wedge \psi) \wedge \varphi$.

(1)	$\Gamma \vdash \gamma \to (\varphi \to (\psi \to \chi))$	(2.12)
(2)	$\Gamma \vdash \gamma \to \varphi$	(2.4)
(3)	$\Gamma \vdash \gamma \to (\psi \to \chi)$	(2.42): (1), (2)
(4)	$\Gamma \vdash \gamma \to \psi$	(2.13)
(5)	$\Gamma \vdash \gamma \to \chi$	(2.42): (3), (4)
(6)	$\Gamma \vdash ((\varphi \to (\psi \to \chi)) \wedge \psi) \to (\varphi \to \chi)$	(EXPORTATION): (5)
(7)	$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to (\psi \to (\varphi \to \chi))$	(EXPORTATION): (6)

□

Lemma 2.3.11.

$$\Gamma \vdash \big((\varphi \to (\psi \to \chi)) \wedge (\varphi \to \psi)\big) \wedge \varphi \to \chi \tag{2.57}$$
$$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \tag{2.58}$$
$$\Gamma \vdash (\varphi \to \psi) \wedge (\varphi \to \chi) \to (\varphi \to \psi \wedge \chi) \tag{2.59}$$
$$\Gamma \vdash (\varphi \to \psi) \to ((\varphi \to \chi) \to (\varphi \to \psi \wedge \chi)) \tag{2.60}$$
$$\Gamma \vdash (\varphi \to \chi) \wedge (\psi \to \chi) \to (\varphi \vee \psi \to \chi) \tag{2.61}$$
$$\Gamma \vdash (\varphi \to \chi) \to ((\psi \to \chi) \to (\varphi \vee \psi \to \chi)) \tag{2.62}$$
$$\Gamma \vdash (\varphi \wedge (\varphi \to \psi) \to \chi) \to (\varphi \wedge \psi \to \chi) \tag{2.63}$$
$$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \to (\varphi \to (\psi \to \chi)) \tag{2.64}$$
$$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \leftrightarrow (\varphi \to (\psi \to \chi)) \tag{2.65}$$

Proof. Let us denote $\pi := \big((\varphi \to (\psi \to \chi)) \wedge (\varphi \to \psi)\big) \wedge \varphi$.

(2.57):

(1)	$\Gamma \vdash \pi \to \varphi$	(2.4)
(2)	$\Gamma \vdash \pi \to (\varphi \to \psi)$	(2.13)
(3)	$\Gamma \vdash \pi \to (\varphi \to (\psi \to \chi))$	(2.12)
(4)	$\Gamma \vdash \pi \to \psi$	(2.42): (1), (2)
(5)	$\Gamma \vdash \pi \to (\psi \to \chi)$	(2.42): (1), (3)
(6)	$\Gamma \vdash \pi \to \chi$	(2.42): (4), (5)

(2.58):

(1)	$\Gamma \vdash \pi \to \chi$	(2.57)
(2)	$\Gamma \vdash (\varphi \to (\psi \to \chi)) \wedge (\varphi \to \psi)\big) \to (\varphi \to \chi)$	(EXPORTATION): (1)
(3)	$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$	(EXPORTATION): (2)

(2.59): Let $\gamma := (\varphi \to \psi) \wedge (\varphi \to \chi) \wedge \varphi$.

(1)	$\Gamma \vdash \gamma \to \varphi$	(2.4)
(2)	$\Gamma \vdash \gamma \to (\varphi \to \psi)$	(2.12)
(3)	$\Gamma \vdash \gamma \to (\varphi \to \chi)$	(2.13)
(4)	$\Gamma \vdash \gamma \to \psi$	(2.42): (1), (2)
(5)	$\Gamma \vdash \gamma \to \chi$	(2.42): (1), (3)
(6)	$\Gamma \vdash \gamma \to \psi \wedge \chi$	(2.21): (4), (5)
(7)	$\Gamma \vdash (\varphi \to \psi) \wedge (\varphi \to \chi) \to (\varphi \to \psi \wedge \chi)$	(EXPORTATION): (6)

(2.60): Apply (EXPORTATION) to (2.59).

(2.61): Let $\gamma := (\varphi \to \chi) \wedge (\psi \to \chi)$.

(1)	$\Gamma \vdash \gamma \to (\varphi \to \chi)$	(WEAKENING)
(2)	$\Gamma \vdash \varphi \to (\gamma \to \chi)$	(2.41): (1)
(3)	$\Gamma \vdash \gamma \to (\psi \to \chi)$	(2.4)
(4)	$\Gamma \vdash \psi \to (\gamma \to \chi)$	(2.41): (3)
(5)	$\Gamma \vdash \varphi \vee \psi \to (\gamma \to \chi)$	(2.47): (2), (4)
(6)	$\Gamma \vdash \gamma \to (\varphi \vee \psi \to \chi)$	(2.41): (5)

(2.63):

(1)	$\Gamma \vdash \varphi \to \varphi$	(2.7)
(2)	$\Gamma \vdash \psi \to (\varphi \to \psi)$	(2.5)
(3)	$\Gamma \vdash \varphi \wedge \psi \to \varphi \wedge (\varphi \to \psi)$	(2.34): (1), (2)
(4)	$\Gamma \vdash (\varphi \wedge (\varphi \to \psi) \to \chi) \to (\varphi \wedge \psi \to \chi)$	(2.43): (3)

(2.64):

(1)	$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \to (\varphi \to ((\varphi \to \psi) \to \chi))$	(2.56)
(2)	$\Gamma \vdash (\varphi \to ((\varphi \to \psi) \to \chi)) \to (\varphi \wedge (\varphi \to \psi) \to \chi)$	(2.54)
(3)	$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \to (\varphi \wedge (\varphi \to \psi) \to \chi)$	(SYLLOGISM): (1), (2)
(4)	$\Gamma \vdash (\varphi \wedge (\varphi \to \psi) \to \chi) \to (\varphi \wedge \psi \to \chi)$	(2.63)
(5)	$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \to (\varphi \wedge \psi \to \chi)$	(SYLLOGISM): (3), (4)
(6)	$\Gamma \vdash (\varphi \wedge \psi \to \chi) \to (\varphi \to (\psi \to \chi))$	(2.53)
(7)	$\Gamma \vdash ((\varphi \to \psi) \to (\varphi \to \chi)) \to (\varphi \to (\psi \to \chi))$	(SYLLOGISM): (5), (6)

(2.65): Apply (2.58), (2.64), (2.20) and the definition of $\leftrightarrow$. $\square$

Lemma 2.3.12.

$$\Gamma \vdash (\varphi \to \psi) \to (\varphi \to \chi) \quad \textit{implies} \quad \Gamma \vdash \varphi \to (\psi \to \chi) \tag{2.66}$$

$$\Gamma \vdash \varphi \to (\psi \to \chi) \textit{ and } \Gamma \vdash \varphi \to (\chi \to \gamma) \quad \textit{implies} \quad \Gamma \vdash \varphi \to (\psi \to \gamma) \tag{2.67}$$

$$\Gamma \vdash \varphi \to \psi \textit{ and } \Gamma \vdash \chi \to \gamma \quad \textit{implies} \quad \Gamma \vdash (\varphi \to \chi) \to (\psi \to \gamma) \tag{2.68}$$

$$\Gamma \vdash \varphi \to (\psi \land \chi \to \gamma) \quad \textit{iff} \quad \Gamma \vdash \varphi \to (\psi \to (\chi \to \gamma)) \tag{2.69}$$

$$\Gamma \vdash \varphi \land \chi \to \psi \textit{ and } \Gamma \vdash \varphi \to \psi \lor \chi \quad \textit{implies} \quad \Gamma \vdash \varphi \to \psi \tag{2.70}$$

Proof. (2.66):

(1)	$\Gamma \vdash (\varphi \to \psi) \to (\varphi \to \chi)$	(Assumption)
(2)	$\Gamma \vdash (\varphi \to \psi) \land \varphi \to \chi$	(IMPORTATION)
(3)	$\Gamma \vdash \varphi \land (\varphi \to \psi) \to \chi$	(2.31): (2)
(4)	$\Gamma \vdash (\varphi \land (\varphi \to \psi) \to \chi) \to (\varphi \land \psi \to \chi)$	(2.63)
(5)	$\Gamma \vdash \varphi \land \psi \to \chi$	(MODUS-PONENS): (3), (4)
(6)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	(EXPORTATION): (5)

(2.67):

(1)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	(Assumption)
(2)	$\Gamma \vdash (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$	(2.58)
(3)	$\Gamma \vdash (\varphi \to \psi) \to (\varphi \to \chi)$	(MODUS-PONENS): (1), (2)
(4)	$\Gamma \vdash \varphi \to (\chi \to \gamma)$	(Assumption)
(5)	$\Gamma \vdash (\varphi \to (\chi \to \gamma)) \to ((\varphi \to \chi) \to (\varphi \to \gamma))$	(2.58)
(6)	$\Gamma \vdash (\varphi \to \chi) \to (\varphi \to \gamma)$	(MODUS-PONENS): (4), (5)
(7)	$\Gamma \vdash (\varphi \to \psi) \to (\varphi \to \gamma)$	(SYLLOGISM): (3), (6)
(8)	$\Gamma \vdash \varphi \to (\psi \to \gamma)$	(2.66): (7)

(2.68):

(1)	$\Gamma \vdash \psi \to \varphi$	(Assumption)
(2)	$\Gamma \vdash ((\varphi \to \chi) \land \psi) \to (\psi \to \varphi)$	(2.23): (1)
(3)	$\Gamma \vdash ((\varphi \to \chi) \land \psi) \to (\varphi \to \chi)$	(WEAKENING-CONJ)
(4)	$\Gamma \vdash ((\varphi \to \chi) \land \psi) \to (\psi \to \chi)$	(2.67): (2), (3)
(5)	$\Gamma \vdash ((\varphi \to \chi) \land \psi) \to \psi$	(2.4)
(6)	$\Gamma \vdash ((\varphi \to \chi) \land \psi) \to \chi$	(2.42): (4), (5)
(7)	$\Gamma \vdash \chi \leftrightarrow \gamma$	(Assumption)
(8)	$\Gamma \vdash (\chi \leftrightarrow \gamma) \to (\chi \to \gamma)$	(WEAKENING-CONJ)

(9)	$\Gamma \vdash \chi \to \gamma$	(MODUS-PONENS): (7), (8)
(10)	$\Gamma \vdash ((\varphi \to \chi) \wedge \psi) \to \chi \to \gamma$	(2.23): (9)
(11)	$\Gamma \vdash ((\varphi \to \chi) \wedge \psi) \to \gamma$	(2.42): (6), (10)
(12)	$\Gamma \vdash (\varphi \to \chi) \to (\psi \to \gamma)$	(EXPORTATION): (11)

(2.69): " $\Rightarrow$ "

(1)	$\Gamma \vdash \varphi \to (\psi \wedge \chi \to \gamma)$	(Assumption)
(2)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \gamma$	(IMPORTATION): (1)
(3)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \varphi \wedge (\psi \wedge \chi)$	(2.26)
(4)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \gamma$	(SYLLOGISM): (3), (2)
(5)	$\Gamma \vdash (\varphi \wedge \psi) \to (\chi \to \gamma)$	(EXPORTATION): (4)
(6)	$\Gamma \vdash \varphi \to (\psi \to (\chi \to \gamma))$	(EXPORTATION): (5)

" $\Leftarrow$ "

(1)	$\Gamma \vdash \varphi \to (\psi \to (\chi \to \gamma))$	(Assumption)
(2)	$\Gamma \vdash \varphi \wedge \psi \to (\chi \to \gamma)$	(IMPORTATION): (1)
(3)	$\Gamma \vdash (\varphi \wedge \psi) \wedge \chi \to \gamma$	(IMPORTATION): (2)
(4)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to (\varphi \wedge \psi) \wedge \chi$	(2.27)
(5)	$\Gamma \vdash \varphi \wedge (\psi \wedge \chi) \to \gamma$	(SYLLOGISM): (4), (3)
(6)	$\Gamma \vdash \varphi \to (\psi \wedge \chi \to \gamma)$	(EXPORTATION): (5)

(2.70):

(1)	$\Gamma \vdash \varphi \wedge \chi \to \psi$	(Assumption)
(2)	$\Gamma \vdash \varphi \to (\chi \to \psi)$	(EXPORTATION): (1)
(3)	$\Gamma \vdash \varphi \to \psi \vee \chi$	(Assumption)
(4)	$\Gamma \vdash \chi \to (\varphi \to \psi)$	(2.41): (2)
(5)	$\Gamma \vdash \psi \to (\varphi \to \psi)$	(2.5)
(6)	$\Gamma \vdash \psi \vee \chi \to (\varphi \to \psi)$	(2.47): (5), (4) $\square$
(7)	$\Gamma \vdash \varphi \to (\varphi \to \psi)$	SYLLOGISM: (2), (6)
(8)	$\Gamma \vdash \varphi \to \varphi$	(2.7)
(9)	$\Gamma \vdash \varphi \to \varphi \wedge (\varphi \to \psi)$	(2.20)
(10)	$\Gamma \vdash \varphi \wedge (\varphi \to \psi) \to \psi$	(2.10)
(11)	$\Gamma \vdash \varphi \to \psi$	SYLLOGISM: (9), (10)

Lemma 2.3.13.

$$\Gamma \vdash \varphi \to \neg\neg\varphi \tag{2.71}$$
$$\Gamma \vdash \neg\varphi \to \neg\neg\neg\varphi \tag{2.72}$$
$$\Gamma \vdash \varphi \wedge \neg\varphi \to \psi \tag{2.73}$$
$$\Gamma \vdash \psi \to \neg(\varphi \wedge \neg\varphi) \tag{2.74}$$
$$\Gamma \vdash \varphi \to (\neg\bot) \tag{2.75}$$
$$\Gamma \vdash \varphi \to (\neg\varphi \to \psi) \tag{2.76}$$
$$\Gamma \vdash \neg\varphi \to (\varphi \to \psi) \tag{2.77}$$
$$\Gamma \vdash (\varphi \to (\varphi \wedge \neg\varphi)) \to \neg\varphi \tag{2.78}$$
$$\Gamma \vdash (\varphi \to \psi) \to (\neg\psi \to \neg\varphi) \tag{2.79}$$
$$\Gamma \vdash (\neg\varphi \vee \psi) \to (\varphi \to \psi) \tag{2.80}$$

Proof. (2.71): Immediately, by (2.9).

(2.72): Immediately, by (2.71).

(2.73):

(1)	$\Gamma \vdash \varphi \wedge (\varphi \to \bot) \to \bot$	(2.10)
(2)	$\Gamma \vdash \bot \to \psi$	EXFALSO
(3)	$\Gamma \vdash \varphi \wedge (\varphi \to \bot) \to \psi$	SYLLOGISM: (1), (2)

(2.74):

(1)	$\Gamma \vdash \varphi \wedge (\varphi \to \bot) \to \bot$	(2.10)
(2)	$\Gamma \vdash (\varphi \wedge (\varphi \to \bot) \to \bot) \to (\psi \to (\varphi \wedge (\varphi \to \bot) \to \bot))$	(2.5)
(3)	$\Gamma \vdash \psi \to (\varphi \wedge (\varphi \to \bot) \to \bot)$	MODUS-PONENS: (1), (2)

(2.75):

(1)	$\Gamma \vdash \bot \to \bot$	EXFALSO
(2)	$\Gamma \vdash \varphi \to (\bot \to \bot)$	(2.23): (1)

(2.76): Immediately, by applying (2.53) to (2.73) and MODUS-PONENS.

(2.77): Immediately, by applying (2.41) to (2.76).

(2.78):

(1)	$\Gamma \vdash (\varphi \to (\varphi \wedge \neg\varphi)) \to (\varphi \to (\varphi \wedge \neg\varphi))$	(2.7)
(2)	$\Gamma \vdash \varphi \wedge \neg\varphi \to \neg\varphi$	(2.4)
(3)	$\Gamma \vdash (\varphi \to (\varphi \wedge \neg\varphi)) \to (\varphi \to \neg\varphi)$	(2.44): (1), (2)
(4)	$\Gamma \vdash (\varphi \to (\varphi \to \bot)) \to (\varphi \wedge \varphi \to \bot)$	(2.54)

(5)	$\Gamma \vdash \varphi \to \varphi \wedge \varphi$	(CONTRACTION-CONJ)
(6)	$\Gamma \vdash (\varphi \wedge \varphi \to \bot) \to (\varphi \to \bot)$	(2.43): (5)
(7)	$\Gamma \vdash (\varphi \to (\varphi \to \bot)) \to (\varphi \to \bot)$	(SYLLOGISM): (4), (6)
(8)	$\Gamma \vdash (\varphi \to (\varphi \wedge \neg\varphi)) \to \neg\varphi$	(SYLLOGISM): (3), (7)

(2.79):

(1)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to (\varphi \to \psi)$	(2.12)
(2)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to \neg\psi$	(2.14)
(3)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to \varphi$	(2.13)
(4)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to \psi$	(2.42): (1), (3)
(5)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to \psi \wedge \neg\psi$	(2.21): (2), (4)
(6)	$\Gamma \vdash \psi \wedge \neg\psi \to \bot$	(2.73)
(7)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \wedge \varphi \to \bot$	SYLLOGISM: (5), (6)
(8)	$\Gamma \vdash ((\varphi \to \psi) \wedge \neg\psi) \to (\varphi \to \bot)$	EXPORTATION: (7)
(9)	$\Gamma \vdash (\varphi \to \psi) \to (\neg\psi \to \neg\varphi)$	EXPORTATION: (8)

(2.80):

(1)	$\Gamma \vdash \neg\varphi \to (\varphi \to \psi)$	(2.77)
(2)	$\Gamma \vdash \psi \to (\varphi \to \psi)$	(2.5)
(3)	$\Gamma \vdash (\neg\varphi \vee \psi) \to (\varphi \to \psi)$	(2.47): (1), (2)

□

The following lemma groups theorems and derived rules which will be necessary in Section 3.4:

Lemma 2.3.14.

$$\Gamma \vdash \varphi \leftrightarrow \varphi \tag{2.81}$$

$$\Gamma \vdash \varphi \leftrightarrow \psi \quad \textit{implies} \quad \Gamma \vdash \psi \leftrightarrow \varphi \tag{2.82}$$

$$\Gamma \vdash \varphi \leftrightarrow \psi \textit{ and } \Gamma \vdash \psi \leftrightarrow \chi \quad \textit{implies} \quad \Gamma \vdash \varphi \leftrightarrow \chi \tag{2.83}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi' \textit{ and } \Gamma \vdash \psi \leftrightarrow \psi' \quad \textit{implies} \quad (\Gamma \vdash \varphi \to \psi \textit{ iff } \Gamma \vdash \varphi' \to \psi') \tag{2.84}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi' \textit{ and } \Gamma \vdash \psi \leftrightarrow \psi' \quad \textit{implies} \quad \Gamma \vdash \varphi \vee \psi \leftrightarrow \varphi' \vee \psi' \tag{2.85}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi' \textit{ and } \Gamma \vdash \psi \leftrightarrow \psi' \quad \textit{implies} \quad \Gamma \vdash \varphi \wedge \psi \leftrightarrow \varphi' \wedge \psi' \tag{2.86}$$

$$\Gamma \vdash \varphi \leftrightarrow \varphi' \textit{ and } \Gamma \vdash \psi \leftrightarrow \psi' \quad \textit{implies} \quad \Gamma \vdash (\varphi \to \psi) \leftrightarrow (\varphi' \to \psi') \tag{2.87}$$

$$\Gamma \vdash \varphi \quad \textit{iff} \quad \Gamma \vdash \varphi \leftrightarrow \neg(\psi \wedge \neg\psi) \tag{2.88}$$

$$\Gamma \vdash \neg\varphi \leftrightarrow (\varphi \to \varphi \wedge \neg\varphi) \tag{2.89}$$

$$\Gamma \vdash \varphi \quad \textit{iff} \quad \Gamma \vdash \varphi \leftrightarrow \neg\bot \tag{2.90}$$

$$\Gamma \vdash \neg\varphi \leftrightarrow (\varphi \to \bot) \tag{2.91}$$

Proof. (2.81): Immediately, by the definition of $\leftrightarrow$, (2.7) and (2.20).

(2.82):

(1)	$\Gamma \vdash (\varphi \to \psi) \wedge (\psi \to \varphi)$	(Assumption)
(2)	$\Gamma \vdash (\varphi \to \psi) \wedge (\psi \to \varphi) \to (\psi \to \varphi) \wedge (\varphi \to \psi)$	(PERMUTATION)
(3)	$\Gamma \vdash (\psi \to \varphi) \wedge (\varphi \to \psi)$	(MODUS-PONENS): (1), (2)

(2.83):

(1)	$\Gamma \vdash (\varphi \to \psi) \wedge (\psi \to \varphi)$	(Assumption)
(2)	$\Gamma \vdash (\psi \to \chi) \wedge (\chi \to \psi)$	(Assumption)
(3)	$\Gamma \vdash \varphi \to \psi$	(2.20): (1)
(4)	$\Gamma \vdash \psi \to \chi$	(2.20): (2)
(5)	$\Gamma \vdash \varphi \to \chi$	(SYLLOGISM): (3), (4)
(6)	$\Gamma \vdash \psi \to \varphi$	(2.20): (1)
(7)	$\Gamma \vdash \chi \to \psi$	(2.20): (2)
(8)	$\Gamma \vdash \chi \to \varphi$	(SYLLOGISM): (6), (7)
(9)	$\Gamma \vdash (\varphi \to \chi) \wedge (\chi \to \varphi)$	(2.20): (7), (12)

(2.84):

" $\Rightarrow$ "

(1)	$\Gamma \vdash \varphi \to \psi$	(Assumption)
(2)	$\Gamma \vdash (\varphi \to \varphi') \wedge (\varphi' \to \varphi)$	(Assumption)
(3)	$\Gamma \vdash \varphi' \to \varphi$	(2.20): (2)
(4)	$\Gamma \vdash (\psi \to \psi') \wedge (\psi' \to \psi)$	(Assumption)
(5)	$\Gamma \vdash \psi \to \psi'$	(2.20): (4)
(6)	$\Gamma \vdash \varphi' \to \psi$	(SYLLOGISM): (3), (1)
(7)	$\Gamma \vdash \varphi' \to \psi'$	(SYLLOGISM): (6), (5)

" $\Leftarrow$ "

(1)	$\Gamma \vdash \varphi' \to \psi'$	(Assumption)
(2)	$\Gamma \vdash (\varphi \to \varphi') \wedge (\varphi' \to \varphi)$	(Assumption)
(3)	$\Gamma \vdash \varphi \to \varphi'$	(2.20): (2)
(4)	$\Gamma \vdash (\psi \to \psi') \wedge (\psi' \to \psi)$	(Assumption)
(5)	$\Gamma \vdash \psi' \to \psi$	(2.20): (4)
(6)	$\Gamma \vdash \varphi \to \psi'$	(SYLLOGISM): (3), (1)
(7)	$\Gamma \vdash \varphi \to \psi$	(SYLLOGISM): (6), (5)

(2.85):

(1)	$\Gamma \vdash (\varphi \to \varphi') \wedge (\varphi' \to \varphi)$	(Assumption)
(2)	$\Gamma \vdash \varphi \to \varphi'$	(2.20): (1)
(3)	$\Gamma \vdash (\psi \to \psi') \wedge (\psi' \to \psi)$	(Assumption)
(4)	$\Gamma \vdash \psi \to \psi'$	(2.20): (3)
(5)	$\Gamma \vdash \varphi \vee \psi \to \varphi' \vee \psi'$	(2.48): (2), (4)
(6)	$\Gamma \vdash \varphi' \to \varphi$	(2.20): (1)
(7)	$\Gamma \vdash \psi' \to \psi$	(2.20): (3)
(8)	$\Gamma \vdash \varphi' \vee \psi' \to \varphi \vee \psi$	(2.48): (6), (7)
(9)	$\Gamma \vdash (\varphi \vee \psi \to \varphi' \vee \psi') \wedge (\varphi' \vee \psi' \to \varphi \vee \psi)$	(2.20): (5), (8)

(2.86):

