Università degli Studi di Genova
Dipartimento di Matematica
PhD Program in Mathematics and Applications
Coordinated by Prof. Stefano Vigni

PhD Thesis in Mathematical Logic

Investigations of Proof Theory and Automated Reasoning for Non-classical Logics

Cosimo Perini Brogi

Supervised by
Prof. Sara Negri
Prof. Giuseppe Rosolini

Defended on July 12, 2022

Abstract

This thesis presents some new results in structural proof theory for modal, intuitionistic, and intuitionistic modal logics.

The first part introduces three original Gentzen-style natural deduction calculi for, respectively, intuitionistic verification-based epistemic states – namely, belief and knowledge operators – and intuitionistic strong Löb logic for arithmetical provability. For each of these calculi strong normalisation results are proven w.r.t. several systems of proof rewritings, which are considered on the basis of their structural relevance, e.g. for establishing the related subformula principles, or for providing a categorical semantics of normal deductions. The presentation of new and original sequent calculi for a wide family of interpretability logics closes this first part of the thesis. These sequent systems are modularly designed by recurring to internalisation techniques which make possible their fine grained structural analysis, this way establishing both their semantic and structural completeness.

The second part has a more applicative nature. It presents first an implementation in the HOL Light proof assistant of an internal theorem prover and countermodel constructor for Gödel-Löb logic, relying on a previous computerised proof of modal completeness for that logic within the same formal environment. The design of that proof search algorithm is surveyed, and examples of both its interactive and automated use are shown. An overview of an ongoing automation-oriented implementation in UniMath of the basics of univalent universal algebra closes this second part of the thesis. The coding style and methodology used are discussed besides some concrete formalisation examples of algebraic structures.

Finally, two appendices describe the logical engine underlying each of the proof assistants that are used for the results presented in the second part, namely classical higher order logic for HOL Light, and univalent type theory for UniMath.

— ◇ —

In memory of Maria Magini,
structuring element of each Cosmos.

— ◇ —

Acknowledgements

I would like to thank my supervisors, Prof. Sara Negri and Prof. Giuseppe "Pino" Rosolini: They let me pursue my own research interests with no fear of wrecking, since I could have always relied on their expertise.

I am also thankful to the coordinator of the PhD Program, Prof. Stefano Vigni, for his kind and absolute helpfulness during these years at the Department of Mathematics in Genoa: In many situations, his and Pino's support and advice played a key and rebalancing role for my academic progress.

I am grateful to the referees of this thesis, Prof. Martin Hyland and Dr. Eugenio Orlandelli: I have been honoured by their acceptance to review my manuscript, and I care to express all my gratitude to them for their detailed and constructive feedback on my work.

I wish also to thank the people whom I worked with: Prof. Gianluca Amato, Prof. Marco Maggesi, and Prof. Maurizio Parton, since our discussions and collaboration were invaluable. In particular, I want to express my deepest gratitude to Marco, for hosting me at the University of Florence, and, even more, for his guidance in the jungle of implicit traditions and non-optional social conventions characterising academic life.

During this three years – and a bit more – journey, I was lucky to have met, in person and online, many further inspiring researchers.

I am thankful to Prof. Nicola Olivetti for our online discussions with Sara on structural proof theory for interpretability logics, which were very important for achieving some of the definite results presented here.

I am grateful to Dr. Reuben Rowe for our chatting during the Proof Society Summer School at Swansea University – and, in particular, on the way to Rhosili – which was more than enlightening and made me achieve a concrete insight on the directions my PhD would have taken.

I wish to thank the members of the Genoa Logic Group and, in particular, Prof. Riccardo Camerlo for keeping the activities at the DiMa going on during these unusual years.

Many thanks also to Dr. Ulrich von der Ohe and Dr. Rémi Bignalet-Cazalet for being kindest office-mates, and for the pleasant hours we spent together in Room 810 talking about maths, natural languages, and culture. Special thanks to Ulrich for our Matrix sessions during the confinement months, and for his contagious enthusiasm about free software.

My endless gratitude goes to my family and friends for helping me in moving forward.

In particular, I wish to thank my close friend Marianna, since her encouragement has been a cipher for my personal development; I am glad to have find in her a kindred academic who truly values rational nonconformity.

Most of all, I care to thank Laura Brogi, my mother "generalessa von Kurtewitz": Your courage, strength, and intelligence nurtured my mind and spirit. We both achieved this, together, and I wouldn't have been able to cope with all that without you by my side.

I have dedicated this thesis to your mother, my stellar grandmother Maria Magini: The life years that the three of us shared have marked an invariant in all possible Cosmoses.

Contents

Prologue

General introduction

This thesis can be thought to dwell at the intersection of two research topics: structural proof theory and automated reasoning. It is then organised in two parts, dealing with each of the two subjects.

Overall, its focus is on mathematical logics that behave differently from the usual way of reasoning in mathematical practice. The classically trained mathematician is generally interested in the truth value of a statement: If that statement is true, then she calls it a theorem; if it is false, she says to have refuted it. The logic underlying this reasoning is called classical: It is bivalent and based on the assumption that every statement is either true or false. *Non-classical logics* challenge that perspective, by investigating the *ways* a statement can be true or false [Pri08]. This is formally done by means of different logical operators, or by changing the classical behaviour of the standard ones.

Here we will discuss exclusively modal and intuitionistic logics, as well as their interplay. More exotic non-classical logics do exist and have a wide range of applications, but mainly outside the mathematical realm. In fact, our interest will be directed towards those non-classical systems that deal with notions that are somehow related to mathematics; in more details: verification and formal provability; interpretability over a base formal theory; constructive and algorithmic reasoning in mathematics.

Modal logics [vB16] are interested in the modes in which a statement is true (or false): It is clear that, outside everyday mathematics, a statement can be contingently true; true in the past, and false at a different time; or true after performing a task – or process; or even true (or false) according to the background knowledge of an agent. In all these cases, our language is enriched by modal expressions, which are formally rendered by modal operators. Modal logics are now a powerful tool for investigations in computer science, philosophy, linguistics and the foundations of mathematics, since many mathematical constructions can be described in abstract terms by means of modalities. It is then not unusual to read about *mathematical* modal logics in scientific literature [Gol06].

But even before the advent of modal systems, non-classical logics have been applied in mathematical inquiries. *Intuitionistic mathematics* [vDT88] have been developed since the first decades of 20th century to challenge the metaphysical assumptions underlying the traditional mathematical practice. On logical terms, such a challenge is made concrete my rejecting the principle of bivalence of classical logic: It is not possible to claim that any statement is either true or false, since the truth of a proposition is mathematically established only by producing a *constructive proof*, and we know that in some cases we are not able to prove or refute a conjecture. Accordingly, *intutionistic logic* [Mos21, Iem20] describes the behaviour of the standard logical operators in very different terms from their classical analogous, by emphasising the *constructive and algorithmic* nature of proving and refuting a mathematical statement.

Then there exist their mutual interactions: *Intuitionistic modal logics* [Sim94, dPGM04] introduce modal operators besides intuitionistic ones, and in recent years are experiencing a new dawn in virtue of their applications to theoretical computer science, as well as to meta-mathematical inquiries in intuitionistic arithmetic.

In this thesis these kinds of logics – modal, intuitionistic, and intuitionistic modal – will be investigated by the methods and techniques of proof theory. But we will also use further non-classical systems – respectively, higher order logic and univalent type theory – for performing deductions in a modal system – namely, Gödel-Löb logic – and computations in mathematical structures – namely, basic constructions in universal algebra – by recourse to proof assistants that do their job thanks to knowledge coming from structural proof theory applied to computer science and computerised mathematics.

Structure of the thesis and its contents

The thesis is organised in two parts, for a total of seven chapters and two appendices. Chapter 1 is introductory: It briefly collects all the material that is shared by at least two of the subsequent chapters, and recalls the basics of normal modal logics and provability logics; sequent calculi in the G3 paradigm and their extensions for non-classical reasoning; natural deduction for intuitionistic propositional logic; type theory from the perspective of the proof-as-programs paradigm, together with a minimalist primer on category theory for natural deduction systems.

The remaining chapters offer original contributions.

- ▷ In Part I, Chapter 2 introduces a natural deduction system for the logic of intuitionistic belief axiomatised in [AP16]. The calculus is shown to strongly normalise w.r.t. several systems of rewritings, that are based on fulfilled proof-theoretic expectations and categorical considerations.

  Chapter 3 extends the results about intuitionistic belief to a natural deduction system for intutionistic knowledge, that we have designed on the basis of the BHK explanation given in [AP16] for verification-based epistemic states. Even for this system, we prove strong normalisation of deductions as well as further proof-theoretic *desiderata*.

Chapter 4 extends the natural deduction for intuitionistic belief into a new natural deduction calculus for intuitionistic provability logic based on the strong version of Löb axiom investigated first in [dJV95]. This extension is rather modular, so that the normalisation results for deductions in this intuitionistic provability logic are obtained by adapting the proof strategies developed in the previous chapters.

Chapter 5 closes the first part of the thesis by presenting a uniform family of labelled sequent calculi for interpratability logics over a classical propositional system, based on the semantics surveyed in [JRMV20]. This chapter contains the description of what is probably the largest class of analytic calculi for those mathematical modal logics currently available, and presents detailed proofs of their excellent behaviour from the point of view of structural proof theory. Most notably, we produce a uniform cut-elimination procedure showing that the rule of cut is admissible in any of the calculi under investigation.

▷ In Part II, we move from structural proof theory to its applications in automated reasoning.

Chapter 6 describes a theorem prover and countermodel constructor for Gödel-Löb logic (GL) developed within the theorem prover HOL Light. It surveys the code that is now part of the official distribution of HOL Light, and gives the main details of how the GL-theorem prover has been designed on the basis of the methodology of structural proof theory for modal logics analysed in [Neg14b]. The chapter contains also some hands-on examples of proof search for a modal formula given as input to the theorem prover, as well as some reflections on the possibility to widen the implementation presented here to develop automated theorem provers in HOL Light for many other non-classical logics by working on the same methodological idea.

Chapter 7 gives an overview of the UniMath library for universal algebra that we developed with the intent of making an efficient use of the computational relevance of computerisation of mathematics: We wanted to highlight the demonstrative contents of our formalisation by leaving to the computer all the trivial steps of computation that are involved in the development of formal proofs about mathematical structures. We have been largely inspired by Henk Barendregt's Poincaré principle [Bar92], and we have instantiated it by our definition of terms of an algebra over a signature, which does not recur to Coq inductive type constructors. Therefore, we present our implementation of universal algebra from the very basics of the subject, and then proceed with the definition of the categorical structures determined by the basic notions – namely, signatures, algebras, and equational algebras. Some examples of concrete algebraic structures rephrased in our formalism close the chapter, which contains also some methodological remarks and perspectives on future work on the library.

▷ Finally, there are the two appendices describing the logical theories underlying the proof assistants that we used in the second part of the thesis.

Appendix A gives a concise introduction to the deductive engine of HOL Light, and contains some personal considerations about the main characteristics of that theorem prover, namely its flexibility and minimalist nature. We recall the primitive rules on top of which any formal proof can be carried out in HOL Light, and discuss its proof development environment based on rules, tactics and tacticals. Finally, we point to the official documentation of HOL Light, and we recall some worth noticing examples of use of that theorem prover in both academic and industrial research.

Appendix B proposes a formal definition of intuitionistic type theory as a single conclusion sequent calculus, and discuss its extension by the univalence principle. This is the logical kernel of UniMath, that we describe as a very minimal system for univalent type theory. We collect some general information about univalent reasoning and its relevance for the foundations of mathematics and for theoretical computer science, and we point to the most recent literature about this new research field, with an emphasis on meta-theoretical investigations.

Part of the results presented in this thesis have already been published. The papers in which they appear are the following:

- Chapter 2: The natural deduction calculus for intuitionistic belief, its strong normalisation and corresponding categorical semantics can be found in [PB21b]; some preliminary results are also discussed in [PB21a] and [PB19].

- Chapter 6: The formalisation of modal completeness of $\mathbb{GL}$ appeared first in [MPB21]; the theorem prover and countermodel constructor is described in [MPB22]. The code surveyed in the chapter is freely available from the official HOL Light distribution [Har22].

- Chapter 7: The implementation of universal algebra is surveyed in [AMPB21]. A preliminary version appeared in [AMPPB20]. The version discussed here is part of the official UniMath library [VAG+22].

1

Preliminary material

This chapter recalls the basic tools that are common in investigations concerning modal logics, structural proof theory, and type theory.

The material in these pages can be thought of as a minimal set of concepts, ideas, and results that will recur often in the main parts of this dissertation, or that will be common to more than one chapter to follow: They are collected here since thay are the unavoidable preliminaries for the subsequent presentation.

No aim of completeness leads the following exposition, and the reader might find further and more detailed results among the appropriate references that will be cited in the next few pages.

1.1. Normal modal logics

Modal notions are ubiquitous in natural languages as well as in science. In all areas of inquiring, concepts having modal character appear, and their study under the light of logic dates back to Aristotle's *De Interpretatione, 12-13*.

In general terms, a modality expresses a mode in which a statement may be true (or false). Modal logics arise then as a very general framework to model those modes, and their peculiar characteristic is the possibility to expand the descriptive power of standard – i.e. classical or intuitionistic – logics.

An informal logical treatment of modalities runs across the Middle Ages in the works of several philosophers discussing problems in metaphysics, theology and proto-linguistics; they were largely inspired by Aristotle's previous reflections on necessity, possibility, and Fate. After the advent of modern mathematical logic, we are able to rephrase their modal arguments by means of modal logical systems obtained by enriching the base language and deductive apparatus for standard logic by new non-truth-functional operators that are usually called modalities, or modal operators.

The father of modern modal logic is C.I. Lewis whose pioneering [Lew18] contains almost the whole syntactic "toolbox" that we use to develop modal systems nowadays.[1] In the 1930s, modal concepts appear in the study of foundations of mathematics thanks to Ivan Orlov [Orl28] and Kurt Gödel [Göd86][2], while Jan Łukasiewicz investigated formal semantics of modal concepts starting from the 1920s [Łu30].

In the following decades, many further modalities are standing in the spotlight: G.H. von Wright introduces deontic logics to study obligations and permissions [vW51]; Arthur Prior uses modalities dealing with time [Pri57]; Jaakko Hintikka introduces multi-modal systems for studying epistemic notions [Hin62]. These are also the years when a powerful unifying tool arises, namely relational semantics, independently developed by Saul Kripke, Stig Kanger, and Jaakko Hintikka.[3]

It is also because of this new general semantic framework that many other research fields have started to inquire on and apply modalities. More prominently, dynamic logic marks the advent of modal logic for computer science, thanks to R.W. Floyd [Flo67], Vaughan Pratt [Pra76], and Krister Segerberg [Seg82]. By now, modal operators modelling script runs, branching and linear time flows, conditionals, knowledge and its evolution have appeared in any branch of theoretical computer science, as well as formal hardware and software verification, game theory, and social choice theory [vB16].

Modern coalgebraic methods [KP11] are furthermore progressing towards a unifying theoretical account on modal phenomena in terms of categorical notions. On the more applicative side, description logics [Sch93] are used in several domains involving computational information flow and knowledge representation, from molecular biology [MB06] to the modelling of decentralised and distributed systems [BFS20].

The interested reader may find several good textbooks on modal logic – e.g. [CZ97], [Pop94], and [Hum15]: She may consider [BvB07] for an excellent starting point for surveying more advanced topics in theory and applications of modalities. Finally, an exceptionally clear history of modern modal logic is provided by [Bal21]. Here, we limit ourselves to collect the very basic theoretical ingredients for syntax and semantics of normal modal logics. The contents of the present section will be used partially in Chapters 3–5, and, more extensively, in Chapter 6. The latter provides also a computer formalisation of all the notions and results collected here.

1.1.1. Elementary syntax

Let's start by recalling the formal definitions of a propositional modal language and its formulas.

[1] Previous investigations were carried out in [Mac06].

[2] See Section 1.2 below.

[3] See Section 1.1.3. Previously, an algebraic interpretation was proposed by Alfred Tarski, J.C.C. McKinsey and Bjarni Jónsson: Refer [CZ97, Ch. 7] for a modern presentation.

DEFINITION 1.1.1. A (mono)modal propositional language $\mathcal{L}_\Box$ is given by:

- a denumerable infinite set $\texttt{Atm}$ of propositional atoms $p_0, p_1, \ldots$;
- propositional operators $\rightarrow, \bot, \wedge, \vee$;
- a unary modal operator $\Box$;
- auxiliary symbols "(" and ")".

DEFINITION 1.1.2. The set of formulas $\texttt{Form}$ of $\mathcal{L}_\Box$ is inductively defined as follows

- for each p_i, p_i is a formula;
- $\bot$ is a formula;
- if A and B are formulas, so are $A \rightarrow B, A \wedge B, A \vee B$;
- if A is a formula, so is $\Box A$;
- nothing else is a formula.

Truth, negation, coimplication are defined in the standard way: $\top := \bot \rightarrow \bot$, $\neg A := A \rightarrow \bot$, and $A \leftrightarrow B := (A \rightarrow B) \wedge (B \rightarrow A)$, respectively. When we are reasoning in a classical setting, we can also consider a further modal operator $\Diamond$, that we define as $\Diamond A := \neg\Box\neg A$.

We will often recur to the following formalism to define the admissible expressions of a given language:
The formulas of our language are inductively defined b y the following grammar

$$\texttt{Form} := \ p \mid A \wedge B \mid A \vee B \mid A \rightarrow B \mid A \leftrightarrow B \mid \neg A \mid \top \mid \bot \mid \Box A \ ,$$

where $p \in \texttt{Atm}$ and $A, B \in \texttt{Form}$.

The inductive definition of subformula easily follows:

DEFINITION 1.1.3. The set $Sb(A)$ of subformulas of a given formula A is defined by:

- $A \in Sb(A)$;
- if p occurs in A, then $p \in Sb(A)$;
- if $B \circ C \in Sb(A)$, then $B \in Sb(A)$ and $C \in Sb(A)$, where $\circ \in \{\rightarrow, \wedge, \vee\}$;
- if $\Box B \in Sb(A)$, then $B \in Sb(A)$

Let the degree of a formula be the number of logical operators in it.

DEFINITION 1.1.4. Let $A \in \texttt{Form}$ and $\sigma : \texttt{Atm} \longrightarrow \texttt{Form}$. Then ${}^{\sigma}A$, i.e. the formula obtained from A by simultaneously substituting every occurrence in A of each $p \in Sb(A)$ with $\sigma(p)$ is defined by induction on the degree of A:

- ${}^{\sigma}p :\equiv \sigma(p)$;
- ${}^{\sigma}\bot :\equiv \bot$;
- ${}^{\sigma}(B \circ C) :\equiv {}^{\sigma}B \circ {}^{\sigma}C$ for $\circ \in \{\rightarrow, \wedge, \vee\}$;
- ${}^{\sigma}(\Box A) :\equiv \Box({}^{\sigma}A)$.

1.1.2. AXIOMATIC CALCULI

Any logic is made of a language along with a calculus and a semantics.

For the moment, we start by characterising a normal modal logic as the set of all formulas of $\mathcal{L}_{\Box}$ that are derivable in an axiomatic calculus extending any sound and complete axiomatic system for classical *or* intuitionistic propositional logic by means of specific axiom schemas and inference rules.

The minimal normal modal system is usually denoted by $\mathbb{K}$.

DEFINITION 1.1.5. The axiomatic calculus $\mathbb{K}$ is defined by the following axiom schemas:

- standard schemas for propositional logic;
- schema K: $\Box(A \rightarrow B) \rightarrow \Box A \rightarrow \Box B$;

along with the following inference rules:

MP: $\dfrac{A \rightarrow B \qquad A}{B}$;

NR: $\dfrac{A}{\Box A}$.

A **normal modal system** $\mathbb{S}$ is then any axiomatic extension of $\mathbb{K}$ that is closed under MP, NR and substitution. When the basic propositional logic is intuitionistic we denote the resulting system by $\mathbb{IS}$.

A **formal derivation** in $\mathbb{S}$ is a finite sequence of formulas where each one is an instance of an axiom schema of $\mathbb{S}$, or has been obtained from the previous formulas in the sequence by applying one of the rules of $\mathbb{S}$.

We write $\mathbb{S} \vdash A$ when A is the last formula of a formal derivation in $\mathbb{S}$. Derivability from a set of hypotheses is then defined as follows:

DEFINITION 1.1.6. Given a set Γ of formulas of $\mathcal{L}_{\Box}$, we say that A is $\mathbb{S}$-derivable from Γ, and write $\Gamma \vdash_{\mathbb{S}} A$, when there exists a finite sequence $A_1, \ldots, A_n$ of formulas such that $A_n \equiv A$ and, for $1 \leq k \leq n$, A_k is an instance of an axiom schema of $\mathbb{S}$, or it belongs to Γ, or it is obtained from previous formulas in the sequence by applying MP, or it is obtained by applying NR to a previous formula in the sequence *non-depending from any formula in* Γ.

For such a notion, the standard version of the **deduction theorem** holds:[4]

$$\Gamma, A \vdash_{\mathbb{S}} B \quad \text{iff} \quad \Gamma \vdash_{\mathbb{S}} A \to B$$

That, as it is known, has a prominent role in helping the development of formal derivation by pencil and paper.

Finally, let's convey to say that a set Γ of formulas is $\mathbb{S}$-**consistent** when $\Gamma \nvdash_{\mathbb{S}} \bot$.

1.1.3. Relational semantics

Turning now to the model-theoretic side, it is usual to deal with normal modal logics by relying on relational semantics, which is also called Kripke semantics, or possible world semantics.[5]

Definition 1.1.7. A modal frame $\mathcal{F} = \langle W, R \rangle$ is made of a non-empty set W "of possible worlds" and a binary "accessibility" relation $R \subseteq W \times W$. $\langle W, R \rangle$ is said to be finite if W is so. [6]

Definition 1.1.8. The notion of model is defined as follows:

- Let $\mathcal{F} = \langle W, R \rangle$ be a frame. An evaluation function v on $\mathcal{F}$ is a function

  $$v : W \times \texttt{Atm} \longrightarrow \{0, 1\}$$

  associating to each $i \in W$ and each $p \in \texttt{Atm}$ a truth-value $v(i, p) \in \{0, 1\}$.
- A model $\mathcal{M}$ is a triple $\langle W, R, v \rangle$ where $\mathcal{F} = \langle W, R \rangle$ is a frame and v is an evaluation function on $\mathcal{F}$. When that is in the case, we say that $\mathcal{M}$ is based on $\mathcal{F}$.