(1)	$\Gamma \vdash (\varphi \to \varphi') \wedge (\varphi' \to \varphi)$	(Assumption)
(2)	$\Gamma \vdash \varphi \to \varphi'$	(2.20): (1)
(3)	$\Gamma \vdash (\psi \to \psi') \wedge (\psi' \to \psi)$	(Assumption)
(4)	$\Gamma \vdash \psi \to \psi'$	(2.20): (3)
(5)	$\Gamma \vdash \varphi \wedge \psi \to \varphi' \wedge \psi'$	(2.34): (2), (4)
(6)	$\Gamma \vdash \varphi' \to \varphi$	(2.20): (1)
(7)	$\Gamma \vdash \psi' \to \psi$	(2.20): (3)
(8)	$\Gamma \vdash \varphi' \wedge \psi' \to \varphi \wedge \psi$	(2.34): (6), (7)
(9)	$\Gamma \vdash (\varphi \wedge \psi \to \varphi' \wedge \psi') \wedge (\varphi' \wedge \psi' \to \varphi \wedge \psi)$	(2.20): (5), (8)

(2.87):

(1)	$\Gamma \vdash (\varphi \to \varphi') \wedge (\varphi' \to \varphi)$	(Assumption)
(2)	$\Gamma \vdash \varphi \to \varphi'$	(2.20): (1)
(3)	$\Gamma \vdash (\psi \to \psi') \wedge (\psi' \to \psi)$	(Assumption)
(4)	$\Gamma \vdash \psi \to \psi'$	(2.20): (3)
(5)	$\Gamma \vdash (\varphi \to \psi) \to (\varphi' \to \psi')$	(2.68): (2), (4)
(6)	$\Gamma \vdash \varphi' \to \varphi$	(2.20): (1)
(7)	$\Gamma \vdash \psi' \to \psi$	(2.20): (3)
(8)	$\Gamma \vdash (\varphi' \to \psi') \to (\varphi \to \psi)$	(2.68): (6), (7)
(9)	$\Gamma \vdash ((\varphi \to \psi) \to (\varphi' \to \psi')) \wedge ((\varphi' \to \psi') \to (\varphi \to \psi))$	(2.20): (5), (8)

(2.88):

" $\Rightarrow$ "

(1)	$\Gamma \vdash \varphi$	(Assumption)
(2)	$\Gamma \vdash \neg(\psi \wedge \neg\psi) \to \varphi$	(2.23): (1)
(3)	$\Gamma \vdash \varphi \to \neg(\psi \wedge \neg\psi)$	(2.74)
(4)	$\Gamma \vdash (\neg(\psi \wedge \neg\psi) \to \varphi) \wedge (\varphi \to \neg(\psi \wedge \neg\psi))$	(2.20): (2), (3)

" $\Leftarrow$ "

(1)	$\Gamma \vdash (\neg(\psi \wedge \neg\psi) \to \varphi) \wedge (\varphi \to \neg(\psi \wedge \neg\psi))$	(Assumption)
(2)	$\Gamma \vdash \neg(\psi \wedge \neg\psi) \to \varphi$	(2.20): (1)
(3)	$\Gamma \vdash \neg(\psi \wedge \neg\psi)$	(2.10)
(4)	$\Gamma \vdash \varphi$	(MODUS-PONENS): (3), (2)

(2.89):

(1)	$\Gamma \vdash \neg\varphi \wedge \varphi \to \varphi \wedge \neg\varphi$	(PERMUTATION)
(2)	$\Gamma \vdash \neg\varphi \to (\varphi \to \varphi \wedge \neg\varphi)$	(EXPORTATION): (1)
(3)	$\Gamma \vdash (\varphi \to \varphi \wedge \neg\varphi) \to \varphi$	(2.78)
(4)	$\Gamma \vdash (\neg\varphi \to (\varphi \to \varphi \wedge \neg\varphi)) \wedge ((\varphi \to \varphi \wedge \neg\varphi) \to \varphi)$	(2.20): (2), (3)

(2.90):

" $\Rightarrow$ "

(1)	$\Gamma \vdash \varphi$	(Assumption)
(2)	$\Gamma \vdash \neg\bot \to \varphi$	(2.23): (1)
(3)	$\Gamma \vdash \varphi \to \neg\bot$	(2.75)
(4)	$\Gamma \vdash (\neg\bot \to \varphi) \wedge (\varphi \to \neg\bot)$	(2.20): (2), (3)

" $\Leftarrow$ "

(1)	$\Gamma \vdash (\neg\bot \to \varphi) \wedge (\varphi \to \neg\bot)$	(Assumption)
(2)	$\Gamma \vdash \neg\bot \to \varphi$	(2.20): (1)
(3)	$\Gamma \vdash \neg\bot$	(2.10)
(4)	$\Gamma \vdash \varphi$	(MODUS-PONENS): (3), (2)

(2.91):

(1)	$\Gamma \vdash (\neg\varphi \to (\varphi \to \bot))$	(2.7)
(2)	$\Gamma \vdash (\varphi \to \bot) \to \neg\varphi$	(2.7)
(3)	$\Gamma \vdash (\neg\varphi \to (\varphi \to \bot)) \wedge ((\varphi \to \bot) \to \neg\varphi)$	(2.20): (1), (2)

$\square$

2.4 Deduction theorem

In the following, we prove the deduction theorem, an useful tool for obtaining proofs in Hilbert-style systems.

Theorem 2.4.1. *Let Γ be a set of formulas and φ, ψ be formulas. Then*

$$\Gamma \cup \{\varphi\} \vdash \psi \textit{ iff } \Gamma \vdash \varphi \to \psi.$$

Proof. "$\Leftarrow$" Suppose that $\Gamma \vdash \varphi \to \psi$. We get that

(1)	$\Gamma \vdash \varphi \to \psi$	hypothesis
(2)	$\Gamma \cup \{\varphi\} \vdash \varphi \to \psi$	Proposition 2.2.5.(i)
(3)	$\Gamma \cup \{\varphi\} \vdash \varphi$	Definition 2.2.1.(ii)
(4)	$\Gamma \cup \{\varphi\} \vdash \psi$	(MP): (2), (3).

"$\Rightarrow$" Let

$$\Sigma := \{\psi \in Form \mid \Gamma \vdash \varphi \to \psi\}.$$

We have to prove that $Thm(\Gamma \cup \{\varphi\}) \subseteq \Sigma$. The proof is by induction on $\Gamma \cup \{\varphi\}$-theorems.

(i) Assume that ψ is an axiom or a formula from Γ. Then

(1)	$\Gamma \vdash \psi$	Definition 2.2.1.(i), (ii)
(2)	$\Gamma \vdash \psi \to (\varphi \to \psi)$	Proposition 2.2.5.(ii)
(3)	$\Gamma \vdash \varphi \to \psi$	(MP): (1), (2).

Hence, $\psi \in \Sigma$.

(ii) Assume that $\psi = \varphi$. Then $\varphi \to \psi = \varphi \to \varphi$ is a theorem, by (2.7), so $\Gamma \vdash \varphi \to \psi$. Hence, $\psi \in \Sigma$.

(iii) Σ is closed to (MODUS PONENS).

We suppose that ψ, $\psi \to \chi \in \Sigma$ and we have to prove that $\chi \in \Sigma$. We have that

(1)	$\Gamma \vdash \varphi \to \psi$	induction hypothesis
(2)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	induction hypothesis
(3)	$\Gamma \vdash \varphi \to \chi$	(2.42): (1), (2).

Hence, $\chi \in \Sigma$. $\square$

(iv) Σ is closed to (SYLLOGISM).

We suppose that $\psi \to \chi$, $\chi \to \gamma \in \Sigma$ and we have to prove that $\psi \to \gamma \in \Sigma$. We have that

(1)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	induction hypothesis
(2)	$\Gamma \vdash \varphi \to (\chi \to \gamma)$	induction hypothesis
(5)	$\Gamma \vdash \varphi \to (\psi \to \gamma)$	(2.67): (1), (2).

(v) Σ is closed to (IMPORTATION).
We suppose that $\psi \to (\chi \to \gamma) \in \Sigma$ and we have to prove that $\psi \wedge \chi \to \gamma \in \Sigma$. We have that:

(1)	$\Gamma \vdash \varphi \to (\psi \to (\chi \to \gamma))$	(Assumption)
(2)	$\Gamma \vdash \varphi \to (\psi \wedge \chi \to \gamma)$	(2.69): (1)

(vi) Σ is closed to (EXPORTATION).
We suppose that $\psi \wedge \chi \to \gamma \in \Sigma$ and we have to prove that $\psi \to (\chi \to \gamma) \in \Sigma$. We have that:

(1)	$\Gamma \vdash \varphi \to (\psi \wedge \chi \to \gamma)$	(Assumption)
(2)	$\Gamma \vdash \varphi \to (\psi \to (\chi \to \gamma))$	(2.69): (1)

(vii) Σ is closed to (EXPANSION).
We suppose that $\psi \to \chi \in \Sigma$ and we have to prove that $\gamma \vee \psi \to \gamma \vee \chi \in \Sigma$. We have that:

(1)	$\Gamma \vdash \varphi \to (\psi \to \chi)$	induction hypothesis
(2)	$\Gamma \vdash \psi \to (\varphi \to \chi)$	(2.41): (1)
(3)	$\Gamma \vdash \chi \to \gamma \vee \chi$	(2.3)
(4)	$\Gamma \vdash \psi \to (\varphi \to \gamma \vee \chi)$	(2.44): (2), (3)
(5)	$\Gamma \vdash \gamma \wedge \varphi \to \gamma \vee \chi$	(2.6)
(6)	$\Gamma \vdash \gamma \to (\varphi \to \gamma \vee \chi)$	(EXPORTATION): (5)
(7)	$\Gamma \vdash \gamma \vee \psi \to (\varphi \to \gamma \vee \chi)$	(2.47): (6), (4)
(8)	$\Gamma \vdash \varphi \to (\gamma \vee \psi \to \gamma \vee \chi)$	(2.41): (7)

$\square$

2.5 Disjunctive theories, consistent and complete pairs

The notions and results obtained in this section will be used when giving the first proof of the completeness theorem.

Definition 2.5.1. *Let Γ be a set of formulas. We say that Γ is:*

(i) ***deductively-closed****, if $\Gamma \vdash \varphi$ implies $\varphi \in \Gamma$.*

(ii) ***consistent****, if $\Gamma \nvdash \bot$.*

(iii) ***disjunctive****, if $\Gamma \vdash \varphi \vee \psi$ implies $\Gamma \vdash \varphi$ or $\Gamma \vdash \psi$.*

Definition 2.5.2. *Let Γ be a set of formulas. We say that Γ is a* ***disjunctive theory****, if Γ is deductively-closed, consistent and disjunctive.*

Definition 2.5.3. *Let Γ, Δ be sets of formulas. The pair (Γ, Δ) is called* ***consistent****, if there are no $G_1, \ldots, G_n \in \Gamma$ and $D_1, \ldots, D_m \in \Delta$, such that $\vdash G_1 \wedge \ldots \wedge G_n \rightarrow D_1 \vee \ldots \vee D_m$.*

Remark 2.5.4. *(i) If Γ is consistent, then $(\Gamma, \emptyset)$ is consistent.*

(ii) If (Γ, Δ) is consistent, then Γ is consistent.

Proof. (i) Assume by reductio ad absurdum that $(\Gamma, \emptyset)$ is not consistent. Then, there would exist $G_1, \ldots, G_n \in \Gamma$ such that $\vdash G_1 \wedge \ldots \wedge G_n \rightarrow \bot$. But, applying Deduction Theorem and Proposition 2.2.5(i), we get that $\Gamma \vdash \bot$, which is false, since Γ is consistent.

(ii) Assume now that Γ is not consistent. By Proposition 2.2.9, this would mean that there exist $G_1, \ldots, G_n \in \Gamma$ such that $\{G_1, \ldots, G_n\} \vdash \bot$. Applying Deduction Theorem, this reduces to $\vdash G_1 \wedge \ldots \wedge G_n \rightarrow \bot$, so there exist $\{G_1, \ldots, G_n\} \subseteq \Gamma$ and $\emptyset \subseteq \Delta$, which contradict the consistency of the pair (Γ, Δ).

$\square$

Definition 2.5.5. *Let (Γ, Δ) be sets of formulas. The pair (Γ, Δ) is called* ***complete****, if for any $\varphi \in Form$, either $\varphi \in \Gamma$, or $\varphi \in \Delta$.*

Definition 2.5.6. *Complete pairs are consistent pairs of the from $(\Gamma, Form \setminus \Gamma)$.*

In the following lemma, we prove that disjunctive theories and complete pairs are in a bijection.

Lemma 2.5.7. *(i) If (Γ, Δ) is a complete pair, then Γ is a disjunctive theory.*

(ii) If Γ is a disjunctive theory, then $(\Gamma, Form \setminus \Gamma)$ is a complete pair.

Proof. (i) We prove that Γ has the three properties of a disjunctive theory.

(a) **deductively-closed**: Let φ a formula such that $\Gamma \vdash \varphi$. We proceed by reductio ad absurdum and assume that $\varphi \notin \Gamma$. Since the pair (Γ, Δ) is complete, it follows that $\varphi \in \Delta$. Let $G_1 \ldots G_n$ be the formulas from Γ which occur in the derivation of φ.

Then, we have:

(1) $\{G_1, \ldots G_n\} \vdash \varphi$ Assumption

(2) $\vdash G_1 \to \ldots \to G_n \to \varphi$ DEDUCTION THEOREM: (1)

(3) $\vdash G_1 \wedge \ldots \wedge G_n \to \varphi$ (IMPORTATION): (2)

By the assumption we made, that $\varphi \in \Delta$, it follows that there are $G_1, \ldots, G_n \in \Gamma$ and $D_1 = \varphi \in \Delta$, such that $\vdash G_1 \wedge \ldots \wedge G_n \to D_1$, so we obtained that the pair (Γ, Δ) is not consistent.

This is in contradiction to the hypothesis, so we can conclude that the assumption we made is false, and we proved that Γ is deductively-closed.

(b) **consistent**: This is obvious, by Remark 2.5.4(ii).

(c) **disjunctive**: Let $\Gamma \vdash \varphi \vee \psi$. Again, we proceed by contraposition and assume that $\Gamma \nvdash \varphi$ and $\Gamma \nvdash \psi$. This implies that $\varphi, \psi \notin \Gamma$ and, as (Γ, Δ) is complete, it must be that $\varphi, \psi \in \Delta$. We now have $G_1 = \varphi \vee \psi \in \Gamma$ and $D_1 = \varphi, D_2 = \psi \in \Delta$, such that $G_1 \to D_1 \vee D_2$, which contradicts the consistency of the pair (Γ, Δ).

In conclusion, the assumption we made is false, so we proved that Γ is disjunctive.

(ii) We show that $(\Gamma, Form \setminus \Gamma)$ is consistent, and then by Definition 2.5.6, it is immediate that $(\Gamma, Form \setminus \Gamma)$ is complete.

We assume, by reductio ad absurdum, that $(\Gamma, Form \setminus \Gamma)$ is not consistent, thus there exist $G_1, \ldots, G_n \in \Gamma$ and $D_1 \ldots, D_k \in Form \setminus \Gamma$, such that:

(1) $\vdash G_1 \wedge \ldots \wedge G_n \to D_1 \vee \ldots \vee D_k$

(2) $\{G_1 \wedge \ldots \wedge G_n\} \vdash D_1 \vee \ldots \vee D_k$ DEDUCTION THEOREM: (1)

By Definition 2.2.1(ii) and by (2.20) applied inductively, we have that $\Gamma \vdash G_1 \wedge \ldots \wedge G_n$. By Proposition 2.2.5(iii), we get that $\Gamma \vdash D_1 \vee \ldots \vee D_k$ and it can be immediately proved by induction on k, that this implies $\Gamma \vdash D_i$, for some $i = \overline{1, k}$. Considering that Γ is deductively-closed, we derive that $D_i \in \Gamma$, which is in contradiction with the hypothesis that $D_i \in Form \setminus \Gamma$.

We conclude that the assumption we made was false, so $(\Gamma, Form \setminus \Gamma)$ is a complete pair.

$\square$

Lemma 2.5.8. *Let (Γ, Δ) be a consistent pair and φ be a formula. Then, at least one of the following holds:*

(i) $(\Gamma \cup \{\varphi\}, \Delta)$ *is consistent*

(ii) $(\Gamma, \Delta \cup \{\varphi\})$ *is consistent*

Proof. We proceed by contraposition and assume that none of the two statements is true for the formula φ. Thus, we would have:
$\vdash G_1 \wedge \ldots \wedge G_n \wedge \varphi \to D_1 \vee \ldots \vee D_m$ and $\vdash H_1 \wedge \ldots \wedge H_p \to E_1 \vee \ldots \vee E_q \vee \varphi$
In the sequel, we denote $G_1 \wedge \ldots \wedge G_n$ by γ, $D_1 \vee \ldots \vee D_m$ by δ, $H_1 \wedge \ldots \wedge H_p$ by χ and $E_1 \vee \ldots \vee E_q$ by η. We have that:

(1)	$\vdash \gamma \wedge \varphi \to \delta$	Assumption
(2)	$\vdash \delta \to \delta \vee \eta$	WEAKENING
(3)	$\vdash \gamma \wedge \varphi \to \delta \vee \eta$	SYLLOGISM: (1), (2)
(4)	$\vdash (\gamma \wedge \chi) \wedge \varphi \to \gamma \wedge \varphi$	(2.20): WEAKENING, (2.4)
(5)	$\vdash (\gamma \wedge \chi) \wedge \varphi \to \delta \vee \eta$	SYLLOGISM: (4), (3)
(6)	$\vdash \chi \to \eta \vee \varphi$	Assumption
(7)	$\vdash \eta \vee \varphi \to \delta \vee (\eta \vee \varphi)$	(2.3)
(8)	$\vdash \chi \to \delta \vee (\eta \vee \varphi)$	SYLLOGISM: (6), (7)
(9)	$\vdash \delta \vee (\eta \vee \varphi) \to (\delta \vee \eta) \vee \varphi$	(2.27)
(10)	$\vdash \chi \to (\delta \vee \eta) \vee \varphi$	SYLLOGISM: (9), (10)
(11)	$\vdash \gamma \wedge \chi \to \chi$	(2.4)
(12)	$\vdash \gamma \wedge \chi \to (\delta \vee \eta) \vee \varphi$	SYLLOGISM: (11), (10)
(13)	$\vdash \gamma \wedge \chi \to \delta \vee \eta$	(2.70): (5), (12)

This contradicts the hypothesis that (Γ, Δ) is consistent.
The assumption we made was false, so we can derive the conclusion. □

Lemma 2.5.9. *Let (Γ, Δ) be a consistent pair. Then, there exists a complete pair (Γ', Δ'), such that $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$.*

Proof. We apply 2.5.8 to add all the formulas in *Form* one by one to either Γ or Δ, without making the pair inconsistent. This way, we obtain a complete pair (Γ', Δ'), with $\Gamma \subseteq \Gamma'$, $\Delta \subseteq \Delta'$. In order to do this, we consider an enumeration of the formulas $\varphi_0, \ldots, \varphi_n$ and inductively define the following family of formula-set pairs:

$$(\Gamma_0, \Delta_0) = (\Gamma, \Delta)$$

$$(\Gamma_{n+1}, \Delta_{n+1}) = \begin{cases} (\Gamma_n \cup \{\varphi\}, \Delta_n), & \text{if } (\Gamma_n \cup \{\varphi\}, \Delta_n) \text{ is consistent.} \\ (\Gamma_n, \Delta_n \cup \{\varphi\}), & \text{otherwise.} \end{cases}$$

By construction, we have that $\Gamma_0 \subseteq \Gamma_i$ and $\Delta_0 \subseteq \Delta_i$, for any $i \geq 0$ and that any pair (Γ_i, Δ_i) of the constructed family is consistent.
We now have to prove that $(\Gamma^+, \Delta^+) = (\bigcup \Gamma_i, \bigcup \Delta_i)$ is a consistent pair.
Let's proceed by reductio ad absurdum and assume that the above pair is not consistent.
Then, there would exist $G_1, \ldots, G_n \in \Gamma^+$ and $D_1, \ldots, D_m \in \Delta^+$ such that $\vdash G_1 \wedge \ldots \wedge G_n \to D_1 \vee \ldots \vee D_m$. Now, considering that we have $\Gamma_0 \subseteq \ldots \subseteq \Gamma_i \subseteq \ldots$ and $\Delta_0 \subseteq \ldots \subseteq \Delta_i \subseteq \ldots$, we obtain that there is a $i \in \mathbb{N}$, such that $G_1, \ldots, G_n \in \Gamma_i$ and $D_1, \ldots, D_m \in \Delta_i$, which means that (Γ_i, Δ_i) is not consistent, which is false.
We've reached a contradiction, so we can conclude that the pair (Γ^+, Δ^+) is consistent and, as it is obviously a partition of $Form$, we have completed the proof. $\square$

Chapter 3

Intuitionistic Propositional Logic - semantics and completeness

3.1 Kripke semantics

The first semantics we present for the IPL is the Kripke semantics, introduced by Kripke, in the seminal paper [9]. Our presentation is based on Mints' and Fitting's textbooks [3, 12].

Definition 3.1.1. *An **intuitionistic propositional Kripke model** is a tuple (W, R, V), where W is a non-empty set, R is a reflexive and transitive binary relation on W and $V : Var \times W \to \{0, 1\}$ is a function assigning truth values to variables. V is assumed to be **monotone** with respect to R, thus:*

$$V(p, w) = 1 \text{ and } Rww' \text{ implies } V(p, w') = 1$$

Definition 3.1.2. *A pair (W, R) is called a **intuitionistic propositional Kripke frame**.*

The relation from the definition above corresponds to a time succession relation between states of knowledge. Hence, it is natural that the monotonicity property has to hold for any formula (not just variables), as it will be proved later in this section.

Definition 3.1.3. *The truth value $V(\varphi, w) \in \{0, 1\}$ of an arbitrary formula φ in a world $w \in W$, in a model (W, R, V) is defined inductively, as follows:*

(i) $V(p, w)$, *for any variable p whose truth is already established*

(ii) $V(\bot, w) := 0$

(iii) $V(\varphi \wedge \psi, w) := V(\varphi, w) \wedge V(\psi, w)$

(iv) $V(\varphi \vee \psi, w) := V(\varphi, w) \vee V(\psi, w)$

(v) $V(\varphi \to \psi, w) = 1$ *iff for all* $w' \in W, Rww'$ *and* $V(\varphi, w') = 1$ *imply* $V(\psi, w') = 1$.

The above definition corresponds to the intuitive behaviour of the connectives.
For example, from the information we have in the state corresponding to the world w, we can infer $\varphi \to \psi$, iff in any state reachable in the future (by adding new information), the fact of knowing φ gives evidence for the truth of ψ.

Remark 3.1.4. $V(\neg\varphi, w) = 1$ *iff* $V(\varphi, w') = 0$*, for all* $w' \in W$*, such that* Rww'.

Definition 3.1.5.

(i) *A formula* φ *is* ***true at the world w in M****, if* $V(\varphi, w) = 1$.
Notation: $M, w \vDash \varphi$

(ii) *A formula* φ *is* ***valid in a model*** $M := (W, R, V)$*, if, for all* $w \in W, \varphi$ *is true at* w.
Notation: $M \vDash \varphi$

(iii) *A formula* φ *is* ***valid****, if* φ *is valid in all propositional intuitionistic models.*
Notation: $\vDash \varphi$

Definition 3.1.6. *Using the above introduced notation, we rewrite the conditions in Definition 3.1.3, as follows:*

(i) $M, w \vDash p$*, for any variable* p *whose truth is already established*

(ii) $M, w \nvDash \bot$

(iii) $M, w \vDash \varphi \wedge \psi$ *iff* $M, w \vDash \varphi$ *and* $M, w \vDash \psi$

(iv) $M, w \vDash \varphi \vee \psi$ *iff* $M, w \vDash \varphi$ *or* $M, w \vDash \psi$

(v) $M, w \vDash \varphi \to \psi$ *iff for all* $w' \in W, Rww'$ *and* $M, w' \vDash \varphi$ *imply* $M, w' \vDash \psi$.

Definition 3.1.7. *Let* M *be a model,* w *be a world in* M *and* Γ *be a set of formulas. We say that* M, w *forces* Γ *(and denote it by* $M, w \vDash \Gamma$*), if* φ *is true at the world* w *in* M *for all* $\varphi \in \Gamma$.