Definition 1.1.9. The relation $\Vdash$ holding between a model $\mathcal{M} = \langle W, R, v \rangle$, a world $i \in W$ and a formula A of $\mathcal{L}_\Box$ when 'A is forced by i in $\mathcal{M}$' is inductively defined on the degree of A:

- $i \Vdash_{\mathcal{M}} p$ iff $v(i, p) = 1$;
- $i \nVdash_{\mathcal{M}} \bot$ for any $i, \mathcal{M}$;
- $i \Vdash_{\mathcal{M}} B \to C$ iff $i \nVdash_{\mathcal{M}} B$ or $i \Vdash_{\mathcal{M}} C$;
- $i \Vdash_{\mathcal{M}} B \wedge C$ iff $i \Vdash_{\mathcal{M}} B$ and $i \Vdash_{\mathcal{M}} C$;
- $i \Vdash_{\mathcal{M}} B \vee C$ iff $i \Vdash_{\mathcal{M}} B$ or $i \Vdash_{\mathcal{M}} C$;

[4] See [HN12] for an exhaustive discussion on this most debated property of normal modal systems.

[5] [Cop02] provides a fascinating history of this interpretation of modal operators.

[6] We formally write iRj to mean that j is accessible to i.

- $i \Vdash_{\mathcal{M}} \Box B$ iff for all $j \in W$, if iRj, then $j \Vdash_{\mathcal{M}} B$.

The previous definitions are acceptable when dealing with modal logics based on a classical propositional system. When moving to the intuitionistic setting, some things need to be modified.

DEFINITION 1.1.10. An intuitionistic modal model[7] $\mathcal{M} := \langle W, \leq, R, v\rangle$ is made of

- a preorder $\leq$ on a non-empty set W;
- a binary relation R on W;
- an evaluation function $v : W \times \texttt{Atm} \to \{0, 1\}$ such that for any $i, j \in W$ and $p \in \texttt{Atm}$, if $i \leq j$, then, if $v(i, p) = 1$, $v(j, p) = 1$

The right forcing condition $\Vdash$ in the intuitionistic modal setting has been debated for a long time, and no universal agreement has been achieved, nor should be expected, since the choice heavily depends on the concepts the single logician strives to capture by an appropriate modality. Indeed, it is common to distinguish between *intuitionistic* modal logic [Sim94, PS86, FS77] and *constructive* modal logic [dPGM04, Lit14] on the basis of the expected behaviour of the modal operators – for our purposes, the $\Box$-modality only.

Part of the present work is committed to a proof-theoretic analysis of modal logics based on an intuitionistic propositional system. We will not dwell on these subtleties, so that the appropriate definition of $\Vdash$ will be given in due time for each specific logic considered.

1.1.4. CANONICAL MODELS, FINITE MODELS, BISIMULATIONS

The real interest in relational semantics, actually, is not revealed by the notion of validity of theorems in a single model, but in a whole *class of frames*.

DEFINITION 1.1.11. We say that:

- A formula A is true in a model $\mathcal{M} = \langle W, (\leq,)R, v\rangle$ iff it is forced by any $w \in W$; formally, one writes $\vDash_{\mathcal{M}} A$ iff $w \Vdash_{\mathcal{M}} A$ for any $w \in W$.
- A formula A is valid in a frame $\mathcal{F}$ iff it is true in any model based on that frame, i.e. iff if is forced by any world w.r.t. any evaluation function on $\mathcal{F}$; formally, one writes $\vDash_{\mathcal{F}} A$ iff $\vDash_{\mathcal{M}} A$ for any $\mathcal{M}$ based on $\mathcal{F}$.
- A formula A in a class of frames $\mathfrak{C}$ iff A is valid in any frame belonging to $\mathfrak{C}$; formally, one writes $\vDash_{\mathfrak{C}} A$ iff $\vDash_{\mathcal{F}} A$ for any frame $\mathcal{F} \in \mathfrak{C}$.

[7] See Sect. 2.1 for a concrete example of such a structure

Proving that a normal modal system $\mathbb{S}$ is sound and complete w.r.t. possible world semantics means establishing that the set of theorems of $\mathbb{S}$ corresponds to the set of formulas that are valid in a class of frames whose accessibility relations satisfy a specific group of (first- or higher-order) properties corresponding to the modal schemas characterising $\mathbb{S}$. It is common to prove that adequacy theorem by means of the **canonical model construction**.

THEOREM 1.1.12. *Let $\mathbb{S}$ be an axiomatic calculus for a normal modal logic. Then the following hold:*

Soundness: *If $\mathbb{S} \vdash A$, then A is valid in all the frames satisfying the properties corresponding to the characteristic schemas of $\mathbb{S}$.*

Completeness: *If $\mathbb{S} \not\vdash A$, there exists a countermodel to A which is based on a frame satisfying the properties corresponding to the characteristic schemas of $\mathbb{S}$.*

Proof sketch. Soundness is usually proven by induction on the derivation of A.

The construction of the canonical models plays the main role in proving completeness. The proof strategy can be summarised as follows:

1. Define a set of formulas Γ to be $\mathbb{S}$-maximal when it is $\mathbb{S}$-consistent, but none of its proper extension is so. Let $\mathrm{MAX}_{\mathbb{S}}$ denote the collection of $\mathbb{S}$-maximal consistent sets. It is not hard to prove that any $\Gamma \in \mathrm{MAX}_{\mathbb{S}}$ contains any formula that is $\mathbb{S}$-derivable from Γ and that it also behaves as a standard evaluation for (classical) propositional connectives.

2. An extension lemma can then be proven by a Lindenbaum construction, assuring that any $\mathbb{S}$-consistent set can be extended to an $\mathbb{S}$-maximal set, so that $\mathrm{MAX}_{\mathbb{S}} \neq \varnothing$.

3. To each system $\mathbb{S}$ a canonical model is associated, for which $W = \mathrm{MAX}_{\mathbb{S}}$. To be more precise, set

   - a canonical frame $\mathcal{F}_{\mathbb{S}} = \langle \mathrm{MAX}_{\mathbb{S}}, (\subseteq,) R_{\mathbb{S}} \rangle$, where

     $$\Gamma R_{\mathbb{S}} \Delta \quad \text{iff,} \quad \text{when} \quad \Gamma \vdash_{\mathbb{S}} \Box A, \quad \Delta \vdash_{\mathbb{S}} A \text{ for any formula } A;$$

   - a canonical model $\mathcal{M}_{\mathbb{S}} = \langle \mathrm{MAX}_{\mathbb{S}}, (\subseteq,) R_{\mathbb{S}}, v_{\mathbb{S}} \rangle$ where

     $$v_{\mathbb{S}}(\Gamma, p) = 1 \quad \text{iff} \quad \Gamma \vdash_{\mathbb{S}} p.$$

   By the extension lemma it is also provable that if $\Gamma \in \mathrm{MAX}_{\mathbb{S}}$ and $\Gamma \nvdash_{\mathbb{S}} \Box B$, then there exists a $\Delta \in \mathrm{MAX}_{\mathbb{S}}$ such that $\Gamma R_{\mathbb{S}} \Delta$ and $\Delta \nvdash_{\mathbb{S}} B$.

4. By induction on the degree of A, a truth-lemma is proven:

For any $\Gamma \in MAX_{\mathbb{S}}$ *and any formula* A, $\Gamma \Vdash_{\mathcal{M}_{\mathbb{S}}} A$ *iff* $A \in \Gamma$.

5. Consider now a formula A such that $\mathbb{S} \nvdash A$. Then $\{\neg A\}$ is $\mathbb{S}$-consistent; by the extension lemma, $\{\neg A\} \subseteq \Gamma \in \mathrm{MAX}_{\mathbb{S}}$ for some Γ, so that $\Gamma \nvdash_{\mathbb{S}} A$ and, by the truth-lemma, $\Gamma \nVdash_{\mathcal{M}_{\mathbb{S}}} A$.
6. This suffices to prove the completeness of $(\mathbb{I})\mathbb{K}$: For its normal extensions it is enough to show that the related canonical model does satisfy the required property.

⊠

For many normal modal systems, moreover, the previous adequacy theorem can be strengthened to *finite frames*. It is common to refer to that result as the **finite model property** (FMP) of a given system $\mathbb{S}$.

Proving that the property does hold for a specific $\mathbb{S}$ requires further notions that will not be necessary for the purposes of the present work, including filtration results that are highly sensitive to the logic under investigation. Notice, however, that if $\mathbb{S}$ satisfies the finite model property, it is decidable. See e.g. [CZ97, Sim94] for more details.

Let us conclude this survey by recalling a central notion in model theory for modal logic.

DEFINITION 1.1.13. Let $\mathcal{M}$ e $\mathcal{M}'$ two relational models. Let B be a non-empty binary relation between their world sets, resp. W e W'. B is said to be a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$ if, whenever wBw', we have

Atomic harmony: $x \Vdash_{\mathcal{M}} p$ iff $x' \Vdash_{\mathcal{M}'} p$ for any $p \in \mathtt{Atm}$;

Zig: if wRy, then there exists a world $y' \in W'$ such that yBy' and $w'R'y'$;

Zag: if $w'R'y'$, then there exists a world $y \in W$ such that yBy' and wRy.

If there is a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$, one says that those models are bisimilar.

Bisimulation subsumes several model-theoretic notions, including model isomorphism.

The key result about bisimilar models is indeed the following:

LEMMA 1.1.14. *Modal forcing is invariant under bisimulation.*

Proof. Let $\mathcal{M}$,$\mathcal{M}'$ be bisimilar via B. Let wBw'. A straightforward induction on the degree of A proves that

$$w \Vdash_{\mathcal{M}} A \qquad \text{iff} \qquad w' \Vdash_{\mathcal{M}'} A,$$

as desired.

⊠

1.2. Gödel-Löb provability logic

The origin of provability logic dates back to a short paper by Gödel [Göd86] where propositions about provability are formalised by means of a unary operator B with the aim of giving a classical reading of intuitionistic logic.

The resulting system corresponds to the logic $\mathbb{S}4$, and the proposition Bp is interpreted as 'p is *informally* provable' – as claimed by Gödel himself. This implies that $\mathbb{S}4$ can be considered a provability logic lacking an appropriate semantics.

At the same time, that work opened the question of finding an adequate modal calculus for the formal properties of the provability predicate used in Gödel's incompleteness theorems. That problem has been settled since 1970s for many formal systems of arithmetic by means of Gödel-Löb logic GL.

The corresponding axiomatic calculus $\mathbb{GL}$ consists of the axiomatic system for classical propositional logic, extended by the distributivity axiom schema K, the necessitation rule NR, and the axiom schema GL

$$\Box(\Box A \to A) \to \Box A \ .$$

This schema consists of a formal version of Löb's theorem, which states that an arithmetical sentence S is provable in a theory T iff in T the reflection formula $Bew(\ulcorner S \urcorner) \to S$ is provable, where $Bew(x)$ denotes the formal provability predicate for T. Indeed, Löb's theorem holds for a wide class of arithmetical theories satisfying the so-called Hilbert-Bernays-Löb (HBL) provability conditions:

(τ1) IF $\mathsf{T} \vdash S$, then $\mathsf{T} \vdash Bew(\ulcorner S \urcorner)$;

(τ2) $\mathsf{T} \vdash Bew(\ulcorner S \to T \urcorner) \to (Bew(\ulcorner S \urcorner) \to Bew(\ulcorner T \urcorner))$;

(τ3) $\mathsf{T} \vdash Bew(\ulcorner S \urcorner) \to Bew(\ulcorner Bew(\ulcorner S \urcorner) \urcorner)$;

(τ4) $Bew(\ulcorner S \urcorner)$ is in Σ_1;

(τ5) If S is in Σ_1, then $\mathsf{T} \vdash S \to Bew(\ulcorner S \urcorner)$.

The formal analogy with the axiomatic calculus $\mathbb{GL}$ is clear.

The semantic counterpart of that calculus is – through the Kripke formalism – the logic of irreflexive transitive finite frames. Moreover, the calculus can be interpreted arithmetically in a sound and complete way. In other terms, $\mathbb{GL}$ solves the problem raised in Gödel's paper by identifying a propositional formal system for provability in all arithmetical theories that satisfies the previously mentioned HBL conditions.

Published in 1976, Solovay's arithmetical completeness theorem [Sol76] is in this sense a milestone result in the fields of proof theory and modal logic.[8] As $\mathbb{GL}$ is arithmetically complete, it is capable of capturing and identifying *all relevant properties of formal provability for arithmetic* in a very simple system, which is decidable and neatly characterised.

Such a deep result, however, uses in an essential way the *modal* completeness of $\mathbb{GL}$: Solovay's technique basically consists of an arithmetisation of a relational countermodel for a given formula that is not a theorem of $\mathbb{GL}$, from which it is possible to define an appropriate arithmetical formula that is not a theorem of the mathematical system.

In contemporary research, this is still the main strategy to prove arithmetical completeness for other modalities for provability and related concepts, in particular for interpretability logics. In spite of this, for many theories of arithmetic – including Heyting Arithmetic – this technique cannot be applied, and no alternatives are currently known.

In this section, the basic definitions and properties of GL are recalled. This contents gives an elementary background for Chapters 4–6.

1.2.1. Axiomatisation and relational semantics

Definition 1.2.1. The axiomatic calculus $\mathbb{GL}$ is given by

- axiom schemas for classical propositional logic;
- schema K $\quad \Box(A \to B) \to (\Box A \to \Box B)$;
- schema GL $\quad \Box(\Box A \to A) \to \Box A$, also called Löb schema;

together with the following inference rules:

- MP $\quad \dfrac{A \to B \qquad A}{B}$;
- NR $\quad \dfrac{A}{\Box A}$.

It is not hard to see that $\mathbb{GL} \vdash \Box A \to \Box\Box A$. Actually, it is possible to give an equivalent definition of $\mathbb{GL}$ – that we will call $\mathbb{K4LR}$ – which extends $\mathbb{K}$ by means of the schema

$$4: \quad \Box A \to \Box\Box A$$

together with the Löb rule

[8] Actually, four different schools originated from the first years of research leading to the arithmetical completeness theorem: In the US, George Boolos, Craig Smòrynski and Robert Solovay were the principal investigators on provability logic; in the Netherlands, Albert Visser and Dick de Jongh led a very active research group; in the USSR, Sergei Artemov started an extremely prolific school on provability logic and ordinal analysis; in Italy, Roberto Magari, Franco Montagna, Giovanni Sambin e Silvio Valentini focused their research on algebraic and proof-theoretic treatments of GL.

$$\frac{\Box A \to A}{A}\, LR$$

On the semantic side, GL is adequate to the class of relational frames which are **irreflexive, transitive and finite**, which we denote by *ITF*.

THEOREM 1.2.2. *The following hold:*

(**Soundness**) *If $\mathbb{GL} \vdash A$, then A is valid in any frame in ITF.*

(**Completeness**) *If $\mathbb{GL} \nvdash A$, then there exists a countermodel for A in ITF.*

Proof. Soundness is proven by induction on the derivation of A.

There are several way to prove completeness for $\mathbb{GL}$: See [Boo95] and its implementation described in Chapter 6 below for a proof based on a variation of the canonical model construction.

$\boxtimes$

Notice, however, that it is possible to prove both a weaker and a stronger completeness result: $\mathbb{GL}$ is complete w.r.t. the class of transitive and Noetherian frames,[9] which *ITF* is a proper subclass of, and w.r.t. the class of irreflexive finite trees, which are a proper subclass of *ITF*.

1.2.2. ARITHMETICAL REALISATION

We want now to make clear how a modal formula $\Box A$ can be interpreted as asserting the formal provability of an arithmetical sentence corresponding to A. This is achieved by means of the following

DEFINITION 1.2.3. Let T be an arithmetical theory over a classical basis satisfying the provability conditions HBL. A realisation of a modal formula into an arithmetical *sentence* of T consists of a function $*$ that commutes with the propositional connectives and such that

$$(\Box A)^* := Bew(\ulcorner A^* \urcorner),$$

where the mapping $\ulcorner \cdot \urcorner$ denotes the Gödelianisation of arithmetical formulas, and $Bew(x)$ is the standard provability predicate for T.[10]

Proving that any realisation preserves the theorems of $\mathbb{GL}$ is a rather easy task.

LEMMA 1.2.4 (Arithmetical soundness). *Let T be a classical arithmetical theory satisfying the provability conditions HBL. If $\mathbb{GL} \vdash A$ then $T \vdash A^*$ for any realisation $*$.*

[9] We say that a relation R is Noetherian on W when for any $X \subseteq W$ there exists a $w \in X$ such that for no $x \in X$ wRx.

[10] Refer to e.g. [Boo95, Ch. 2] for more details.

Proof. Straightforward induction on the height of the derivation in $\mathbb{K}4\mathbb{GL}$.

⊠

This soundness lemma suffices to study some phenomena of formal arithmetical theories from an abstract point of view, by considering their modal counterparts. For instance, by working in $\mathbb{GL}$ we can state the modal version of very well-know meta-mathematical results.

LEMMA 1.2.5. *The following hold:*

(i) Formal second incompleteness theorem:

$$\mathbb{GL} \vdash \Box\Diamond\bot \to \Box\bot;$$

(ii) Arithmetical ignorance about unprovability statements:

$$\mathbb{GL} \vdash \Box\bot \leftrightarrow \Box\Diamond p;$$

(iii) Unprovability of consistency implies undecidability of consistency:

$$\neg(\Box\Box\bot) \to \neg(\Box\neg\Box\bot) \wedge \neg(\Box\neg\neg\Box\bot);$$

(iv) Undecidability of Gödel's sentence:

$$\Box(A \leftrightarrow \neg\Box A) \wedge \neg\Box\Box\bot \to \neg\Box A \wedge \neg\Box\neg A;$$

(v) Gödel's sentence is equiconsistent with a consistency statement:

$$\Box(A \leftrightarrow \neg\Box A) \leftrightarrow \Box(A \leftrightarrow \neg\Box\bot).$$

Proof. All the items are proven by direct reasoning within the axiomatic calculus.

⊠

Another central result about Gödel-Löb logic is the fixpoint theorem, which provides an abstract, structural, and extensional version of self-reference in arithmetical theories.[11]

THEOREM 1.2.6 (Modal fixpoint, Sambin 1976, Smorynski 1979). *Let us say that a modal formula A is modalised in $p \in \mathtt{Atm}$ iff all occurrences of p in A are within the scope of $\Box$. Furthermore, let us define $\boxdot B := B \wedge \Box B$. Then for any modal formula A modalised in p, it is possible to explicitly find a modal sentence H made of the same atoms of A except for p such that*

$$\mathbb{GL} \vdash \boxdot(p \leftrightarrow A) \leftrightarrow \boxdot(p \leftrightarrow H).$$

[11] The original proof is due to Sambin and, independently, de Jongh, but several other proofs have been carried out over the years, including some of constructive nature.

Proof. See e.g. [Boo95, Ch. 8].

⊠

Soundness of Gödel-Löb logic allows to reason in a simple modal system to prove relevant facts about arithmetic. However, GL is capable of providing relevant information about arithmetical theories even in an indirect way, namely by considering what is not provable in that modal system. This is possible thanks to Solovay's arithmetical completeness theorem:

THEOREM 1.2.7 (**Arithmetical completeness, Solovay 1976**). *For any modal formula A, if $\mathbb{GL} \nvdash A$, then there exists an arithmetical realisation $*$ such that $\mathsf{T} \nvdash A^*$ for any classical T satisfying the provability conditions HBL.*

Proof. Refer to [Sol76], or [Boo95, Ch. 9].

⊠

The proof of Solovay's theorem makes essential use of the modal completeness of $\mathbb{GL}$: The proof strategy consists of an arithmetical endcoding of a countermodel for A, and a construction of an appropriate realisation $*$ from that encoded countermodel via Gödel-Carnap diagonal lemma for arithmetical theories.

More prominently, it is relevant to notice that this strategy relies on the *classical nature* of the theory T. At present time, no other proof strategy is known for proving arithmetical completeness of modal logic for arithmetical theories. This implies, in particular, that a provability logic for intuitionistic arithmetic is still unknown. However, recent advances on that field seem promising: For instance, it is known that the intuitionistic version of GL – i.e. $\mathbb{IGL}$, the axiomatic system obtained by adding the rule NR together with schemas K and GL to intuitionistic propositional calculus – is *not* the provability logic for Heyting arithmetic HA.[12] But if we consider the intuitionistic version of strong Löb logic – that we recall in Chapter 4 – then we obtain a modal system that is sound and complete w.r.t. $\mathsf{HA} + (A \to \Box_{(\mathsf{HA}^s)*}A)$, where HA^s is an axiomatisation of intuitionistic arithmetic with only small axioms, and $(\mathsf{HA}^s)*$ is the unique theory such that provably in HA: $(\mathsf{HA}^s)* = (\mathsf{HA}^s) + (A \to \Box_{(\mathsf{HA}^s)*}A)$.[13]

Furthermore, arithmetical completeness of $\mathbb{GL}$ can be extended to a modal treatment of different mathematical notions by means of proper extensions of the logic and the language underlying Gödel-Löb system: More on this is described in Chapter 5.

Finally, it is worth noticing that proof theory for provability logic has been considered for long very complex. It suffices to mention that the history of

[12] While the present thesis was already under review, the preprint [Moj22] proposed a definitive solution to the open problem for HA. See also the related footnote at the beginning of Chapter 4.

[13] See [VZ19] for the definitions and the main result.

the design of an analytic sequent calculus for GL – beginning with Valentini's proof of cut elimination for calculus GLS [Val83] – has been characterised by controversy, erroneous claims and alleged proof gaps. The situation changed with the advent of internalisation techniques, that we briefly recall in Section 1.3.2.[14] The theorem prover for GL that we describe in Chapter 6 is based indeed on a sequent calculus designed by an explicit internalisation of the relational semantics that we recalled in the previous pages.

1.3. SEQUENT CALCULI FOR NON-CLASSICAL LOGICS

In very abstract terms, a *proof system* consists of a set of starting formal expressions together with inference rules. Its principal aim is to find proofs of valid expressions w.r.t. a given logic L. A proof (or derivation) in a proof system is obtained by application of the inference rules to starting expressions, followed by further application of the inference rules to the conclusion, and so on, recursively. A theorem (or lemma) in such a system is the formal expression obtained after a finite run of the procedure just sketched.

These definitions capture the axiomatic calculi of Section 1.1.2, and the associated derivability relation $\vdash_{\mathbb{S}}$.

The proof-theoretic paradigm behind axiomatic systems could be called *synthetic*: Proof search in such systems is not guided by the components of the formula one wishes to prove. The human prover – as well as an hypothetical proof assistant dealing with a formal derivability relation w.r.t. this paradigm – has to guess both the correct instances of the axiom schemas and the correct application order of inference rules required in the proof. No explicit tip is given to her for mechanically finding a proof. Thus, this kind of proof systems has automation shortcomings that a naive computerisation is unable to solve.[15]

A better paradigm is provided by sequent calculi, introduced first in [Gen35a, Gen35b]. That work marks the advent of structural proof theory and the definite shift from investigations in synthetic proof systems to *analytic* ones.

1.3.1. G3cp SEQUENT CALCULUS

Gerhard Gentzen's original calculi have been further refined in [Ket45] and [Kle52] into the so-called G3-style systems.

In those systems, a **sequent** is a formal expression with shape

$$\Gamma \Rightarrow \Delta,$$

[14] In any case, this has happened in parallel with the development of different proof strategies for proving cut elimination from standard Gentzen style sequent calculi, and a more fine-grained analysis of proofs for the original systems thanks to the systematic use of proof assistants, as witnessed by the recent [GRS21].

[15] See also Section 6.2 for some remarks on proof search w.r.t. an explicit implementation of $\vdash_{\mathbb{GL}}$.

where Γ, Δ are finite multisets – i.e. finite lists modulo permutations – of formulas of a given language. The symbol $\Rightarrow$ reflects in the object language the deducibility relation at the meta-level. Γ is called the **antecedent** of the sequent; Δ is its **consequent**.

A derivation in a G3-style sequent calculus is a finite rooted tree labelled with sequents such that:

- its leaves are labelled by **initial sequents** (the starting formal expressions of the abstract proof system);
- its intermediate nodes are labelled by sequents obtained from the sequent(s) labelling the node(s) directly above by a correct application of an inference rule of the calculus;
- its root is the conclusion of the derivation, and it is called the **end-sequent**.

Figure 1.1 summarises the calculus G3cp for classical propositional logic. For each rule, one distinguishes:

- its **main formula**, which is the formula occurring in the conclusion and containing the logical connective naming the rule;
- its **active formulas**, which are the formulas occurring in the premise(s) of the rule;
- its **context**, which consists of the formulas occurring in the premise(s) *and* the conclusion, untouched by the rule.

G3-style systems are the best available option for (efficiently) automating decision procedures: Once an adequate – i.e. sound and complete – G3-calculus for a given logic L has been defined, in order to decide whether a formula is a theorem of L it suffices to start a *root-first* proof search of that very formula in the related G3-calculus.

This is so because, by design, good G3-style systems satisfy the following desiderata:

1. **Analyticity:** Each formula occurring in a derivation is a subformula of the formulas occurring lower in the derivation branch. This means that *no guesses are required* to the prover when developing a formal proof in the G3-calculus;
2. **Avoiding of backtracking:** For each rule of the system, derivability of the conclusion implies the derivability of the premise(s). *This invertibility of all the rules is what avoids backtracking* on the proof search. It also means that at each step of the proof search procedure *no bit of information gets lost*, so that the tentative construction of *one* derivation tree is enough to decide derivability of a sequent;

Initial sequents:

$p, \Gamma \Rightarrow \Delta, p$

Propositional rules:

$$\frac{}{\bot, \Gamma \Rightarrow \Delta}\ \mathcal{L}\bot$$

$$\frac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\wedge \qquad \frac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B}\ \mathcal{R}\wedge$$

$$\frac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\vee \qquad \frac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B}\ \mathcal{R}\vee$$

$$\frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\neg \qquad \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ \mathcal{R}\neg$$

$$\frac{\Gamma \Rightarrow \Delta, A \qquad B, \Gamma \Rightarrow \Delta}{A \to B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\!\to \qquad \frac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \to B}\ \mathcal{R}\!\to$$

Figure 1.1: Rules of the calculus G3cp

3. **Termination:** Each proof search must come to an end. If the final state of the procedure does generate a derivation, the end-sequent is a theorem indeed; otherwise, it is generally possible to extract a *refutation* of the sequent from the failed proof search.[16]

G3cp does satisfy these desiderata because of its *structural properties*. By this term, it is common to denote a small set of meta-results that can be condensed in a "canonical form theorem" for G3cp-derivations: Each derivation in G3cp can be rearranged so that it acquires a well-defined structure. The structure itself is what guides the prover in the proof search of a given formula.

THEOREM 1.3.1. *Let us say that a rule of a sequent calculus is admissible if from the derivability of its premise(s) the derivability of its conclusion follows.*

Then, for G3cp *the following hold:*

- *Generalised initial sequents $A, \Gamma \Rightarrow \Delta, A$ are derivable.*

- *The structural rules of weakening*

[16] Notice, however, that sometimes invertibility of a rule could break termination of the proof search, as witnessed by $\mathcal{L} \to$ in G3ip for intuitionistic propositional logic [TS00].

$$\frac{\Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta}\,\mathcal{L}Wk \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, A}\,\mathcal{R}Wk$$

are admissible.

- *The structural rules of contraction*

$$\frac{A, A, \Gamma \Rightarrow \Delta}{A, \Gamma \Rightarrow \Delta}\,\mathcal{L}Ctr \qquad \frac{\Gamma \Rightarrow \Delta, A, A}{\Gamma \Rightarrow \Delta, A}\,\mathcal{R}Ctr$$

are admissible.

- *The cut rule*

$$\frac{\Gamma \Rightarrow \Delta, A \qquad A, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,Cut$$

is admissible.

Proof. Refer to e.g. [NvP08, Ch. 3], or [TS00, Ch. 3-4].

⊠

Admissibility of *Cut* deserves some further remarks.
It is rather usual to define a G3-style sequent calculus by considering *Cut* as the only structural rule of the system. This is so because that rule corresponds to the argumentation by Modus Ponens, i.e. to the rule MP of the synthetic paradigm. Having *Cut* among the defining rules of the calculus makes the proof of completeness w.r.t. an already given axiomatic system much easier. Afterwards, admissibility of *Cut* is proven to guarantee analyticity of the sequent calculus.

There are, however, at least two ways to prove that *Cut* is indeed admissible.
The easiest way is to prove that the cut-free version of the G3-system for a logic L is complete w.r.t. the semantics for L itself. This method establishes admissibility of *Cut* as a **normal form theorem** for G3-derivations.
A stronger result is obtained by *effectively transforming* each derivation in the full G3-style sequent calculus – including the cut rule – into a derivation where no application of *Cut* occurs: This version of admissibility is called **cut-elimination theorem**, or, in Gentzen's own words, *Hauptsatz*.

Thus, the algorithmic version of admissibility of cut should be rephrased as follows:

THEOREM 1.3.2 (Cut-elimination for G3cp). *A constructive procedure can be defined to transform any derivation of a sequent* $\Gamma \Rightarrow \Delta$ *in* G3cp *into a cut-free derivation of the same sequent in* G3cp.

There is nothing special in G3cp: The analogous of Theorems 1.3.1 and 1.3.2 do hold for first-order G3c as well as for its intuitionistic counterpart. More generally, the seminal work collected [NvP08, Ch. 6] shows that any (co)geometric theory can be formalised in a G3-style sequent calculus satisfying those excellent structural properties.

Chapter 5 contains a detailed proof of cut-elimination for a G3-style sequent calculus for a mathematical modal logic, and makes precise all the previous remarks by considering that specific case study.

1.3.2. Formal relational semantics

Besides the previously mentioned "computational" properties, G3-style sequent calculi could be expected to satisfy more principled desiderata.

Among those, **modularity** is a property that fits the spirit of (normal) modal logics: The rules of G3-systems for those logics should be designed so that *stronger logics should be captured by only adding rules to weaker systems*, without any further rearrangement of the calculus. That requirement might clash with another computational desideratum, namely **efficiency**: A resource aware proof search should provide an *optimal decision procedure* for a given logic L w.r.t. the "best" complexity classification for L.

Satisfaction of all those desiderata by a pure Gentzen-style sequent calculus has been considered for long almost a mirage for logics involving extra-mathematical reasoning.[17]

Times have changed with the advent of *internalisation* techniques of semantic notions in sequent calculi for non-classical logics.

The starting point of that perspective is still the basic G3-paradigm, but the formalism of sequent systems is extended either by

- enriching the language of the calculi themselves (**explicit internalisation**); or by
- enriching the structure of sequents (**implicit internalisation**).

The implicit approach adds structural connectives to sequents, other than '$\Rightarrow$' and commas.

Hypersequents, in more detail, enrich sequents by the connective '|' to handle several sequents in parallel. The rules are defined so that the occurring sequents may interact with each other.[18]

Hypersequent calculi were fully exploited first in [Avr96], on the basis of preliminary and embryonic formalisations in [Kri59], [MF74, Min92], and [Pot83]. Further refinements – not limited to modal logics – include [Res05], [Kur13], [MS14], and [GLO$^+$17].

[17] In spite of this, several advances have been made in investigating both limits and potentiality of that formalism, as witnessed by [Iem19].

[18] The connective '|' is informally interpreted as a disjunction operator at the meta-level.

That formalism naturally generalises to **nested sequents**, that enrich the structure of sequents by the structural connective '[ ]', allowing the design of inference rules operating at any deep in the entire tree of sequents defined by means of the additional connective itself.

Nested sequents were introduced first in [Kas94], and independently in [Brü09], and in [Pog09].[19] Nested sequent calculi for other non-classical logics have since then further developed in e.g. [OP15].

Most of G3-style calculi obtained this way provide in general very efficient decision procedures for the related logics, but they are sometimes rather hard to design, and might lack the desideratum of modularity.

The situation is reversed by the explicit approach.
Implicit internalisation uses specific items to represent semantic elements; the formulas of the basic language are then **labelled** by those items, and have shape e.g. $x : A$. That expression formalises the forcing relation of Def. 1.1.9, and classical propositional rules operate within the scope of labels. Furthermore, to handle modal operators, the antecedent of any sequent may now contain **relational atoms** of shape xRy, or any other expression borrowed from the semantics for the logic under investigation.

The rules for the modalities formalise the forcing condition for each modal connective. For instance, the labelled sequent calculus G3K is defined by the rules in Figure 1.2.

Extensions of the minimal normal modal logic K are obtained by rules for relational atoms, formalising the characteristic properties of each specific extension. For instance, the system G3K4 for the logic K4 is defined by adding to G3K the rule

$$\frac{xRz, xRy, yRz, \Gamma \Rightarrow \Delta}{xRy, yRz, \Gamma \Rightarrow \Delta}\ \textit{Trans}$$

Labelled sequent calculi do satisfy in general the basic desiderata of G3-style systems, and are rather modular. However, since analyticity holds in a less strict version[20] than the subformula principle mentioned in the previous definition, termination of proof search is sometimes hard to prove, and in many cases produces complexity results far from being optimal.

Their origin dates back to [Kan57], followed by the refinements in [Kri65], [Fit13], and [Gab96]. However, labelled sequent calculi have established as a well-structured methodology after [Neg05]. Extensions, refinements, and further results have since then obtained for a plethora of logics.[21]

In this thesis, internalisation is considered only in its explicit version, for reasons that appear clear in Chapter 5. However, the connections between

[19] The latter deals with nested sequents under the name 'tree-hypersequents', and a uniform treatment of many normal modal logics is presented in [Pog10].

[20] See Section 5.5.

[21] Refer to e.g. [OC18, Orl21, NO19], [Neg17, NP21], [Pog16], [DON18], [GNOR18, GNO18], [GO21].

Initial sequents:

$x : p, \Gamma \Rightarrow \Delta, x : p$

Propositional rules:

$$\frac{}{x : \bot, \Gamma \Rightarrow \Delta}\ \mathcal{L}\bot$$

$$\frac{x : A, x : B, \Gamma \Rightarrow \Delta}{x : A \land B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\land \qquad \frac{\Gamma \Rightarrow \Delta, x : A \qquad \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \land B}\ \mathcal{R}\land$$

$$\frac{x : A, \Gamma \Rightarrow \Delta \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \lor B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\lor \qquad \frac{\Gamma \Rightarrow \Delta, x : A, x : B}{\Gamma \Rightarrow \Delta, x : A \lor B}\ \mathcal{R}\lor$$

$$\frac{\Gamma \Rightarrow \Delta, x : A}{x : \neg A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\neg \qquad \frac{x : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \neg A}\ \mathcal{R}\neg$$

$$\frac{\Gamma \Rightarrow \Delta, x : A \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \to B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\!\to \qquad \frac{x : A, \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \to B}\ \mathcal{R}\!\to$$

Modal rules:

$$\frac{y : A, xRy, x : \Box A, \Gamma \Rightarrow \Delta}{xRy, x : \Box A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\Box \qquad \frac{xRy, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \Box A}\ \mathcal{R}\Box_{(y!)}$$

where the annotation $(y!)$ in $\mathcal{R}\Box$ states that the label y does not occur in Γ, Δ.

Figure 1.2: Rules of the calculus G3K

the two approaches are being investigated in the literature: The reader is referred to [Fit12], [KL16], [GNO18], and [GR12].

1.4. Natural deduction for propositional logic

Natural deduction calculi are, among the proof systems currently available, those that are closest to the informal reasoning of mathematical practice.

They have been developed by Gentzen as a formal model for the notion of *deduction under hypotheses*. Systems of natural deduction for classical and intuitionistic logics have been introduced first in his [Gen35a], and since then are usually denoted by NK and NJ, respectively.

The key feature of these systems is the *absence of logical axioms* from the calculus: The whole deductive apparatus only rests on inference rules that encode the atomic deductive steps involved in any logical argument. Furthermore, these rules reflect the operational meaning of the logical connectives of a given base language. Indeed, to each connective, a pair of rules is associated:

- The **elimination rule** instructs the prover on how to use in a formal proof a formula having that connective as main logical operator;
- The **introduction rule** instructs the prover on how to construct a deduction of a formula having that connective as main logical operator.

In principle, these constitute the (formal model of the) whole deductive engine behind the great leap that characterises the practice of arguing from premisses to conclusions that in everyday mathematics is called 'proof'. The same is clearly achieved also by the synthetic paradigm behind axiomatic calculi. However, what makes the real difference here is that Gentzen's natural deduction determines a new paradigm by which it is not only possible to assure that the conclusion one reaches in a proof is correct, but also to justify – and to make explicit by formalising it in a natural way – *how* the correct conclusion has been reached from the premises – i.e. to trace the logical connection from the premises to the conclusion by recurring, as an internal justificatory tool, to the actual use of the logical connectives in the mathematical practice.

In these sections we recall the rules of natural deduction for intuitionistic and classical logics. Our focus will be on the propositional fragment of NJ, since in subsequent chapters we will build on top of that calculus further systems for modal logics. Moreover, we discuss here the structural analysis of formal proofs in this paradigm, and collect the main results from the existing literature on the topic, starting with the notion of normal deduction, due to Dag Prawitz [Pra65, Pra71].

1.4.1. Basic natural deduction systems

As previously stated, there are no axioms in a natural deduction system. The starting point of any formal proof is thus an assumption: Any formula A is per se a proof, whose premise and conclusion are A itself. This starting points are then developed according to the rules in Figure 1.3, defining the propositional fragment of NJ, here denoted by NJp. In those rules,[22] Γ denotes a set of (occurrences of) formulas, which are called active assumptions of the deduction. Notice that in $\to\mathcal{I}$ and $\vee\mathcal{E}$ some assumptions are bracketed and labelled by numbers that are repeated on the side of the rule: This formalism denotes the discharge of assumptions. In NJp several as well as

[22] We omitted rules for negation, since we can define $\neg A$ as $A \to \bot$.

$$\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\end{matrix} \qquad \begin{matrix}\Delta\\ \vdots\\ B\end{matrix}}{A \wedge B}\,\wedge\mathcal{I} \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ A \wedge B\end{matrix}}{A}\,\wedge\mathcal{E}_1 \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ A \wedge B\end{matrix}}{B}\,\wedge\mathcal{E}_2$$

$$\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\end{matrix}}{A \vee B}\,\vee\mathcal{I}_1 \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ B\end{matrix}}{A \vee B}\,\vee\mathcal{I}_2 \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ A \vee B\end{matrix} \qquad \begin{matrix}\Delta, [A]^1\\ \vdots\\ C\end{matrix} \qquad \begin{matrix}\Theta, [B]^2\\ \vdots\\ C\end{matrix}}{C}\,\vee\mathcal{E}{:}1{,}2$$

$$\dfrac{\begin{matrix}\Gamma, [A]^1\\ \vdots\\ B\end{matrix}}{A \to B}\,\to\mathcal{I}{:}1 \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ A \to B\end{matrix} \qquad \begin{matrix}\Delta\\ \vdots\\ A\end{matrix}}{B}\,\to\mathcal{E}$$

$$\dfrac{\begin{matrix}\Gamma\\ \vdots\\ \bot\end{matrix}}{A}\,\bot_J \qquad \dfrac{}{\top}\,\top_J$$

Figure 1.3: Rules for NJp

no occurrences of the same assumption can be labelled by the same number, and so discharged by a single application of the rule. These possibilities correspond to the structural rules of contraction (multiple discharge) and weakening (vacuous discharge) in Gentzen's sequent calculi [NvP08].

Full NJ is obtained by adding the following rules for first-order quantifiers:

$$\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A\end{matrix}}{\forall x.A}\ \forall\mathcal{I}_{(x!)} \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ \forall x.A\end{matrix}}{A[x:=t]}\ \forall\mathcal{E}$$

$$\dfrac{\begin{matrix}\Gamma\\ \vdots\\ A[x:=t]\end{matrix}}{\exists x.A}\ \exists\mathcal{I} \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ \exists x.A\end{matrix} \qquad \begin{matrix}\Delta,[A]^1\\ \vdots\\ C\end{matrix}}{C}\ \exists\mathcal{E}{:}1_{(x!)}$$

In $\forall\mathcal{I}$ the side condition $(x!)$ imposes that x must not occur free in the assumptions of Γ. Similarly, for $\exists\mathcal{E}$ the side condition requires that x must not occur free in C and in Δ.

NK(p) extends NJ(p) by the classical rule for absurdity:

$$\dfrac{\begin{matrix}[\neg A]^1,\Gamma\\ \vdots\\ \bot\end{matrix}}{A}\ \bot_C : 1$$

Finally, the identity predicate can be handled by adding the following rules to the basic system, be it classical or intuitionistic, this way obtaining $\mathsf{NK}^=$ or $\mathsf{NJ}^=$, respectively:

$$\dfrac{}{t=t}\ {=}\mathcal{I} \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ A[x:=t]\end{matrix} \qquad \begin{matrix}\Delta\\ \vdots\\ t=s\end{matrix}}{A[x:=s]}\ {=}\mathcal{E}$$

1.4.2. Normalisation

Gentzen's paradigm succeeds in endowing the notion of proof with a precise and natural mathematical structure, namely that of a derivation tree having active assumptions as leaves, the conclusion of the proof as root, and each intermediate node labelled by a formula occurrence derived from the nodes above it by a correct application of the inference rules.

Could one say something mathematically relevant about these structures? The first question to address is whether it is possible to find a simplest structural arrangement of a given deduction. Indeed, one of the key features of deductions in Gentzen's paradigm is that branches can be grafted in correspondence of a formula occurring as the root of a given deduction, and as an active assumption of another one. This is a highly practical move that captures the mathematician's common way of proving complex theorems: She achieves the goal by proving intermediate lemmas. In sequent calculi, grafting is localised by the *Cut* rule. Cut elimination assures then that any derivation of a sequent can be rearranged into a derivation which uses *Cut* in no place. Therefore, it can be considered a theorem assuring the existence of *canonical derivations* in sequent calculi.

An analogous canonical form theorem holds for deductions in $\mathsf{NJ(p)}^{=}$ and $\mathsf{NK(p)}^{=}$: It was established in [Pra65] in terms of a *normalisation process* of deductions in those systems. Whenever a deduction f_1 is grafted onto another one, say f_2, a structural *detour* may occur: It is possible that the conclusion of f_1 has been obtained by an $\mathcal{I}$-rule, while the corresponding assumption of f_2 is the major premise of an $\mathcal{E}$-rule. The resulting deduction lacks a certain directedness, and therefore its structure is unnecessarily complex.

The main aim of normalisation is to eliminate this kind – as well as more general forms – of useless complexities in the structure of deductions, by manipulating them until simplest and most direct deductions are obtained. This manipulation is made precise by the simplification steps – also called conversions or proof-rewritings, see Section 1.5.1 – of NJp-deductions summarised in Figure 1.4.[23]

If no simplification step needs to be performed in a deduction, then we will call that deduction **normal.** The main question that structural proof theory may ask about deductions is then the following:

> If $\Gamma \vdash A$ in a given natural deduction calculus, is it possible to have a normal deduction in the same calculus of A from Γ – i.e. without detours and permutations?[24]

The answer is: Yes, definitely. But there are several ways to reach such a positive answer. The simplest justification for the latter is given by showing that, whenever A is deducible from Γ, it is always possible to find a normal deduction, without giving an explicit instance of such a canonical formal proof. This is a weakest – and non-constructive – normalisation result, that is usually called **normal form theorem**.

A **normalisation theorem**, on the contrary, would perform an algorithmic procedure on any deduction of A from Γ to explicitly construct a normal deduction of the same conclusion from the same assumptions. By involving a procedure, a proof-theorist might further wonder whether the order of the

[23] For the other systems, further simplifications are required: See [Pra71] for their definition.

[24] Notice that the set of simplifications can be made smaller if some connectives can be defined by others. Thus, for NKp one could consider only some detours and the permutations for the $\vee$, $\bot$-free language. However, this cannot happen for NJp, since intuitionistic connectives are not interdefinable. This implies that proving normalisation of deductions in NKp is much easier than in NJp, and the structure of normal NKp-deductions is also easier to identify than that of normal NJp-deductions. See immediately below and Section 1.4.3.