Definition 3.1.8. *Let* Γ *be a set of formulas and* φ *be a formula. We say that* φ *is a* ***local semantic consequence*** *of* Γ*, if for all models* M*, and all worlds* w *in* M*, we have:*

$$M, w \vDash \Gamma \text{ implies } M, w \vDash \varphi$$

Notation 3.1.9. *We denote the above defined notion by:* $\Gamma \vDash \varphi$ *and use the terminology:* Γ *forces* φ.

Remark 3.1.10. $\vDash \varphi$ *iff* $\emptyset \vDash \varphi$, *by the fact that the empty set has no models and the implication in Definition 3.1.7*

Definition 3.1.11. *Let* $\Gamma, \Delta \subseteq Form$. *We say that* Γ *forces* Δ *(and denote it by* $\Gamma \vDash \Delta$*), if* Γ *forces any formula* $\varphi, \varphi \in \Delta$.

Lemma 3.1.12. *For any axiom* φ, $\vDash \varphi$.

Proof. Let M be an arbitrary model and w an arbitrary world in M. We have to prove that for any axiom φ, $M, w \vDash \varphi$:

(i) CONTRACTION-$\vee$

$M, w \vDash \varphi \vee \varphi \to \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi \vee \varphi$ imply $M, w' \vDash \varphi$
iff for all w' in M, Rww' and $(M, w' \vDash \varphi$ or $M, w' \vDash \varphi)$ imply $M, w' \vDash \varphi$, which is obviously true.

(ii) CONTRACTION-$\wedge$

$M, w \vDash \varphi \to \varphi \wedge \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi$ imply $M, w' \vDash \varphi \wedge \varphi$
iff for all w' in M, Rww' and $M, w' \vDash \varphi$ imply $M, w' \vDash \varphi$ and $M, w' \vDash \varphi$, which is obviously true.

(iii) WEAKENING-$\vee$

$M, w \vDash \varphi \to \varphi \vee \psi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi$ imply $M, w' \vDash \varphi \vee \psi$
iff for all w' in M, Rww' and $M, w' \vDash \varphi$ imply $M, w' \vDash \varphi$ or $M, w' \vDash \psi$, which is obviously true.

(iv) WEAKENING-$\wedge$

$M, w \vDash \varphi \wedge \psi \to \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi \wedge \psi$ imply $M, w' \vDash \varphi$
iff for all w' in M, Rww' and $M, w' \vDash \varphi$ and $M, w' \vDash \psi$ imply $M, w' \vDash \varphi$, which is obviously true.

(v) PERMUTATION-$\vee$

$M, w \vDash \varphi \vee \psi \to \psi \vee \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi \vee \psi$ imply $M, w' \vDash \psi \vee \varphi$
iff for all w' in M, Rww' and $(M, w' \vDash \varphi$ or $M, w' \vDash \psi)$ imply $M, w' \vDash \psi$ or $M, w' \vDash \varphi$, which is obviously true.

(vi) PERMUTATION-$\wedge$

$M, w \vDash \varphi \wedge \psi \to \psi \wedge \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \varphi \wedge \psi$ imply $M, w' \vDash \psi \wedge \varphi$
iff for all w' in M, Rww' and $M, w' \vDash \varphi$ and $M, w' \vDash \psi$ imply $M, w' \vDash \psi$ and $M, w' \vDash \varphi$, which is obviously true.

(vii) EXFALSO

$M, w \vDash \bot \to \varphi$ iff for all w' in M, Rww' and $M, w' \vDash \bot$ imply $M, w' \vDash \varphi$
which is true, because, by Definition 3.1.6, $M, w \nvDash \bot$,
for any world w in any model M.

$\square$

Lemma 3.1.13. *In this lemma, we prove some trivial properties of the local semantic consequence relation. Let Γ, Δ be sets of formulas. Then:*

(i) For any formula $\varphi \in \Gamma$,

$$\Gamma \vDash \varphi.$$

(ii) Assume that $\Delta \subseteq \Gamma$. Then for any formula φ,

$$\Delta \vDash \varphi \textit{ implies } \Gamma \vDash \varphi.$$

(iii) For any formula φ,

$$\vDash \varphi \textit{ implies } \Gamma \vDash \varphi.$$

(iv) Assume that $\Gamma \vDash \Delta$. Then for any formula φ,

$$\Delta \vDash \varphi \textit{ implies } \Gamma \vDash \varphi.$$

(v) Assume that $\Delta \vDash \Gamma$ and $\Gamma \vDash \Delta$. Then for any formula φ,

$$\Delta \vDash \varphi \textit{ iff } \Gamma \vDash \varphi.$$

Proof. (i) Immediately, by Definition 3.1.8.

(ii) Immediately, by Definitions 3.1.7 and 3.1.8.

(iii) $\vDash \varphi$ iff for any model M and any world w in M, $M, w \vDash \varphi$
Let M be a model and w a world in M, such that $M, w \vDash \Gamma$.
We get that $M, w \vDash \Gamma$ implies $M, w \vDash \varphi$ iff $\Gamma \vDash \varphi$.

(iv) Let M be a model and w be a world in M, such that: $M, w \vDash \Gamma$.
By the hypothesis that $\Gamma \vDash \Delta$, we have that $M, w \vDash \psi$, for any formula $\psi \in \Delta$, and, by Defition 3.1.7, we derive that $M, w \vDash \Delta$. Since $\Delta \vDash \varphi$, we get that $M, w \vDash \varphi$, which gives evidence of the conclusion $\Gamma \vDash \varphi$.

(v) Apply (iv) twice.

□

Lemma 3.1.14. *(Monotonicity of valuation) Let (W, R, V) be a model. Then for any $w, w' \in W$ and formula φ:*

$$Rww' \text{ and } V(\varphi, w) = 1 \text{ imply } V(\varphi, w') = 1.$$

Proof. The proof is by induction on formulas. Assume Rww'.

(i) If φ is a variable, then the monotonicity is granted by the assumption in Definition 3.1.1

(ii) If $\varphi = \bot$, we don't have to prove anything, by Definition 3.1.3, $V(\bot, w) = 0$, for all w in W.

For the connective cases, by induction hypothesis, we have:

$$V(\psi, w) = 1 \text{ implies } V(\psi, w') = 1 \text{ and } V(\chi, w) = 1 \text{ implies } V(\chi, w') = 1.$$

(iii) $\varphi = \psi \wedge \chi$

$$\begin{array}{lll} V(\varphi, w) = 1 & \text{iff} & V(\psi, w) = 1 \text{ and } V(\chi, w) = 1 \\ & \text{implies} & V(\psi, w') = 1 \text{ and } V(\chi, w') = 1 \\ & \text{iff} & V(\psi \wedge \chi, w') = 1 \\ & \text{iff} & V(\varphi, w') = 1 \end{array}$$

(iv) $\varphi = \psi \vee \chi$

$$\begin{array}{lll} V(\varphi, w) = 1 & \text{iff} & V(\psi, w) = 1 \text{ or } V(\chi, w) = 1 \\ & \text{implies} & V(\psi, w') = 1 \text{ or } V(\chi, w') = 1 \\ & \text{iff} & V(\psi \vee \chi, w') = 1 \\ & \text{iff} & V(\varphi, w') = 1 \end{array}$$

(v) $\varphi = \psi \to \chi$
Assume that $V(\varphi, w) = 1$ and let w'' such that $Rw'w''$ and $V(\psi, w'') = 1$.
By the transitivity of R, we get that Rww'', and since $V(\varphi, w) = 1$ we obtain $V(\chi, w'') = 1$ and then follows the conclusion $V(\varphi, w') = 1$.

□

3.2 Kripke completeness theorem

In this section we present a detailed proof of the completeness theorem. Firsly, we prove the soundness implication, by a straightforward induction on Γ-theorems. After that, we define the so-called canonical model, whose special property known as the main semantic lemma (Lemma 3.2.3) is a crucial tool in the completeness proof.

Theorem 3.2.1. *(Soundness Theorem) Any Γ-theorem is a local semantic consequence of Γ:*

$$\Gamma \vdash \varphi \textit{ implies } \Gamma \vDash \varphi,$$

for all $\varphi \in Form$ and $\Gamma \subseteq Form$.

Proof. Let $\Sigma := \{\varphi \in Form \mid \Gamma \vDash \varphi\}$. We have to prove that $Thm(\Gamma) \subseteq \Sigma$.
The proof is by induction on Γ-theorems.

(i) By Lemmas 3.1.12 and 3.1.13(iii), $\varphi \in \Sigma$, for any axiom φ.

(ii) By Lemma 3.1.13(i), $\Gamma \subseteq \Sigma$.

Let M be a model and w be a world in M, such that $M, w \vDash \Gamma$.
In the sequel, we prove that Σ is closed to all the deduction rules:

(iii) MODUS-PONENS

$\Gamma \vDash \varphi$ and $\Gamma \vDash \varphi \to \psi$	imply	$M, w \vDash \varphi$ and $M, w \vDash \varphi \to \psi$	by Definition 3.1.8
	implies	$M, w \vDash \varphi$ implies $M, w \vDash \psi$	by reflexivity of R
	imply	$M, w \vDash \psi$	

(iv) SYLLOGISM:

$\Gamma \vDash \varphi \to \psi$ and $\Gamma \vDash \psi \to \chi$	imply	$M, w \vDash \varphi \to \psi$ and $M, w \vDash \psi \to \chi$	by Definition 3.1.8
	iff	for all w', Rww' and $M, w' \vDash \varphi$ implies $M, w' \vDash \psi$ and for all w'', Rww'' and $M, w'' \vDash \psi$ implies $M, w'' \vDash \chi$	
	iff	for all w', Rww' and $M, w' \vDash \varphi$ implies $M, w' \vDash \chi$	
	iff	$M, w \vDash \varphi \to \chi$	

(v) EXPORTATION:

$\Gamma \vDash \varphi \land \psi \to \chi$	implies	$M, w \vDash \varphi \land \psi \to \chi$	by Definition 3.1.8
	iff	for all w', Rww' and $M, w' \vDash \varphi \land \psi$ imply $M, w' \vDash \chi$	

Let w' such that Rww' and $M, w' \vDash \varphi$ and let w'' such that $Rw'w''$ and $M, w'' \vDash \psi$.
By transitivity of R, we have that Rww''. We prove that $M, w'' \vDash \chi$.

We have that $Rw'w''$ and $M, w' \vDash \varphi$ and, by Lemma 3.1.14, we get $M, w'' \vDash \varphi$.

$$\begin{array}{lll} M, w'' \vDash \varphi \text{ and } M, w'' \vDash \psi & \text{imply} & M, w'' \vDash \varphi \wedge \psi \\ & \text{implies} & M, w'' \vDash \chi \\ & \text{implies} & M, w' \vDash \psi \to \chi \\ & \text{implies} & M, w \vDash \varphi \to (\psi \to \chi) \end{array}$$

(vi) IMPORTATION:

$$\begin{array}{lll} \Gamma \vDash \varphi \to (\psi \to \chi) & \text{implies} & M, w \vDash \varphi \to (\psi \to \chi) \quad \text{by Definition 3.1.8} \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ imply } M, w' \vDash \psi \to \chi \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ imply} \\ & & \text{for all } w'', Rw'w'' \text{ and } M, w'' \vDash \psi \text{ imply } M, w'' \vDash \chi \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ and} \\ & & \text{for all } w'', Rw'w'' \text{ and } M, w'' \vDash \psi \text{ imply } M, w'' \vDash \chi \end{array}$$

Let w' such that Rww' and $M, w' \vDash \varphi$ and $M, w' \vDash \psi$.

By reflexivity of R, we have that $Rw'w'$.

So we have that $M, w' \vDash \varphi \wedge \psi$, and we can conclude that $M, w \vDash \varphi \wedge \psi \to \chi$.

(vii) EXPANSION:

$$\begin{array}{lll} \Gamma \vDash \varphi \to \psi & \text{implies} & M, w \vDash \varphi \to \psi \quad \text{by Definition 3.1.8} \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ imply } M, w' \vDash \psi \\ & \text{implies} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ imply } M, w' \vDash \chi \text{ or } M, w' \vDash \psi \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \varphi \text{ imply } M, w' \vDash \chi \vee \psi \\ & \text{iff} & \text{for all } w', Rww' \text{ and } M, w' \vDash \chi \vee \varphi \text{ imply } M, w' \vDash \chi \vee \psi \end{array}$$

□

Definition 3.2.2. *The* ***canonical model*** $M_0 = (W_0, R_0, V_0)$ *is defined as follows:*

(i) W_0 *is the set of all disjunctive theories.*

(ii) R_0 *is the subset relation.*

(iii) $V_0(v, \Gamma) = 1$ *iff* $v \in \Gamma$.

Lemma 3.2.3. $M_0, \Gamma \vDash \varphi$ *iff* $\varphi \in \Gamma$

Proof. The proof is by induction on φ.

(i) φ is a variable. The conclusion is immediate by Definition 3.2.2(iii).

(ii) $\varphi = \bot$. By Definition 3.1.6(ii), $M_0, \Gamma \nvDash \bot$ and by Definition 2.5.1(i, ii).
For the following structural cases, assume that the induction hypothesis holds for ψ and χ, thus:

$$M_0, \Gamma \vDash \psi \text{ iff } \psi \in \Gamma$$
$$M_0, \Gamma \vDash \chi \text{ iff } \chi \in \Gamma$$

(iii) $\varphi = \psi \wedge \chi$.

$M_0, \Gamma \vDash \varphi$	iff	
$M_0, \Gamma \vDash \psi \wedge \chi$	iff	
$M_0, \Gamma \vDash \psi$ and $M_0, \Gamma \vDash \chi$	iff	(by induction hypothesis)
$\psi \in \Gamma$ and $\chi \in \Gamma$	iff	(by Definition 2.2.1(ii) and Γ deductively-closed)
$\Gamma \vdash \psi$ and $\Gamma \vdash \chi$	iff	(by (2.20), (WEAKENING) and (2.4), (MODUS-PONENS))
$\Gamma \vdash \psi \wedge \chi$	iff	(Γ deductively-closed)
$\psi \wedge \chi \in \Gamma$	iff	
$\varphi \in \Gamma$		

(iv) $\varphi = \psi \vee \chi$.

$M_0, \Gamma \vDash \varphi$	iff	
$M_0, \Gamma \vDash \psi \vee \chi$	iff	
$M_0, \Gamma \vDash \psi$ or $M_0, \Gamma \vDash \chi$	iff	(by induction hypothesis)
$\psi \in \Gamma$ or $\chi \in \Gamma$	iff	(by Definition 2.2.1(ii) and Γ deductively-closed)
$\Gamma \vdash \psi$ or $\Gamma \vdash \chi$	iff	(by (WEAKENING), (2.3), (MODUS PONENS) and Γ disjunctive theory)
$\Gamma \vdash \psi \vee \chi$	iff	(Γ deductively-closed)
$\psi \vee \chi \in \Gamma$	iff	
$\varphi \in \Gamma$		

(v) (a) "$\Leftarrow$" If $\varphi \in \Gamma$, then, by Definition 3.2.2(ii), we have that $\varphi \in \Gamma'$, for all Γ' such that $R_0\Gamma\Gamma'$.
Now we have two cases:

(1) If $\psi \in \Gamma'$, applying the MODUS-PONENS rule and using the fact that Γ' is deductively closed, we get that $\chi \in \Gamma'$.
By the induction hypothesis, this is equivalent to $M_0, \Gamma' \vDash \psi$ implies $M_0, \Gamma' \vDash \chi$, for all Γ' such that $R_0\Gamma\Gamma'$ iff $M_0, \Gamma \vDash \psi \to \chi$ iff $M_0, \Gamma \vDash \varphi$.

(2) If $\psi \notin \Gamma'$, applying the left implication of the induction hypothesis, we immediately obtain a contradiction.

(b) "$\Rightarrow$" If $\varphi \notin \Gamma$, we construct Γ', such that $R_0\Gamma\Gamma'$, $\Gamma' \vDash \psi$ and $\Gamma' \nvDash \chi$.
Applying the two implications of the induction hypothesis, this reduces to $\psi \in \Gamma'$ and $\chi \notin \Gamma'$.

We consider the pair $(\Gamma \cup \{\psi\}, \{\chi\})$ and prove in the sequel that it is consistent. Assume by reductio ad absurdum that the pair is not consistent.
Then, there would exist a finite subset of $\Gamma \cup \{\psi\}$, which we'll denote by Δ, such that:

(1)	$\vdash G_1 \wedge \ldots \wedge G_n \to \chi$	
(2)	$\vdash G_1 \to \ldots \to G_n \to \chi$	EXPORTATION
(3)	$\Delta \vdash \chi$	DEDUCTION-THEOREM
(4)	$\Gamma \cup \{\psi\} \vdash \chi$	2.2.5(i): (3)
(5)	$\Gamma \vdash \psi \to \chi$	DEDUCTION-THEOREM

This is impossible by our assumption that $\varphi \notin \Gamma$ and by the fact that Γ is deductively-closed.
Thus, by Lemma 2.5.9, there exists a complete pair (Γ', Δ'), such that $\Gamma \cup \{\psi\} \subseteq \Gamma'$ and $\{\chi\} \subseteq \Delta'$.
Then, Γ' is the disjunctive theory we need, since $R_0\Gamma\Gamma'$, $\psi \in \Gamma'$ and $\chi \notin \Gamma'$.

□

Theorem 3.2.4. *(completeness theorem) For any set of formulas Γ and any formula φ:*

$$\Gamma \vdash \varphi \textit{ iff } \Gamma \vDash \varphi.$$

Proof. "$\Rightarrow$" was proved in Theorem 3.2.1.
"$\Leftarrow$" We proceed by reductio ad absurdum and assume φ to be a formula such that $\Gamma \nvdash \varphi$. Then the pair $(\Gamma, \{\varphi\})$ is consistent and, by Lemma 2.5.9, there exists a complete pair (Γ', Δ), such that $\Gamma \subseteq \Gamma'$ and $\varphi \in \Delta$. Hence, $\varphi \notin \Gamma'$, which, by Lemma 3.2.3, is equivalent to $M_0, \Gamma' \nvDash \varphi$. Also applying Lemma 3.2.3, we get that $M_0, \Gamma' \vDash \Gamma$, and so we can conclude that $\Gamma \nvDash \varphi$, which contradicts the hypothesis that $\Gamma \vDash \varphi$.
Hence, the assumption we made is false, and it follows the implication we want to prove. □

In the following, we make use of the soundness theorem to prove that some classical theorems are not intuitionistically true. In doing so, we build examples of Kripke models which refute the following formulas:

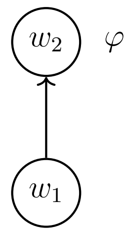
(i) $\varphi \vee \neg\varphi$

(ii) $\neg\neg\varphi \to \varphi$

(iii) $\neg(\varphi \wedge \psi) \to \neg\varphi \vee \neg\psi$

(iv) $\neg\neg(\varphi \vee \psi) \to \neg\neg\varphi \vee \neg\neg\psi$

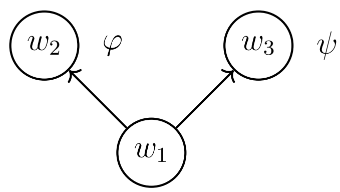
(i)



We consider the Kripke model $\mathcal{M}$ with $W = \{w_1, w_2\}$, $R = \{(w_1, w_1), (w_1, w_2), (w_2, w_2)\}$ and $V(\varphi, w_1) = 0, V(\varphi, w_2) = 1$. We have that $\mathcal{M}, w_1 \nvDash \varphi$ and, since Rw_1w_2 and $\mathcal{M}, w_2 \vDash \varphi$, we get that $\mathcal{M}, w_1 \nvDash \neg\varphi$. Hence, we proved that $\mathcal{M}, w_1 \nvDash \varphi \vee \neg\varphi$.

(ii) We can use the same counter-model from (i), because we have that $\mathcal{M}, w_1 \vDash \neg\neg\varphi$, but $\mathcal{M}, w_1 \nvDash \varphi$, so we conclude that $\mathcal{M}, w_1 \nvDash \neg\neg\varphi \to \varphi$.

(iii)



We consider the Kripke model $\mathcal{M}$ with $W = \{w_1, w_2, w_3\}$, $R = \{(w_1, w_1), (w_1, w_2), (w_1, w_3), (w_2, w_2), (w_3, w_3)\}$ and $V(\varphi, w_1) = 0, V(\varphi, w_2) = 1, V(\varphi, w_3) = 0, V(\psi, w_1) = 0, V(\psi, w_2) = 0, V(\psi, w_3) = 1$. Since $Rw_1w_1, Rw_1w_2, Rw_1w_3$ and $\mathcal{M}, w_i \nvDash \varphi \wedge \psi$, for all $i \in \{1, 2, 3\}$, we have that $\mathcal{M}, w_1 \vDash \neg(\varphi \wedge \psi)$. Assume by reductio ad absurdum that $\mathcal{M}, w_1 \vDash \neg\varphi \vee \neg\psi$. Then, we would have that either $\mathcal{M}, w_1 \vDash \neg\varphi$, or $\mathcal{M}, w_1 \vDash \neg\psi$. In both of the cases, we easily get to a contradiction: for the first one, we use Rw_1w_2 and $V(\varphi, w_2) = 1$ to conclude, and analogously, we contradict the assumption in the second case, by Rw_1w_3 and $V(\psi, w_3) = 1$. Thus, $\mathcal{M}, w_1 \nvDash \neg(\varphi \wedge \psi) \to \neg\varphi \vee \neg\psi$.

(iv) We use the same counter-model from (iii). We have that $\mathcal{M}, w_2 \vDash \varphi \vee \psi$ and $\mathcal{M}, w_3 \vDash \varphi \vee \psi$, from which it follows that $\mathcal{M}, w_1 \vDash \neg\neg(\varphi \vee \psi)$. Assume now by reductio ad absurdum that $\mathcal{M}, w_1 \vDash \neg\neg\varphi \vee \neg\neg\psi$ and we have to analyze the two cases. If $\mathcal{M}, w_1 \vDash \neg\neg\varphi$, we get a contradiction with the assumptions that Rw_1w_3 and $V(\varphi, w_3) = 0$ and symmetrically, for the second one, we use that Rw_1w_2 and $V(\psi, w_2) = 0$ to contradict the assumption.
Therefore, we conclude that $\mathcal{M}, w_1 \nvDash \neg\neg(\varphi \vee \psi) \to \neg\neg\varphi \vee \neg\neg\psi$.

3.3 Algebraic semantics

Throughout the following section, we present a different semantics for IPL, the so-called algebraic semantics, using Heyting algebras. Our exposition is based on the presentation in [4, 5].

3.3.1 Heyting algebras

In the sequel, we give some preliminary definitions and properties of Heyting algebras, which are necessary for the upcoming sections.

Definition 3.3.1. *A structure of the form* $(H, \vee, \wedge, \rightarrow, 0, 1)$ *is called* ***Heyting algebra****, if it satisfies the following properties:*

(i) $(H, \vee, \wedge, 0, 1)$ *is a bounded lattice* (A.0.7)

(ii) $a \leq b \rightarrow c$ *iff* $a \wedge b \leq c$*, for all* $a, b, c \in H$

Proposition 3.3.2. *The following properties are satisfied in a Heyting algebra:*

(i) $x \rightarrow y = 1$ *iff* $x \leq y$

(ii) $x \wedge (x \rightarrow y) = x \wedge y$

(iii) $x \rightarrow y \wedge z = (x \rightarrow y) \wedge (x \rightarrow z)$

(iv) $x \rightarrow x = 1$

Proof. (i) We have that:

$$\begin{array}{lll} x \rightarrow y = 1 & \text{iff} \quad 1 \leq x \rightarrow y & \text{by (A.0.9)(iii)} \\ & \text{iff} \quad 1 \wedge x \leq y & \\ & \text{iff} \quad x \leq y & \text{by (A.0.9)(iv)} \end{array}$$

(ii) Let $u \in H$. We have that:

$$\begin{array}{lll} u \leq x \wedge (x \rightarrow y) & \text{iff} \quad u \leq x \text{ and } u \leq x \rightarrow y & \text{by (A.0.9)(ii)} \\ & \text{iff} \quad u \leq x \text{ and } u \wedge x \leq y & \\ & \text{iff} \quad u \leq x \text{ and } u \leq y & \\ & \text{iff} \quad u \leq x \wedge y & \text{by (A.0.9)(ii)} \end{array}$$

Hence, by (A.0.9)(i), $x \wedge (x \rightarrow y) = x \wedge y$.

(iv) Applying (i), our goal reduces to $x \leq x$, which is true by the reflexivity of the $\leq$ relation. □

Definition 3.3.3. *A nonempty set* F *of the Heyting algebra* H *is called* ***filter****, if:*

(i) $x, y \in F$ *implies* $x \wedge y \in F$

(ii) $x \in F$ *and* $x \leq y$ *imply* $y \in F$

Lemma 3.3.4. *For any filter* F*, we have that* $1 \in F$*.*

Proof. This is trivial, since $x \leq 1$, for all $x \in H$. $\square$

Definition 3.3.5. *A set* F *of the Heyting algebra* H *is called* ***deductive system****, if:*

(i) $1 \in F$

(ii) $x, x \to y \in F$ *implies* $y \in F$

Proposition 3.3.6. *If* F *is a subset of the Heyting algebra* H*, then the following are equivalent:*

(i) F *is a filter*

(ii) F *is a deductive system*

Proof. (i) $(i) \Rightarrow (ii)$:
If $x, x \to y \in F$, then $x \wedge (x \to y) \in F$ and, since by Proposition 3.3.2(ii), $x \wedge (x \to y) = x \wedge y$, we have that $x \wedge y \in F$. But, by Definition A.0.5, $x \wedge y \leq y$, so $y \in F$.
It is obvious that $1 \in F$, since $x \leq 1$, for all $x \in H$. Hence, we proved that F is a deductive system.

(ii) $(ii) \Rightarrow (i)$:
Assume that $x \leq y$ and $x \in F$. Then, $x \to y = 1 \in F$, so we can derive that $y \in F$.
Now, let $x, y \in F$. Since $y \leq x \to y$, we have that $x \to y \in F$.
By, Proposition 3.3.2((iii), (ii)), we have that $x \to x \wedge y = (x \to x) \wedge (x \to y) = 1 \wedge (x \to y) = x \to y$. Now, since $x \in F$ and $x \to (x \wedge y) \in F$, we can conclude that $x \wedge y \in F$.

$\square$

Definition 3.3.7. *Let* X *be a subset of the Heyting algebra* H*. The filter generated by* X *is the intersection of all the filters of* H *containing* X*.*

Notation 3.3.8. *The filter generated by* X *is denoted in the sequel by* $[X)$*.*

Proposition 3.3.9. *Let* X *be a nonempty subset of the Heyting algebra* H*. Then:*

$$[X) = \{a \mid \textit{there exist } x_1, \ldots, x_n \in X, x_1 \wedge \ldots \wedge x_n \leq a\}$$

Proof. Let's denote the set $\{a \mid \text{there exist } x_1, \ldots, x_n \in X, x_1 \wedge \ldots \wedge x_n \leq a\}$ by S. We have to prove that S is a filter containing X and that for any other filter F, such that $X \subseteq F$, we have that $S \subseteq F$.
To prove that S is a filter of A, let first $x, y \in S$. Thus, we have that there exist $x_1, \ldots x_n \in X$ such that $x_1 \wedge \ldots \wedge x_n \leq x$ and $y_1, \ldots, y_m \in X$ such that $y_1 \wedge \ldots \wedge y_m \leq y$. Hence, there exist $x_1, \ldots, x_n, y_1 \ldots, y_m \in X$ such that $x_1 \wedge \ldots \wedge x_n \wedge y_1 \wedge \ldots \wedge y_m \leq x \wedge y$, which gives evidence of the fact that $x \wedge y \in S$.
Now, let $a_1 \in S$ and $a_1 \leq a_2$. There exist $x_1, \ldots, x_n \in X$, such that $x_1 \wedge \ldots \wedge x_n \leq a_1 \leq a_2$, so $a_2 \in S$.
Next, to prove that $X \subseteq S$, let $x \in X$ and we show that $x \in S$. This is because we can consider x itself to be the witness in the condition of S, since $x \leq x$.
Finally, let F be a filter of H, such that $X \subseteq F$.
Let $x \in S$. Thus, there exist $x_1, \ldots, x_n \in X \subseteq F$, such that $x_1 \wedge \ldots \wedge x_n \leq x$. Since $x_1, \ldots, x_n \in F$, we have that $x_1 \wedge \ldots \wedge x_n \in F$ and, by Definition 3.3.3(ii), we have that $x \in F$.
Hence, we proved that $S \subseteq F$. □

Lemma 3.3.10. *If F is a filter of H and $a \to b \notin F$, then $b \notin [F \cup \{a\})$.*

Proof. Assume by reductio ad absurdum that $b \in [F \cup \{a\})$.
By Proposition 3.3.9, we have that there exist $x_1, \ldots, x_n \in F \cup \{a\}$, such that $x_1 \wedge \ldots \wedge x_n \leq b$.
Assuming that $x_1 = \ldots = x_n = a$, we get that $a \leq b$, which is equivalent to $a \to b = 1$. We would get that $1 \notin F$, which is false, by Proposition 3.3.6.
Thus, there exists a subset S of $\{x_1, \ldots, x_n\}$, such that $S \subseteq F \setminus \{a\}$.
By Definition 3.3.3(i), we have that $\bigwedge_{x \in S} x \in F$, so there is an $y = \bigwedge_{x \in S} x \in F$, such that $y \wedge a \leq b$, i.e. $y \leq a \to b$. By Definition 3.3.3(ii), we conclude that $a \to b \in F$, which contradicts the hypothesis. Hence, $b \notin [F \cup \{a\})$. □

Definition 3.3.11. *A filter F is called **proper** filter if $0 \notin F$.*

Lemma 3.3.12. *If F is a filter of H and $\neg a \notin F$, then $[F \cup \{a\})$ is a proper filter.*

Proof. By Lemma 3.3.10, $a \to 0 = \neg a \in F$ implies $0 \notin [F \cup \{a\})$. □

Definition 3.3.13. *A proper filter P of H is called **prime** filter if:*

$$x \vee y \in P \textit{ implies } x \in P \textit{ or } y \in P.$$

Proposition 3.3.14. *Let F be a filter of the Heyting algebra H and $a \notin F$. Then, there is a prime filter P of H, such that $F \subseteq P$ and $a \notin P$.*

Proof. Let $\mathcal{F}$ be the set of all filters F' of H, such that $a \notin F'$.
We prove that $\mathcal{F}$ is an inductively ordered set, in order to apply Zorn's lemma on it.
Thus, we have to prove that, any nonempty totally ordered subset of C of $\mathcal{F}$ is upper-bounded by the union of its elements. Obviously, we have that, for any $F \in C$, $F \subseteq \bigcup C$ and $a \notin \bigcup C$.
Hence, all we need to prove is that $\bigcup C$ is itself a filter:

(i) Let $x, y \in \bigcup C$. We show that $x \wedge y \in \bigcup C$.
Since $x, y \in \bigcup C$, there are $F_1, F_2 \in C$, such that $x \in F_1$ and $y \in F_2$.
C being partially ordered, we have that $F_1 \subseteq F_2$ or $F_2 \subseteq F_1$. Without loss of generality, assume that $F_1 \subseteq F_2$ holds. Then, we get that $x, y \in F_2$, so we can use F_2's first filter property to conclude that $x \wedge y \in F_2$ and, by $F_2 \subseteq \bigcup C$, we conclude that $x \wedge y \in \bigcup C$.

(ii) Let $x \in \bigcup C$ and y such that $x \leq y$. We prove that $y \in \bigcup C$.
Since $x \in \bigcup C$, there is a $F \in C$, such that $x \in F$.
We apply F's second filter property to derive that $y \in F \in \bigcup C$ and, by $F \subseteq \bigcup C$, we conclude that $y \in \bigcup C$.

Applying Zorn's Lemma to $(\mathcal{F}, \subseteq)$, we get that there is a maximal element of $\mathcal{F}$. We denote it by P and we show that P is a prime filter.
Assume by reductio ad absurdum that P is not prime, so there exist $x, y \in H$, such that $x \vee y \in P$, $x \notin P$ and $y \notin P$. We have that $P \subset [P \cup \{x\})$ and $P \subset [P \cup \{y\})$ and, by the maximality of P, we get that $a \in [P \cup \{x\})$ and $a \in [P \cup \{y\})$. By Proposition 3.3.9, there exist $u, v \in P$, such that $u \wedge x \leq a$ and $v \wedge y \leq a$. Let's denote $c = u \wedge v$. We have that:

$x \wedge c \leq u \wedge x \leq a$ and $y \wedge c \leq v \wedge y \leq a$ by Definition A.0.5
iff $x \leq c \to a$ and $y \leq c \to a$
iff $x \vee y \leq c \to a$ by (A.0.9)(v)
iff $(x \vee y) \wedge c \leq a$

But, since $x \vee y \in P$ and $c \in P$, we have that $(x \vee y) \wedge c \in P$, so, by Definition 3.3.3(ii), we get that $a \in P$. This is a contradiction, so P has to be prime. Obviously, $F \subseteq P$ and $a \notin P$. $\square$

Corollary 3.3.15. *Assume that $a \in H, a \neq 1$. Then there exists a prime filter P of H such that $a \notin P$.*

Proof. We first prove that $\{1\}$ is a filter. This is trivial:

(i) It reduces to showing that $1 \wedge 1 = 1$, which is true by Proposition A.0.1(iii).

(ii) It follows immediately by Proposition A.0.1(iii).

Now, applying Proposition 3.3.14 to the filter $\{1\}$, we get the conclusion. $\square$

Notation 3.3.16. *We denote by $\mathcal{P}(H)$ the set of all the prime filters of the Heyting algebra H.*

Corollary 3.3.17.

$$\bigcap_{P\in\mathcal{P}(H)} P = \{1\}.$$

Proof. We prove this by double inclusion.
"$\subseteq$" Let $a \in \bigcap_{P\in\mathcal{P}(H)} P$ and assume that $a \neq 1$. Then, by Corrolary 3.3.15, we would get that there is a prime filter P of H, such that $a \notin P$. Thus, we obtained a contradiction, so it has to be that $a = 1$.
"$\supseteq$" This second inclusion is trivial, by applying Lemma 3.3.4 to any prime filter P of H. □

3.3.2 Algebraic models

Let H be a Heyting algebra.

Definition 3.3.18. *An* ***algebraic interpretation*** *in H is a function $\overline{h} : Form \to H$ satisfying the following properties:*

(i) $\overline{h}(\perp) = 0$;

(ii) $\overline{h}(\varphi \wedge \psi) = \overline{h}(\varphi) \wedge \overline{h}(\psi)$ *for all* $\varphi, \psi \in Form$;

(iii) $\overline{h}(\varphi \vee \psi) = \overline{h}(\varphi) \vee \overline{h}(\psi)$ *for all* $\varphi, \psi \in Form$;

(iv) $\overline{h}(\varphi \to \psi) = \overline{h}(\varphi) \to \overline{h}(\psi)$ *for all* $\varphi, \psi \in Form$.

Lemma 3.3.19. *Let $\overline{h} : Form \to H$ be an algebraic interpretation. Then*

(i) $\overline{h}(\neg\varphi) = \neg\overline{h}(\varphi)$ *for all* $\varphi \in Form$;

(ii) $\overline{h}(\top) = 1$.

Proof. (i) $\overline{h}(\neg\varphi) = \overline{h}(\varphi \to 0) = \overline{h}(\varphi) \to \overline{h}(0) = \overline{h}(\varphi) \to 0 = \neg\overline{h}(\varphi)$

(ii) $\overline{h}(\top) = \overline{h}(\neg\perp) = \neg\overline{h}(\perp) = \neg 0 = 1$

□

Definition 3.3.20. *An* ***algebraic model*** *is a pair $(H, \overline{h})$, where H is a Heyting algebra and $\overline{h}$ is an algebraic interpretation in H.*

Definition 3.3.21. *Let φ be a formula.*

*(i) We say that φ is **true in an algebraic model** $(H, \overline{h})$, if $\overline{h}(\varphi) = 1$.*
Notation: $(H, \overline{h}) \vDash_{alg} \varphi$

*(ii) We say that φ is **valid** in H, if φ is true in any algebraic model $(H, \overline{h})$.*
Notation: $H \vDash_{alg} \varphi$.

*(iii) We say that φ is **algebraically valid**, if φ is valid in every Heyting algebra H.*
Notation: $\vDash_{alg} \varphi$.

Definition 3.3.22. *Let Γ be a set of formulas.*

*(i) We say that Γ is **true in an algebraic model** $(H, \overline{h})$, if $(H, \overline{h}) \vDash_{alg} \varphi$ for any $\varphi \in \Gamma$.*
Notation: $(H, \overline{h}) \vDash_{alg} \Gamma$.

*(ii) We say that Γ is **valid** in H, if Γ is true in any algebraic model $(H, \overline{h})$.*
Notation: $H \vDash_{alg} \Gamma$.

*(iii) We say that Γ is **algebraically valid**, if Γ is valid in every Heyting algebra H.*
Notation: $\vDash_{alg} \Gamma$.

Definition 3.3.23. *Let $\Gamma \cup \{\varphi\}$ be a set of formulas. We say that φ is an algebraic semantic consequence of Γ if for any algebraic model $(H, \overline{h})$,*

$$(H, \overline{h}) \vDash_{alg} \Gamma \textit{ implies } (H, \overline{h}) \vDash_{alg} \varphi.$$

Notation: $\Gamma \vDash_{alg} \varphi$.

The following lemma collects some trivial properties.

Lemma 3.3.24. *Let $\Gamma \cup \{\varphi\}$ be a set of formulas.*

(i) $\emptyset \vDash_{alg} \varphi$ iff $\vDash_{alg} \varphi$.

(ii) If $\varphi \in \Gamma$, then $\Gamma \vDash_{alg} \varphi$.

(iii) If $\vDash_{alg} \varphi$, then $\Gamma \vDash_{alg} \varphi$.

Proof. (i) "$\Rightarrow$" Since $\emptyset$ is true in any algebraic model, we can immediately derive the conclusion. "$\Leftarrow$" Conversely, let $(H, \overline{h})$ be an algebraic model such that $(H, \overline{h}) \vDash_{alg} \emptyset$. Since φ is algebraically valid, we have that $(H, \overline{h}) \vDash_{alg} \varphi$.

(ii) Let $(H, \overline{h})$ be an algebraic model such that $(H, \overline{h}) \vDash_{alg} \Gamma$.
Since $\varphi \in \Gamma$, by Definition 3.3.22(i), we get that $(H, \overline{h}) \vDash_{alg} \varphi$.

(iii) Let $(H, \overline{h})$ be an algebraic model such that $(H, \overline{h}) \vDash_{alg} \Gamma$.
Since φ is algebraically valid, we have that $(H, \overline{h}) \vDash_{alg} \varphi$.

$\square$

3.4 Lindenbaum-Tarski algebra

We define the Lindenbaum-Tarksi algebra, a very important example of Heyting algebra, which we make use of, when proving the algebraic completeness theorem.

Definition 3.4.1. *We define a binary relation $\sim$ on Form:*

$$\varphi \sim_\Gamma \psi \textit{ iff } \Gamma \vdash \varphi \leftrightarrow \psi.$$

Lemma 3.4.2. *We prove that $\sim$ is an equivalence relation on Form.*

Proof. (i) Reflexivity follows by (2.81).

(ii) Symmetry follows by (2.82).

(iii) Transitivity follows by (2.83).

□

Let $Form/\sim_\Gamma$ be the factor set. We denote the equivalence class of a formula φ by $\widehat{\varphi}_\Gamma$.

Lemma 3.4.3. *Let $\varphi, \varphi', \psi, \psi'$ be formulas. If $\varphi \sim_\Gamma \varphi'$ and $\psi \sim_\Gamma \psi'$, then:*

$$\Gamma \vdash \varphi \to \psi \textit{ iff } \Gamma \vdash \varphi' \to \psi'.$$

Proof. By (2.84). □

Lemma 3.4.4. *We consider the following binary relation $\leq_\Gamma$ on $Form/\sim_\Gamma$ and prove that it is an order relation:*

$$\widehat{\varphi}_\Gamma \leq_\Gamma \widehat{\psi}_\Gamma \textit{ iff } \Gamma \vdash \varphi \to \psi.$$

Proof. (i) Reflexivity follows by (2.7).

(ii) Antisymmetry is trivial (by Definition 3.4.1).

(iii) Transitivity follows by applying SYLLOGISM.

□

Definition 3.4.5. *We define the $\vee$, $\wedge$ and $\to$ operations on $Form/\sim_\Gamma$ as follows:*

$$\widehat{\varphi}_\Gamma \vee \widehat{\psi}_\Gamma = \widehat{\varphi \vee \psi}_\Gamma$$
$$\widehat{\varphi}_\Gamma \wedge \widehat{\psi}_\Gamma = \widehat{\varphi \wedge \psi}_\Gamma$$
$$\widehat{\varphi}_\Gamma \to \widehat{\psi}_\Gamma = \widehat{\varphi \to \psi}_\Gamma$$

Next, we have to prove that the above defined operations are well-defined:

Lemma 3.4.6. *Let $\varphi, \varphi', \psi, \psi'$ be formulas such that $\varphi \sim_\Gamma \varphi'$ and $\psi \sim_\Gamma \psi'$. Then we have that:*

(i) $\widehat{\varphi \vee \psi}_\Gamma = \widehat{\varphi' \vee \psi'}_\Gamma$

(ii) $\widehat{\varphi \wedge \psi}_\Gamma = \widehat{\varphi' \wedge \psi'}_\Gamma$

(iii) $\widehat{\varphi \to \psi}_\Gamma = \widehat{\varphi' \to \psi'}_\Gamma$

Proof. (i) By (2.85).

(ii) By (2.86).

(iii) By (2.87).

□

Proposition 3.4.7. *$(Form/\sim_\Gamma, \leq)$ is a lattice, where $\widehat{\varphi}_\Gamma \vee \widehat{\psi}_\Gamma = \widehat{\varphi \vee \psi}_\Gamma$ and $\widehat{\varphi}_\Gamma \wedge \widehat{\psi}_\Gamma = \widehat{\varphi \wedge \psi}_\Gamma$.*

Proof. By WEAKENING-DISJ, WEAKENING-CONJ, (2.3), (2.4). □

Lemma 3.4.8. *For any $\varphi, \psi, \chi \in E$, we have that:*

$$\widehat{\varphi}_\Gamma \leq_\Gamma \widehat{\psi}_\Gamma \to \widehat{\chi}_\Gamma \text{ iff } \widehat{\varphi}_\Gamma \wedge \widehat{\psi}_\Gamma \leq_\Gamma \widehat{\chi}_\Gamma.$$

Proof. We have that:

$$\begin{aligned}
\widehat{\varphi}_\Gamma \leq_\Gamma \widehat{\psi}_\Gamma \to \widehat{\chi}_\Gamma \quad &\text{iff} \quad \widehat{\varphi}_\Gamma \leq_\Gamma \widehat{\psi \to \chi}_\Gamma \\
&\text{iff} \quad \vdash \varphi \to (\psi \to \chi) \\
&\text{iff} \quad \vdash \varphi \wedge \psi \to \chi \\
&\text{iff} \quad \widehat{\varphi \wedge \psi}_\Gamma \leq_\Gamma \widehat{\chi}_\Gamma \\
&\text{iff} \quad \widehat{\varphi}_\Gamma \wedge \widehat{\psi}_\Gamma \leq_\Gamma \widehat{\chi}_\Gamma
\end{aligned}$$

□

Proposition 3.4.9. *The factor set $Form/\sim_\Gamma$ is a Heyting algebra, where:*

(i) $\widehat{\varphi}_\Gamma \vee \widehat{\psi}_\Gamma = \widehat{\varphi \vee \psi}_\Gamma$, $\widehat{\varphi}_\Gamma \wedge \widehat{\psi}_\Gamma = \widehat{\varphi \wedge \psi}_\Gamma$, $\widehat{\varphi}_\Gamma \to \widehat{\psi}_\Gamma = \widehat{\varphi \to \psi}_\Gamma$

(ii) $\widehat{\bot}_\Gamma$ *is the first element*

(iii) $\widehat{\neg\bot}_\Gamma$ *is the last element*

Proof. (i) By Lemmas 3.4.6, 3.4.8 and Proposition 3.4.7.

(ii) By EXFALSO.

(iii) By (2.75).

□

Proposition 3.4.10. *For any $\varphi \in Form$, we have that:*

(i) $\Gamma \vdash \varphi$ *iff* $\widehat{\varphi}_\Gamma = 1$

(ii) $\widehat{\neg\varphi}_\Gamma = \widehat{\varphi}_\Gamma \to 0$

Proof.

By (2.90).

By (2.91). □

Definition 3.4.11. *The Heyting algebra* $(Form/\sim_\Gamma, \vee, \wedge, \to, 0, 1)$ *is called the Lindenbaum-Tarski algebra of IPL.*

Proposition 3.4.12. *Define the following mapping:*

$$\overline{h}_{LT\Gamma} : Form \to Form/\sim_\Gamma, \quad \overline{h}_{LT\Gamma}(\varphi) = \widehat{\varphi}_\Gamma.$$

Then $LT_\Gamma = (Form/\sim_\Gamma, \overline{h}_{LT\Gamma})$ *is an algebraic model satisfying the following:*

(i) $LT_\Gamma \vDash_{alg} \Gamma$.

(ii) For any formula φ, $LT_\Gamma \vDash_{alg} \varphi$ *iff* $\Gamma \vdash \varphi$.

Proof. (i) Let $\varphi \in \Gamma$. We have to prove that $LT_\Gamma \vDash_{alg} \varphi$, which is equivalent to $\widehat{\varphi}_\Gamma = 1$. By Proposition 3.4.10(i), the goal reduces to proving that $\Gamma \vdash \varphi$ and this is true, since φ is an element of Γ.

(ii) "$\Rightarrow$" Assume that $LT_\Gamma \vDash_{alg} \varphi$. By explicitating $\overline{h}_{LT\Gamma}$ and applying Proposition 3.4.10(i), we obtain $\Gamma \vdash \varphi$.
"$\Leftarrow$" If $\Gamma \vdash \varphi$, we can apply the reverse implication from Proposition 3.4.10(i), to get that $\widehat{\varphi}_\Gamma = 1$, which gives evidence of our conclusion that $LT_\Gamma \vDash_{alg} \varphi$.

□

3.5 Algebraic completeness theorem

Hereby we present a second proof of the completeness theorem, using the algebraic semantics. Let Γ be a set of formulas and φ be a formula.

Theorem 3.5.1. *[algebraic completeness] For any set of formulas* Γ *and any formula* φ,

$$\Gamma \vdash \varphi \textit{ iff } \Gamma \vDash_{alg} \varphi.$$

Proof. "$\Rightarrow$":

We proceed by induction on Γ-theorems.

Firstly, we have to prove that any element of Γ is an algebraic semantic consequence of Γ, which is true by Proposition 3.3.24(ii).

Next, we prove that any axiom is an algebraic semantic consequence of Γ.

Let H be a Heyting algebra and h be an algebraic interpretation.

(i) CONTRACTION-$\vee$:

$$\begin{aligned} h(\varphi \vee \varphi \to \varphi) = 1 \quad &\text{iff} \quad h(\varphi \vee \varphi) \to h(\varphi) = 1 \\ &\text{iff} \quad (h(\varphi) \vee h(\varphi)) \to h(\varphi) = 1 \\ &\text{iff} \quad h(\varphi) \to h(\varphi) = 1 \qquad \text{by Definition A.0.1(iii)} \end{aligned}$$

which is true, by Proposition 3.3.2(iv).

(ii) CONTRACTION-$\wedge$:

$$\begin{aligned} h(\varphi \to \varphi \wedge \varphi) = 1 \quad &\text{iff} \quad h(\varphi) \to h(\varphi \wedge \varphi) = 1 \\ &\text{iff} \quad h(\varphi) \to (h(\varphi) \wedge h(\varphi)) = 1 \\ &\text{iff} \quad h(\varphi) \to h(\varphi) = 1 \qquad \text{by Definition A.0.1(iii)} \end{aligned}$$

which is true, by Proposition 3.3.2(iv).

(iii) WEAKENING-$\vee$:

$$\begin{aligned} h(\varphi \to \varphi \vee \psi) = 1 \quad &\text{iff} \quad h(\varphi) \to h(\varphi \vee \psi) = 1 \\ &\text{iff} \quad h(\varphi) \to (h(\varphi) \vee h(\psi)) = 1 \\ &\text{iff} \quad h(\varphi) \leq h(\varphi) \vee h(\psi) \qquad \text{by Proposition 3.3.2(i)} \end{aligned}$$

which is true, by Definition A.0.5.