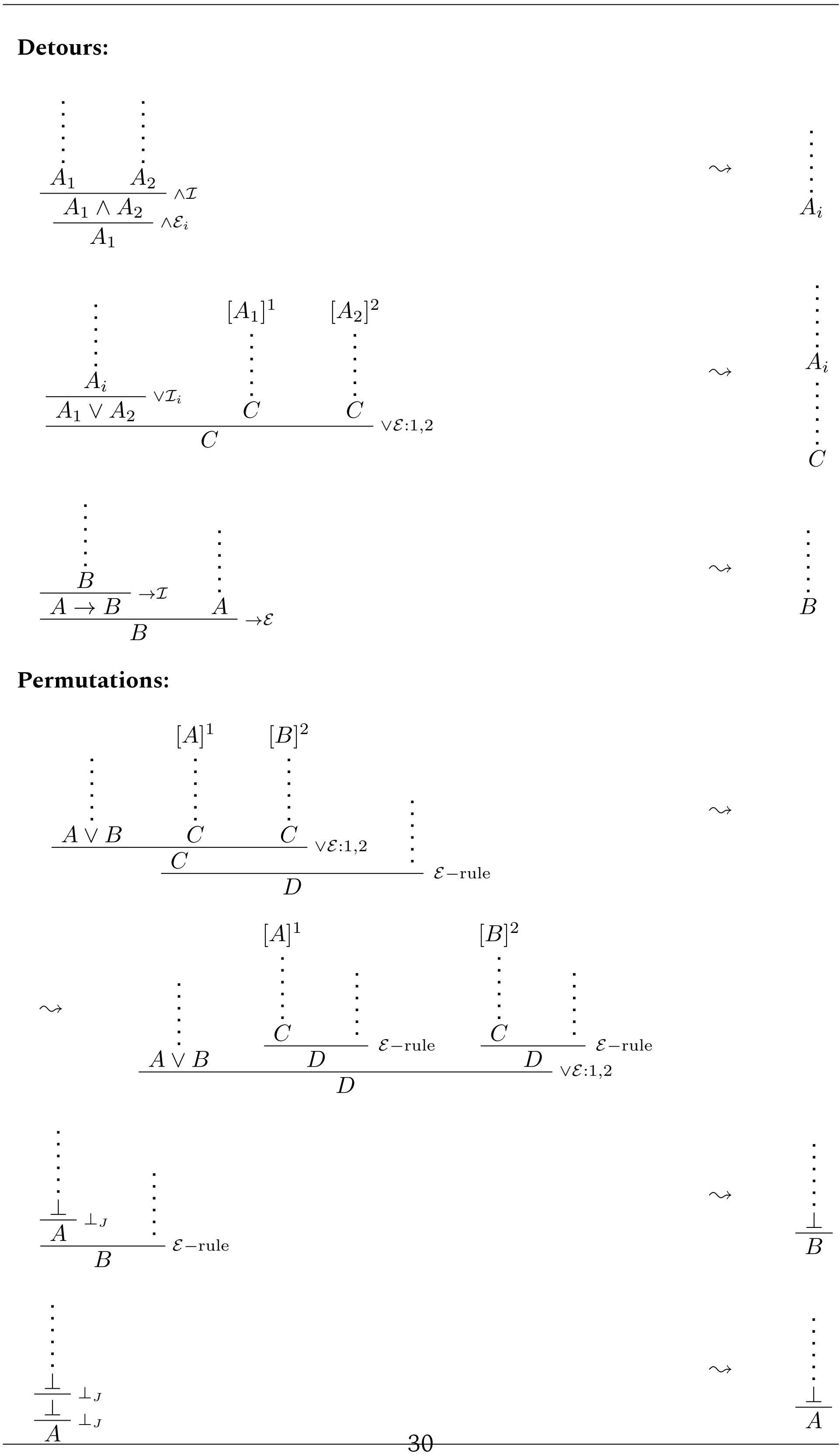


Figure 1.4: Simplifications for NJp-deductions

single algorithmic steps is important for achieving the desired normal deduction from the input deduction. A **strong normalisation theorem** would guarantees that the termination of the normalisation algorithm is independent from the order of the single algorithmic steps.[25]

For $\mathsf{NJ}(\mathsf{p})^{=}$ and $\mathsf{NK}(\mathsf{p})^{=}$ we know that the latter strongest result does hold:

THEOREM 1.4.1 (Strong normalisation, Prawitz 1965–71). *Given a deduction of A from Γ in $\mathsf{NJ}(\mathsf{p})^{=}$ ($\mathsf{NK}(\mathsf{p})^{=}$) every simplification sequence based on the conversions in Figure 1.4 (and further rewritings for first-order quantifiers) terminates in a normal deduction.*

Proof. Refer to [Pra71].

⊠

1.4.3. ANALYTICITY

As stated in Section 1.3.1, cut elimination assures the analyticity of sequent calculi. Similarly, the (strong) normalisation theorem guarantees that any deduction in $\mathsf{NJ}(\mathsf{p})^{=}$ (or $\mathsf{NK}(\mathsf{p})^{=}$) can be turned into a normal deduction; an inspection of the inference rules reveals that normal deductions share a single and precise structure (Figure 1.5).

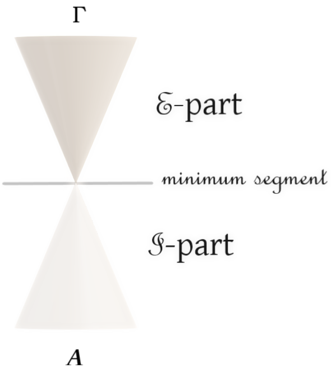


Figure 1.5: The regular structure of a normal deduction of A from Γ in NJ.

With no regards to Γ and A, any normal deduction of A from Γ consists of a first part made of a (possibly empty) sequence of elimination rules, followed by a (possible) part consisting of an application of $\bot_J$; and finally a part made of a (possibly empty) sequence of introduction rules.[26]

[25] Sometimes, one requires also that the normalisation algorithm is **confluent**, i.e. that for any input deduction there exists a unique normal deduction as output of the normalisation process.

[26] The structure is in fact a bit more complex than this: See e.g. Section 2.3.1 for a precise definition of these parts, based on [Pra65].

Thus we might loosely say that according to the natural deduction model we can always start the deductive process by analysing the constituents of the assumptions Γ and then proceed in building complex formulas by combining those constituents until the conclusion A is reached. That informal account has a most relevant structural analogous:

THEOREM 1.4.2 (**Subformula principle, Prawitz 1965–71**). *Every formula occurring in a normal deduction is a subformula of the conclusion, or of some active assumption.*

Therefore, good natural deduction systems are expected to share the analyticity property with good sequent calculi, since, as for the latter, such a property is essential for proving by purely proof-theoretic methods relevant meta-theorems of the former, including consistency, decidability, as well as more logic-specific properties. That the subformula principle does hold for $\mathsf{NJ(p)}^{=}$ and $\mathsf{NK(p)}^{=}$ is shown by a little thought on the structure of normal deduction, and a proof sketch was given first in [Pra65]. Developing a more precise proof requires, however, some care and several preliminary lemmas of combinatorial nature. In the first chapter of the present work we present a detailed proof of the subformula principle for a conservative extension of NJp dealing with modal operators; in it, all the intermediate steps are made explicit.

1.4.4. PERSPECTIVES FROM MATHEMATICAL PHILOSOPHY

Normalisation of natural deduction is far from a purely technical result: It is a glowing instance of a mathematical theorem that is able to shed light – and to justify also – a precise philosophical position.

At the beginning of Section 1.4, we asserted that

($\star$) the rules of natural deduction *reflect* the operational meaning of logical operators.

The philosophical branch dealing with semantic notions has progressively shifted from a referentialist-representationalist account of meaning – that, in mathematics is embodied, in a sense, by Tarskian semantics based on truth tables for propositional logic – to a pragmatist one – whose most basic assumptions are embodied by Micheal Dummett's position that a semantics of logical operators must rest on their use.[27]

If we move into the latter philosophical position, we see that the vague verb 'reflect' used in ($\star$) could be naturally substituted by the more precise 'define'. But in what sense does an inference rule of natural deduction define a logical concept?

[27] The Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic can been seen as an implicit mathematisation of this perspective, as clearly explained in [Dum00].

The idea of considering natural deduction rules as definitions was notably challenged by Arthur Prior [Pri60]. His "*tonk* argument" shows that introducing a (pair of) rule(s) in natural deduction-style does not suffice to provide a well-defined logical notion. Further, but not clearly specified, requirements need to be satisfied.

Neul Belnap proposed a purely proof-theoretic solution: Good natural deduction rules do explicitly define a logical notion whenever they are *conservative extending* and *uniquely defining*. His [Bel62] gives strong philosophical arguments supporting the idea that the rules for NJ and NK do satisfy these two criteria. For arguing that the propositional rules uniquely define their related connectives no deep analysis is required; on the contrary, without normalisation and the subformula principle the argument supporting the claim about conservative extension lacks a certain mathematical strength.

Form a philosophical perspective we can read each possible simplification – and, in particular, each detour – as a place in which the prover is unnecessarily appealing to a logical notion during the deductive process.[28] For if A is deducible from Γ, then, if we extend the basic language underlying a given good proof system with a new logical concept $*$ that is not used in A nor in Γ, in the *normal* deduction of A from Γ – which exists whenever the normalisation theorem holds – no possible rule for $*$ can occur – by the subformula principle – so that the extended calculus including the rule(s) for $*$ is a conservative extension of the original one indeed: It does give the prover new reasoning methods with no clash with the preexisting deductive apparatus.

Technical treatments of natural deduction are contained in [NvP08], as well as in [TS00]. For the development of Gentzen's paradigm characterising both natural deduction and sequent calculi the reader is referred to [vP18]. A marvellous historical account on the origins of natural deduction even before Gentzen's systematisation is contained in [Pel99].

1.5. Type theory

λ-calculus was proposed in the early 1930s by Alonzo Church as a foundation of logic and mathematics in which the concept of *abstraction* is taken as primitive. In 1936, Church's students Stephen Kleene and Barkley Rosser proved that the 1933 version of this system was inconsistent, but the pure formulation turned out to be surprisingly rich.

Progressively, pure λ-calculus revealed to be a successful model for the concept of *effectively computable function*, and, although the class of λ-definable

[28] This remains implicit in Gentzen's original justification for the rules of natural deduction in [Gen35a]. An explicit connection with detour elimination is made first in [Pra65]. Recently, Prawitz suggested in [Pra19] that the notion of detour elimination could be generalised into a less formalism-sensitive concept of analytically valid argument, that is closer to Gentzen's original informal account of natural deduction.

numerical functions was proved to be identical with that of Turing computable ones, it differs from Turing's model being what is now known as a (higher order) functional programming language, while Turing machines resemble programs in imperative programming languages.

Typed versions of λ-calculus were introduced first by Haskell Curry in 1934 (for the related system of combinatory logic in fact) and by Church in 1940, after he had given a series of lectures attended by Alan Turing during his doctorate studies in Princeton.[29]

Types are syntactic objects which can be assigned to λ-terms, providing a partial specification of the represented algorithm. Moreover, types are useful to check the safety of the very algorithms coded as λ-terms, and to improve the efficiency of this coding. In spite of this, Curry's and Church's works on typed systems were characterized by two different approaches to typing λ-terms:

- Curry assumed terms of type-free λ-calculus, and associated to each term a set of possible types;
- Church adopted an annotated version of λ-terms, so that there exist different terms of different type, although they are "morphologically identical".

Curry's approach, which is also called 'type assignment', corresponds to the programming paradigm of writing a program without typing at all, and then to check by means of a compiler whether this program can receive a type (and this will happen if the program is correct). This is what usually happens in programming with *implicit typing*.

Church's typing, on the other hand, corresponds to the programming paradigm of writing a program together with its type, so that the type-checking results easier. In literature this is known as *explicit typing*. In this work, only systems with this kind of typing are discussed, and they will be called 'typed systems'.

Several versions of λ-calculus with types have been developed within both paradigms, and the evolution of this kind of systems has been (and is being) widespread, with mutual interactions with proof theory[30], theoretical computer science, and computerised mathematics.

[29] This (and further) technical and historical information on type theory can be found in e.g. [HS08], [Hin97], [Bar92], [Bel12].

[30] [GTL89] gives a friendly introduction to F including an exposition of a categorial semantics for this *polymorphic λ-calculus*; it corresponds to second-order logic and dates back to 1970. By means of a slightly modified computability predicate, Jean-Yves Girard proved strong normalisation for F and its extension F_ω, which corresponds to Takeuti's GLC sequent-calculus for higher-order logic: Girard's proofs of normalization for these systems solved also Takeuti's conjecture about cut-elimination in GLC, and (progressively extended) have become standard techniques in proof theory.

On the latter field, constructive versions of type theory play a prominent role.

In dependent type theory, the correspondence between types and propositions is more systematically exploited than in systems we will consider in this section. With an eye at Chapter 7 and Appendix B, let us mention that this intuitionistic version of typed systems reveals interesting connections with homotopy theory and higher category theory, catalysing the development of homotopical interpretation of types and indirectly legitimizing, from a constructive point of view, the univalent approach to foundations of mathematics started in 2013 [The13] with the help of various kind of proof assistants.

Types were in fact introduced first in Russell and Whitehead's *Principia Mathematica*, and since then no clear definition of what kind of object a type should be: From grammatical categories to spaces in the sense of homotopy theory, various but closely interrelated interpretations have been given.[31]

In this work, no specific semantic interpretation of λ-calculi is given, but, since its relevance for proof theory and influence in development modern computerised mathematics, we will briefly survey an elementary typed system[32] according to the proofs-as-programs paradigm, sometimes called Curry-Howard correspondence. When moving to full NJ with equality, that correspondence is systematically exploited in the material of Chapter 7, whose theoretical foundations lie in (univalent) dependent type theory, briefly recalled in Appendix B.

Similarly, the theory underlying the computerisations presented of Chapter 6 – consisting of a classic axiomatic extension of the type theory in [Chu40] with polymorphic variables – is more appropriately defined in Appendix A.

1.5.1. Proofs as programs

For the reader's sake, let's start with a partial glossary to summarize in Figure 1.6 the proofs-as-programs view-point.

This allows to overload notation, so that we can inductively define the class of types $\mathtt{T}$ as the formulas of our propositional language:

$$A \in \mathtt{T} \coloneqq p \mid A \wedge B \mid A \vee B \mid A \to B \mid \bot \mid \top,$$

where p belongs to a denumerable set of type variables, corresponding to $\mathtt{Atm}$.

In the present setting, $A \wedge B$ is called a **product type**, $A \vee B$ a (**weak**) **sum type**, $A \to B$ a **function type**, $\bot$ is the **empty type**, and $\top$ is the **unit type**.

[31] See [Lam58] for a first "type-theoretic" approach to grammatical categories. [Lon00] gives a concise history of set-theoretic interpretation of typed systems and of their "standard models". [Law69, Law71] introduced first the notions of *hyperdoctrine* and *elementary topos* as logico-categorical models for λ-abstraction; for a unified exposition of logic and type theory by means of fibred categories see [Jac99].

[32] That type theory is used in the first three chapters of Part I as a base-calculus for proof-terms in NJp-deductions, to be appropriately extended to each modal logic under investigation.

TYPE THEORY	LOGIC
type	proposition
term	proof
type constructor	logic connective
constructor	introduction rule
destructor	elimination rule
redex	proof detour
reduction	normalization
normal form	normal proof
inhabitation	provability

Figure 1.6: Proofs-as-programs paradigm, in brief

DEFINITION 1.5.1. The set of typed-terms Λ^T of the basic simple type theory is inductively defined by the following grammar:

$$\begin{aligned} f : A \in \Lambda^T \coloneqq \quad & x : A \mid ((f_1 : A \to B)(f_2 : A)) : B \mid (\lambda x : A.f_1 : B) : A \to B \mid \\ & (f_1 : A, f_2 : B) : A \wedge B \mid (\pi^1.(f : A \wedge B)) : A \mid (\pi^2.(f : A \wedge B)) : B \mid \\ & (\mathsf{in}_1(f : A)) : A \vee B \mid (\mathsf{in}_2(f : B)) : A \vee B \mid \\ & (\mathsf{C}((f : A \vee B), x : A.(f_1 : C), y : B.(f_2 : C))) : C \mid \\ & (\bot_J(f : \bot)) : A \mid * : \top \qquad . \end{aligned}$$

Any expression of the form $f : A$ is said **type-assignment** of the subject f to the type A.
A finite set of type-assignments $\Gamma = \{x_1 : A_1, \ldots, x_m : A_m\}$ whose subjects are term-variables and which is consistent, i.e. no variable is subject of more than one assignment, is said a (type-)**context**.

It is easy to see that any type assignment corresponds to a labelled deduction in NJp based on the following mapping:

A	$\longmapsto$	$x_i^A,$	where i is the parcel of the hypothesis A
$\dfrac{\begin{matrix} \vdots\, f_1 & \vdots\, f_2 \\ A & B \end{matrix}}{A \wedge B}\wedge\mathcal{I}$	$\longmapsto$	$(t : A, s : B),$	where $t : A$, $s : B$ correspond to f_1 and f_2 resp.
$\dfrac{\begin{matrix} \vdots\, f' \\ A \wedge B \end{matrix}}{A}\wedge\mathcal{E}_1$	$\longmapsto$	$\pi_1.(t : A \wedge B),$	where $t : A \wedge B$ corresponds to f'

$\dfrac{\overset{\vdots\; f'}{A \wedge B}}{B}\,\wedge\mathcal{E}_2$	$\longmapsto$	$\pi_2.(t : A \wedge B)$,	where $t : A \wedge B$ corresponds to f'
$\dfrac{\overset{\vdots\; f'}{B}}{A \to B}\,\to\mathcal{I}$	$\longmapsto$	$\lambda x_i : A.t : B$,	where $t : B$ corresponds to f' and i is the parcel of discharged hypotheses A
$\dfrac{\overset{\vdots\; f_1}{A \to B} \qquad \overset{\vdots\; f_2}{A}}{B}\,\to\mathcal{E}$	$\longmapsto$	$t : A \to Bs : A$,	where $t : A \to B$, $s : A$ correspond to f_1 and f_2 resp.
$\dfrac{\overset{\vdots\; f'}{A}}{A \vee B}\,\vee\mathcal{I}_1$	$\longmapsto$	$\mathsf{in}_1.t : A$,	where $t : A$ corresponds to f'
$\dfrac{\overset{\vdots\; f'}{B}}{A \vee B}\,\vee\mathcal{I}_2$	$\longmapsto$	$\mathsf{in}_2.t : B$,	where $t : B$ corresponds to f'
$\dfrac{\overset{\vdots\; f'}{A \vee B} \qquad \overset{\overset{[A]}{\vdots\; f_1}}{C} \qquad \overset{\overset{[B]}{\vdots\; f_2}}{C}}{C}\,\vee\mathcal{E}$	$\longmapsto$	$\mathsf{C}(t, (x : A.t_1), (y : B.t_2))$	where C bounds all occurrences of x in t_1 and all occurrences of y in t_2, and t, t_1, t_2 correspond to f', f_1, f_2, resp.
$\dfrac{\overset{\vdots\; f'}{\bot}}{A}\,\bot_J$	$\longmapsto$	$(\bot_J t) : A$	where t corresponds to f'
$\dfrac{}{\top}\,\top_J$	$\longmapsto * : \top$		

Thus, one could rephrase any proof in NJp in type theoretic terms, which can be thought as proof-names for deductions. This justifies the notation overload for deductions and terms of the type theory. Also notice that in the term-constructor for $\to$-types the discharging of hypothesis A is denoted by the bounding of the variable $x : A$ in λ-abstraction. Similarly, the term-destructor for $\vee$-types bounds the variables $x : A$ and $y : B$, as expected.

A sequent-style formalisation on this theory of NJp-deductions could then been defined by decorating the sequent-style version of NJ given in [NvP08, Sect. 1.3] with proof-names.

The sequents of that type theory will be called **judgements**. For instance, the judgements corresponding to the rules for implication in NJ are[33]

$$\frac{\Gamma_1 \Rightarrow f : A \to B \qquad \Gamma_2 \Rightarrow g : A}{\Gamma_1 \cup \Gamma_2 \Rightarrow (fg) : B} \qquad \text{with } \Gamma_1 \cup \Gamma_2 \text{ consistent.}$$

$$\frac{\Gamma \Rightarrow f : A}{\Gamma - (x : A) \Rightarrow (\lambda x.f) : A \to B} \qquad \text{when } \Gamma \text{ is consistent with } x : A.$$

When a judgement $\Gamma \Rightarrow f : A$ is provable in the type theory, the term f is said to **inhabit** the type A in context Γ.

The normalisation results that are mentioned in Section 1.4.2 acquire now a precise procedural meaning: Eliminating a detour or performing a permutation corresponds to executing a process for computing values; normalisation assures that a computation can always be continued to a final result.

This is made precise by rephrasing the detour eliminations and permutations of Section 1.4.2 in type-theoretic manners.

DEFINITION 1.5.2. Let Ξ denote the reflexive and transitive closure of the relation $>_\Xi \subseteq \Lambda^{\mathrm{T}} \times \Lambda^{\mathrm{T}}$ defined as follows:[34]

Detours:
- $(\lambda x.f)g \; >_\Xi \; f[x := g]$
- $\pi_i(f_1, f_2) \; >_\Xi \; f_i$ for $i = 1, 2$
- $\mathsf{C}(\mathsf{in}_i f, x.f_1, y.f_2) \; >_\Xi \; f_i[x_i := f]$ for $i = 1, 2$

Permutations:
- $\mathsf{C}(f, x.f_1, y.f_2)g \; >_\Xi \; \mathsf{C}(f, x.f_1 g, y.f_2 g)$
- $\pi_i \mathsf{C}(f, x.f_1, y.f_2) \; >_\Xi \; \mathsf{C}(f, x.\pi_i f_1, y.\pi_i f_2)$ for $i = 1, 2$
- $\mathsf{C}(\mathsf{C}(f, x.f_1, y.f_2), u.g_1, v.g_2) \; >_\Xi \; \mathsf{C}(f, x.\mathsf{C}(f_1, u.g_1, v.g_2), y.\mathsf{C}(f_2, u.g_1, v.g_2))$
- $\bot_J(f)g >_\Xi \bot_J(f)$
- $\pi_i \bot_J(f) >_\Xi \bot_J(f)$ for $i = 1, 2$
- $\mathsf{C}(\bot_J(f), x.f_1, y.f_2) >_\Xi \bot_J(f)$
- $\bot_J(\bot_J(f)) >_\Xi \bot_J(f)$.

where $g[x_i := f]$ denotes the substitution of f for all free occurrences of x_i in g.

A Ξ-normal form is just a term of Λ^{T} that cannot be rewritten w.r.t. Ξ. For this system of rewritings it is possible to prove the properties that the reader of Section 1.4 would expect:[35]

LEMMA 1.5.3. *Ξ has the Church-Rosser Property.*

[33] The judgements for the other typed terms follow the structure of the corresponding rules in NJ, as the reader might expect. See the beginning of Chapter 7 for the formal definition of the type theory corresponding to (first-order) NJ.

[34] Type annotations are omitted for the sake of readability.

[35] Refer to [SU06] for the proofs.

As a consequence, we have that a denotational calculus of Ξ-reduction is consistent: The Church-Rosser property implies, for any term, the uniqueness of Ξ-normal form, so that denotational consistency is shown by the fact that if $x : A \not\equiv y : A$ are Ξ-normal forms, then if we could prove that $x : A$ and $y : A$ are equal modulo Ξ, by the Church-Rosser property there would exists an $f : A$ such that $x : A$ rewrites to $f : A$ and $y : A$ rewrites to $f : A$, *contra* our assumption.

Normalisation of deductions in NJp can also be easily rephrased in type-theoretic manners:

THEOREM 1.5.4 (**Weak Normalization**). *For every $t : A \in \Lambda^{\mathsf{T}}$, there exists a Ξ-normal form.*

THEOREM 1.5.5 (**Strong Normalization**). *For every $t : A \in \Lambda^{\mathsf{T}}$, all Ξ-rewritings of $f : A$ terminate.*

Thus, the proofs-as-programs paradigm turns the structural concept of proof simplifications, as well as the structural normalisation of deductions, into programming and algorithmic notions, expressed in the language of typed λ-calculus.