(iv) WEAKENING-$\wedge$:

$$\begin{aligned} h(\varphi \wedge \psi \to \varphi) = 1 \quad &\text{iff} \quad (h(\varphi) \wedge h(\psi)) \to h(\varphi) = 1 \\ &\text{iff} \quad (h(\varphi) \wedge (h(\psi)) \to h(\varphi)) = 1 \qquad \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad h(\varphi) \wedge h(\psi) \leq h(\varphi), \text{ which is true.} \end{aligned}$$

which is true, by Definition A.0.5.

(v) PERMUTATION-$\vee$:

$$\begin{aligned} h(\varphi \vee \psi \to \psi \vee \varphi) = 1 \quad &\text{iff} \quad h(\varphi \vee \psi) \to h(\psi \vee \varphi) = 1 \\ &\text{iff} \quad (h(\varphi) \vee h(\psi)) \to (h(\psi) \vee h(\varphi)) = 1 \\ &\text{iff} \quad (h(\varphi) \vee h(\psi)) \to (h(\varphi) \vee h(\psi)) = 1 \quad \text{by Definition A.0.1(i)} \end{aligned}$$

which is true, by Proposition 3.3.2(iv).

(vi) PERMUTATION-$\wedge$:

$$\begin{aligned} h(\varphi \wedge \psi \to \psi \wedge \varphi) = 1 \quad &\text{iff} \quad h(\varphi \wedge \psi) \to h(\psi \wedge \varphi) = 1 \\ &\text{iff} \quad (h(\varphi) \wedge h(\psi)) \to (h(\psi) \wedge h(\varphi)) = 1 \\ &\text{iff} \quad (h(\varphi) \wedge h(\psi)) \to (h(\varphi) \wedge h(\psi)) = 1 \quad \text{by Definition A.0.1(i)} \end{aligned}$$

which is true, by Proposition 3.3.2(iv).

(vii) EX FALSO QUODLIBET:

$$\begin{aligned} h(\bot \to \varphi) = 1 \quad &\text{iff} \quad h(\bot) \to h(\varphi) = 1 \\ &\text{iff} \quad h(\bot) \le h(\varphi) && \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad 0 \le h(\varphi) \end{aligned}$$

which is true, by Definition A.0.7.

Now, we prove that the deduction-rules generate algebraic semantic consequences of Γ:

(i) MODUS-PONENS:

$$\begin{aligned} h(\varphi \to \psi) = 1 \quad &\text{iff} \quad h(\varphi) \to h(\psi) = 1 \\ &\text{iff} \quad h(\varphi) \le h(\psi) && \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad 1 \le h(\psi) && \text{by the induction hypothesis} \\ &\text{iff} \quad h(\psi) = 1 && \text{by Proposition A.0.9(iii)} \end{aligned}$$

(ii) SYLLOGISM:

By the induction hypothesis, we have that $h(\varphi \to \psi) = 1$ and $h(\psi \to \chi) = 1$. Hence, by Proposition 3.3.2(i), we get that $h(\varphi) \le h(\psi)$ and $h(\psi) \le h(\chi)$, the transitivity of $\le$ and the reverse implication of Proposition 3.3.2(i), we immediately get the conclusion.

(iii) EXPORTATION:

$$\begin{aligned} h(\varphi \wedge \psi \to \chi) = 1 \quad &\text{iff} \quad h(\varphi) \wedge h(\psi) \le h(\chi) \\ &\text{iff} \quad h(\varphi) \le h(\psi) \to h(\chi) \\ &\text{iff} \quad h(\varphi) \to (h(\psi) \to h(\chi)) = 1 && \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad h(\varphi \to (\psi \to \chi)) = 1 \end{aligned}$$

(iv) IMPORTATION:

$$\begin{aligned} h(\varphi \to (\psi \to \chi)) = 1 \quad &\text{iff} \quad h(\varphi) \le (h(\psi) \to h(\chi)) \\ &\text{iff} \quad h(\varphi) \wedge h(\psi) \le h(\chi) \\ &\text{iff} \quad h(\varphi) \wedge h(\psi) \to h(\chi) = 1 && \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad h(\varphi \wedge \psi \to \chi) = 1 \end{aligned}$$

(v) EXPANSION:

$$\begin{aligned} h(\varphi \to \psi) = 1 \quad &\text{iff} \quad h(\varphi) \le h(\psi) \\ &\text{implies} \quad h(\chi) \vee h(\varphi) \le h(\chi) \vee h(\psi) && \text{by Proposition A.0.9(vi)} \\ &\text{iff} \quad h(\chi \vee \varphi) \le h(\chi \vee \psi) \\ &\text{iff} \quad h(\chi \vee \varphi) \to h(\chi \vee \psi) = 1 && \text{by Proposition 3.3.2(i)} \\ &\text{iff} \quad h(\chi \vee \varphi \to \chi \vee \psi) = 1 \end{aligned}$$

"$\Leftarrow$": By Proposition 3.4.12(i), we have that $LT_\Gamma \vDash_{alg} \Gamma$ and since $\Gamma \vDash_{alg} \varphi$, we get that $LT_\Gamma \vDash_{alg} \varphi$. Now, applying Proposition 3.4.12(ii), we conclude that $\Gamma \vdash \varphi$. □

3.6 Kripke models and algebraic models

In the last section of this chapter, we establish a correspondence between Kripke models and algebraic models and use it to prove the equivalence between Kripke and algebraic validity.

3.6.1 From Kripke models to algebraic models

Definition 3.6.1. *Let (W, R) be a Kripke frame. A subset A of W is called closed if:*

$$w \in A \text{ and } Rww' \text{ imply } w' \in A.$$

Notation 3.6.2. *We denote by $\mathcal{H}$ the set of all closed subsets of W.*

Notation 3.6.3. *Let $A, B \in \mathcal{H}$. We denote by $A \to B$ the greatest closed subset of W contained in $(W \setminus A) \cup B$.*

Lemma 3.6.4. *For any $A, B, X \in \mathcal{H}$:*

$$X \subseteq A \to B \text{ iff } A \cap X \subseteq B.$$

Proof. " $\Rightarrow$ "

$$\begin{aligned}
X \subseteq A \to B \quad &\text{implies} \quad X \subseteq (W \setminus A) \cup B \\
&\text{implies} \quad A \cap X \subseteq A \cap [(W \setminus A) \cup B] = A \cap B \\
&\text{implies} \quad A \cap X \subseteq B
\end{aligned}$$

" $\Leftarrow$ "

$$\begin{aligned}
A \cap X \subseteq B \quad &\text{implies} \quad (A \cap X) \cup (X \setminus A) \subseteq (X \setminus A) \cup B \subseteq (W \setminus A) \cup B \\
&\text{implies} \quad X \subseteq (W \setminus A) \cup B \\
&\text{implies} \quad X \subseteq A \to B \text{ (since } X \text{ is closed and by the maximality of } A \to B)
\end{aligned}$$

□

Corollary 3.6.5. *$(\mathcal{H}, \cap, \cup, \to, \emptyset, W)$ is a Heyting algebra.*

Lemma 3.6.6. *Consider the Kripke model $\mathcal{M} = (W, R, V)$. Let $h : Form \to \mathcal{H}$ be such that $h(\varphi) = \{w \in W \mid V(\varphi, w) = 1\}$, for any $\varphi \in Form$.*
Then, $\mathcal{H}_\mathcal{M} = (\mathcal{H}, h)$ is an algebraic model.

Proof. (i) $h(v) = \{w \in W \mid V(v, w) = 1\} \in \mathcal{H}$, by Definition 3.6.1 and by the monotonicity of V

(ii) $h(\perp) = \{w \in W \mid V(\perp, w) = 1\} = \emptyset$

(iii) $h(\varphi \wedge \psi) = \{w \in W \mid V(\varphi \wedge \psi, w) = 1\} = \{w \in W \mid V(\varphi, w) = 1 \text{ and } V(\psi, w) = 1\} = \{w \in W \mid V(\varphi, w) = 1\} \cap \{w \in W \mid V(\psi, w) = 1\} = h(\varphi) \cap h(\psi)$

(iv) $h(\varphi \vee \psi) = \{w \in W \mid V(\varphi \vee \psi, w) = 1\} = \{w \in W \mid V(\varphi, w) = 1 \text{ or } V(\psi, w) = 1\} = \{w \in W \mid V(\varphi, w) = 1\} \cup \{w \in W \mid V(\psi, w) = 1\} = h(\varphi) \cup h(\psi)$

(v) Let us show first that,

$$(*)\ A \subseteq h(\varphi \to \psi) \text{ iff } A \cap h(\varphi) \subseteq h(\psi) \text{ for any } A \in \mathcal{H}.$$

" $\Rightarrow$ " Assume that $A \subseteq h(\varphi \to \psi)$.
$h(\varphi \to \psi) = \{w \mid V(\varphi \to \psi, w) = 1\} = \{w \mid \text{for all } w' \in W, \text{ such that } Rww', V(\varphi, w') = 1 \text{ implies } V(\psi, w') = 1\}$
Let $w \in A \cap h(\varphi)$. Since $A \subseteq h(\varphi \to \psi)$, we have that $w \in h(\varphi \to \psi)$.
On the other hand, $w \in h(\varphi)$ implies that $V(\varphi, w) = 1$ and since Rww, thus we get that $V(\psi, w) = 1$, i.e. $\psi \in h(\psi)$. Hence we've reached our conclusion that $A \cap h(\varphi) \subseteq h(\psi)$.
" $\Leftarrow$ " Conversely, suppose that $A \cap h(\varphi) \subseteq h(\psi)$.
Let $w \in A$ and assume that Rww' and $V(\varphi, w') = 1$, so $w' \in h(\varphi)$.
Since A is closed, $w \in A$ and Rww', we have that $w' \in A$, so $w' \in A \cap h(\varphi) \subseteq h(\psi)$.
Thus, $V(\psi, w') = 1$, so we proved that $w \in h(\varphi \to \psi)$ which gives evidence that $A \subseteq h(\varphi \to \psi)$.

Applying Lemma 3.6.4 to (*), we deduce:

$$(**)\ A \subseteq h(\varphi \to \psi) \text{ iff } A \subseteq h(\varphi) \to h(\psi) \text{ for any } A \in \mathcal{H}.$$

We prove finally $h(\varphi \to \psi) = h(\varphi) \to h(\psi)$ by double inclusion:

"$\subseteq$" Let $w \in h(\varphi \to \psi)$. We have to prove that there exists a closed subset A, such that $A \subseteq W \setminus h(\varphi) \cup h(\psi)$ and $w \in A$. Applying (**) to $h(\varphi \to \psi)$, we conclude that we can take the witness to be just $h(\varphi \to \psi)$ and our goal is finished.

"$\supseteq$" Let $w \in h(\varphi) \to h(\psi)$. Thus, there exists a closed subset A such that $A \subseteq W \setminus h(\varphi) \cup h(\psi)$ and $w \in A$. Applying (**) to A, we get that $A \subseteq h(\varphi \to \psi)$ and, since $w \in A$, we can conclude that $w \in h(\varphi \to \psi)$, which is what we needed to prove.

□

Proposition 3.6.7. *For any formula φ, $\mathcal{M} \vDash \varphi$ iff $\mathcal{H}_{\mathcal{M}} \vDash_{alg} \varphi$.*

Proof. " $\Rightarrow$ ": Assume that $\mathcal{M} \vDash \varphi$, that is, for any $w \in W$, $V(\varphi, w) = 1$. Hence, we have that $h(\varphi) = W = 1$, which is exactly what we needed to prove.
" $\Leftarrow$ ": If $\mathcal{H}_{\mathcal{M}} \vDash_{alg} \varphi$, we have that $h(\varphi) = 1 = W$, so for any $w \in W$, $V(\varphi, w) = 1$. Thus, we conclude that $\mathcal{M} \vDash \varphi$. □

3.6.2 From algebraic models to Kripke models

Let $\mathcal{H} = (H, \overline{h})$ be an algebraic model.
Then $(\mathcal{P}(H), \subseteq)$ is an Intuitionistic Kripke frame. Define the function $V_{\overline{h}} : Form \times \mathcal{P}(H) \to L_2$, such that:

$$V_{\overline{h}}(\varphi, F) = 1 \text{ iff } \overline{h}(\varphi) \in F, \text{ for any formula } \varphi \text{ and filter } F \in \mathcal{P}(H).$$

Proposition 3.6.8. *$\mathcal{M}_{\mathcal{H}} = (\mathcal{P}(H), \subseteq, V_{\overline{h}})$ is a Kripke model.*

Proof. Let φ, ψ be formulas and F be a prime filter of H. Then:

(i) $V_{\overline{h}}(v, F) = 1$ iff $\overline{h}(v) \in F$

(ii) $V_{\overline{h}}(\bot, F) = 1$ iff $\overline{h}(\bot) \in F$ iff $0 \in F$, which is false, since F is a proper filter

(iii) "$\wedge$":

$$\begin{aligned} V_{\overline{h}}(\varphi \wedge \psi, F) = 1 \quad & \text{iff} \quad \overline{h}(\varphi \wedge \psi) \in F \\ & \text{iff} \quad \overline{h}(\varphi) \wedge \overline{h}(\psi) \in F \\ & \text{iff} \quad \overline{h}(\varphi) \in F \text{ and } \overline{h}(\psi) \in F, \text{ by Definition 3.3.3(i, ii)} \\ & \qquad \text{and by } h(\varphi) \wedge h(\psi) \leq h(\varphi), h(\psi) \\ & \text{iff} \quad V_{\overline{h}}(\varphi, F) = 1 \text{ and } V_{\overline{h}}(\psi, F) = 1 \end{aligned}$$

(iv) "$\vee$":

$$\begin{aligned} V_{\overline{h}}(\varphi \vee \psi, F) = 1 \quad & \text{iff} \quad \overline{h}(\varphi \vee \psi) \in F \\ & \text{iff} \quad \overline{h}(\varphi) \vee \overline{h}(\psi) \in F \\ & \text{iff} \quad \overline{h}(\varphi) \in F \text{ or } \overline{h}(\psi) \in F, \text{ by } F \text{ prime and by Definition 3.3.3(ii)} \\ & \qquad \text{and by } \overline{h}(\varphi), \overline{h}(\psi) \leq \overline{h}(\varphi) \vee \overline{h}(\psi) \\ & \text{iff} \quad V_{\overline{h}}(\varphi, F) = 1 \text{ or } V_{\overline{h}}(\psi, F) = 1 \end{aligned}$$

(v) "$\rightarrow$":

We have to prove that the following two conditions are equivalent:

(a) $V_{\overline{h}}(\varphi \to \psi, F) = 1$

(b) For any $F' \in \mathcal{K}$, if $F \subseteq F'$ and $V_{\overline{h}}(\varphi, F') = 1$, then $V_{\overline{h}}(\psi, F') = 1$.

$(a) \Rightarrow (b)$:
Assume that $V_{\overline{h}}(\varphi \to \psi, F) = 1$, so $\overline{h}(\varphi \to \psi) \in F$.
Let $F' \in \mathcal{K}$ be such that $F \subseteq F'$ and $V_{\overline{h}}(\varphi, F') = 1$.
We have that $\overline{h}(\varphi) \to \overline{h}(\psi) = \overline{h}(\varphi \to \psi) \in F'$ and $\overline{h}(\varphi) \in F'$ and, by Proposition 3.3.6, we get that F' is a deductive system, hence we can derive that $\overline{h}(\psi) \in F'$, i.e. $V_{\overline{h}}(\psi, F') = 1$.

$(b) \Rightarrow (a)$:
Now suppose $V_{\overline{h}}(\varphi \to \psi, F) \neq 1$, that is $\overline{h}(\varphi) \to \overline{h}(\psi) = \overline{h}(\varphi \to \psi) \notin F$.
By Lemma 3.3.10, we have that $\overline{h}(\psi) \notin [F \cup \{\overline{h}(\varphi)\})$ and, by Proposition 3.3.14, there is a prime filter F', such that $[F \cup \{\overline{h}(\varphi)\}) \subseteq F'$ and $\overline{h}(\psi) \notin F'$.
Since $F \subseteq [F \cup \{\overline{h}(\varphi)\})$, we found a filter $F' \in \mathcal{K}$, such that $F \subseteq F'$, $V_{\overline{h}}(\varphi, F') = 1$ and $V_{\overline{h}}(\psi, F') = 0$, which contradicts our hypothesis.

□

Proposition 3.6.9. *For any formula φ, $\mathcal{H} \vDash_{alg} \varphi$ iff $\mathcal{M}_{\mathcal{H}} \vDash \varphi$.*

Proof. " $\Rightarrow$ ": Assume that $\mathcal{H} \vDash_{alg} \varphi$, so $\overline{h}(\varphi) = 1$. Let F be a prime filter of H. We have to show that $V_{\overline{h}}(\varphi, F) = 1$, which reduces to $1 \in F$. This is true, by Lemma 3.3.4.
" $\Leftarrow$ ": Assume now that for any $F \in \mathcal{P}(H)$, we have that $\overline{h}(\varphi) \in F$. That is, $\overline{h}(\varphi) \in \bigcap_{P \in \mathcal{P}(H)} P$. By Corrolary 3.3.17, we get that $\overline{h}(\varphi) = 1$, which is exactly what we needed to prove. □

3.6.3 Equivalence between Kripke and algebraic validity

We prove the equivalence between the notions of Kripke and algebraic validity.

Theorem 3.6.10. *For any formula φ, the following are equivalent:*

(i) $\vDash \varphi$

(ii) $\vDash_{alg} \varphi$

Proof. "$\Rightarrow$": Since $\vDash \varphi$, we have that $\mathcal{M} \vDash \varphi$, for any Kripke model $\mathcal{M}$. In particular, let $\mathcal{H} = (H, \overline{h})$ be an algebraic model. We have that $\mathcal{M}_{\mathcal{H}} \vDash \varphi$ and, by Propostion 3.6.9, this is equivalent to $\mathcal{H} \vDash_{alg} \varphi$, which concludes our goal.
"$\Leftarrow$": Now we have that for any algebraic model $\mathcal{H} = (H, \overline{h})$, $\mathcal{H} \vDash_{alg} \varphi$. Instantiating with the algebraic model from Lemma 3.6.6, we obtain that $\mathcal{H}_{\mathcal{M}} \vDash_{alg} \varphi$. Finally, applying Proposition 3.6.7, we get the conclusion. □

Chapter 4

Lean formalization

In the sequel, we present the Lean formalization of the previous definitions and results. The full code is available at [16].

4.1 Lean overview

Lean is a functional programming language and interactive theorem prover [2], which was launched in 2013 at Microsoft Research. The founder and principal developer of the project is Leonardo de Moura. Since its first releases, Lean has gained an excellent reputation among theorem provers, more and more areas of mathematics being formalized by means of it. All of these formalizations are available in the Mathlib library, which is maintained by the Lean community.

The underlying theory of Lean is based on a version of dependent type theory, known as the calculus of inductive constructions [1]. Thus, type-checking is the mechanism which assists the user in their approach to prove mathematical statements, either by directly constructing proof terms or by using Lean's so-called tactic-mode.

In the context of this thesis, an worth-mentioning aspect is that Lean relies on constructive reasoning, at a meta level. However, one can use classical logic mechanisms as well. Our formalization makes use of this type of reasoning when performing proofs by reductio ad absurdum, at the declaration of noncomputable functions or instances (this will be explicitly emphasized in the presentation) and also along with the use of the axiom of choice.

4.2 Language and syntax

4.2.1 Language

We start by formalizing the language of Intuitionistic Propositional Logic in Lean.
In the file *Formula.lean*, we define the propositional variables as a structure over $\mathbb{N}$.
Structures (or records) are used to define non-recursive inductive data types, containing only one constructor. And this is also the case here: we want the Var type to be a wrapper over the set of natural numbers, so we formalize it as a structure with a single field, called *val*, which specifies the index of the variable:

```
structure Var where
    val : Nat
```

Worth-mentioning is also the fact that the constructor of the above structure is named *mk* by default, since we didn't provide a name for it.

Formulas are defined as an inductive type, in which each constructor corresponds to a way of building objects of type $Formula$: the first non-recursive case uses the above defined structure type and simply encapsulates it in a $Formula$ term; then, the *bottom* constructor is meant to construct the false, whilst the following three constructors are functions which request as arguments two terms of type $Formula$, and output a new term of this type, corresponding to the connective:

```
inductive Formula where
| var : Var → Formula
| bottom : Formula
| and : Formula → Formula → Formula
| or : Formula → Formula → Formula
| implication : Formula → Formula → Formula
```

Next, we introduce some standard notations and define the derived connectives for negation and equivalence, as well as the truth:

```
notation "⊥" => bottom

infixl:60 " ∧∧ " => and

infixl:60 " ∨∨ " => or

infixr:50 (priority := high) " ⇒ " => implication
```

```
def equivalence (φ ψ : Formula) := (φ ⇒ ψ) ∧∧ (ψ ⇒ φ)
infix:40 " ⇔ " => equivalence

def negation (φ : Formula) : Formula := φ ⇒ ⊥
prefix:70 " ∼ " => negation

def top : Formula := ∼⊥
notation " ⊤ " => top
```

To declare a symbol denoting false, we use *notation*, which is the most flexible command for declaring operation notations. This is suitable for notations of functions with zero up to unlimited number of arguments. For binary operations (conjunction, disjunction, implication, equivalence), we use the *infix* command and its variants with the $-l$ or $-r$ suffices, to specify the associativity of the operator. The precedence is given by the value right after the command - the greater this value, the tighter the operator binds. Finally, to define a notation for the unary negation operation, we make use of the *prefix* command.

4.2.2 Proof system

In the file *Syntax.lean*, we define the proof system, using again an inductive type, with constructors for each axiom and deduction rule. Notice the curly brackets around the constructors arguments. They specify the fact that those arguments are implicit, i.e. Lean will try to infer them from the context, unless we precede the name of the function by @, when we call it, in order to mention the arguments explicitly.

```
inductive Proof (Γ : Set Formula) : Formula → Type where
| premise {φ} : φ ∈ Γ → Proof Γ φ
| contractionDisj {φ} : Proof Γ (φ ∨∨ φ ⇒ φ)
| contractionConj {φ} : Proof Γ (φ ⇒ φ ∧∧ φ)
| weakeningDisj {φ ψ} : Proof Γ (φ ⇒ φ ∨∨ ψ)
| weakeningConj {φ ψ} : Proof Γ (φ ∧∧ ψ ⇒ φ)
| permutationDisj {φ ψ} : Proof Γ (φ ∨∨ ψ ⇒ ψ ∨∨ φ)
| permutationConj {φ ψ} : Proof Γ (φ ∧∧ ψ ⇒ ψ ∧∧ φ)
| exfalso {φ} : Proof Γ (⊥ ⇒ φ)
| modusPonens {φ ψ} : Proof Γ φ → Proof Γ (φ ⇒ ψ) → Proof Γ ψ
| syllogism {φ ψ χ} : Proof Γ(φ ⇒ ψ) → Proof Γ(ψ ⇒ χ) → Proof Γ(φ ⇒ χ)
| exportation {φ ψ χ} : Proof Γ (φ ∧∧ ψ ⇒ χ) → Proof Γ (φ ⇒ ψ ⇒ χ)
| importation {φ ψ χ} : Proof Γ (φ ⇒ ψ ⇒ χ) → Proof Γ (φ ∧∧ ψ ⇒ χ)
| expansion {φ ψ χ} : Proof Γ (φ ⇒ ψ) → Proof Γ (χ ∨∨ φ ⇒ χ ∨∨ ψ)
```

And we declare the specific notation for Γ-theorems:

```
infix:25 " ⊢ " => Proof
```

4.2.3 Theorems and derived deduction rules

Having the proof system defined, we prove the theorems and derived deduction rules from Section 2.3. Below we provide some examples of how we transposed theorems and deduction rules into Lean functions. These correspond to (2.20), (2.34), (2.43), (2.53).

```
def conjIntroRule : Γ ⊢ φ → Γ ⊢ ψ → Γ ⊢ φ ∧∧ ψ :=
    fun p1 p2 => modusPonens p2 (modusPonens p1 conjIntro)

def andImplDistrib : Γ ⊢ φ ⇒ ψ → Γ ⊢ χ ⇒ γ → Γ ⊢ φ ∧∧ χ ⇒ ψ ∧∧ γ :=
    fun p1 p2 => conjImplIntroRule (extraPremiseConjIntroLeft1 p1)
    (extraPremiseConjIntroLeft2 p2)

def implExtraHypRev : Γ ⊢ φ ⇒ ψ → Γ ⊢ (ψ ⇒ χ) ⇒ (φ ⇒ χ) := fun p =>
    exportation (conjImplComm (syllogism (andImplDistrib p implSelf)
    modusPonensAndTh2))

def exportationTh : Γ ⊢ (φ ∧∧ ψ ⇒ χ) ⇒ φ ⇒ (ψ ⇒ χ) :=
    exportation (exportation (modusPonensExtraHyp (conjImplIntroRule
    andElimLeftRight conjElimRight) andElimLeftLeft))
```

Next, we present the formalization of Proposition 2.2.5(i). We perform structural induction on Δ-theorems and this can be accomplished by using the *induction* tactic.

```
lemma subset_proof : Δ ⊆ Γ → Δ ⊢ φ → Γ ⊢ φ :=
  by
    intros Hsubseteq Hdelta
    induction Hdelta with
    | premise Hvp => exact (premise (Set.mem_of_mem_of_subset Hvp Hsubseteq))
    | contractionDisj => exact contractionDisj
    | contractionConj => exact contractionConj
    | weakeningDisj => exact weakeningDisj
    | weakeningConj => exact weakeningConj
    | permutationDisj => exact permutationDisj
    | permutationConj => exact permutationConj
    | exfalso => exact exfalso
```

```
  | modusPonens _ _ ih1 ih2 => exact (modusPonens ih1 ih2)
  | syllogism _ _ ih1 ih2 => exact (syllogism ih1 ih2)
  | exportation _ ih => exact (exportation ih)
  | importation _ ih => exact (importation ih)
  | expansion _ ih => exact (expansion ih)
```

The only non-trivial case here is the premise one, for which we used the Mathlib theorem *Set.mem_of_mem_of_subset*:

```
theorem mem_of_mem_of_subset {x : α} {s t : Set α} (hx : x ∈ s) (h : s ⊆ t):
    x ∈ t := h hx.
```

Then, for the axioms cases, there is nothing to prove, and for the deduction rules ones, we simply apply the induction hypothesis to get the conclusion.

To prove Proposition 2.2.9, we define the following function which outputs the set of the premises that were used in a Γ-proof:

```
noncomputable instance {φ ψ : Formula} : Decidable (φ = ψ) :=
    @default _ (Classical.decidableInhabited _)

noncomputable def usedPremises {φ : Formula} : Proof Γ φ → Finset Formula
| premise Hvp => {φ}
| contractionDisj | contractionConj | weakeningDisj | weakeningConj
  | permutationDisj | permutationConj | exfalso => ∅
| modusPonens p1 p2 | syllogism p1 p2 => usedPremises p1 ∪ usedPremises p2
| exportation p | importation p | expansion p => usedPremises p
```

Notice the instance preceding the above definition and the *noncomputable* keyword in front of them. We need the equality operation on formulas to be decidable, to perform the union operation on the third case of the *usedPremises* function. Because of the fact that the instance depends on the axiom *Classical.decidableInhabited*, both the instance itself and the definition which synthesizes it have to be declared *noncomputable*.

Then, we prove by a straightforward induction that φ can be proved from the above obtained set. We only mention the statement of this lemma here:

```
noncomputable def toFinitePremises {φ : Formula} (p : Proof Γ φ) :
    Proof (@usedPremises Γ φ p).toSet φ
```

And finally, we are able to prove Proposition 2.2.9, as follows:

```
lemma finset_proof (p : Proof Γ φ) :
  ∃ (Ω : Finset Formula), Ω.toSet ⊆ Γ /\ Nonempty (Ω.toSet ⊢ φ) :=
  by
    exists usedPremises p
    apply And.intro
    · induction p with
    | premise Hvp => unfold usedPremises; simp; assumption
    | contractionDisj | contractionConj | weakeningDisj | weakeningConj
        | permutationDisj | permutationConj | exfalso =>
        unfold usedPremises; simp
    | modusPonens p1 p2 ih1 ih2 | syllogism p1 p2 ih1 ih2 =>
        unfold usedPremises; simp; apply And.intro; assumption'
    | importation p ih | exportation p ih | expansion p ih =>
        unfold usedPremises
        assumption
    · apply Nonempty.intro
      apply toFinitePremises
```

4.2.4 Deduction theorem

The proof of the deduction theorem mirrors the theoretical proof detailed in Section 2.3. In Lean, we implemented two separate definitions for the two implications of the theorem. We first give the statements of the necessary theorems and derived deduction rules:

```
def implSelf : Γ ⊢ φ ⇒ φ

def extraPremise : Γ ⊢ φ → Γ ⊢ ψ ⇒ φ

def modusPonensExtraHyp : Γ ⊢ φ ⇒ ψ → Γ ⊢ φ ⇒ (ψ ⇒ χ) → Γ ⊢ φ ⇒ χ

def syllogism_th : Γ ⊢ φ ⇒ (ψ ⇒ χ) → Γ ⊢ φ ⇒ (χ ⇒ γ) →
                   Γ ⊢ φ ⇒ (ψ ⇒ γ)

def imp_extra_hyp : Γ ⊢ φ ⇒ (ψ ⇒ (χ ⇒ γ)) → Γ ⊢ φ ⇒ (ψ ∧∧ χ ⇒ γ)

def exp_extra_hyp : Γ ⊢ φ  ⇒ (ψ ∧∧ χ ⇒ γ) → Γ ⊢ φ  ⇒ (ψ ⇒ (χ ⇒ γ))
```

```
def disjIntroAtHyp : Γ ⊢ φ ⇒ χ → Γ ⊢ ψ ⇒ χ → Γ ⊢ φ ∨∨ ψ ⇒ χ

def disjOfAndElimLeft : Γ ⊢ (φ ∧∧ ψ) ⇒ (φ ∨∨ γ)

def implConclTrans : Γ ⊢ φ ⇒ (ψ ⇒ χ) → Γ ⊢ χ ⇒ γ → Γ ⊢ φ ⇒ (ψ ⇒ γ)

def permuteHyps : Γ ⊢ φ ⇒ (ψ ⇒ χ) → Γ ⊢ ψ ⇒ (φ ⇒ χ)

def disjIntroRight : Γ ⊢ ψ ⇒ φ ∨∨ ψ
```

Again, we have to declare the following function definitions *noncomputable*, as they depend on the decidability of the membership relation on sets of formulas. Below is the necessary noncomputable instance, followed by the proofs of the two implications of the deduction theorem:

```
noncomputable instance {φ : Formula} {Γ : Set Formula} : Decidable (φ ∈ Γ) :=
  @default _ (Classical.decidableInhabited _)

noncomputable def deductionTheorem_left {φ ψ : Formula} (p : Γ ∪ {φ} ⊢ ψ) :
  Γ ⊢ φ ⇒ ψ :=
  match p with
  | premise Hvp =>
    if Hvpin : ψ ∈ Γ then
    extraPremise (premise Hvpin)
    else
    have Heq : ψ = φ :=
    by
        cases Hvp
        · contradiction
        · assumption
    by rw [Heq]; exact implSelf
  | contractionDisj => extraPremise contractionDisj
  | contractionConj => extraPremise contractionConj
  | weakeningDisj => extraPremise weakeningDisj
  | weakeningConj => extraPremise weakeningConj
  | permutationDisj => extraPremise permutationDisj
  | permutationConj => extraPremise permutationConj
  | exfalso => extraPremise exfalso
  | modusPonens p1 p2 => modusPonensExtraHyp (deductionTheorem_left p1)
                          (deductionTheorem_left p2)
  | syllogism p1 p2 => syllogism_th (deductionTheorem_left p1)
```

```
                          (deductionTheorem_left p2)
  | importation p => imp_extra_hyp (deductionTheorem_left p)
  | exportation p => exp_extra_hyp (deductionTheorem_left p)
  | expansion p =>
    permuteHyps (disjIntroAtHyp (exportation disjOfAndElimLeft)
                (implConclTrans (permuteHyps (deductionTheorem_left p))
                 disjIntroRight))
```

```
noncomputable def deductionTheorem_right {φ ψ : Formula} (p : Γ ⊢ φ ⇒ ψ) :
  Γ ∪ {φ} ⊢ ψ :=
    let p1 : φ ∈ Γ ∪ {φ} := by rw [Set.mem_union]; apply Or.inr
                                 apply Set.mem_singleton
    modusPonens (premise p1) (subset_proof (Set.subset_union_left Γ {φ}) p)
```

4.2.5 Utilitary lemmas

Next, we shall take a look at some utilitary lemmas, which we omitted in the theoretical section, as they state trivial results and have also immediate proofs. Most of these lemmas formalize the repeated application of already defined results, over a list of formulas, and we can easily prove them by induction on *List*. For example, the lemmas below provide chained application of the deduction theorem:

```
lemma deductionTheorem_left_ind {Γ : List Formula} {Δ : Set Formula}
  {φ : Formula} :
    Δ ∪ Γ.toFinset ⊢ φ → Δ ⊢ Γ.foldr Formula.implication φ :=
    by
        revert Δ
        induction Γ with
        | nil => intros Δ Hdelta
                 rw [List.toFinset_nil, Finset.coe_empty, Set.union_empty] at
                    Hdelta
                 assumption
        | cons h t ih => intros Δ Hdelta
                         have Haux : Δ ∪ {h} ∪ (List.toFinset t).toSet ⊢ φ :=
                          by
                            rw [List.toFinset_cons, Finset.insert_eq,
                                Finset.coe_union, Finset.coe_singleton,
                                ←Set.union_assoc] at Hdelta
                                assumption
                            exact (deductionTheorem_left (@ih (Δ ∪ {h}) Haux))
```

```
lemma deductionTheorem_right_ind {Γ : List Formula} {Δ : Set Formula}
  {φ : Formula}:
    Δ ⊢ Γ.foldr Formula.implication φ → Δ ∪ Γ.toFinset ⊢ φ :=
    by
        revert Δ
        induction Γ with
        | nil => intros Δ Hdelta
                 simp
                 assumption
        | cons h t ih => intros Δ Hdelta
                         let Hih := @ih (Δ ∪ {h}) (deductionTheorem_right
                                                    Hdelta)
                         rw [List.toFinset_cons, Finset.insert_eq,
                             Finset.coe_union, Finset.coe_singleton,
                             ←Set.union_assoc]
                         assumption
```

Notice the use of the *revert* tactic in the two proofs above. This tactic moves a hypothesis into the goal, yielding an implication. We made use of it in this case, in order to obtain a more general induction hypothesis, which holds for any set of formulas Δ. For brevity of exposition, we only give the statements of the other similar results proved:

```
lemma exportation_ind {Γ : List Formula} {Δ : Set Formula} {φ : Formula} :
    Δ ⊢ Γ.foldr Formula.and ⊤ ⇒ φ → Δ ⊢ Γ.foldr Formula.implication φ

lemma importation_ind {Γ : List Formula} {Δ : Set Formula} {φ : Formula} :
    Δ ⊢ Γ.foldr Formula.implication φ → Δ ⊢ Γ.foldr Formula.and ⊤ ⇒ φ
```

Other technical lemmas we proved arose from the conversions between *List* and *Finset*. The finset generated by the *List.toFinset* function is a permutation of the initial list, so we need some trivial results about finite conjunction and disjunction over a list being syntactically equivalent modulo permutations of the list. We give below the Mathlib definition of the *Perm* type, followed by the inductive proof of the conjunctive result, and only mention the statement of the disjunctive one, as the proofs are very similar.

```
inductive Perm : List α → List α → Prop
| nil : Perm [] []
| cons (x : α) {l₁ l₂ : List α} : Perm l₁ l₂ → Perm (x :: l₁) (x :: l₂)
| swap (x y : α) (l : List α) : Perm (y :: x :: l) (x :: y :: l)
| trans {l₁ l₂ l₃ : List α} : Perm l₁ l₂ → Perm l₂ l₃ → Perm l₁ l₃
```

```
lemma permutationConj_ind (l1 l2 : List Formula) (Hperm : l1 ~ l2) :
    Nonempty(∅ ⊢ List.foldr Formula.and ⊤ l1 ⇒ List.foldr Formula.and ⊤ l2):=
    by
        induction Hperm with
        | nil => apply Nonempty.intro; apply implSelf
        | @cons _ _ _ _ ihequiv => apply Nonempty.intro
                                   apply conjImplIntroRule weakeningConj
                                        (syllogism conjElimRight
                                        (Classical.choice ihequiv))
        | swap => apply Nonempty.intro
                  apply andAssocComm2
        | @trans _ _ _ _ _ ihequiv12 ihequiv23 => apply Nonempty.intro
                                                  apply syllogism
                                                  (Classical.choice
                                                  ihequiv12) (Classical.choice
                                                  ihequiv23)

lemma permutationDisj_ind (l1 l2 : List Formula) (Hperm : l1 ~ l2) :
    Nonempty (∅ ⊢ List.foldr Formula.or ⊥ l1 ⇒ List.foldr Formula.or ⊥ l2)
```

We consider the following auxiliary lemma worth-mentioning, because it makes use of another form of induction in Lean. Although *Finset* is not an inductive type, we can proceed by induction on a term of this type, using the Mathlib theorem *Finset.induction_on*, which requires two proofs of the predicate holding for the empty finset and for a nonempty finset, to conclude the predicate holds for any finset. Below is the predicate we want to prove:

```
def pfoldrAndUnion (Φ Ω : Finset Formula) :=
    Nonempty (∅ ⊢ List.foldr Formula.and (~⊥) (Φ ∪ Ω).toList ⇒
    List.foldr Formula.and (~⊥) Φ.toList ∧∧
    List.foldr Formula.and (~⊥) Ω.toList)
```

We give here only the statements of the two lemmas for the induction cases:

```
lemma foldrAndUnion_empty (Ω : Finset Formula) : pfoldrAndUnion ∅ Ω

lemma foldrAndUnion_insert (φ : Formula) (Φ Ω : Finset Formula)
    (Hnotin:φ ∉ Φ) (Hprev : pfoldrAndUnion Φ Ω) :
    pfoldrAndUnion (insert φ Φ) Ω
```

And this is finally the inductive proof:

```
lemma foldrAndUnion (Φ Ω : Finset Formula) : pfoldrAndUnion Φ Ω :=
  by
    induction Φ using Finset.induction_on with
    | empty => exact foldrAndUnion_empty Ω
    | @insert φ Φ Hnotin Hprev => exact foldrAndUnion_insert φ Φ Ω
                                  Hnotin Hprev
```

4.2.6 Disjunctive theories, consistent and complete pairs

Deductive-closure, consistency, disjunctiveness, disjunctive theories, consistent and complete pairs are defined in the file *Completeness.lean*, in the same manner as in the theoretical section of the thesis. However, a worth-mentioning detail here is the fact that, because of our design option of modeling proofs as a *Type* instead of *Prop*, we have to use the *Sum* and *Nonempty* types here.

```
def dedClosed {Γ : Set Formula} := ∀ (φ : Formula), Γ ⊢ φ → φ ∈ Γ

def consistent {Γ : Set Formula} := Γ ⊢ ⊥ → False

def disjunctive {Γ : Set Formula} :=
  ∀ (φ ψ : Formula), Γ ⊢ φ ∨∨ ψ → Sum (Γ ⊢ φ) (Γ ⊢ ψ)

def disjunctiveTheory {Γ : Set Formula} :=
  @dedClosed Γ /\ @consistent Γ /\ Nonempty (@disjunctive Γ)

def consistentPair {Γ Δ : Set Formula} :=
  ∀ (Φ Ω : Finset Formula), Φ.toSet ⊆ Γ → Ω.toSet ⊆ Δ →
  (∅ ⊢ Φ.toList.foldr Formula.and (∼⊥) ⇒ Ω.toList.foldr Formula.or ⊥ →
  False)

def completePair {Γ Δ : Set Formula} :=
  @consistentPair Γ Δ /\ ∀ (φ : Formula),(φ ∈ Γ /\ φ ∉ Δ) ∨ (φ ∈ Δ /\
  φ ∉ Γ)
```

Another technical detail is that, in the *consistentPair* definition, in order to represent finite conjunctions and disjunctions, we converted the finsets into lists, and made use of Lean's *foldr* function.

Below we give the statement of the lemma which formalizes Proposition 2.5.8, claiming that given a consistent pair, any formula can be added to one of the sets in the pair, preserving the consistency:

```
lemma add_preserves_cons :
    @consistentPair Γ Δ → ∀ (φ : Formula), @consistentPair ({φ} ∪ Γ) Δ ∨
                                            @consistentPair Γ ({φ} ∪ Δ)
```

For the proof of Lemma 2.5.9, we need to define the indexed family of formula-set pairs, thus:

```
def family (nf : Nat → Formula) (n : Nat) : Set Formula × Set Formula :=
    match n with
    | .zero => @add_formula_to_pair Γ Δ (nf 0)
    | .succ n => @add_formula_to_pair (family nf n).fst (family nf n).snd
                (nf (n + 1))
```

To have access to an enumeration of formulas, we pass as the first argument a function which assigns, to any natural number, a formula. Then, we inductively build the family, by adding the formulas to one of the sets in the pair, whilst preserving the consistency. Without loss of generality, we defined the function to add the formula to the first set in the pair, if possible:

```
def add_formula_to_pair (φ : Formula) : Set Formula × Set Formula :=
    if @consistentPair ({φ} ∪ Γ) Δ then (({φ} ∪ Γ), Δ)
    else (Γ, {φ} ∪ Δ)
```

The enumeration of formulas is not required to be bijective, a surjection from Nat to $Formula$ is sufficient in this case, as we don't have any restriction for adding the formulas only once. Classicaly, the existence of an injective function from a type α to a type β gives evidence that there is a surjection from β to α. Hence, in the file $Formula.lean$, we define an injective function from $Formula$ to Nat.

To construct the injection, we use Cantor's pairing function, which we multiply by two, for ease of formalization:

```
def pairing (x y : ℕ) := (x + y) * (x + y + 1) + 2 * x
```

Then, we associate a numerical identifier to any connective symbol and encode formulas into natural numbers by recursively applying the pairing function on the structure of the formula, as follows:

```
def encode_form : Formula → ℕ
| var v => pairing 0 (v.val + 1)
| bottom => 0
```

```
| φ ∧∧ ψ => pairing (pairing (encode_form φ) 1) (encode_form ψ)
| φ ∨∨ ψ => pairing (pairing (encode_form φ) 2) (encode_form ψ)
| φ ⇒ ψ => pairing (pairing (encode_form φ) 3) (encode_form ψ)
```

This way of dealing with the countability of *Formula* is one of the major differences between our approach and the one in [6]. After proving the injectivity of our encoding function, we are able to define an instance of *Countable* for our *Formula* type. The Mathlib definition of the *Countable* type-class is as follows:

```
class Countable (α : Sort u) : Prop where
    exists_injective_nat' : ∃ f : α → ℕ, Injective f
```

So we immediately define the *Countable* instance for *Formula*, based on the proof of the encoding's injectivity:

```
instance : Countable Formula := inject_Form.countable
```

Now, having the surjective enumeration at hand, we can get a step closer to the final construction of the complete pair in Lemma 2.5.9, by proving that any formula φ is contained in the family-pair with the index $fn(\varphi)$, where by fn we denote the injective encoding of formulas into natural numbers:

```
lemma vp_in_ΓiΔi (φ : Formula) (fn : Formula → Nat) (fn_inj : fn.Injective)
  (nf : Nat → Formula) (nf_inv : nf = fn.invFun) :
   φ ∈ (@family Γ Δ nf (fn φ)).fst \/ φ ∈ (@family Γ Δ nf (fn φ)).snd :=
    by
        have Hleftinv : ∀ (φ : Formula), nf (fn φ) = φ :=
            by intros φ; simp [nf_inv, fn.leftInverse_invFun fn_inj φ]
        conv =>
            congr
            repeat {lhs; rw [←Hleftinv φ]}
        exact nf_in_ΓiΔi nf (fn φ)
```

Notice the use of the *conv* tactic in the above proof. This allows targeted rewriting. In our case, we produce two subgoals, one for each member of the disjunction, by using the *congr* tactic and then by *lhs* we focus on the left hand-side of the goals, and perform the desired rewriting there. In Mathlib, the inverse of a function is noncomputably defined as follows:

```
noncomputable def invFun {α : Sort u} {β} [Nonempty α] (f : α → β) :
    β → α :=
    fun y ↦ if h : (∃ x, f x = y) then h.choose else Classical.arbitrary α
```

So this is why we can count on this inverse for any function, regardless of its bijectivity. Notice also that the injectivity of fn gives evidence of $invFun$ being the so-called $left-inverse$.

It is also crucial to prove that the pair-family we defined is increasing:

```
lemma increasing_family {nf : Nat → Formula} (i j : Nat) : i <= j →
    (@family Γ Δ nf i).fst ⊆ (@family Γ Δ nf j).fst /\
    (@family Γ Δ nf i).snd ⊆ (@family Γ Δ nf j).snd
```

Next, we define the component-wise union of the indexed pair-family:

```
def consistent_family_union (_ : @consistentPair Γ Δ) (nf : Nat → Formula) :
    Set Formula × Set Formula :=
    ({φ | ∃ i : Nat, φ ∈ (@family Γ Δ nf i).fst},
     {φ | ∃ i : Nat, φ ∈ (@family Γ Δ nf i).snd})
```

Using the increasing property, we show that for any consistent pair (Γ, Δ) contained component-wise in the union, we have that for any φ in the first set of the union-pair, and for any ψ in the second set of the union-pair, there is an index i such that $\varphi \in \Gamma_i$ and $\psi \in \Delta_i$.

```
lemma finset_subset_union_mem_local {Hcons : @consistentPair Γ Δ}
  (nf : Nat → Formula) (fn : Formula → Nat) (fn_inj : fn.Injective)
  (nf_inv : nf = fn.invFun) (Φ Ω : Finset Formula)
  (Hincl1 : Φ.toSet ⊆ (@consistent_family_union Γ Δ Hcons nf).fst)
  (Hincl2 : Ω.toSet ⊆ (@consistent_family_union Γ Δ Hcons nf).snd) :
  ∃ (i : Nat), ((∀ (φ : Formula), φ ∈ Φ.toSet → φ ∈ (@family Γ Δ nf i).fst)
              /\ ∀ (φ : Formula), φ ∈ Ω.toSet → φ ∈ (@family Γ Δ nf
                                                     i).snd)
```

Finally, having all these auxiliary results proved, we present the main aspects of the Lean proof of Lemma 2.5.9. First of all, we make use of the $FormulaCountable$ instance, in order to get access to the $exists\_injective\_nat$ function, which provides us with two additional hypotheses for the injection. These will be necessary later in the proof, when calling the $vp\_in\_\Gamma_i\Delta_i$ function.

```
lemma consistent_incl_complete :
  @consistentPair Γ Δ → (∃ (Γ' Δ' : Set Formula), Γ ⊆ Γ' ∧ Δ ⊆ Δ' ∧
  @completePair Γ' Δ') :=
    by
      intros Hcons
      let ⟨fn, fn_inj⟩ := exists_injective_nat Formula
```

```
        let nf := fn.invFun
```

Of course, we use the union of the family and prove that it is a valid witness for our statetment:

```
exists (@consistent_family_union Γ Δ Hcons nf).fst,
       (@consistent_family_union Γ Δ Hcons nf).snd
```

The most interesting part is proving the union-pair is a partition of the set of all formulas. For this, we reason by cases on $vp\_in\_\Gamma_i\Delta_i$. The two cases are analogous, so we present only one of them. Notice the essential use of the *increasing* property here:

```
apply And.intro
· exists (fn φ)
· intro x
  by_cases Horder : (fn φ) ≤ x
  · let Hdisj := consistent_disj (family_cons Hcons nf x)
    rw [Set.disjoint_left] at Hdisj
    exact (Hdisj (Set.mem_of_subset_of_mem (And.left (increasing_family (fn φ)
        x Horder)) Hgamma))
  · intro Hsndx
    simp only [not_le] at Horder
    let Hsnd := Set.mem_of_mem_of_subset Hsndx
                (And.right (increasing_family x (fn φ) (Nat.le_of_lt Horder)))
    let Hdisj := consistent_disj (family_cons Hcons nf (fn φ))
    rw [Set.disjoint_left] at Hdisj
    let Hdisj := Hdisj Hgamma
    contradiction
```

4.3 Kripke semantics

In the file *Semantics.lean*, we define the semantics of the language. The first definition we need is, of course, that of a Kripke model:

```
structure KripkeModel (W : Type) where
    R : W → W → Prop
    V : Var → W → Prop
    refl (w : W) : R w w
    trans (w1 w2 w3 : W) : R w1 w2 → R w2 w3 → R w1 w3
    monotonicity (v : Var) (w1 w2 : W) : R w1 w2 → V v w1 → V v w2
```

We define the Kripke model as a parameterized structure, where the parameter W represents the space of worlds. Thus, the worlds of a model are in Lean terms of type W. The first two fields of the structure mirror the elements from Definition 3.1.2: the accessibility relation R is a binary relation over terms of type W, V is the valuation function, which takes two arguments - a variable and an inhabitant of type W. Then, the last three fields are meant to formalize the properties of the relation R (reflexivity and transitivity) and the monotonicity of the valuation.