Adding specific constructors for $\forall$- and $\exists$-rules that occur in NJ, this paradigm can be extended to a type theory λP_1, which expresses the computational meaning of full NJ based on a single sorted language.[36] It is worth noticing, however, that it is possible to define a correspondence between λP_1 and typed λ-calculus with $\rightarrow$-types only by means of a contracting map, so that the latter is indeed the core of all computational significance of NJ-deductions.

Moreover, the computational reading of proofs in natural deduction systems are not limited to intuitionistic logic: The proof-as-programs paradigm extends to classical logic by means of type theories with control operators [Gri89, DN03], as well as arithmetical theories, by means of Gödel's system T [AF98], and second-order theories, by means of polymorphic type theories, like Girard's F_ω [MKO95]. The theory behind the proof assistant Coq consists of a powerful combination of polymorphic and dependent type theory known as calculus of (co-inductive) constructions. This is further generalised to the so-called full higher-order dependent type theory, whose categorical shades are discussed in [Jac99].

1.5.2. COMPUTATIONAL TRINITARIANISM

The correspondence between type theory and natural deduction is further enlightened by extending the perspective to cover the most algebraic foundations of mathematics currently available, namely category theory.

Slightly dramatising, Robert Harper has written [Har11]

[36] The λP_1 calculus is a restricted version of the dependent type theory discussed in Appendix B; see [SU06, Ch. 8] for an introduction.

> The central dogma of computational trinitarianism holds that Logic, Languages, and Categories are but three manifestations of one divine notion of computation. There is no preferred route to enlightenment: each aspect provides insights that comprise the experience of computation in our lives.

In less emphatic terms, the correspondence between proofs and programs is extended to consider arrows in categories which have enough structure to capture the behaviour of logical operators. The key ideas behind the so-called computational trinitarianism – or Curry-Howard-Lambek correspondence – would be summarised by the partial glossary of Figure 1.7.

Logic	Type Theory	Category Theory
proposition	type	object
proof	term	arrow
theorem	inhabitant	element-arrow
conjunction	product type	product
true	unit type	terminal object
implication	function type	exponential
disjunction	sum type	(weak) coproduct
false	empty type	(weak) initial object

Figure 1.7: Computational trinitarianism

We recall now the basics of categoric theory in order to make explicit the meaning of this trilogy. From a philosophical view-point, category theory has been introduced by Samuel Eilenberg and Saunders Mac Lane in [EM45] as a rearrangement of mathematics on the basis of the notions of (natural) transformations, and their compositions, which unchain the power of abstract algebra developed in the 1930s by making explicit the relevance, for a mathematical structure, of its relationships with other structures of the same kind, i.e. the Daedalus of structure transformations, or *homomorphism*, existing among them.[37]

Definition 1.5.6. A category $\mathscr{C}$ consists of the following data:

- a collection of **objects** $\mathscr{C}_0$, generally denoted $\mathscr{C}$ also;
- a collection of **arrows** $\mathscr{C}_1$;
- to each arrow $f : \mathscr{C}_1$,[38] a pair of objects $\langle \mathsf{source}(f), \mathsf{target}(f) \rangle$ is associated. When $X : \mathscr{C}_0$ is the source of f and $Y : \mathscr{C}_0$ is the target of f one

[37] Refer to [Mar08] for the history and philosophy of category theory. Many enlightening observations are widespread in the pages of [Mac13].

[38] Here and in the rest of this section we overload the type-theoretic notation to denote membership: Such a move is justified by the correspondence we are discussing in these pages.

writes $f : X \to Y$. The collection of arrows from X to Y is denoted by $\mathsf{hom}(X, Y)$;

- to each object $X : \mathscr{C}_0$, an arrow $id_X : \mathsf{hom}(X, X)$ is associated. This is called the **identity arrow** on X;
- to each pair of arrows f, g such that $\mathsf{source}(g) = \mathsf{target}(f)$ is associated an arrow $g \circ f : \mathsf{source}(f) \to \mathsf{target}(g)$, called the **composite arrow** of f with g. When $\mathsf{source}(g) = \mathsf{target}(f)$, f, g are called composable.

These data need to satisfy specific conditions:

Associativity: For any composable arrows f, g and g, h, we have

$$h \circ (g \circ f) = (h \circ g) \circ f;$$

Identity: For any $f : X \to Y$, we have

$$id_Y \circ f = f = f \circ id_X.$$

In this thesis we will work with **locally small categories** only, i.e. we assume that for any category $\mathscr{C}$ and any $X, Y : \mathscr{C}_0$, the collection $\mathsf{hom}(X, Y)$ is indeed a set. Moreover, we will overload the notation and write $\mathscr{C}$ for $\mathscr{C}_0$ whenever the context makes clear we are dealing with the objects of $\mathscr{C}$.

We turn now to arrows between categories, i.e. functors, and arrows between functors, i.e. natural transformations.

DEFINITION 1.5.7. Let $\mathscr{C}$ and $\mathscr{D}$ be categories. A functor $\mathfrak{F} : \mathscr{C} \to \mathscr{D}$ consists of the following data:

- A function $\mathfrak{F}_0 : \mathscr{C} \to \mathscr{D}$, generally denoted $\mathfrak{F}$ also.
- For each $X, Y : \mathscr{C}$, a function $\mathfrak{F}_{X,Y} : \mathsf{hom}_{\mathscr{C}}(X, Y) \to \mathsf{hom}_{\mathscr{D}}(\mathfrak{F}X, \mathfrak{F}Y)$, generally denoted $\mathfrak{F}$ also;

satisfying the following conditions:

(i) For each $X : \mathscr{C}$, we have $\mathfrak{F}(id_X) = 1_{\mathfrak{F}(X)}$; and

(ii) For each $X, Y, Z : \mathscr{C}$, and $f : \mathsf{hom}_{\mathscr{C}}(X, Y)$ and $g : \mathsf{hom}_{\mathscr{C}}(Y, Z)$, we have $\mathfrak{F}(g \circ f) = \mathfrak{F}g \circ \mathfrak{F}f$.

DEFINITION 1.5.8. For functors $\mathfrak{F}, \mathfrak{G} : \mathscr{C} \to \mathscr{D}$, a natural transformation $\gamma : \mathfrak{F} \Rightarrow \mathfrak{G}$ consists of the following data:

- For each $X : \mathscr{C}$, an arrow $\gamma_X : \mathsf{hom}_{\mathscr{D}}(\mathfrak{F}X, \mathfrak{G}X)$, giving the **components** of γ;

satisfying the **naturality** condition

For each $X, Y : \mathscr{C}$ and $f : \mathsf{hom}_{\mathscr{C}}(X, Y)$, we have $\mathfrak{G}f \circ \gamma_X = \gamma_Y \circ \mathfrak{F}f$.

It is common to rephrase equations between arrows of a given category by recurring to graphical tools: A **commutative diagram** is a graph having objects as vertices and arrows as directed edges, in which the composite arrow obtained from any connected path depends only on the starting and ending point of path itself.
For instance, the previous naturality condition could be graphically rendered by the commuting diagram

$$\begin{tikzcd}
\mathfrak{F}X \arrow[r, "\gamma_X"] \arrow[d, "\mathfrak{F}f"'] & \mathfrak{G}X \arrow[d, "\mathfrak{G}f"] \\
\mathfrak{F}Y \arrow[r, "\gamma_Y"'] & \mathfrak{G}Y
\end{tikzcd}$$

The identity morphism on x will be simply rendered by $\begin{tikzcd} x \arrow[r, equal] & x \end{tikzcd}$.

After the very basic definitions just given, we can summarise the further notions that trinitarianism requires by the following

DEFINITION 1.5.9. Given a category $\mathscr{C}$ we define the following objects

- **Product of objects** X, Y**:** Object $X \times Y$ of $\mathscr{C}$ together with projection arrows $\pi_1 : X \times Y \to X$, $\pi_2 : X \times Y \to Y$ such that any diagram

$$\begin{tikzcd}
& Z \arrow[dl, "f"'] \arrow[d, "\langle f,g\rangle"] \arrow[dr, "g"] & \\
X & X \times Y \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & Y
\end{tikzcd}$$

  commutes for a unique $\langle f, g\rangle : Z \to X \times Y$.

- **Terminal object:** Object 1 of $\mathscr{C}$ such that for any other $X : \mathscr{C}$, there exists a unique arrow $!_X : X \to 1$.

- **Product of arrows** $f_1 : X_1 \to Y_1$**,** $f_2 : X_2 \to Y_2$**:** Unique arrow $f_1 \times f_2 : X_1 \times X_2 \to Y_1 \times Y_2$ making the following diagram commute

$$\begin{tikzcd}
X_1 & X_1 \times X_2 \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "f_1 \times f_2"] & X_2 \arrow[d, "f_2"] \\
Y_1 & Y_1 \times Y_2 \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & Y_2
\arrow[from=1-1, to=2-1, "f_1"']
\end{tikzcd}$$

- **Exponential of objects** Y **and** X**:** An object Y^X of $\mathscr{C}$ together with an arrow $eval : X \times Y^X \to Y$ such that any diagram

$$\begin{tikzcd}
X \times Z \arrow[d, "id_X \times \lambda_f"'] \arrow[dr, "f"] & \\
X \times Y^X \arrow[r, "eval"'] & Y
\end{tikzcd}$$

commutes for a unique $\lambda_f : Z \to Y^X$.

- **Element of an object** X**:** An arrow $x : 1 \to X$.
- **Coproduct of objects** X, Y**:** Object $X+Y$ of $\mathscr{C}$ together with injection arrows $\iota_1 : X \to X+Y$, $\iota_2 : Y \to X+Y$ such that any diagram

$$\begin{tikzcd}
X \arrow[r, "\iota_1"] \arrow[dr, "f"'] & X+Y \arrow[d, "{[f,g]}"] & Y \arrow[l, "\iota_2"'] \arrow[dl, "g"] \\
& Z &
\end{tikzcd}$$

  commutes for a unique $[f,g] : X+Y \to Z$.
- **Initial object:** Object 0 of $\mathscr{C}$ such that for any other $X : \mathscr{C}$, there exists a unique arrow $0_X : 0 \to X$.

The defining properties of these categorical constructions correspond to rewritings of deductions in $\mathsf{NJ(p)}$, as noticed in [LS88, Ch. I.8]. Notice, however, that the system of rewriting imposed by the universal properties of initial object and coproducts *is a proper extension* of Ξ discussed in Section 1.4.2. Therefore we will refer to the categorical counterparts of $\bot$ and $\vee$ w.r.t. the system Ξ as **weak initial object** and **weak coproduct**, respectively.

The reader is referred to [LS88] and [Jac99] for further results connecting type theory, logic, and category theory. It is relevant to note that computational trinitarianism has been extended in recent times to cover also topological notions, as well as concepts coming from physics, as witnessed by [FKS20]. These aspects are in any case far beyond the scope of our aims here.

Part I

Structural Proof Theory

Introduction to Part I

Wherever there is communication, there are proofs.
The Proof Society, *The Proof Manifesto*, 2017

David Hilbert's *Beweistheorie* originated from the discussions on the foundations of mathematics at the beginning of 20th century. Its general aim is to study mathematical proofs as real mathematical objects, and its first steps can be traced back to the works Gottlob Frege, the father of mathematical logic.

Hilbert's main aim was to study mathematical proofs in order to guarantee to mathematics a granitic foundation by means of an unquestionable proof of its consistency. His original research program [Zac19] has been partially led to success in the 1930s by Gerhard Gentzen, whose proof of consistency of arithmetic has correctly been considered a milestone in the history of mathematical logic, and marked the advent of a new topic in the field, namely ordinal analysis [RS20].

But Gentzen's work had another revolutionary impact on mathematical logic, by establishing the birth of *structural proof theory* [vP18]. With his natural deduction systems and sequent calculi of [Gen35a, Gen35b], Gentzen introduced a new paradigm which substituted the axiomatic approach in the mathematical modelling of mathematical reasoning: Thanks to those new models it is possible to study the structure, abstract properties, and interactions of mathematical proofs. Actually, his very proof of the consistency of arithmetic is based on a result about the structure of formal derivations in sequent calculi, namely the *cut-elimination theorem*, which correctly deserved the name of '*Hauptsatz*' – 'central result', in German.

Its counterpart for natural deduction was later established by Dag Prawitz [Pra65], as a *normalisation theorem* for derivations, which later inspired analogous results for type theories and functional programming languages.

In this first part of the thesis, we present three original natural deduction calculi for intuitionistic modal logics. The first and the second one deal with Sergei Artemov and Tudor Protopopescu's logics for verification-based epistemic states [AP16]; the third one is built on top of the first to model the reasoning for one of the few known provability logics for intuitionistic arithmetic, namely Dick de Jongh and Albert Visser's intuitionistic strong Löb logic [dJV95].

The final chapter deals with a modular family of G3-style sequent calculi for classical interpretability logics – i.e. classical modal logics that, besides formal arithmetical provability, are capable of identifying the relevant abstract properties of formal interpretability over an arithmetical base theory [JRMV20, Vis90].

While our natural deduction systems are based on Gentzen's original paradigm, the sequent calculi for interpretability logics we are proposing follow the well-established methodology of internalising semantic notions into the proof system by recurring to labels. Nevertheless, for each of the systems under investigations we propose a detailed structural analysis, and prove that they all satisfy the main proof-theoretic *desiderata*. In particular, for each of the natural deduction calculi we prove a (strong) *normalisation result* – as well as some relevant corollaries – and for the whole family of labelled sequent systems we provide a *uniform cut-elimination algorithm*.

The structure of the present part of the thesis is as follows:

- ▷ Chapter 2 introduces the natural deduction system IEL^- for intuitionistic belief designed with the intent of a natural type-theoretic translation. The latter is then used to prove a strong normalisation theorem for IEL^--deductions w.r.t. different systems of proof simplifications. From normalisation w.r.t. the most comprehensive of those systems we derive the subformula principle for normal deductions in IEL^-, which we prove by adapting and filling the details of the proof sketch for propositional intuitionistic logic given in [Pra71]. Many other properties of the logic for intuitionistic belief are then proven as corollaries to those structural results. Finally, we consider a categorical interpretation of normal IEL^--deductions w.r.t. a smaller system of proof simplifications, as an example of proof relevant semantics for our calculus.

- ▷ Chapter 3 extends IEL^- into a natural deduction calculus for intuitionistic knowledge, that we denote by IEL. This extension consists of the introduction of a new elimination rule for the modal operator, which is based on the account of intuitionistic factivity of knowledge described in [AP16]. For IEL we prove a strong normalisation theorem and the subformula principle on the basis of an extended version of the largest system of proof rewritings used for intuitionistic belief. As in the previous chapter, we prove by purely proof-theoretic methods some relevant properties of the logic for intuitionistic knowledge, namely: consistency, decidability, disjunction property, □-primality, and modal disjunction property.

- ▷ Chapter 4 extends IEL^- towards a different direction, namely a natural deduction calculus for intuitionistic strong Löb logic [dJV95], that we denote by ISL. With this chapter, we shift our attention to modal logic for provability and interpretability. Our calculus is obtained by adding to IEL^- a special elimination rule for the modality corresponding to the axiom for strong Löb induction. For ISL we prove a strong normalisation theorem w.r.t. specific systems of proof rewritings as a preliminary result for further proof-theoretic investigations on that provability logic. A discussion of those perspectives on future research closes the chapter.

▷ Finally, Chapter 5 presents our family of labelled sequent calculi for classical interpretability logics, modularly designed on the basis of generalised Veltman semantics for those logics [JRMV20]. We prove that each of our calculi enjoys excellent structural properties, namely: admissibility of weakening, contraction and, more relevantly, cut. To the best of our knowledge, this is the most extensive class of analytic proof systems for modal logics of interpretability currently available, and none of the calculi already present in the literature satisfies as many proof-theoretic desiderata as ours.

2

Natural deduction for intuitionistic belief

When does a mathematician believe in a mathematical statement? In general terms, she believes that a statement is true when it has been proven. This does not mean, however, that this very mathematician has proven that statement, nor that she is capable of producing a proof of it by herself.

Beliefs about mathematics are generated by the existence of proofs, and, in many cases, such an existence does not presuppose an explicit witness for provability.

In more practical terms: It is rather usual to believe that a mathematical statement is a theorem of a given theory even though we do not have a direct access to any explicit proof of it. Sometimes, we rely on the literature to check a given claim; in other cases, e.g. when we are new to a given field, we trust an authority – a lecturer, a supervisor, an expert – about the validity of a statement. It is not unusual even to trust a computer about the truth of a mathematical claim.

A very recent episode is quite enlightening on that point.

Peter Scholze and Dustin Clausen are leading very advanced research in a "grand unification project" called condensed mathematics. The field is highly specialised, and it is not surprising that only few mathematicians are able to work on those concepts to prove some meaningful claim in that theory.

But it is astonishing that Scholze and Clausen *themselves* were not sure that they had correctly proven a theorem – the famous liquid tensor experiment [Sch21] – of their own theory, until their self-questioned proof has been developed in the proof assistant Lean [MU21].

Such a formalisation required efforts from several researchers around Europe, and the resulting formal proof is even less comprehensible to humans than the original proof by Scholze and Clausen.

However it has been that *verification* that convinced them to trust in the liquid tensor experiment.

A constructive account for truth requires that a statement can be considered true only if it has been proven. The liquid tensor experiment episode supports such an account; moreover, it sheds some light on the meaningfulness of a verificationist account of epistemic states. According to that view, a rational agent should believe a statement when it has been *at least* verified.

[AP16] extends the standard constructive-intuitionistic analysis of truth to cover also epistemic states. In few words, they claim that if proving a statement A assures that A is true, then we might *believe* that A is true even though we do not have a direct proof of A: A correct verification of A suffices. Thus belief is mainly considered as the result of a process of verification.

At the same time, it is clear that a proof of a mathematical statement is a most strict type of verification. Hence, their proposed intuitionistic account for belief validates the following principle of 'constructivity of truth':

$$A \to \Box A,$$

where $\Box$ is the modal operator for belief in our formal language.

Even at the informal setting, that principle is clearly validated by contemporary mathematical practice: When a statement A is proven, then its very proof can be checked by another mathematician, both by 'pencil and paper', and by relying on computer proof assistants. It is that very checking that lets the mathematical community believe that A holds indeed.

Now, the standard intuitionistic account for truth is embodied by the Brouwer-Heyting-Kolmogorov (BHK) interpretation of intuitionistic logic. That interpretation is based on a semantic reading of propositional variables as problems (or tasks), and of logical connectives as operations on *proofs*. In this way, it provides a semantics of mathematical statements in which the computational aspects of proving and refuting are highlighted.[1]

In spite of being named after L.E.J. Brouwer, this approach is rather away from the deeply philosophical attitude at the origin of intuitionism: In BHK interpretation, reasoning intuitionistically is similar to a safe mode of program execution which always terminates; on the contrary, according to the founders of intuitionism, at the basis of the mathematical activity there is a continuous mental process of construction of objects starting with the flow of time underlying the chain of natural numbers, and intuitionistic reasoning is what structures that process.[2]

This reading of the mathematical activity is formally captured by Kripke semantics for intuitionistic logic [Kri65]: Relational structures based on preorders capture the informal idea of a process of growth of knowledge in time which characterises the mental life of the mathematician.

It is worth noticing that the focuses of these semantics are quite different: BHK interpretation stresses the importance of the concept of proof in the semantics for intuitionistic logic; Kripke's approach highlights the epistemic process behind the provability of a statement.

[1] The reader is referred to [Tro69] for an introduction.

[2] For instance, Dummett's [Dum00] advocates a purely philosophical justification of the whole current of intuitionistic mathematics.

The lines of reasoning of [AP16] are the following: In the BHK interpretation we have an implicit notion of proof whose epistemic aspects are modelled by Kripke structures; the construction of a proof for a specific proposition that we carried out as a cumulative mental process gives us sufficient reason for (at least) believing in that proposition. It is then possible to cover also traditional epistemic states of belief and knowledge within such a framework, once we recognise the correct clauses for corresponding modal operators: Proving A assures that A is true; proving $\Box A$ is weaker, for we may believe A even though we do not have a direct proof of A itself.

The insight of BHK interpretation is formally captured by the rules of natural deduction for intuitionistic logic, and, even more precisely, by the associated λ-calculus with types of the proofs-as-programs paradigm.[3] In spite of this, [AP16] discusses only axiomatic calculi and possible world semantics for intuitionistic epistemic logics, so that the stimulating question of considering the proof-theoretic/computational aspects of epistemic states remained at an informal level and a very beginning stage.

In the present chapter, on the contrary, we propose a natural deduction system IEL^- designed with the intent of translating it into a functional calculus for deductions. In a sense, that defines a formal counterpart to Artemov and Protopopescu's reading of the belief operator by extending the proofs-as-programs paradigm between NJp and simple type theory to handle that modal operator and its corresponding constructor in the proof-term calculus.

The resulting modal λ-calculus is then used to prove a strong normalisation theorem for IEL^-, from which the subformula property is derived. As corollaries, many properties of the logic under investigation are proven, namely: consistency, decidability, disjunction property, admissibility of the reflection rule, modal disjunction property, and $\Box$-primality.

The chapter is organised as follows: Section 2.1 recalls the axiomatic calculus for intuitionistic belief and its Kripke semantics, as defined in [AP16]. In Section 2.2, the natural deduction system IEL^- is introduced, and its logical equivalence with the axiomatic calculus is proven. Then proof-rewritings are defined by using a corresponding modal λ-calculus for IEL^-, in order to handle deductions without detours and permutation conversions. Strong normalisation is proven w.r.t. these rewritings. In Section 2.3, analyticity of IEL^- is established by proving the subformula property for normal deductions; furthermore, purely proof-theoretic proofs of known properties of intuitionistic belief are given. Finally, in Section 2.4, I discuss a categorical interpretation of the belief modality and a potential proof-theoretic semantics for IEL^- based on that interpretation.

[3] See Section 1.5.1, and [SU06] for deeper results in the field.

WRITING AND REVISION NOTES. This chapter is based on [PB21b] and [PB21a]. Nevertheless, many changes have been made to that material: In particular, the proof of the strong normalisation theorem for IEL^- given here substantially differs from the one developed in [PB21a], for being much clearer and simpler than the latter, which was based on a rather technical CPS-translation of IEL^- in NJp. More prominently, we present here an original and fully detailed proof of the subformula property for our system of natural deduction on the basis of the techniques sketched in [Pra71]. That, in turn, let us present very neat proof-theoretic properties of the logic for intuitionistic belief established in [AP16] by model-theoretic means.

2.1. AXIOMATIC CALCULUS FOR INTUITIONISTIC BELIEF

Let's start by recalling the syntax and relational semantics for the logic of intuitionistic belief as introduced in [AP16].