The extended valuation function (on formulas) is defined as follows:

```
def val {W : Type} (M : KripkeModel W) (w : W) : Formula → Prop
| Formula.var p => M.V p w
| ⊥ => False
| φ ∧∧ ψ => val M w φ /\ val M w ψ
| φ ∨∨ ψ => val M w φ \/ val M w ψ
| φ ⇒ ψ => ∀ (w' : W), M.R w w' /\ val M w' φ → val M w' ψ
```

And we can easily prove that the negation connective is interpreted by the valuation function in the same way as stated in Remark 3.1.4. The *simp* tactic suffices here, as the proof follows simply by the implication case in the definition of the valuation function.

```
lemma val_neg {W : Type} (M : KripkeModel W) (w : W) (φ : Formula) :
    val M w (∼φ) ↔ ∀ (w' : W), M.R w w' → ¬ val M w' φ := by simp [val]
```

The notions of truth and validity of a formula, forcing, local semantic consequence and set forcing are formalized according to Definitions 3.1.5, 3.1.7, 3.1.8 and 3.1.11, as follows:

```
def true_in_world {W : Type} (M : KripkeModel W) (w : W) (φ : Formula): Prop :=
    val M w φ

def valid_in_model {W : Type} (M : KripkeModel W) (φ : Formula) : Prop :=
    ∀ (w : W), val M w Φ

def valid (φ : Formula) : Prop :=
    ∀ (W : Type) (M : KripkeModel W), valid_in_model M Φ

def model_sat_set {W : Type}(M : KripkeModel W)(Γ : Set Formula)(w : W):Prop:=
    ∀ (φ : Formula), φ ∈ Γ → val M w φ

def sem_conseq (Γ : Set Formula) (φ : Formula) : Prop :=
    ∀ (W : Type) (M : KripkeModel W) (w : W), model_sat_set M Γ w → val M w φ
infix:50 " ⊨ " => sem_conseq
```

```
def set_forces_set (Γ Δ : Set Formula) : Prop :=
    ∀ (φ : Formula), φ ∈ Δ → Γ ⊨ φ
```

Below, we present the Lean formalizations of the basic properties in Lemma 3.1.13:

```
lemma elem_sem_conseq (Γ : Set Formula) (φ : Formula) : φ ∈ Γ → Γ ⊨ φ :=
    by { intros Helem _ _ _  Ha; exact (Ha φ Helem) }

lemma subseteq_sem_conseq (Γ Δ : Set Formula) (φ : Formula) :
  Δ ⊆ Γ → Δ ⊨ φ → Γ ⊨ φ :=
    by { intros Hsubseteq Hdelta _ _ _ Ha; apply Hdelta
         intros _ Ha'; apply Ha;
         apply Set.mem_of_mem_of_subset Ha' Hsubseteq }

lemma valid_sem_conseq (Γ : Set Formula) (φ : Formula) : valid φ → Γ ⊨ φ :=
    by { intros Hvalid _ _ _ _; apply Hvalid; }

lemma set_conseq (Γ Δ : Set Formula) (φ : Formula) :
  set_forces_set Γ Δ → Δ ⊨ φ → Γ ⊨ φ :=
    by { simp [sem_conseq, model_sat_set]
         intros Hsetval Hdelta _ M w Ha; intros
         apply Hsetval; assumption' }

lemma set_conseq_iff (Γ Δ : Set Formula) (φ : Formula) :
  set_forces_set Γ Δ → set_forces_set Δ Γ → (Δ ⊨ φ ↔ Γ ⊨ φ) :=
    by {
        intros Hsetvalgd Hsetvaldg; apply Iff.intro
        · exact set_conseq _ _ _ Hsetvalgd
        · exact set_conseq _ _ _ Hsetvaldg
        }
```

A semantic result which will be essential for the proof of Soundness theorem is the monotonicity of valuations. We prove it by the induction principle on *Formula*:

```
lemma monotonicity_val (W : Type) (M : KripkeModel W) (w1 w2 : W)
  (φ : Formula) :
  M.R w1 w2 → val M w1 φ → val M w2 φ :=
  by
    intros Hw1w2 Hval
    induction φ with
    | var p => apply M.monotonicity p w1 w2 Hw1w2 Hval
    | bottom => simp [val] at Hval
```

```
| and _ _ ih1 ih2 => apply And.intro
                    · apply ih1 (And.left Hval)
                    · apply ih2 (And.right Hval)
| or _ _ ih1 ih2 => rcases Hval with Hvalpsi | Hvalchi
                    · apply Or.inl; apply ih1; assumption
                    · apply Or.inr; apply ih2; assumption
| implication => simp [val]
                 simp [val] at Hval
                 intros w3 Hw2w3 Hvalw3psi
                 have Hw1w3 : M.R w1 w3 := (M.trans w1 w2 w3 Hw1w2 Hw2w3)
                 apply Hval _  Hw1w3 Hvalw3psi
```

The *var* case follows directly by the monotonicity of the valuation on variables. In the *bottom* case, there is actually nothing to prove, since there is no world of any Kripke model such that the false is true at that world. Then, for the conjunction and disjunction cases, all we have to do is applying the induction hypothesis that match our assumptions. Finally, the implication case does not make use of the induction hypothesis at all, but of the transitivity of the accessibility relation.

4.4 Kripke completeness theorem

4.4.1 Soundness

The main auxiliary result we need for Soundness, apart of Lemma 3.1.14 (presented in the previous section), is Lemma 3.1.12, which is formalized in the file *Soundness.lean*. Below we give the proof of the auxiliary lemma stating that any axiom is valid:

```
lemma axioms_valid (φ : Formula) (ax : Axiom φ) : valid φ :=
  by
    intros W M w
    rcases ax
    · intros w' Hww'val
      apply Or.elim (And.right Hww'val)
      repeat {intros; assumption}
    · intros w' Hww'val
      apply And.intro (And.right Hww'val)
      apply And.right Hww'val
    · intros w' Hww'val
      apply Or.inl (And.right Hww'val)
    · intros w' Hww'val
      apply And.left (And.right Hww'val)
```

```
· intros w' Hww'val
  rcases And.right Hww'val with Hvp | Hpsi
· apply Or.inr Hvp
· apply Or.inl Hpsi
· intros w' Hww'val
  apply And.intro (And.right (And.right Hww'val))
                  (And.left (And.right Hww'val))
· intros w' Hww'val
  exfalso
  apply And.right Hww'val
```

All the cases of the above proof follow easily by the definition of the valuation function. Notice the use of the $exfalso$ tactic on the last goal: this works because we have a false assumption, specifically the fact that there exists a world w' such that $\bot$ is true at w'.

Now we can move to the proof of Soundness theorem, which stays very close to its pen-and-paper version. For the *premise* case, we use the *elem_sem_conseq* lemma from the previous section and for the axiom cases, we apply the above auxiliary result. Notice also the use of the monotonicity of valuation, in the *exportation* case.

```
theorem soundness (Γ : Set Formula) (φ : Formula) : Γ ⊢ φ → Γ ⊨ φ :=
  by
    intros Hlemma
    induction Hlemma with
    | premise Hvp => apply elem_sem_conseq; assumption
    | contractionDisj | contractionConj | weakeningDisj | weakeningConj
    | permutationDisj | permutationConj | exfalso =>
      apply valid_sem_conseq; apply axioms_valid; constructor
    | modusPonens _ _ ih1 ih2 => simp [sem_conseq, val] at ih2
                                 intros _ M _ _
                                 apply ih2
                                 · assumption
                                 · apply M.refl
                                 · apply ih1; assumption
    | syllogism H1 H2 ih1 ih2 => simp [sem_conseq, val] at *
                                 intros _ _ _ Hmodelval _ _ _
                                 apply ih2; assumption'
                                 apply ih1; apply Hmodelval; assumption'
    | exportation H ih => simp [sem_conseq, val] at *
                          intros _ M w1 Hmodelval w2 Hw1w2 _ w3 Hw2w3 _
```

```
                        apply ih; assumption'
                        · apply M.trans w1 w2 w3 Hw1w2 Hw2w3
                        · apply monotonicity_val; assumption'
  | importation H ih => simp [sem_conseq, val] at *
                        intros _ M w1 Hmodelval w2 Hw1w2 _ _
                        apply ih; assumption'; apply M.refl
  | expansion H ih => simp [sem_conseq, val] at *
                      intros _ M w1 Hmodelval w2 Hw1w2 Hvalor
                      rcases Hvalor with Hvalchi | Hvalvp
                      · apply Or.inl; assumption
                      · apply Or.inr; apply ih; assumption'
```

4.4.2 Completeness

The definitions and proofs from this section can be found in the file *Completeness.lean*.
We first formalize the definition of the canonical model, parameterizing the world space in the definition of KripkeModel presented earlier in this section. For the refl, trans, and monotonicity fields of the structure, we have to pass proofs of the set inclusion relation satisfying these properties. These proofs are easily completed, using the corresponding Mathlib theorems.

```
def canonicalModel : KripkeModel (setDisjTh) :=
{
    R := fun (Γ Δ) => Γ.1 ⊆ Δ.1,
    V := fun (v Γ) => Formula.var v ∈ Γ.1,
    refl := fun (Γ) => Set.Subset.rfl
    trans := fun (Γ Δ Φ) => Set.Subset.trans
    monotonicity := fun (v Γ Δ) => by intros; apply Set.mem_of_mem_of_subset
                                        assumption'
}
```

Before moving on to the proof of the completeness theorem, we shall take a look at the main formalization steps of Lemma 3.2.3. It is worth mentioning that the two implications in this lemma cannot be formalized as independent lemmas in Lean, because of the *implication* case, where the proof of the left implication depends on the right implication in the induction hypothesis, and vice versa.
Of course, we proceed by induction on formulas, and use the *revert* tactic, in order to obtain a more general induction hypothesis:

```
lemma main_sem_lemma (Γ : setDisjTh) (φ : Formula) :
  val canonicalModel Γ φ ↔ φ ∈ Γ.1 :=
    by
      revert Γ
      induction φ with
```

The induction cases corresponding to the *var* and *bottom* constructors are trivial, as usual. We present below the formalization of the *and* case, and the *or* case is very similar.

```
| and ψ χ ih1 ih2 => intros Γ
                     let Γth := Γ
                     rcases Γ with ⟨_, ⟨Hded, _⟩⟩
                     apply Iff.intro
                     · intro Hval
                       let Hpsi := Proof.premise ((ih1 Γth).1 Hval.1)
                       let Hchi := Proof.premise ((ih2 Γth).1 Hval.2)
                       apply Hded (ψ ∧∧ χ) (Proof.conjIntroRule Hpsi Hchi)
                     · intro Helem
                       let Hpsi := Proof.modusPonens (Proof.premise Helem)
                                     Proof.weakeningConj
                       let Hchi := Proof.modusPonens (Proof.premise Helem)
                                     Proof.conjElimRight
                       apply And.intro ((ih1 Γth).2 (Hded ψ Hpsi))
                          ((ih2 Γth).2 (Hded χ Hchi))
```

Notice the fact that we make a copy of the Γ disjunctive theory, before applying the *rcases* tactic to structurally decompose it, because we need the original Γ as a named assumption, which can be passed as an argument to the induction hypothesis. Also, since deductive-closure is the only disjunctive theory property we make use of in this case of the proof, we use placeholders for the other components, so that they are introduced as anonymous assumptions.

Now we can move on to the most interesting case of this inductive proof, the *implication* one. For the right implication, the formalization is as follows:

```
intro Helem
simp [val]
intros Φ Φdisj Hincl Hpsi1
let Hdisjthphi : setDisjTh := ⟨Φ, Φdisj⟩
rcases Φdisj with ⟨Hded', ⟨Hcons', Hdisj'⟩⟩
· by_cases ψ ∈ Φ
```

```
· have Hchi : Φ ⊢ χ :=
    by apply Proof.modusPonens (Proof.premise h)
      (Proof.premise (Set.mem_of_mem_of_subset Helem Hincl))
  exact (ih2 Hdisjthphi).2 (Hded' χ Hchi)
· let Hih := (ih1 Hdisjthphi).1 Hpsi1
  contradiction
```

As in the pen-and-paper version of this case, we have to analyze two cases, depending on the membership of ψ to the disjunctive theory. If $\psi \in \Phi$, using the fact that Φ is closed under MODUS-PONENS, we get that $\chi \in \Phi$, hence we can apply the right implication of the induction hypothesis to conclude our goal. On the other hand, if $\psi \notin \Phi$, we have to apply the reverse implication of the induction hypothesis, to our assumption:

```
Hpsi1 : val canonicalModel { val := Φ, property := (_ : dedClosed ∧ consistent
                                            ∧ Nonempty disjunctive) } ψ
```

to obtain a contradiction.

As far as the other implication is concerned, we present here only the last main step of it, specifically, the proof of the fact that $\varphi = \psi \to \chi$ is not true at the world corresponding to the constructed witness. Thus, in the two *have* statements below, we prove that ψ is true at the world Φ, while χ is not true at this world. Notice the use of the two implications of the induction hypothesis, in the *have* statements.

```
have Hphipsi : val canonicalModel { val := Φ, property := Hdisjth' } ψ :=
  by
    have Haux : ψ ∈ Φ :=
      by
        rw [Set.union_subset_iff, Set.singleton_subset_iff] at Hincl1
        exact Hincl1.right
    exact (ih1 Hdisjthphi).2 Haux
have Hphinotchi : val canonicalModel { val := Φ, property := Hdisjth' } χ →
  False :=
  by
    by_cases val canonicalModel { val := Φ, property := Hdisjth' } χ
    · let Hih2 := (ih2 Hdisjthphi).1 h
      rcases Hcompl with ⟨_, Hvp⟩
      let Hvpchi := Hvp χ
      have Hchielem : χ ∈ Ω := by simp at Hincl2; assumption
      rcases Hvpchi with Hphi | Homega
      · rcases Hphi; contradiction
```

```
      · rcases Homega; contradiction
    · assumption
  let Hvalspecaux := Hvalspec Hphipsi
  contradiction
```

Finally, having all the auxiliary results proved, we can present the most important aspects of transposing the Completeness theorem (Theorem 3.2.4) in Lean.
First of all, the formalized statement:

```
theorem completeness {φ : Formula}{Γ : Set Formula}: Γ ⊨ φ ↔ Nonempty(Γ ⊢ φ)
```

Similarly to the last case of the previous lemma, we focus here on the last essential proof step, which consists of proving that Φ forces Γ, but φ is not true at Φ. And of course, this concludes our reductio ad absurdum, since the assumption that $\Gamma \vDash \varphi$ is contradicted.

```
let Hnotconseq : ¬val canonicalModel Hdisjthphi φ :=
  by
    by_cases (val canonicalModel Hdisjthphi φ)
    · exfalso
      let Hin := (main_sem_lemma Hdisjthphi φ).1 h
      rcases Hvp with Hphi | Homega
      · rcases Hphi; contradiction
      · rcases Homega
        have : φ ∈ Φ := Hin
        contradiction
    · assumption
have Hmodelset : model_sat_set canonicalModel Γ Hdisjthphi :=
  by
    intros vp Hvpin
    have Hvpinphi : vp ∈ Φ := by apply Set.mem_of_subset_of_mem Hincl1 Hvpin
    apply elem_sem_conseq Φ
    · assumption
    · intros vp Hvpin
      let Hphi : vp ∈ Φ := by assumption
      let Hmainsem := (main_sem_lemma Hdisjthphi vp).2 Hphi
      assumption
exfalso
let Haux := Hsem (@setDisjTh) canonicalModel Hdisjthphi Hmodelset
contradiction
```

4.5 Algebraic semantics and completeness theorem

4.5.1 Heyting algebras

Switching to the algebraic semantics section of the thesis, we start by formalizing the general definitions on Heyting algebras, from Section 3.3.1, in the file *HeytingAlgebraUtils.lean*. We consider a type α for which there is an instance of the Mathlib *HeytingAlgebra* class:

```
variable {α : Type u} [HeytingAlgebra α]
```

Then, we formalize the following main definitions, using the above α type-variable, to represent the domain of the Heyting algebra:

```
def filter (F : Set α) := (Set.Nonempty F) ∧ (∀ (x y : α), x ∈ F → y ∈ F →
                           x ⊓ y ∈ F) ∧ (∀ (x y : α), x ∈ F → x ≤ y → y∈F)

def deductive_system (F : Set α) := ⊤ ∈ F ∧ (∀ (x y : α), x ∈ F →
                                           x ⇒ y ∈ F → y ∈ F)

abbrev X_filters (X : Set α) := {F // filter F ∧ X ⊆ F}
def X_gen_filter (X : Set α) := {x | ∀ (F : X_filters X), x ∈ F.1}

def proper_filter (F : Set α) := filter F ∧ ⊥ ∉ F

def prime_filter {α : Type} [HeytingAlgebra α] (F : Set α) :=
    proper_filter F ∧ (∀ (x y : α), x ⊔ y ∈ F → x ∈ F ∨ y ∈ F)

def X_filters_not_cont_x (x : α) := {F | filter F ∧ x ∉ F}
```

The first main result of Section 3.3.1 we present here is Proposition 3.3.9, which provides a characterization of the filter generated by X. Its statement is formalized as follows:

```
lemma gen_filter_prop (X : Set α) :
    X_gen_filter X = {a | ∃ (l : List α), l.toFinset.toSet ⊆ X∧inf_list l≤a}
```

The proof is structured in three *have* statements - one for each of the conditions that have to be satisfied by the set in the right-hand side of the proposition's statement. We denote this set by S, in the *let* statement:

```
let S := {a | ∃ (l : List α), l.toFinset.toSet ⊆ X ∧ inf_list l ≤ a}
```

Firstly, we prove that S is a filter:

```
have HSfilter : filter S :=
  by
    apply And.intro
    · exists ⊤; exists []; simp
    · apply And.intro
      · intros x y Hxin Hyin
        rcases Hxin with ⟨l1, ⟨Hsubset1, Hinf1⟩⟩
        rcases Hyin with ⟨l2, ⟨Hsubset2, Hinf2⟩⟩
        exists l1 ++ l2
        simp
        apply And.intro
        · simp at Hsubset1; simp at Hsubset2
          exact And.intro Hsubset1 Hsubset2
        · rw [inf_list_concat]
          exact And.intro (le_trans inf_le_left Hinf1)
                          (le_trans inf_le_right Hinf2)
      · intros x y Hxin Hle
        rcases Hxin with ⟨l, ⟨Hsubset, Hinf⟩⟩
        exists l
        exact And.intro Hsubset (le_trans Hinf Hle)
```

The proof mirrors exactly its pen-and-paper version. However, notice the use of the *inf_list_concat* lemma, which inductively proves the equlity between the infimum over a concatenation of lists and their individual infimums. Of course, the proof is based on the associativity of the infimum operation in a lattice:

```
lemma inf_list_concat (l1 l2 : List α) :
  inf_list (l1 ++ l2) = inf_list l1 ⊓ inf_list l2 :=
    by
      induction l1 with
      | nil => simp
      | cons h t ih => simp; rw[ih]; rw [inf_assoc]
```

Coming back to our proof of the *gen_filter_prop* lemma, the next step is to show that X is a subset of S:

```
have HXin : X ⊆ S :=
  by
    rw [Set.subset_def]
    intro x HxinX
    exists [x]
    simp; assumption
```

Now, the only necessary condition to complete our proof is the minimality of S:

```
have Hmin : ∀ (F : Set α), filter F → X ⊆ F → S ⊆ F :=
  by
    intro F Hfilter Hsubset
    rw [Set.subset_def]
    intro x HxinS
    rcases HxinS with ⟨l, ⟨Hsubset', Hinf⟩⟩
    let Htrans := Set.Subset.trans Hsubset' Hsubset
    have Hinf_list_mem : inf_list l ∈ F := by apply inf_list_mem; assumption'
    exact (Hfilter.right).right (inf_list l) x Hinf_list_mem Hinf
```

For the proof of Lemma 3.3.10, we define an auxiliary lemma, stating that:

```
lemma mem_gen_ins_filter (F : Set α) (Hfilter : filter F) :
    y ∈ X_gen_filter (F ∪ {x}) → ∃ (z : α), z ∈ F /\ x ⊓ z ≤ y
```

To keep the exposition as concise as possible, we do not present here the details of the above result, but only the way we apply it, when proving Lemma 3.3.10. The proof is based on the residuation property of the Heyting algebra (le_himp_iff) and the second property satisfied by a filter (which we obtain by $(Hfilter.right).right$, since $Hfilter.left$ is the nonempty property):

```
lemma himp_not_mem (F : Set α) (Hfilter : filter F)
   (Himp_not_mem : x ⇒ y ∉ F) : y ∉ X_gen_filter (F ∪ {x}) :=
   by
     intro Hcontra
     have Haux : ∃ (z : α), z ∈ F /\ x ⊓ z ≤ y :=
       by apply mem_gen_ins_filter F Hfilter Hcontra
     rcases Haux with ⟨z, ⟨Hzin, Hglb⟩⟩
     rw [inf_comm, ←le_himp_iff] at Hglb
     exact Himp_not_mem ((Hfilter.right).right z (x ⇒ y) Hzin Hglb)
```

The central result of this section is Proposition 3.3.14, stating that, given a filter F and an element $x \notin F$, there exists a prime filter P containing F, such that $x \notin P$. In Lean, this statement transposes to:

```
lemma super_prime_filter (x : α) (F : Set α) (Hfilter : @filter α _ F)
    (Hnotin: x ∉ F):
    ∃ (P : Set α), @prime_filter α _ P /\ F ⊆ P /\ x ∉ P
```

In the following, we present how the main steps of the proof are formalized. First of all, we show that the set of all the prime filters not containing x has an upper bound:

```
have Hzorn : ∃ F' ∈ X_filters_not_cont_x x, F ⊆ F' ∧
             ∀ (F'' : Set α), F'' ∈ X_filters_not_cont_x x → F' ⊆ F'' →
               F'' = F'
```

This is achieved by applying Zorn's lemma, which is formalized as follows in Mathlib:

```
theorem zorn_subset_nonempty (S : Set (Set α))
    (H : ∀ (c) (_ : c ⊆ S), IsChain (· ⊆ ·) c → c.Nonempty →
        ∃ ub ∈ S, ∀ s ∈ c, s ⊆ ub) (x)
    (hx : x ∈ S) : ∃ m ∈ S, x ⊆ m ∧ ∀ a ∈ S, m ⊆ a → a = m
```

where *isChain* is a *Prop* deciding whether a given set is totally ordered. As we have already detailed in the theoretical section of the thesis, the upper bound is the union of all the chain's elements. The rest of the proof aims to prove that this upper bound is a prime filter, and we proceed by contraposition, in doing so. We consider two elements y, z such that $y \notin P$ and $z \notin P$. Then, the first key-step is proving that $P \subset [P \cup \{y\})$ and its analogous $P \subset [P \cup \{z\})$:

```
have Hsubset1 : P ⊂ X_gen_filter (P ∪ {y}) :=
  by
    unfold X_gen_filter
    rw [Set.ssubset_def]
    apply And.intro
    · rw [Set.subset_def]
      intro t Htin
      simp
      intro F' _ Hsubset
      apply Set.mem_of_mem_of_subset (Set.mem_of_mem_of_subset Htin
        (Set.subset_insert y P)) Hsubset
    · rw [Set.subset_def]
      push_neg
      exists y
      apply And.intro
      · simp
        intro F' _ Hsubset
        rw [Set.insert_subset_iff] at Hsubset
        exact Hsubset.left
      · assumption
```

Using this auxiliary statement and the maximality of P, we prove that $x \in [P \cup \{y\})$:

```
have Hxin1 : x ∈ X_gen_filter (P ∪ {y}) :=
  by
    by_cases Hxin : x ∈ X_gen_filter (P ∪ {y})
    · assumption
    · have Hfilter_not_cont: X_gen_filter (P ∪ {y}) ∈ X_filters_not_cont_x x :=
        by
          apply And.intro
          · simp
            exact X_gen_filter_filter (insert y P) (Set.insert_nonempty y P)
          · assumption
            exfalso
            exact Hsubset1.right (Eq.subset (Hmax (X_gen_filter (P ∪ {y}))
              Hfilter_not_cont Hsubset1.left))
```

Now, having also this hypothesis at hand, the proof is concluded by applying a few well-known Heyting algebras properties, as already shown in the theoretical proof.