2.1.1. SYSTEM $\mathbb{IEL}^-$

DEFINITION 2.1.1. $\mathbb{IEL}^-$ is the axiomatic calculus given by:

- Axiom schemas for intuitionistic propositional logic;
- Axiom schema $\mathsf{K} : \Box(A \to B) \to \Box A \to \Box B$;
- Axiom schema of co-reflection $A \to \Box A$;
- Modus Pones $\dfrac{A \to B \qquad A}{B}\, \mathit{MP}$ as the only inference rule.

We write $\Gamma \vdash_{\mathbb{IEL}^-} A$ when A is derivable in $\mathbb{IEL}^-$ assuming the set of hypotheses Γ, and we write $\mathbb{IEL}^- \vdash A$ when $\Gamma = \varnothing$.

We immediately have

PROPOSITION 2.1.2. *The following properties hold:*

(i) Necessitation rule $\dfrac{A}{\Box A}$ *is derivable in* $\mathbb{IEL}^-$*;*

(ii) The deduction theorem holds in $\mathbb{IEL}^-$*;*

(iii) $\mathbb{IEL}^-$ *is a normal intuitionistic modal system.*

Proof. See [AP16].

⊠

As stated before, this system axiomatises the idea of belief as the result of verification within a framework in which truth corresponds to provability, accordingly to the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic. Note also that, in this perspective, the co-reflection schema is valid, while its converse does not hold: If A is true, then it has a proof, hence it is verified; but A can be verified without disclosing a specific proof, therefore the standard epistemic schema $\Box A \to A$ is not valid under this interpretation.[4]

2.1.2. Kripke semantics for $\mathbb{IEL}^-$

Turning to relational semantics, in [AP16] the following class of Kripke models is given.

Definition 2.1.3. A model for $\mathbb{IEL}^-$ is a quadruple $\langle W, \leq, v, E\rangle$ where

- $\langle W, \leq, v\rangle$ is a standard model for intuitionistic propositional logic;
- E is a binary 'knowledge' relation on W such that:
  - if xEy, then $x \leq y$; and
  - if $x \leq y$ and yEz, then xEz; graphically we have

$$\begin{tikzcd} & y \arrow[r, dotted, "E"] & z \\ x \arrow[ur, "\leq"] \arrow[urr, dotted, "E"'] & & \end{tikzcd}$$

- v extends to a forcing relation $\Vdash$ such that
  - $x \Vdash \Box A$ iff $y \Vdash A$ for all y such that xEy.

A formula A is true in a model iff it is forced by each world of that model; we write $\mathbb{IEL}^- \vDash A$ iff A is true in each model for $\mathbb{IEL}^-$.

Notice that this semantics assumes Kripke's original interpretation of intuitionistic reasoning as a growing knowledge – or discovery process – for an epistemic agent: The relation E defines an audit of 'cognitively' $\leq$-accessible states in which the agent can commit a verification.

This semantics is adequate to the calculus:

Theorem 2.1.4. *The following hold:*

(**Soundness**) *If* $\mathbb{IEL}^- \vdash A$*, then* $\mathbb{IEL}^- \vDash A$.

(**Completeness**) *If* $\mathbb{IEL}^- \vDash A$*, then* $\mathbb{IEL}^- \vdash A$.

Proof. Soundness is proven by induction on the derivation of A.

Completeness is proven by a standard construction of a canonical model. See [AP16] for the details.

⊠

[4] Different versions of intuitionistic epistemic states are discussed in [Wil92] and [Pro12].

2.2. Natural deduction for intuitionistic belief

We now introduce a natural deduction system which is logically equivalent to $\mathbb{IEL}^-$, but which is also capable of a computational reading of the epistemic operator and of proofs involving this kind of modality.

Accordingly, the starting point is proving that our of natural deduction calculus IEL^- is sound and complete w.r.t. $\mathbb{IEL}^-$; then, we prove that proofs in IEL^- can be named by means of λ-terms as stated in the proofs-as-programs paradigm for intuitionistic logic. By using this formalism, we prove a strong normalisation theorem for IEL^- stating that detours and permutation conversions can be eliminated from all deductions.

2.2.1. System IEL^-

Definition 2.2.1. Let IEL^- be the calculus extending NJp by the following rule:

$$\dfrac{\begin{matrix}\Gamma_1 \\ \vdots \\ \Box A_1\end{matrix} \qquad \begin{matrix} \\ \vdots \\ \cdots\end{matrix} \qquad \begin{matrix}\Gamma_n \\ \vdots \\ \Box A_n\end{matrix} \qquad \begin{matrix}[A_1, \cdots, A_n], \Delta \\ \vdots \\ B\end{matrix}}{\Box B}\ \Box\mathcal{I}$$

where Γ and Δ are *multisets of formulas*, and $A_1, \cdots, A_n$ are *all* discharged.[5]

We say that B is the major premise of the rule, and each $\Box A_i$ is a minor premise, whose corresponding discharged assumption is A_i.

The rule can be informally read as follows: Suppose that one can prove B by assuming, among others, $A_1, \cdots, A_n$. Then, if all $A_1, \cdots, A_n$ have been verified, then one has enough evidence to state that B is verified, without assuming $A_1, \cdots, A_n$.

As before, we write $\Gamma \vdash_{\mathsf{IEL}^-} A$ when A is derivable in IEL^- assuming the set of hypotheses Γ, and we write $\mathsf{IEL}^- \vdash A$ when $\Gamma = \varnothing$.

Let's immediately check that we are dealing with the same logic presented in [AP16]

Lemma 2.2.2. $\Gamma \vdash_{\mathsf{IEL}^-} A \quad$ *iff* $\quad \Gamma \vdash_{\mathbb{IEL}^-} A$.

Proof. Assume $\Gamma \vdash_{\mathbb{IEL}^-} A$. We proceed by induction on the derivation. We only need to check that the axiom schemas can be derived in the natural deduction calculus:

- Intuitionistic schemas are dealt with NJp rules;

[5] Notice that this calculus differs from the system introduced in [dPR11] by allowing the set Δ of additional hypotheses in the subdeduction of B.

- K: $$\dfrac{\square(A \to B) \qquad \square A \qquad \begin{array}{c}[A \to B, A]\\ \vdots\\ B\end{array}}{\square B}\,{\scriptstyle \square\mathcal{I}}$$;

- co-reflection: $$\dfrac{\dfrac{[A]^1}{\square A}\,{\scriptstyle \square\mathcal{I}}}{A \to \square A}\,{\scriptstyle \to\mathcal{I}:1}$$.

Conversely, assume $\Gamma \vdash_{\mathsf{IEL}^-} A$. We need to consider only the $\square\mathcal{I}$ rule: By induction hypothesis, we have $\Gamma_1 \vdash_{\mathbb{IEL}^-} \square A_1, \cdots, \Gamma_n \vdash_{\mathbb{IEL}^-} \square A_n$, and $A_1, \cdots, A_n, \Delta \vdash_{\mathbb{IEL}^-} B$. Then we have $\Gamma_1, \cdots, \Gamma_n \vdash_{\mathbb{IEL}^-} \square A_1 \wedge \cdots \wedge \square A_n$, and, by the deduction theorem for $\mathbb{IEL}^-$ and ordinary logic, $\Delta \vdash_{\mathbb{IEL}^-} A_1 \wedge \cdots \wedge A_n \to B$.

By co-reflection $\mathbb{IEL}^- \vdash (A_1 \wedge \cdots \wedge A_n \to B) \to \square(A_1 \wedge \cdots \wedge A_n \to B)$, and by K-schema $\mathbb{IEL}^- \vdash \square(A_1 \wedge \cdots \wedge A_n \to B) \to \square(A_1 \wedge \cdots \wedge A_n) \to \square B$. Hence we have $\Delta \vdash_{\mathbb{IEL}^-} \square(A_1 \wedge \cdots \wedge A_n) \to \square B$, whence, by modal logic, we obtain $\Delta \vdash_{\mathbb{IEL}^-} \square A_1 \wedge \cdots \wedge \square A_n \to \square B$, which gives $\Gamma_1, \cdots, \Gamma_n, \Delta \vdash_{\mathbb{IEL}^-} \square B$, as desired.

⊠

As for NJp, a modal λ-calculus is then obtained by decorating IEL^--deductions with proof names.

The λ-term corresponding to $\square\mathcal{I}$ is indeed ruled by the single constructor:

$$\dfrac{\Gamma_1 \vdash f_1 : \square A_1 \qquad \cdots \qquad \Gamma_n \vdash f_n : \square A_n \qquad x_1 : A_1, \cdots, x_n : A_n, \Delta \vdash g : B}{\Gamma_1, \cdots, \Gamma_n, \Delta \vdash (\mathsf{box}[x_1, \cdots, x_n].\, g \,\mathsf{with}\, f_1, \cdots, f_n) \;: \square B}$$

2.2.2. Normalisation

In order to eliminate potential modal detours from IEL^--deduction we introduce the following proof rewriting:

$$\dfrac{\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \square\vec{A}\end{array} \quad \begin{array}{c}[\vec{A}]^1, \vec{C}\\ \vdots\\ B\end{array}}{\square B}\,{\scriptstyle \square\mathcal{I}:1} \quad \begin{array}{c}\Delta\\ \vdots\\ \square\vec{D}\end{array} \quad \begin{array}{c}[B, \vec{D}]^2, \vec{E}\\ \vdots\\ F\end{array}}{\square F}\,{\scriptstyle \square\mathcal{I}:2} \quad \rightsquigarrow \quad \dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \square\vec{A}\end{array} \quad \begin{array}{c}\Delta\\ \vdots\\ \square\vec{D}\end{array} \quad \begin{array}{c}[\vec{A}]^1, \vec{C}\\ \vdots\\ B\\ \\ \end{array} \quad \begin{array}{c}[\vec{D}]^1, \vec{E}\\ \vdots\\ F\end{array}}{\square F}\,{\scriptstyle \square\mathcal{I}:1}$$

This rewriting collapses two applications of $\square\mathcal{I}$ into a single one, making the deduction much simpler, so that can be correctly called a detour elimination. Let's denote it by $\rho_\square$.

It is not hard to prove that IEL^--deductions strongly normalise w.r.t. the rewriting system obtained by adding this detour elimination to the rewriting system Ξ for NJp recalled in Section 1.4:

Lemma 2.2.3. *Deductions in* IEL^- *strongly normalise w.r.t. the rewriting system obtained by adding* $\rho_\Box$ *to* Ξ.

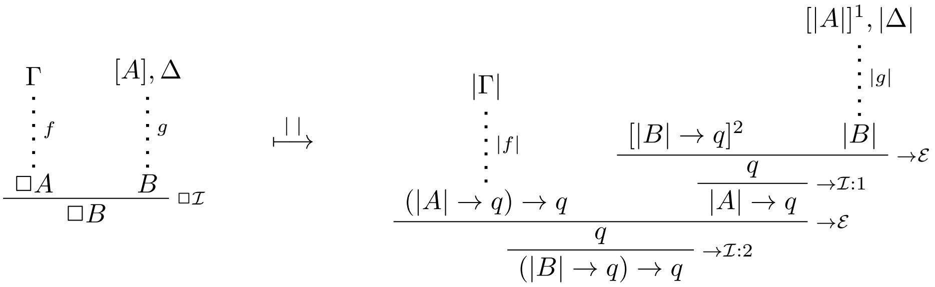
Proof. We define a translation[6] $|-|$ from the λ-calculus of IEL^--deductions to typed λ-calculus with products, sums, unit and empty types:

$$\begin{array}{lll}
|p| & := & p \\
|\bot| & := & \bot \\
|\top| & := & \top \\
|A \to B| & := & |A| \to |B| \\
|A \wedge B| & := & |A| \wedge |B| \\
|A \vee B| & := & |A| \vee |B| \\
|\Box A| & := & (|A| \to q) \to q
\end{array}$$

$$\begin{array}{l}
|x| := x \\
|\bot_J(f)| := \bot_J(|f|) \\
|*| := * \\
|\lambda x.\, f| := \lambda x.|f| \\
|fg| := |f||g| \\
|(f,g)| := (|f|,|g|) \\
|\pi_i(f)| := \pi_i(|f|) \\
|\mathsf{C}(f, x.f_1, y.f_2)| := \mathsf{C}(|f|, x.|f_1|, y.|f_2|) \\
|\mathsf{in}_i(f)| := \mathsf{in}_i(|f|) \\
|\mathsf{box}[x_1, \cdots, x_n].\, g\ \mathsf{with}\ f_1, \cdots, f_n| := \lambda k.|f_1|(\lambda x_1. \cdots |f_n|(\lambda x_n.\, k|g|) \cdots)
\end{array}$$

where q is an arbitrary atom.

For the sake of readability, it is convenient to rephrase the translation of $\Box\mathcal{I}$ in the natural deduction formalism:[7]



To prove strong normalisation it suffices now to prove that $\rho_\Box$ is preserved by this translation, which is almost straightforward. Thus, if IEL^- was not

[6] This function is introduced in [Kak16] to prove detour-elimination for the implicational fragment of basic intuitionistic modal logic $\mathbb{IK}$ by reducing the problem to normalization of simple type theory. Here we adopt the mapping to consider also product, sum, unit and empty types, keeping the original strategy due to [dG02].

[7] We consider w.l.o.g. the case with a single minor premise.

normalizing, then we would have an infinite chain of rewritings starting from, say, $f : A$. But this would lead to an infinite chain of reductions of NJp-deductions w.r.t. Ξ starting from $|f| : |A|$, contradicting strong normalization for NJp w.r.t. Ξ.

$\boxtimes$

The system of rewritings $\Xi + \rho_\Box$ however, does not suffice to establish analyticity of IEL^-. To see that, consider, for instance, the following IEL^--deduction f:

$$\dfrac{\Box(A\to(B\to C))\vee\bot \qquad \dfrac{\Box(A\to(B\to C))^3 \qquad \Box A \qquad \dfrac{A\to(B\to C)^1 \qquad A^2}{B\to C}{\to}\mathcal{E}}{\Box(B\to C)}\Box\mathcal{I}{:}1,2 \qquad \dfrac{\bot^4}{\Box(B\to C)}\bot_J}{\Box(B\to C)}\vee\mathcal{E}{:}3,4$$

Then we see that the IEL^--deduction

$$\dfrac{\begin{matrix}\vdots\, f\\ \Box(B\to C)\end{matrix} \qquad \Box B \qquad \dfrac{B\to C^5 \qquad B^6}{C}{\to}\mathcal{E}}{\Box C}\Box\mathcal{I}{:}5,6$$

is normal w.r.t. $\Xi + \rho_\Box$. In spite of this, the formula $\Box(B\to C)$ is not a subformula of either the conclusion $\Box C$ or of the undischarged hypotheses $\{\Box A, \Box B, \Box(A\to(B\to C))\vee\bot\}$.

Thus we add to $\Xi + \rho_\Box$ the following permutation conversions $\pi_\Box$:

$$\dfrac{\begin{matrix}\Gamma_1\\ \vdots\, f_1\\ \Box A_1\end{matrix} \quad \cdots \quad \dfrac{\vdots\, t \qquad \begin{matrix}\Gamma_i\\ \vdots\, f_i\\ \Box A_i\end{matrix}}{\Box A_i}\mathcal{E}+ \quad \cdots \quad \begin{matrix}\Gamma_n\\ \vdots\, f_n\\ \Box A_n\end{matrix} \qquad \begin{matrix}[A_1,\cdots,A_n],\Delta\\ \vdots\, g\\ B\end{matrix}}{\Box B}\Box\mathcal{I} \qquad \rightsquigarrow$$

$$\rightsquigarrow \qquad \dfrac{\vdots\, t \qquad \dfrac{\begin{matrix}\Gamma_1\\ \vdots\, f_1\\ \Box A_1\end{matrix} \quad \cdots \quad \begin{matrix}\Gamma_i\\ \vdots\, f_i\\ \Box A_i\end{matrix} \quad \cdots \quad \begin{matrix}\Gamma_n\\ \vdots\, f_n\\ \Box A_n\end{matrix} \qquad \begin{matrix}[A_1,\cdots,A_n],\Delta\\ \vdots\, g\\ B\end{matrix}}{\Box B}\Box\mathcal{I}}{\Box B}\mathcal{E}+$$

where $\mathcal{E}+$ is $\vee\mathcal{E}$ or $\bot_J$.

We now prove that IEL^--deductions strongly normalise w.r.t. this extended system of rewritings. Unfortunately, the translation $|-|$ does not preserve the required additional rewritings, but another simple translation does the job.

THEOREM 2.2.4. *Deductions in* IEL^- *strongly normalise w.r.t. the rewriting system* $\Xi + \rho_\Box + \pi_\Box$.

Proof. We define a new translation $\langle - \rangle$ from our modal λ-calculus to simple type theory with products, sums, unit and empty types as follows:

$$\begin{array}{lcl}
\langle \bot \rangle & := & \bot \\
\langle \top \rangle & := & \top \\
\langle p \rangle & := & p \\
\langle A \to B \rangle & := & \langle A \rangle \to \langle B \rangle \\
\langle A \wedge B \rangle & := & \langle A \rangle \wedge \langle B \rangle \\
\langle A \vee B \rangle & := & \langle A \rangle \vee \langle B \rangle \\
\langle \Box A \rangle & := & \langle A \rangle \vee q
\end{array}$$

$$\begin{array}{l}
\langle x \rangle := x \\
\langle * \rangle := * \\
\langle \bot_J(f) \rangle := \bot_J(\langle f \rangle) \\
\langle \lambda x\, f \rangle := \lambda x.\langle f \rangle \\
\langle f g \rangle := \langle f \rangle \langle g \rangle \\
\langle (f, g) \rangle := (\langle f \rangle, \langle g \rangle) \\
\langle \pi_i(f) \rangle := \pi_i(\langle f \rangle) \\
\langle \mathsf{C}(f, x.f_1, y.f_2) \rangle := \mathsf{C}(\langle f \rangle, x.\langle f_1 \rangle, y.\langle f_2 \rangle) \\
\langle \mathsf{in}_i(f) \rangle := \mathsf{in}_i(\langle f \rangle) \\
\langle \mathsf{box}[x_1, \cdots, x_n].\, g \,\mathsf{with}\, f_1, \cdots, f_n \rangle := \mathsf{C}(\langle f_n \rangle, x_n. \cdots \mathsf{C}(\langle f_2 \rangle, x_2.\mathsf{C}(\langle f_1 \rangle, x_1.\mathsf{in}_1(\langle g \rangle), y_1.\mathsf{in}_2(y_1)), y_2.\mathsf{in}_2(y_2)) \cdots, y_n.\mathsf{in}_2(y_n))
\end{array}$$

where q is an arbitrary atom.

As in the proof of the previous normalisation lemma, for the sake of readability, it is convenient to rephrase the translation of $\Box\mathcal{I}$ in the natural deduction formalism:[8]

$$\dfrac{\begin{array}{c}\Gamma_1\\ \vdots\, f_1\\ \Box A_1\end{array} \qquad \begin{array}{c}\Gamma_2\\ \vdots\, f_2\\ \Box A_2\end{array} \qquad \begin{array}{c}[A_1, A_2], \Delta\\ \vdots\, g\\ B\end{array}}{\Box B}\,\Box\mathcal{I} \qquad \xmapsto{\langle\ \rangle}$$

$$\xmapsto{\langle\ \rangle} \quad \dfrac{\begin{array}{c}\langle \Gamma_2 \rangle\\ \vdots\, \langle f_2 \rangle\\ \langle A_2 \rangle \vee q\end{array} \qquad \dfrac{\begin{array}{c}\langle \Gamma_1 \rangle\\ \vdots\, \langle f_1 \rangle\\ \langle A_1 \rangle \vee q\end{array} \qquad \dfrac{\begin{array}{c}[\langle A_1 \rangle]^1, [\langle A_2 \rangle]^2, \langle \Delta \rangle\\ \vdots\, \langle g \rangle\\ \langle B \rangle\end{array}}{\langle B \rangle \vee q}\,\vee\mathcal{I}_1 \qquad \dfrac{[q]^1}{\langle B \rangle \vee q}\,\vee\mathcal{I}_2}{\langle B \rangle \vee q}\,\vee\mathcal{E}{:}1 \qquad \dfrac{[q]^2}{\langle B \rangle \vee q}\,\vee\mathcal{I}_2}{\langle B \rangle \vee q}\,\vee\mathcal{E}{:}2$$

It is now straightforward to check that both $\rho_\Box$ *and* $\pi_\Box$ are preserved by this translation, and recovered by permutation conversions of Ξ.

Since deductions in NJp are strongly normalising w.r.t. Ξ, we have the desired strong normalisation for IEL^--deductions.

$\boxtimes$

[8] This time we can consider w.l.o.g. the case with two minor premises to make the pattern explicit.

COROLLARY 2.2.5. *Every* IEL^-*-deduction uniquely reduces to a normal* IEL^-*-deduction w.r.t.* $\Xi + \rho_\Box + \pi_\Box$.

Proof. We need to prove that every term of the modal λ-calculus for IEL^--deductions has a unique normal form w.r.t. $\Xi + \rho_\Box + \pi_\Box$. That holds if that rewriting system has the Church-Rosser property. It is not hard to see that it does satisfy the *weak* Church-Rosser property: The desired property then follows by Newman's lemma for rewriting systems.[9]

☒

2.3. ANALYTICITY AND OTHER PROPERTIES

In the proofs-as-programs paradigm, a normalisation theorem only states that any execution of a program written in our modal λ-calculus safely terminates. However, from the purely proof-theoretic point of view, Lemma 2.2.3 is less relevant then Theorem 2.2.4, since the latter is much more informative about the structure of the deductions in IEL^-. As Raymond Smullyan once wrote [Smu68]

> The real importance of cut-free proofs is not the elimination of cuts per se, but rather that such proofs obey the subformula principle.

Since normalisation of deductions corresponds to cut-elimination from derivations in sequent calculi [NvP08, Ch. 8], we could rephrase Smullyan's claim to refer to the relevance of (strong) normalisation results.

Subformula property is very easy to state, but giving a detailed proof of it requires some care.

In this section we see how to prove that IEL^--deductions which are normal w.r.t. $\Xi + \rho_\Box + \pi_\Box$ do satisfy that property, and make explicit all the details of the proof sketch of that very property for NJ given in [Pra71]. Afterwards, we derive further properties of the logic for intuitionistic belief.

2.3.1. PROOF OF THE SUBFORMULA PROPERTY

First we need to generalise the notion of proof-rewriting:

DEFINITION 2.3.1 (After [Pra71]). A maximal segment in an IEL^--deduction f is a sequence $A_1, \cdots, A_n$ of formula occurrences in f where:

1. A_1 is the conclusion of an $\mathcal{I}$-rule, or of $\bot_J$;
2. for $1 \leq i < n$, A_i is a minor premise of an $\vee\mathcal{E}$ rule, and A_{i+1} is its conclusion;

[9] A proof of Newman's result relating strong normalization and Church-Rosser property is given in [SU06, Prop. 3.6.2].

3. A_n is the major premise of an $\mathcal{E}$-rule, of $\bot_J$, or a minor premise of $\Box\mathcal{I}$.

Notice that, since in any $\vee\mathcal{E}$ rule, the conclusion is the same formula as the minor premises, all A_i in a maximal segment are (different occurences of) the same formula. Moreover, the rewritings in $\Xi + \rho_\Box + \pi_\Box$ are special instances of maximal segments $A_1, \cdots, A_n$.

By normal deduction we will refer to a deduction in IEL^- with no maximal segments.

Next we need to generalise the notion of branch in a deduction:

DEFINITION 2.3.2 (After [Pra71]). A path in an IEL^--deduction f is a sequence of formula occurrences $A_1, \cdots, A_n$ where:

1. A_1 is either an open assumption of f, or an assumption discharged by $\to\mathcal{I}$;

2. if $1 \leq i < k$, then

   - if A_i is the major premise of $\vee\mathcal{E}$, or a minor premise of $\Box\mathcal{I}$, then A_{i+1} is the corresponding assumption discharged by that rule;
   - if A_i

     a. a premise of $\bot_J$;
     b. a premise of a propositional $\mathcal{I}$-rule;
     c. a major premise of $\Box\mathcal{I}$;
     d. a major premise of an $\mathcal{E}$-rule other than $\vee\mathcal{E}$;
     e. a minor premise of $\vee\mathcal{E}$;

     then A_{i+1} is the conclusion of that rule;

3. A_k is either the end formula of the deduction, or the minor premise of $\to\mathcal{E}$.

We will call any sequence of formula occurrences in a path, in which one is a minor premise of an $\vee\mathcal{E}$ and the following one is its conclusion, a segment.

We can now show that any path in a normal IEL^--deduction has a very neat structure.

LEMMA 2.3.3. *Any path in a normal* IEL^-*-deduction f is made of segments ordered as follows:*

- *first, there are segments where the last formula occurrence is a major premise of an $\mathcal{E}$-rule, or a minor premise of $\Box\mathcal{I}$;*

- *then, there are segments whose last formula occurrence is the premise of a $\bot_J$;*

- *finally, there are segments where the last formula occurrence is a (major) premise of an $\mathcal{I}$-rule.*[10]

We call the first segment ending in the premise of a $\bot_J$, or in the (major) premise of an $\mathcal{I}$-rule the minimum segment *of f, and the formula repeating in it is called the* minimum formula.

Proof. Suppose that the claim were false. Then we would have one of the following cases:

a. a segment ends with a (major) premise of a $\mathcal{I}$-rule, followed by a segment that ends in the major premise of an $\mathcal{E}$-rule;

b. a segment ends with a (major) premise of a $\mathcal{I}$-rule, followed by a segment that ends in a minor premise of $\Box\mathcal{I}$;

c. a segment ends with a (major) premise of a $\mathcal{I}$-rule, followed by a segment that ends in the premise of $\bot_J$;

d. a segment ends with the premise of $\bot_J$, followed by a segment that ends in the premise of $\bot_J$;

e. a segment ends with the premise of $\bot_J$, followed by a segment that ends in the major premise of an $\mathcal{E}$-rule;

f. a segment ends with the premise of $\bot_J$, followed by a segment that ends in the minor premise of $\Box\mathcal{I}$.

Each of those cases defines a maximal segment, contra the hypothesis that f is normal, or an impossible situation, by connective mismatch. Therefore we conclude that the lemma holds.

⊠

At this point we need to consider a

PROPOSITION 2.3.4. *In a path $A_1, \cdots, A_n$ of a normal* IEL^-*-deduction, every formula A in it is either a subformula of A_1, or of A_n.*

Proof. If A occurs before the minimum segment, it is a subformula of A_1, since all rules leading to A either go from major premises of $\mathcal{E}$-rules others than $\vee\mathcal{E}$ to their conclusions, or from major premises of $\vee\mathcal{E}$ to assumptions discharged by that very rule, or from minor premises of $\Box\mathcal{I}$ to the appropriate assumption discharged by that rule, or from minor premises of $\vee\mathcal{E}$ to its conclusion.

If A is the minimum formula, or occurs after the minimum segment, then all the rules after A in the path are $\mathcal{I}$-rules, so that the path proceeds from premises of propositional $\mathcal{I}$-rules, to their conclusions, or from major premises

[10] This means that the structure of normal IEL^- is the same as the one in Figure 1.5.

of $\Box\mathcal{I}$ to their conclusions, or from minor premises of $\vee\mathcal{E}$ to their conclusions, and in each case the premises are subformulas of the conclusions.

⊠

Now we want to prove that every formula in an IEL^--deduction is a subformula of the starting assumption of the path it is on, or of its ending formula. That is implied by the following

PROPOSITION 2.3.5. *Every formula A in an IEL^--deduction f lies in at least one path of f.*

Proof. By induction on the height $\mathfrak{h}(f)$ of f.

If $\mathfrak{h}(f) = 0$, then f is just an assumption, and that single formula makes a path. Otherwise, apply the inductive hypothesis to the subdeductions ending in the premises of the last rule of f. Each premise of that rule lies on a path, and it must be its ending formula. Any path ending in the premise of $\bot_J$, a propositional $\mathcal{I}$-rule, the major premise of $\to\mathcal{E}$, a minor premise of $\vee\mathcal{E}$, or the major premise of $\Box\mathcal{I}$ can be extended to a path in f the conclusion of the rule is on. Every other path in the subdeductions ending in the premises of the last rule in f is still a path in f.

⊠

After the following definition, we can proceed with the last intermediate lemmas towards the main result.

DEFINITION 2.3.6. Given an IEL^--deduction f, and a path in f we define its order $\mathfrak{o}$ as follows:

- the order $\mathfrak{o}$ of a path ending in the end-formula of f is 0;
- the order of a path ending in the minor premise of $\to\mathcal{E}$, whose major premise lies on a path with order $\mathfrak{o}$, is $\mathfrak{o}+1$.

LEMMA 2.3.7. *If a formula A lies on a path of order $\mathfrak{o}$ in a normal IEL^--deduction f, then it is either a subformula of the first formula of that path, of the endformula of that path, or of a formula lying on a path of order $\mathfrak{o}-1$.*

Proof. If $\mathfrak{o}$ equals 0, then the path ends with the end-formula of f, and we rely on Prop. 2.3.4.

Otherwise, the path ends with a minor premise B of $\to\mathcal{E}$. By Prop. 2.3.4, we know that A is a subformula of B or of the first formula in the path. In the former case, it is also a subformula of the major premise, which is on a path of order $\mathfrak{o}-1$, and we are done.

⊠

LEMMA 2.3.8. *If the starting formula A_1 of a path with order $\mathfrak{o}$ in an IEL^--deduction is discharged by $\to\mathcal{I}$ in f, then the conclusion of that rule is on the same path, or on a path of order $< \mathfrak{o}$.*

Proof. Suppose that A_1 is not discharged on the path. Then it must be discharged on a branch between the ending formula A_n of that path and the end-formula of f. All formulas lying there are on paths of order $< \mathfrak{o}$.

$\boxtimes$

THEOREM 2.3.9 (**Subformula principle**). *Every formula B occurring in a normal* IEL^-*-deduction f of A from assumptions Γ is a subformula of A or of some formula in Γ.*

Proof. By induction on the order $\mathfrak{o}$ of a path an occurrence of B is on.

If $\mathfrak{o} = 0$, then B is on a path ending with the end-formula of f. By Prop. 2.3.4, B is a subformula of the end-formula, or of the starting formula of the path. By Lemma 2.3.8, since there are no paths of order less than 0, if that starting formula is discharged by $\to \mathcal{I}$, the conclusion of that very rule is on the same path. Since that conclusion appears after the minimum segment, it is a subformula of the end-formula of f.

Otherwise, by Prop. 2.3.4, B is a subformula of the ending formula of the path, or of its starting formula. If the latter is an assumption discharged on the path, B is a subformula of the ending formula of that path, which, in turn, is a minor premise of $\to \mathcal{E}$. Therefore it is also a subformula of the associated major premise, which lies on a path of order $< \mathfrak{o}$: By applying the inductive hypothesis we are done. If B is a subformula of the starting formula of the path, and that assumption is discharged by $\to \mathcal{I}$ on a different path, by Lemma 2.3.8, the conclusion of that rule is on a path of order $< \mathfrak{o}$, and by applying the inductive hypothesis we are done.

$\boxtimes$

2.3.2. FURTHER PROPERTIES OF INTUITIONISTIC BELIEF

In [AP16], several properties of the logic for intuitionistic belief are established by means of semantic techniques.

Here all those properties are proven by purely proof-theoretic remarks about the natural deduction system IEL^-.

LEMMA 2.3.10. *The logic of intuitionistic belief is consistent.*

Proof. If it was not the case, then we would have $\mathsf{IEL}^- \vdash \bot$. By normalisation, there would be a normal IEL^--deduction of $\bot$ from no hypothesis, which is impossible because of the subformula principle and the absence of any $\mathcal{I}$-rule for $\bot$.

$\boxtimes$

LEMMA 2.3.11. *The calculus* IEL^- *is decidable.*

Proof. By Theorem 2.3.9, proof search in IEL^- is bounded by the degree of the formula one wishes to deduce.

$\boxtimes$

Let's say that a deduction is neutral if it consists of an assumption, or its last rule is not an $\mathcal{I}$-rule.

PROPOSITION 2.3.12. *In any normal and neutral* IEL^-*-deduction of A from Γ, $\Gamma \neq \varnothing$.*

Proof. By induction on the height $\mathfrak{h}$ of the given deduction.

The case $\mathfrak{h} = 0$ is trivial.

If $\mathfrak{h} > 0$, we proceed by considering the last rule of the deduction:

a. if that rule is $\to \mathcal{E}$, we can apply the inductive hypothesis to the major premise, which is neutral by normality of the deduction;

b. if that rule is $\wedge\mathcal{E}$, we can apply the inductive hypothesis to the premise, which is neutral by normality of the deduction;

c. if that rule is $\vee\mathcal{E}$, we can apply the inductive hypothesis to the major premise, which is neutral by normality of the deduction;

d. if that rule is $\bot_J$, we can apply the inductive hypothesis to the premise, which is neutral since there are no $\mathcal{I}$-rules for $\bot$.

⊠

LEMMA 2.3.13 (Canonicity). *In any normal* IEL^-*-deduction of A, the last rule applied is the introduction rule for the main connective of A.*

Proof. By Proposition 2.3.12, since $\Gamma = \varnothing$ here, and the deduction is normal by assumption, its last rule cannot be an elimination.

⊠

COROLLARY 2.3.14 (Admissibility of the reflection rule). *If* $\mathsf{IEL}^- \vdash \Box A$*, then* $\mathsf{IEL}^- \vdash A$*.*

Proof. By Lemma 2.3.13, since $\Box A$ is deducible from no assumption, its last rule in the corresponding normal deduction must be $\Box\mathcal{I}$.

⊠

COROLLARY 2.3.15 (Disjunction property). *If* $\mathsf{IEL}^- \vdash A \vee B$*, then* $\mathsf{IEL}^- \vdash A$ *or* $\mathsf{IEL}^- \vdash B$*.*

Proof. Assume $\mathsf{IEL}^- \vdash A \vee B$. Then consider a normal IEL^--deduction of $A \vee B$. By Lemma 2.3.3, that deduction must end either by $\bot_J$, $\vee\mathcal{I}_1$, or $\vee\mathcal{I}_2$. But since that very deduction has no assumptions, it cannot end by $\bot_J$. Then we have the result.[11]

⊠

[11] Even more directly: It is a straightforward application of Lemma 2.3.13.

COROLLARY 2.3.16 ($\Box$-primality). *If* $\mathsf{IEL}^- \vdash \Box A \vee \Box B$, *then* $\mathsf{IEL}^- \vdash A$ *or* $\mathsf{IEL}^- \vdash B$.

Proof. The result follows by the disjunction property and the admissibility of the reflection rule.

$\boxtimes$

COROLLARY 2.3.17 (Modal disjunction property). *If* $\mathsf{IEL}^- \vdash \Box(A \vee B)$, *then* $\mathsf{IEL}^- \vdash \Box A$ *or* $\mathsf{IEL}^- \vdash \Box B$.

Proof. If we have a deduction of $\Box(A \vee B)$, then by the reflection rule we have a deduction of $A \vee B$. By the disjunction property we have a deduction of A or a deduction of B. In each case, by applying $\Box\mathcal{I}$ we have the desired result.
$\boxtimes$

Finally, notice that the consistency of IEL^- can be established also as a corollary of Lemma 2.3.13, without recurring to the subformula principle.

2.4. PROOF-THEORETIC SEMANTICS FOR IEL^-

If λ-calculus gives a computational semantics of proofs in NJp – and, as we showed in the previous section, in IEL^- also – category theory furnishes the tools for an 'algebraic' semantics which is *proof relevant* – i.e. contrary to traditional algebraic semantics based on Heyting algebras and to relational semantics based on Kripke models, it focuses on the very notion of proof, distinguishing between different deductions of the same formula.

We know that cartesian product models conjunction, and exponential models implication. Any category having products and exponentials for any of its objects is called cartesian closed (CCCat); moreover, if it has also (weak) coproducts – modelling disjunction – it is called bi-cartesian closed (bi-CCCat): $\top$ and $\bot$ correspond to empty product – the terminal object $\mathbf{1}$ – and empty coproduct – the (weak) initial object $\mathbf{0}$ – respectively.[12]

For our calculus, in order to capture the behaviour of the epistemic modality, some more structure is required, which is what will be sketched in the next subsections.

2.4.1. SOME FUNCTORS ON MONOIDAL CATEGORIES

DEFINITION 2.4.1. Given a CCCat $\mathscr{C}$, a monoidal endofunctor consists of a functor $\mathfrak{F} : \mathscr{C} \to \mathscr{C}$ together with

- a natural transformation

$$m_{A,B} : \mathfrak{F}A \times \mathfrak{F}B \to \mathfrak{F}(A \times B);$$

[12] The reader is referred to the classic [LS88] for the details of such completeness result.

- a morphism

$$m_1 : 1 \to \mathfrak{F}1,$$

preserving the monoidal structure of $\mathscr{C}$.[13]

These are called structure morphisms of $\mathfrak{F}$.

It is quite easy to see that a monoidal endofunctor on the category of logical formulas induces a modal operator satisfying K-schema, as proven in [dPR11].

DEFINITION 2.4.2. Given any category $\mathscr{C}$, an endofunctor $\mathfrak{F} : \mathscr{C} \to \mathscr{C}$ is pointed iff there exists a natural transformation

$$\pi : Id_{\mathscr{C}} \Rightarrow \mathfrak{F}$$
$$\pi_A : A \to \mathfrak{F}A$$

$$\begin{tikzcd}
A \arrow[r, "\pi_A"] \arrow[d, "f"'] & \mathfrak{F}A \arrow[d, "\mathfrak{F}f"] \\
B \arrow[r, "\pi_B"'] & \mathfrak{F}B
\end{tikzcd}$$

π is called the point of $\mathfrak{F}$.

In the present setting, a pointed endofunctor on the category of logical formulas 'represents' the co-reflection schema.

Since we want to give a semantics of proofs – and not simply of derivability – in IEL^-, we need a further notion from category theory.

DEFINITION 2.4.3. Given a monoidal category $\mathscr{C}$, and monoidal endofunctors $\mathfrak{F}, \mathfrak{G} : \mathscr{C} \to \mathscr{C}$, a natural transformation $\kappa : \mathfrak{F} \Rightarrow \mathfrak{G}$ is monoidal when the following commute:

$$\begin{tikzcd}
\mathfrak{F}A \times \mathfrak{F}B \arrow[r, "m^{\mathfrak{F}}_{A,B}"] \arrow[d, "\kappa_A \times \kappa_B"'] & \mathfrak{F}(A \times B) \arrow[d, "\kappa_{A \times B}"] \\
\mathfrak{G}A \times \mathfrak{G}B \arrow[r, "m^{\mathfrak{G}}_{A,B}"'] & \mathfrak{G}(A \times B)
\end{tikzcd}$$

and

$$\begin{tikzcd}
1 \arrow[r, "m^{\mathfrak{F}}_1"] \arrow[d, equal] & \mathfrak{F}(1) \arrow[d, "\kappa_1"] \\
1 \arrow[r, "m^{\mathfrak{G}}_1"'] & \mathfrak{G}(1)
\end{tikzcd}$$

[13] See [Mac13, Sect. VII.1] for the corresponding commuting diagrams and the definition of monoidal category.

2.4.2. Categorical interpretation

For our intents, all the previous categorical notions will be packed in a single kind of structures.

Definition 2.4.4. An IEL^{-}-category is given by a bi-CCCat $\mathscr{C}$ together with a monoidal pointed endofunctor $\mathfrak{K}$ whose point κ is monoidal.

The reader is invited to check that by taking the poset reflection of an IEL^{-}-model, one obtains a model for *provability* in the logic of intuitionistic belief according to the usual algebraic semantics for modal logics, i.e. a Heyting algebra with an operator satisfying the axioms and rules of $\mathbb{IEL}^{-}$.

Shifting to the categorical perspective, a general IEL^{-}-model provides thus the tools for an algebraic semantics of deductions in IEL^{-}.

But what kind of identity of proofs does an IEL^{-}-category capture?

For sure, if two IEL^{-}-deductions normalise to the same proofs w.r.t. Ξ, then that equality is adequately modelled by any IEL^{-}-category.

At the beginning of Section 2.3, I claimed that Lemma 2.2.3 does not have a deep proof-theoretic relevance, since normalisation w.r.t. $\Xi + \rho_\Box$ is not enough for deriving the subformula principle for IEL^{-}.

It is not hard to see that $\rho_\Box$ is what is needed to prove that the construction induced by the modal operator preserves arrow composition.

However, that construction cannot be a functor unless one considers the following extensionality principle $\eta_\Box$ for $\Box\mathcal{I}$:

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \Box A\end{array}\qquad [A]}{\Box A}\,\Box\mathcal{I} \qquad\rightsquigarrow\qquad \begin{array}{c}\Gamma\\ \vdots\\ \Box A\end{array}$$

Now, it is not hard to see that IEL^{-}-deductions strongly normalise w.r.t. the rewriting system $\Xi + \rho_\Box + \eta_\Box$.[14]

Henceforth, we can now show the adequacy of this categorical interpretation.

[14] As a matter of fact, there are two ways to see that: One could simply reason as in the proof of Lemma 2.2.3, and showing that $\eta_\Box$ can be simulated in a rewriting system for simple type theory extended by function extensionality $\eta_\to$, which is strongly normalising [SU06, Lem.1.3.11]; or, one could easily prove that in any sequence of $\Xi + \rho_\Box + \eta_\Box$-reductions, $\eta_\Box$-rewritings can be postponed, so that Lemma 2.2.3 implies strong normalisation w.r.t. the system including $\eta_\Box$.

The reader is also invited to notice that even the translation $\langle - \rangle$ is capable of mimicking $\eta_\Box$ in a simple type theory extended by a sum extensionality principle $\eta_\vee$, which in type-theoretic syntax can be defined as

$$\mathsf{C}(f, x.\mathsf{in}_1(x), y.\mathsf{in}_2(y)) \ \rightsquigarrow\ f.$$

Normalisation of IEL^{-}-deductions w.r.t. $\Xi + \rho_\Box + \pi_\Box + \eta_\Box$ is then derived from the fact that $\eta_\vee$-reductions can be postponed in any sequence of $\Xi + \eta_\vee$-rewritings, so that simple type theory is normalising w.r.t. $\Xi + \eta_\vee$ too.

THEOREM 2.4.5 (**Soundness**). *Let $\mathscr{C}$ be an IEL^--category. Then there is a canonical interpretation $[\![-]\!]$ of IEL^- in $\mathscr{C}$ such that*

- *a formula A is mapped to a $\mathscr{C}$-object $[\![A]\!]$;*
- *a deduction f of $A_1, \cdots, A_n \vdash_{\mathsf{IEL}^-} B$ is mapped to an arrow*
$$[\![f]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!];$$
- *for any two deductions f and g which are equal modulo $\Xi{+}\rho_\Box{+}\eta_\Box$-rewritings, we have $[\![f]\!] = [\![g]\!]$.*

Proof. By structural induction on $f : \vec{A} \vdash_{\mathsf{IEL}^-} B$.

The intuitionistic cases are interpreted according to the remarks about bi-CCCats at the beginning of this section.

We overload the notation using $\langle \Box, m, \kappa \rangle$ for the monoidal pointed endofunctor of $\mathscr{C}$, its structure morphisms, and its point.

The deduction

$$\frac{f_1 : \Gamma_1 \vdash \Box A_1 \qquad \cdots \qquad f_n : \Gamma_n \vdash \Box A_n \qquad g : [A_1, \cdots, A_n], C_1, \cdots, C_m \vdash B}{\Box B}$$

is mapped to

$$(\Box[\![g]\!]) \circ m_{[\![A_1]\!], \cdots, [\![A_n]\!], [\![C_1]\!], \cdots, [\![C_m]\!]} \circ [\![f_1]\!] \times \cdots \times [\![f_n]\!] \times \kappa_{[\![C_1]\!]} \times \cdots \times \kappa_{[\![C_m]\!]},$$

where $m_{X_1, \cdots, X_n}$ is defined inductively as

$$m_{X_1, \cdots, X_{n-1}, X_n} := m_{X_1 \times \cdots \times X_{n-1}, X_n} \circ (m_{X_1, \cdots, X_{n-1}}) \times id_{\Box X_n}.$$

It is straightforward to check that identity modulo $\eta_\Box$ holds in the category $\mathscr{C}$ by functoriality of $\Box$.

Identity modulo $\rho_\Box$ is also valid by naturality of m and κ: The reader is invited to check that κ must be monoidal in order to model correctly the following special case[15]

$$\dfrac{\begin{array}{c}\Gamma_1\\ \vdots \\ \dfrac{A_1}{\Box A_1}\,\Box\mathcal{I}\end{array} \qquad \cdots \qquad \begin{array}{c}\Gamma_n\\ \vdots \\ \dfrac{A_n}{\Box A_n}\,\Box\mathcal{I}\end{array} \qquad \begin{array}{c}[A_1, \cdots, A_n]^1, C_1, \cdots, C_m\\ \vdots \\ B\end{array}}{\Box B}\,\Box\mathcal{I}{:}1 \qquad \rightsquigarrow$$

$$\rightsquigarrow \qquad \begin{array}{c}\begin{array}{cccc}\Gamma_1 & \cdots & \Gamma_n & \\ \vdots & & \vdots & \\ A_1 & \cdots & A_n & C_1, \cdots, C_m\end{array}\\ \vdots \\ \dfrac{B}{\Box B}\,\Box\mathcal{I}\end{array} \qquad .$$

[15] Everything reduces to long categorical calculations.