Proposition 3.3.14 has also two important corollaries, which we prove in the sequel. The first one states that given an element x different from the last element of the algebra, there exists a prime filter P such that $x \notin P$. We first prove that $\{\top\}$ is a filter and then, using the *super_prime_filter* lemma, we obtain the witness we needed.

```
lemma super_prime_filter_cor1 (x : α) (Hnottop : x ≠ ⊤) :
  ∃ (P : Set α), @prime_filter α _ P /\ x ∉ P :=
    by
      let Htopfilter : @filter α _ {⊤} :=
        by
          apply And.intro
          · simp
          · simp
            intro x y Hxtop Htople
            rw [Hxtop] at Htople
            rw [top_le_iff] at Htople
            assumption
      let Haux := @super_prime_filter α _ x {⊤} Htopfilter Hnottop
      rcases Haux with ⟨P, ⟨_, ⟨_, _⟩⟩⟩
      exists P
```

The second corollary follows immediately from the first one. It claims that intersecting all the prime filters, we obtain the set $\{\top\}$. We prove this by double inclusion:

```
lemma super_prime_filter_cor2 : Set.sInter (@prime_filters α _) = {⊤} :=
  by
    rw [Set.ext_iff]
    intro x
    apply Iff.intro
    · intro Hincap
      simp
      by_cases Heqtop : x = ⊤
      · assumption
      · exfalso
        let Haux := @super_prime_filter_cor1 α _ x Heqtop
        rcases Haux with ⟨P, ⟨Hprime, Hxnotin⟩⟩
        have Haux' : P ∈ prime_filters :=
            by simp only [prime_filters]; assumption
        exact  Hxnotin (Hincap P Haux')
    · intro Htop
      rw [Htop]
      intro F Hprime
      rcases Hprime with ⟨⟨Hfilter, _⟩, _⟩
      exact @top_mem_filter α _ F Hfilter
```

On the first subgoal, we use the previous corollary in order to obtain a contradiction and on the second one, the essential step is the application of lemma *top_mem_filter*, stating that the last element of a Heyting algebra is contained in any filter:

```
lemma top_mem_filter (F : Set α) (Hfilter : filter F) : ⊤ ∈ F :=
  by
    let Hnempty := Hfilter.1
    have Haux : ∃ (x : α), x ∈ F := by assumption
    rcases Haux with ⟨x, Hxin⟩
    exact Hfilter.2.2 x ⊤ Hxin le_top
```

4.5.2 Algebraic models

The notions from Section 3.3.2 are formalized in the file *HeytingAlgebraSemantics.lean*. Hereby we present the main of them. We start by defining algebraic interpretations and parameterize an algebraic interpretation by a function I, which evaluates the variables, assigning to any of them an element of the Heyting algebra.

```
def AlgInterpretation (I : Var → α) : Formula → α
| Formula.var p => I p
```

```
| Formula.bottom => ⊥
| φ ∧∧ ψ => AlgInterpretation I φ ⊓ AlgInterpretation I ψ
| φ ∨∨ ψ => AlgInterpretation I φ ⊔ AlgInterpretation I ψ
| φ ⇒ ψ => AlgInterpretation I φ ⇨ AlgInterpretation I ψ
```

We've chosen not to explicitly define the notion of algebraic model in Lean, since it would have implied to adjoin the above defined interpretation function to the type α, in a structure of the form:

```
structure AlgModel (α : Type) (I : Var → α) (inst : HeytingAlgebra α) where
    h := AlgInterpretation I
```

We considered this redundant, since an algebraic model is uniquely determined by the variable-interpretation function. Hence, in the following declarations corresponding to the notions in Definitions 3.3.21, 3.3.22 and 3.3.23, the argument $I : Var \to \alpha$ represents the associated algebraic model:

```
def true_in_alg_model (I : Var → α) (φ : Formula) : Prop :=
    AlgInterpretation I φ = Top.Top

def valid_in_alg (φ : Formula) : Prop :=
    ∀ (I : Var → α), true_in_alg_model I φ

def alg_valid (φ : Formula) : Prop :=
    ∀ (α : Type) [HeytingAlgebra α], @valid_in_alg α _ φ

def set_true_in_alg_model (I : Var → α) (Γ : Set Formula) : Prop :=
    ∀ (φ : Formula), φ ∈ Γ → AlgInterpretation I φ = Top.top

def set_valid_in_alg (Γ : Set Formula) : Prop :=
    ∀ (I : Var → α), set_true_in_alg_model I Γ

def set_alg_valid (Γ : Set Formula) : Prop :=
    ∀ (α : Type) [HeytingAlgebra α], @set_valid_in_alg α _ Γ

def alg_sem_conseq (Γ : Set Formula) (φ : Formula) : Prop :=
    ∀ (α : Type)[HeytingAlgebra α](I : Var → α), set_true_in_alg_model I Γ →
    true_in_alg_model I φ
infix:50 " ⊨ₐ " => alg_sem_conseq
```

4.5.3 Lindenbaum-Tarski algebra

In the same *HeytingAlgebraSemantics.lean* file, we proceed by implementing the Lindenbaum-Tarski algebra. First of all, we define the relation from Definition 3.4.1 and its specific notation symbol:

```
def equiv (φ ψ : Formula) := Nonempty (Γ ⊢ φ ⇔ ψ)
infix:50 "∼" => equiv
```

Next, we define a setoid instance for our *Formula* type, by providing a proof of the above defined relation being indeed an equivalence relation and then we can move to defining the $\leq, \wedge, \vee, \rightarrow$ operations on quotients of this setoid. To define quotient conjunction, disjunction and implication, we make use of the built-in $lift_2$ function, which lifts the corresponding binary functions on formulas, to a quotient on both arguments. We give below only the formalization of quotient conjunction. The other quotient operations are defined in a similar manner.

```
def Formula.and_quot (φ ψ : Formula) := Quotient.mk setoid_formula (φ ∧∧ ψ)

def and_quot (φ ψ : Quotient setoid_formula) : Quotient setoid_formula :=
    Quotient.lift₂ Formula.and_quot and_quot_preserves_equiv φ ψ
```

Notice the fact that we have to pass as the second argument of $lift_2$ a proof of our binary operation preserving equivalence. The statement of the corresponding lemma is as follows:

```
lemma and_quot_preserves_equiv (φ ψ φ' ψ' : Formula) : φ ∼ φ' → ψ ∼ ψ' →
    (Formula.and_quot φ ψ = Formula.and_quot φ' ψ')
```

Having this operations defined, we can prove that the quotient type associated to the $\sim$ equivalence relation is a Heyting algebra. We do so by defining a Heyting algebra instance for this type:

```
instance lt_heyting : HeytingAlgebra (Quotient (@setoid_formula Γ))
```

We don't provide the full definition of this instance here, but all the proofs we need to complete its fields are rather trivial.

We define the mapping from Proposition 3.4.12, which associates to a formula its corresponding quotient:

```
def h_quot_var (v : Var) : Quotient (@setoid_formula Γ) := Quotient.mk
  setoid_formula (Formula.var v)
```

```
def h_quot (φ : Formula) : Quotient (@setoid_formula Γ) := Quotient.mk
setoid_formula φ
```

The *h_quot_var* function will be passed as an argument to *AlgInterpretation*, when proving that *h_quot* satisfies the conditions of an algebraic interpretation. The statement of this lemma is as follows:

```
lemma h_quot_interpretation : ∀ (φ : Formula),  h_quot φ =
(@AlgInterpretation (Quotient (@setoid_formula Γ)) _ h_quot_var φ)
```

Then, we are able to prove the two results in Proposition 3.4.12. We mention only their statements below, as the proofs do not contain any technical difficulties:

```
lemma set_true_in_lt :
    @set_true_in_alg_model (Quotient (@setoid_formula Γ)) _ h_quot_var Γ

lemma true_in_lt (φ : Formula) :
    @true_in_alg_model (Quotient (@setoid_formula Γ)) _ h_quot_var φ ↔
    Nonempty (Γ ⊢ φ)
```

4.5.4 Algebraic completeness theorem

In this section, we present the formalization of the algebraic proof of Completeness theorem. Let's focus first on the left implication, i.e. Soundness theorem:

```
theorem soundness_alg (φ : Formula) : Nonempty (Γ ⊢ φ) → alg_sem_conseq Γ φ
```

We provide here the proofs for some of the induction cases. For example, the conjunction contraction case is transposed in Lean as follows:

```
| @contractionConj ψ => intro _ _ I _
                        unfold true_in_alg_model; unfold AlgInterpretation
                        have Haux : AlgInterpretation I (ψ ∧∧ ψ) =
AlgInterpretation I ψ ⊓ AlgInterpretation I ψ := by rfl
                        rw [Haux, himp_eq_top_iff, inf_idem]
```

and for the modus ponens deduction rule, we have:

```
| @modusPonens ψ χ p1 p2 ih1 ih2 => intro α _ I Hsettrue
                                    simp at ih1; simp at ih2
                                    let ih2 := ih2 p2
                                    let ih1 := ih1 p1
```

```
                                          have Haux : AlgInterpretation I (ψ ⇒ χ) =
AlgInterpretation I ψ ⇒ AlgInterpretation I χ := by rfl
                                          let ih2 := ih2 α I Hsettrue
                                          unfold true_in_alg_model at ih2
                                          rw [Haux, ih1, top_himp] at ih2
                                          assumption'
```

Moving now to the reverse implication, the proof is based on the two results in Proposition 3.4.12. Below, we present the full formalization of the equivalence in the completeness theorem:

```
theorem completeness_alg (φ : Formula) :
  alg_sem_conseq Γ φ ↔ Nonempty (Γ ⊢ φ) :=
    by
      apply Iff.intro
      · intro Halg
        rw [←true_in_lt]
        exact Halg (Quotient (@setoid_formula Γ)) h_quot_var set_true_in_lt
      · exact soundness_alg φ
```

4.5.5 Kripke models and algebraic models

We start by establishing the connection from Kripke models to algebraic models. In doing so, we have to define first the notions of closed set, and the Heyting algebra structure which can be built on top of the set of all the closed sets.

Thus, the following *Prop* decides whether a domain set of a Kripke model is closed:

```
def closed {W : Type} (M : KripkeModel W) (A : Set W) : Prop :=
    ∀ (w w' : W), w ∈ A → M.R w w' → w' ∈ A
```

We formalize the set of all closed subsets as a subtype of the *Set W* type, as follows:

```
def all_closed {W : Type} (M : KripkeModel W) := {A // @closed W M A}
```

For the implication operation on closed subsets, we first define the set of all closed sets contained in $W \setminus A \cup B$, where by A, B we denote the two implication operands. Then, the union of the elements in this set is the greatest closed set satisfying our condition:

```
def all_closed_subset {W : Type} (M : KripkeModel W) (A B : all_closed M) :=
    {X | @closed W M X /\ X ⊆ ((@Set.univ W) \ A.1) ∪ B.1}
```

```
def himp_closed {W : Type} {M : KripkeModel W} (A B : all_closed M) :=
    Set.sUnion (@all_closed_subset W M A B)
```

We formalize Corrolary 3.6.5, by defining the corresponding Heyting algebra instance:

```
instance {W : Type} (M : KripkeModel W) : HeytingAlgebra (all_closed M) :=
  { sup := λ X Y => {val := X.1 ∪ Y.1, property := union_preserves_closed X Y}
    le := λ X Y => X.1 ⊆ Y.1
    le_refl := λ _ => Set.Subset.rfl
    le_trans := λ _ _ _ => Set.Subset.trans
    le_antisymm := λ _ _ => by rw [Subtype.ext_iff]; apply Set.Subset.antisymm
    le_sup_left := λ X Y => Set.subset_union_left X.1 Y.1
    le_sup_right := λ X Y => Set.subset_union_right X.1 Y.1
    sup_le := λ _ _ _ => Set.union_subset
    inf := λ X Y => {val := X.1 ∩ Y.1, property := inter_preserves_closed X Y}
    inf_le_left := λ X Y => Set.inter_subset_left X.1 Y.1
    inf_le_right := λ X Y => Set.inter_subset_right X.1 Y.1
    le_inf := λ _ _ _ => Set.subset_inter
    top := {val := @Set.univ W, property := univ_closed}
    le_top := λ X => Set.subset_univ X.1
    himp := λ X Y => {val := himp_closed X Y, property := himp_is_closed X Y}
    le_himp_iff := λ X Y Z => himp_closed_prop Y Z X
    bot := {val := ∅, property := empty_closed}
    bot_le := λ X => Set.empty_subset X.1
    compl := λ X => {val := himp_closed X {val := ∅, property := empty_closed},
                     property := himp_is_closed X {val := ∅,
                                                   property := empty_closed}}
    himp_bot := by simp }
```

The next step is proving that the function from Lemma 3.6.6 is an algebraic interpretation. Except for the implication case, the proof is trivial. We present here the main steps of this last interesting case. The proof is by double inclusion, but before succeeding in doing so, we need to prove an additional *have* statement, which holds only for closed subsets:

```
have Haux : ∀ (A : all_closed M),
    A.1 ⊆ (@h W M (ψ ⇒ χ)).1 ↔ A.1 ∩ (@h W M ψ).1 ⊆ (@h W M χ).1
```

and then apply the residuation property of the closed sets version of the residuation property to obtain that:

```
have Haux' : ∀ (A : all_closed M),
    A.1 ⊆ (@h W M (ψ ⇒ χ)).1 ↔ A.1 ⊆ himp_closed (@h W M ψ) (@h W M χ)
```

This last statement helps us complete both of the inclusions we need to prove.

By this point, we can formalize the first central result of the section - Proposition 3.6.7, which provides a method of constructing an algebraic model of the same capacity of a given Kripke model:

```
lemma kripke_alg {W : Type} {M : KripkeModel W} (φ : Formula) :
  valid_in_model M φ ↔ @true_in_alg_model (all_closed M) _ h_var φ :=
   by
      apply Iff.intro
      · intro Hvalid
        unfold true_in_alg_model
        rw [←h_interpretation]
        simp only [Top.top]
        rw [Subtype.ext_iff, Set.ext_iff]
        simp
        assumption
      · intro Htruealg
        unfold true_in_alg_model at Htruealg
        rw [←h_interpretation] at Htruealg
        simp only [Top.top] at Htruealg
        rw [Subtype.ext_iff, Set.ext_iff] at Htruealg
        simp at Htruealg
        assumption
```

In the sequel, we aim to formalize also the reverse direction, more specifically the switch from an algebraic model to a corresponding Kripke one. We first define the Kripke frame based on the set of all prime filters:

```
def prime_filters_frame (I : Var → α) :
    KripkeModel (@prime_filters α _) :=
    {
    R := λ (F1 F2) => F1.1 ⊆ F2.1,
    V := λ (v F) => I v ∈ F.1,
    refl := λ (F) => Set.Subset.rfl,
    trans := λ (F1 F2 Φ) => Set.Subset.trans,
    monotonicity := λ (v F1 F2) => by intros; apply Set.mem_of_mem_of_subset;
                                      assumption'
    }
```

and prove that the function given by:

```
def Vh (φ : Formula) (F : @prime_filters α _) (I : Var → α) : Prop :=
    AlgInterpretation I φ ∈ F.1
```

is a valuation function for this frame. Now, we can prove Proposition 3.6.9, which establishes the second relation between algebraic and Kripke models:

```
lemma alg_kripke (I : Var → α) (φ : Formula) :
  true_in_alg_model I φ ↔ valid_in_model (prime_filters_frame I) φ :=
    by
      apply Iff.intro
      · intro Htruealg
        intro Hprime
        rcases Hprime with ⟨F, ⟨⟨Hfilter, _⟩, _⟩⟩
        rw [←Vh_valuation]
        unfold Vh
        rw [Htruealg]
        exact @top_mem_filter α _ F Hfilter
      · intro Hvalid
        have Haux : (∀ (w : ↑prime_filters),val (prime_filters_frame I) w φ) →
                    (∀ (w : ↑prime_filters), Vh φ w I) :=
        by
            intro _ _
            rw [Vh_valuation]
            apply Hvalid
        let Hvalid := Haux Hvalid
        unfold Vh at Hvalid
        simp at Hvalid
        rw [←Set.mem_sInter, super_prime_filter_cor2] at Hvalid
        assumption
```

Finally, having this auxiliary results at hand, we can immediately prove the equivalence between Kripke and algebraic validity:

```
theorem alg_kripke_valid_equiv (φ : Formula) :
  alg_valid φ ↔ valid φ :=
    by
      apply Iff.intro
      · intro Halg _ _
        rw [kripke_alg]; apply Halg
      · intro Hvalid _ _ _
        rw [alg_kripke]; apply Hvalid
```

Chapter 5

Conclusions. Future work

We have used the Lean proof assistant to formally verify the completeness of Intuitionistic Propositional Logic. After defining our language, we proceeded by formalizing the Hilbert-style proof system and used it to establish a collection of useful syntactic theorems and derived deduction rules. The next crucial step in our path to completeness was formally specifying the two studied semantics: Kripke and algebraic. With all the definitions so far formalized, the two soundness proofs followed naturally. Then, for the proof of the completeness theorem with respect to the Kripke semantics, we defined the so-called canonical model, based on disjunctive theories and used it in order to complete the proof by contraposition. On the other hand, for the algebraic completeness proof, we made use of the Lindenbaum-Tarski algebra and some of its specific properties.

We conclude the thesis hoping that we have offered a detailed and comprehensive presentation of the Intuitionistic Propositional Logic system, both in its theoretical aspects and in regard to our implementation approach. Not least of all, a key purpose we would like to believe we achieved is demonstrating how statements can be formalized and proved by means of a computer, manner which remains essentially unchanged, regardless if it comes to ordinary mathematical theorems, or claims that network protocols or pieces of software/hardware meet their specifications. As future work, we aim to extend the current formalization to express the language, proof system and semantics of Intuitionistic First-Order Logic and also provide a completeness proof for this more complex system.

Appendix A

Lattices

The following brief exposition of lattices definitions and basic properties is based on the presentation in [14].

Definition A.0.1. *A **lattice** is a nonempty set L together with two binary operations $\vee$ and $\wedge$, which satisfies the following properties:*

(i) commutative laws:

  (a) $x \vee y = y \vee x$

  (b) $x \wedge y = y \wedge x$

(ii) associative laws:

  (a) $x \vee (y \vee z) = (x \vee y) \vee z$

  (b) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

(iii) idempotent laws:

  (a) $x \vee x = x$

  (b) $x \wedge x = x$

(iv) absorption laws:

  (a) $x = x \vee (x \wedge y)$

  (b) $x = x \wedge (x \vee y)$

Definition A.0.2. *A binary relation $\leq$ defined on a set A is a* ***partial order*** *on the set A if the following conditions hold in A:*

(i) $x \leq x$ *(reflexivity)*

(ii) $x \leq y$ *and* $y \leq x$ *imply* $x = y$ *(antisymmetry)*

(iii) $x \leq y$ *and* $y \leq z$ *imply* $x \leq z$ *(transitivity)*

Definition A.0.3. *A nonempty set with a partial order on it is called a* ***partially ordered set****.*

Definition A.0.4. *Let A be a subset of a partially ordered set P. Then:*

(i) *An element p in P is an* ***upper bound*** *for A if $x \leq p$ for every x in A.*

(ii) *An element p in P is the* ***least upper bound*** *of A, if p is an upper bound for A, and $x \leq y$ for every x in A implies $p \leq y$.*

(iii) *An element p in P is a* ***lower bound*** *for A if $p \leq x$ for every x in A.*

(iv) *An element p in P is the* ***greatest lower bound*** *of A, if p is a lower bound for A, and $y \leq x$ for every x in A implies $y \leq p$.*

Now we are ready to provide an alternative definition of a lattice:

Definition A.0.5. *A* ***lattice*** *is a partially ordered set L in which for every x, y in L, there exist both the least upper bound $(x \vee y)$ and the greatest lower bound $(x \wedge y)$.*

Remark A.0.6. *Definitions A.0.1 and A.0.5 are equivalent.*

Definition A.0.7. *A lattice L is called* ***bounded lattice*** *if there exist both a lowest upper bound (which we denote by 1) and a greatest lower bound (which we denote by 0) of L.*

Remark A.0.8. *0 is called the "first" element of the lattice and 1 the "last" element of the lattice.*

Proposition A.0.9. *The following properties are true in a bounded lattice $(L, \vee, \wedge, 0, 1)$, for all $x, y \in L$:*

(i) *for all $z \in L$, ($z \leq x$ iff $z \leq y$) iff $x = y$*

(ii) *for all $z \in L$, $z \leq x$ and $z \leq y$ iff $z \leq x \wedge y$*

(iii) *$1 \leq x$ iff $x = 1$*

(iv) *$\top \wedge x = x$*

(v) *$x \leq z$ and $y \leq z$ iff $x \vee y \leq z$*

(vi) *for all $z \in L$, $x \leq y$ implies $z \vee x \leq z \vee y$*

Bibliography

[1] T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2–3):95–120, 1988.

[2] L. De Moura, S. Kong, J. Avigad, F. Van Doorn, and J. von Raumer. The Lean theorem prover (system description). In A. Felty and A. Middeldorp, editors, *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, number 9195 in Lecture Notes in Computer Science, pages 378–388. Springer, 2015.

[3] M. C. Fitting. *Intuitionistic Logic, Model Theory and Forcing*. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1968.

[4] G. Georgescu. Notes on Heyting Algebras (in Romanian). Lecture Notes, University of Bucharest, 1995.

[5] G. Georgescu. Notes on Intuitionistic Logic (in Romanian). Lecture Notes, University of Bucharest, 1995.

[6] H. Guo, D. Chen, and B. Bentzen. Verified completeness in Henkin-style for intuitionistic propositional logic. In B. Bentzen, B. Liao, D. Liga, R. Markovich, B. Wei, M. Xiong, and T. Xu, editors, *Logics for AI and Law*, pages 36–48, London, 2023. College Publications.

[7] K. Gödel. Über eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958.

[8] S. C. Kleene. *Introduction to Metamathematics*. Bibliotheca Mathematica, edited by N. G. De Bruijn and G. van Dantzig and J. de Groot. North-Holland, Amsterdam, 1952.

[9] S. A. Kripke. Semantical analysis of modal logic i. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 9:67–96, 1963.

[10] S. A. Kripke. Semantical analysis of intuitionistic logic. In *Formal Systems and Recursive Functions*, volume 40, pages 92–130. North-Holland, 1965.

[11] S. Kuznetsov. Propositional Intuitionistic Logic. Lecture Notes, University of Pennsylvania, 2017.

[12] G. Mints. *A Short Introduction To Intuitionistic Logic*. The University Series in Mathematics. Kluwer, New York, 2000.

[13] H. Rasiowa and R. Sikorski. *The mathematics of metamathematics*. Panstwowe Wydawnictwo Naukowe, Warsaw, 1963.

[14] H. P. Sankappanavar and S. Burris. A Course in Universal Algebra. *Graduate Texts in Mathematics*, 78:56, 1981.

[15] A. S. Troelstra. *Metamathematical Investigation Of Intuitionistic Arithmetic And Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer, Berlin Heidelberg, 1973.

[16] D. Trufas. Intuitionistic Logic in Lean. https://github.com/DafinaTrufas/Intuitionistic-Logic-Lean, 2024.

[17] D. Van Dalen. *L.E.J. Brouwer - Topologist, intuitionist, philosopher: How mathematics is rooted in life*. Springer, London, 2012.