$\boxtimes$

It remains to show that this interpretation is also complete.

THEOREM 2.4.6 (**Completeness**). *If the interpretation of two IEL^--deductions is equal in all IEL^--categories, then the two deductions are equal modulo $\Xi + \rho_\Box + \eta_\Box$-rewritings.*

Proof. We proceed by constructing a term model for the modal λ-calculus for IEL^--deductions. Consider the following category $\mathcal{M}$:

- its objects are formulas;
- an arrow $f : A \to B$ is an IEL^--deduction of B from A modulo $\Xi + \rho_\Box + \eta_\Box$-rewritings;
- identities are given by assuming a hypothesis;
- composition is given by transitivity of deductions.

Then $\mathcal{M}$ has a bi-cartesian closed structure given by Ξ-rewritings for NJp.

Moreover, the modal operator $\Box$ induces a functor $\mathfrak{K}$ by mapping A to $\Box A$, and

$$\begin{array}{c} A \\ \vdots \\ B \end{array} \quad \overset{\mathfrak{K}}{\longmapsto} \quad \dfrac{\Box A \qquad \begin{array}{c} [A] \\ \vdots \\ B \end{array}}{\Box B}\,\Box\mathcal{I}$$

which preserves identities by $\eta_\Box$, and preserves composition as a special case of $\rho_\Box$.

The structure morphism is given by

$$\dfrac{\dfrac{\Box A \wedge \Box B}{\Box A}\wedge\mathcal{E} \qquad \dfrac{\Box A \wedge \Box B}{\Box B}\wedge\mathcal{E} \qquad \dfrac{[A] \qquad [B]}{A \wedge B}\wedge\mathcal{I}}{\Box(A \wedge B)}\,\Box\mathcal{I}$$

whose properties follow as a special case of $\rho_\Box$.

The point is given by $\dfrac{A}{\Box A}$ and its characteristic property is given as a special case of $\rho_\Box$. Finally, such a point is monoidal by $\rho_\Box$ up to Ξ-rewritings.

Thus, if an equation between interpreted IEL^--deductions holds in all IEL^--categories, then it holds also in $\mathcal{M}$, so that those deductions are equal whenever they normalise to the same proof w.r.t. $\Xi + \rho_\Box + \eta_\Box$-rewritings.

$\boxtimes$

Related work

The calculus IEL^- presented here is an original natural deduction system for Artemov and Protopopescu's logic of intuitionistic belief.

Before the papers [PB21b, PB21a], the only proof-theoretic investigations on the intuitionistic logics of [AP16] involved standard sequent calculi as witnessed by [KY16, SS19].

Sequent calculi internalising relational semantics for those logics are developed in [Fio21] with an eye towards complexity-related aspects.

The paper [Rog21] is closer to the perspective of the material presented here. In that work, a natural deduction system for the $\bot, \vee$-free fragment of the logic for intuitionistic belief is discussed, and a categorical interpretation of the belief modality is given in terms of applicative functors, described first in [MP08], along with an algebraic semantics for a first-order extension of the logic for intuitionistic knowledge. Since the tensorial strength and the co-reflection schemas are "naturally" interderivable over the minimal intuitionistic modal logic IK, the applicative functor based interpretation is equivalent to the one discussed in the present work. On the purely proof-theoretic side, the calculus discussed in [Rog21] differs from IEL^- for extending the natural deduction system IK of [dPR11] with a further modal introduction rule mimicking co-reflection. Our preliminary work [PB19] had already discussed normalisation and a categorical reading of that formalisation option, and expressed some of its limits. These become explicit when $\mathbb{IEL}^-$ is considered as a starting point for axiomatising various logics involving co-reflection, not necessarily related to the epistemic reading of the modality.

By adding different sorts of elimination rules, our IEL^- characterises indeed as an invariant kernel of several potential natural deduction systems capturing modal notions even outside the original verificationist account of belief and knowledge. An evidence for this claim is given by the very next two chapters.

3

Natural deduction for intuitionistic knowledge

Consider the following scenario.

Your favourite top-rated mathematician has posted on her blog a proof-sketch of a conjecture A in a very specialised mathematical field. Her informal argument is convincing, and her expertise justifies your trust in the conjecture. You are not an expert on that field, but she is; you do not have direct access to a detailed proof of A, but is reasonable for you to believe that A does hold.

This is completely in line with the verificationist account of *belief* in [AP16] that we have seen in Chapter 2. But how could you safely claim that you *know* that A is a theorem indeed?

In philosophy, knowledge is usually defined as *justified true belief* (JTB). These conditions are considered – in general, but not universally [IS18] – to be individually necessary and jointly sufficient for defining knowledge: One cannot know something without believing it (condition *B*); and one does not know a thing unless she also has reasons for believing it (condition *J*). Condition *B* lets us overload the formal epistemic notation, so that we can use the same $\Box$-modality to capture both epistemic states. Condition *J* implies that co-reflection $A \to \Box A$ is still valid when dealing with knowledge. What about condition *T*?

Even at the very intuitive level, the standard way of capturing condition *T* – i.e. adding the schema $\Box A \to A$ to the modal principles of out theory – reveals disastrous in the present setting: It causes the modal collapse of the whole language.

It is then appropriate to ask how to formalise the factivity of knowledge required by condition *T* according to Artemov and Protopopescu's verificationist epistemology. In order to answer that question, let us return to the scenario at the beginning of the chapter, and expand it a little bit.

Some time has passed since the mathematician's post about the conjecture A. You are again surfing the Net looking for new results in the field A belongs to, and come across another post about that very conjecture. This time, it is a famous top-rated computer scientist's blog announcing that A has been refuted. And there's more: Her refutation has been computerised, and the formal proof is freely available on the Net. If you know the programming language at the base of her computer program for refuting A – a proof assistant, or a highly specialised theorem prover – you can read the proof, and see that $\neg A$ holds, so that your original trust in A was misplaced.

In [AP16] five different schemas are proposed to capture this idea:

1. $\neg A \rightarrow \neg\Box A$;
2. $\neg(\Box A \wedge \neg A)$;
3. $\neg\Box\bot$;
4. $\neg\neg(\Box A \rightarrow A)$;
5. $\Box A \rightarrow \neg\neg A$.

Principle 3, despite being the weakest among 1-5, is arguably the closest to a naive intuitionistic version of condition T – "you cannot known something false" – while principle 4 is the double negation of the classical schema for condition T, and principle 1 is its contrapositive. Principle 5 is chosen by Artemov and Protopopescu to extend $\mathbb{IEL}^-$, for intuitionistic *belief*, to the logic of intuitionistic *knowledge* $\mathbb{IEL}$, but, as they notice, all of those principles are intuitionistically equivalent over $\mathbb{IEL}^-$, and each of them is intuitionistically weaker that the classical schema for condition T.

This verificationist version of factivity of knowledge allows a rational agent to know that a proposition A holds even if she has not a proof of A. This is not surprising at all, if we consider empirical knowledge, and the previous scenario about the conjecture A suggests that the same is valid for mathematical knowledge: Even if the proof of $\neg A$ were classified, or written in a computer program completely unknown to you, you would still know that A does not hold, and your original belief in A would have been false.

What [AP16] is proposing is, in a sense, a formal BHK-based version of Robert Nozick's general definition of *knowledge as belief tracking the truth of the matter under discussion* [Noz81].[1]

In more plain terms: The verificationist account of epistemic states in [AP16] is inspired by the intuitionistic interpretation of mathematical rationality, but its potential applications are not limited to the realm of mathematics.

Instead of talking about conjectures, proof sketches, and computer checked proofs, one could consider the following much more ordinary scenario.
Any stopped clock tells the right time twice a day. That statement has become a platitude. But look at such a clock at the right moment, and according to the

[1] Notice, however, that Nozick's account denies also the closure under known logical entailment – i.e. the schema K – that is endorsed in [AP16]. This denying is due to Nozick's original aim to criticise the basic assumptions of [Get63] questioning the *JTB* theory on knowledge. That Nozickian position about K is, in any case, challenged by several views that still adopt Nozick's proposal for defining knowledge: See e.g. [Lip00].

JTB theory, you should *know* the time, since your belief is (uselessly) justified and true.

On the contrary, from the verificationist view-point, this example fails to be considered knowledge producing: If my belief about the time is actual knowledge, then, if it hadn't be that very time, I wouldn't have believed it was. The intuitionistic version of this argument is captured by any of the modal schemas 1-5 above, but principle 2 is, as far as we see, the one that better reveals the pattern of reasoning shared by the clock and the conjecture examples.

In the present chapter, a natural deduction calculus for intuitionistic knowledge is introduced, consisting of an extension of the system IEL^- – already discussed in Chapter 2 – by an appropriate $\Box\mathcal{E}$-rule. The resulting system is denoted by IEL.

The presentation loosely follows the same structure as Chapter 2: After recalling Artemov and Protopopescu's axiomatic calculus and relational semantics for intuitionistic knowledge in Section 3.1, the natural deduction system IEL is introduced and proven to be equivalent to its axiomatic counterpart in Section 3.2. Its final subsection discusses an original proof of a strong normalisation theorem for IEL-deductions, and presents a modal λ-calculus for handling proofs in IEL in a computational manner via an algebraic notation. That extension of the proof-as-programs paradigm for the knowledge operator is rather natural, and requires no deep tweaks for the calculus considered in Section 2.2, as the reader will see. Section 3.3 deals with relevant properties of normal IEL-deductions: on the lines of Section 2.3, a proof of the subformula principle is given, and all the results syntactically proven for IEL^- are shown to hold also for IEL by using the same proof-theoretic techniques.

Writing and Revision Notes. This chapter exclusively contains unpublished original material on the logic of intuitionistic knowledge. The results and proofs here presented have been developed in parallel with their analogous of Chapter 2. The design principles leading to IEL^- have guided the definition of IEL, and the proof strategies for the meta-properties of the former have been adapted to the analogous properties of the latter. Thus everything reveals a certain modularity of this approach.

3.1. Axiomatic calculus for intuitionistic knowledge

The axiomatisation and relational semantics for the logic of intuitionistic knowledge is introduced in [AP16] as follows.

3.1.1. SYSTEM $\mathbb{IEL}$

DEFINITION 3.1.1. $\mathbb{IEL}$ is the axiomatic calculus extending $\mathbb{IEL}^-$ by the axiom schema corresponding to the principle of intuitionistic factivity of knowledge

$$\Box A \to \neg\neg A.$$

Thus, $\mathbb{IEL}$ is defined as the system made of

- Axiom schemas for intuitionistic propositional logic;
- Axiom schema $\mathsf{K} : \Box(A \to B) \to \Box A \to \Box B$;
- Axiom schema of co-reflection $A \to \Box A$;
- Axiom schema of intuitionistic reflection $\Box A \to \neg\neg A$;
- Modus Pones $\dfrac{A \to B \qquad A}{B}\,\textit{MP}$ as the only inference rule.

As usual, we write $\Gamma \vdash_{\mathbb{IEL}} A$ when A is derivable in $\mathbb{IEL}$ assuming the set of hypotheses Γ, and we write $\mathbb{IEL} \vdash A$ when $\Gamma = \varnothing$.

This system defines, as for intuitionistic belief, a normal modal intuitionistic logic:

PROPOSITION 3.1.2. *The following hold:*

(i) Necessitation rule $\dfrac{A}{\Box A}$ *is derivable in* $\mathbb{IEL}$*;*

(ii) The deduction theorem holds in $\mathbb{IEL}$*;*

(iii) $\mathbb{IEL}$ *is a normal intuitionistic modal system;*

(iv) *1.* $\mathbb{IEL} \vdash \neg A \to \neg\Box A$*;*

2. $\mathbb{IEL} \vdash \neg(\Box A \wedge \neg A)$*;*

3. $\mathbb{IEL} \vdash \neg\Box\bot$*;*

4. $\mathbb{IEL} \vdash \neg\neg(\Box A \to A)$*;*

Proof. See [AP16].

⊠

Notice that intuitionistic reflection is classically equivalent to the schema T for classical reflection $\Box A \to A$. The former is justified by the BHK reading of the knowledge modality as the result of a process of verification, and, in turn, makes knowledge justify the *mere possibility* of proof. Provability of $\neg\Box\bot$, however, shows that $\mathbb{IEL}$ is not capable of distinguishing between knowledge from provably consistent belief: This means that philosophical criticisms to $\mathbb{S}5$-based epistemic logics are biting $\mathbb{IEL}$ too.

3.1.2. Kripke semantics for $\mathbb{IEL}$

The paper [AP16] proposes the following class of Kripke models for intuitionistic knowledge.

Definition 3.1.3. A model for $\mathbb{IEL}$ is a model for $\mathbb{IEL}^-$ $\langle W, \leq, v, E\rangle$ (as in Def. 2.1.3), where the relation E satisfies the seriality condition

- for any $x \in W$, there exists a $y \in W$ such that xEy.

We write $\mathbb{IEL} \vDash A$ iff A is true in each model for $\mathbb{IEL}$.

The seriality condition on E is what assures the consistency of audits, since $w \nVdash \Box\bot$ for any world w.

This extended semantics is still adequate to the calculus:

Theorem 3.1.4. *The following hold:*

(**Soundness**) *If* $\mathbb{IEL} \vdash A$*, then* $\mathbb{IEL} \vDash A$*.*

(**Completeness**) *If* $\mathbb{IEL} \vDash A$*, then* $\mathbb{IEL} \vdash A$*.*

Proof. Soundness is proven by induction on the derivation of A.

Completeness is proven by a standard construction of a canonical model. See [AP16] for the details.

☒

3.2. Natural deduction for intuitionistic knowledge

The natural deduction that we are going to present is a modular extension of IEL^- that, in a sense, balances the structural asymmetry of the latter by means of a rule for eliminating modal formulas. The harmony lacking to IEL^- is thus recovered in IEL.

For this calculus, a strong normalisation theorem is proven w.r.t. an extended system of proof-rewritings that also considers $\Box\mathcal{I}$-$\Box\mathcal{E}$ detours. As for IEL^-, a modal λ-calculus is naturally defined by adding a destructor for $\Box$-types.

3.2.1. System IEL

The natural deduction system for intuitionistic knowledge under discussion is obtained by extending IEL^- with an elimination rule for the epistemic modality. That rule is the operational counterpart of intuitionistic reflection, and captures the intuitionistic principle of factivity of knowledge described in the introduction of this chapter.

Definition 3.2.1. Let IEL be the system extending the natural deduction calculus IEL^- of Def. 2.2.1 by the following elimination rule:

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \Box A\end{array}\qquad\begin{array}{c}[A],\Delta\\ \vdots\\ \bot\end{array}}{\bot}\;\Box\mathcal{E}$$

where Γ and Δ are *multisets of formulas*, and A is discharged by $\Box\mathcal{E}$.

We say that $\Box A$ is the major premise of the rule, and $\bot$ is its minor premise; the discharged assumption of the rule is A.

The rule closely corresponds to the truth-tracking definition of knowledge: Suppose that one can prove $\Box A$, i.e. she is allowed to believe A. But if it is possible to prove that A is false, then something among the overall assumptions must be reviewed.

As usual, we write $\Gamma \vdash_{\mathsf{IEL}} A$ when A is derivable in IEL assuming the set of hypotheses Γ, and we write $\mathsf{IEL} \vdash A$ when $\Gamma = \varnothing$.

It is easy to show that the axiomatic system $\mathbb{IEL}$ and the new natural deduction calculus IEL are logically equivalent.

LEMMA 3.2.2. $\Gamma \vdash_{\mathsf{IEL}} A$ *iff* $\Gamma \vdash_{\mathbb{IEL}} A$.

Proof. The proof strategy is the standard one, and the same as that for intuitionistic belief. By Lemma 2.2.2, it suffices to consider intuitionistic reflection only, in both paradigms.

The schema $\Box A \to \neg\neg A$ is clearly deducible in IEL by applying $\to \mathcal{E}$, then $\Box\mathcal{E}$, and $\to \mathcal{I}$ twice.

Conversely, consider the $\Box\mathcal{E}$ rule: By inductive hypothesis $\Gamma \vdash_{\mathbb{IEL}} \Box A$ and $A, \Delta \vdash_{\mathbb{IEL}} \bot$. Then by *MP* and intuitionistic reflection $\Gamma \vdash_{\mathbb{IEL}} \neg\neg A$; moreover, by the deduction theorem, $\Delta \vdash_{\mathbb{IEL}} \neg A$. An application of *MP* gives the result.

⊠

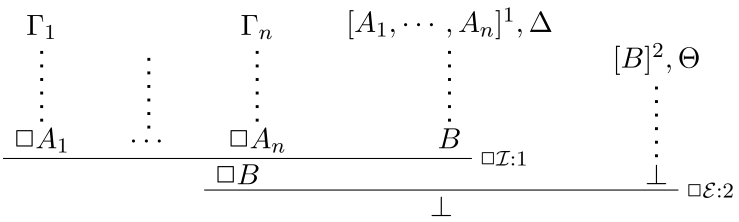
It is straightforward to extend the modal λ-calculus for IEL^{-} by decorating $\Box\mathcal{E}$ with proof names.

The λ-term corresponding to $\Box\mathcal{E}$ gives the eliminator for modal terms, as expected:

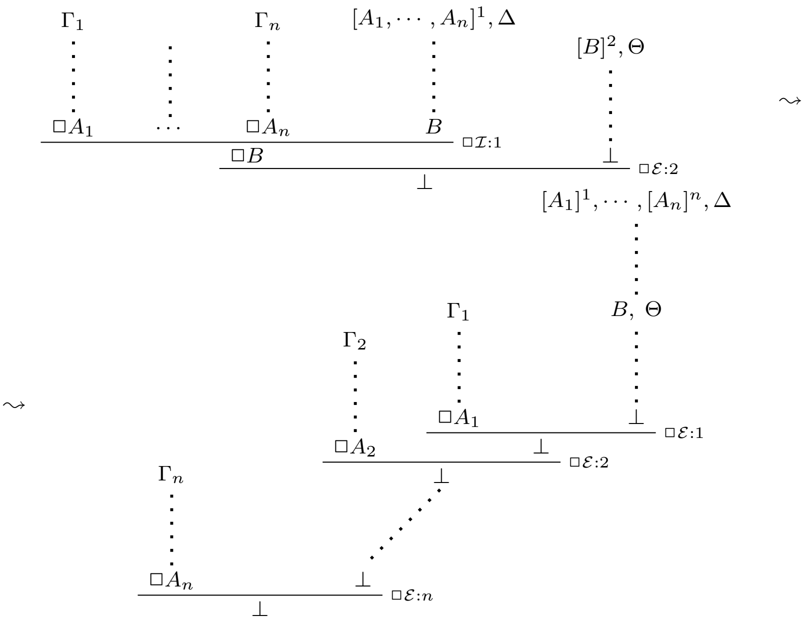
$$\dfrac{\Gamma \vdash f : \Box A \qquad x : A, \Delta \vdash g : \bot}{\Gamma, \Delta \vdash \ (\mathsf{unbox}\, f\, \mathsf{with}\, x.g) \ : \bot}$$

3.2.2. NORMALISATION

The presence of an introduction *and* an elimination rule for the modal operator requires to consider potential detours with the following general pattern



This situation is easily handled by the following rewriting, that we denote by $\delta_\Box$:



We now prove that IEL-deductions strongly normalise w.r.t. the system of rewritings $\Xi + \rho_\Box + \pi_\Box + \delta_\Box$.

All we need to do is to specialise the proof strategy for Theorem 2.2.4.

THEOREM 3.2.3. *Deductions in* IEL *strongly normalise w.r.t. the rewriting system* $\Xi + \rho_\Box + \pi_\Box + \delta_\Box$.

Proof. Tweak the translation $\langle - \rangle$ in the proof of Theorem 2.2.4 as follows:

$$\begin{array}{lcl}
\langle \bot \rangle & := & \bot \\
\langle \top \rangle & := & \top \\
\langle p \rangle & := & p \\
\langle A \to B \rangle & := & \langle A \rangle \to \langle B \rangle \\
\langle A \wedge B \rangle & := & \langle A \rangle \wedge \langle B \rangle \\
\langle A \vee B \rangle & := & \langle A \rangle \vee \langle B \rangle \\
\langle \Box A \rangle & := & \langle A \rangle \vee \bot
\end{array}$$

$$
\begin{aligned}
&\langle x\rangle := x\\
&\langle *\rangle := *\\
&\langle \bot_J(f)\rangle := \bot_J(\langle f\rangle)\\
&\langle \lambda x\, f\rangle := \lambda x.\langle f\rangle\\
&\langle fg\rangle := \langle f\rangle\langle g\rangle\\
&\langle (f,g)\rangle := (\langle f\rangle,\langle g\rangle)\\
&\langle \pi_i(f)\rangle := \pi_i(\langle f\rangle)\\
&\langle \mathsf{C}(f,x.f_1,y.f_2)\rangle := \mathsf{C}(\langle f\rangle,x.\langle f_1\rangle,y.\langle f_2\rangle)\\
&\langle \mathsf{in}_i(f)\rangle := \mathsf{in}_i(\langle f\rangle)\\
&\langle \mathsf{box}[x_1,\cdots,x_n].\,g\,\mathsf{with}\,f_1,\cdots,f_n\rangle := \mathsf{C}(\langle f_n\rangle,x_n.\cdots\mathsf{C}(\langle f_2\rangle,x_2.\mathsf{C}(\langle f_1\rangle,x_1.\mathsf{in}_1(\langle g\rangle),y_1.\mathsf{in}_2(y_1)),y_2.\mathsf{in}_2(y_2))\cdots,y_n.\mathsf{in}_2(y_n))\\
&\langle \mathsf{unbox}\,f\,\mathsf{with}\,x.g\rangle := \mathsf{C}(\langle f\rangle,x.\langle g\rangle,y.y).
\end{aligned}
$$

For the sake of readability, it is again convenient to rephrase the translation of $\Box\mathcal{E}$ in the natural deduction formalism:

$$
\dfrac{\begin{matrix}\Gamma\\ \vdots\, f\\ \Box A\end{matrix}\qquad\begin{matrix}[A],\Delta\\ \vdots\, g\\ \bot\end{matrix}}{\bot}\,\Box\mathcal{E}
\quad\overset{\langle\ \rangle}{\longmapsto}\quad
\dfrac{\begin{matrix}\langle\Gamma\rangle\\ \vdots\, \langle f\rangle\\ \langle A\rangle\vee\bot\end{matrix}\qquad\begin{matrix}[\langle A\rangle]^1,\langle\Delta\rangle\\ \vdots\, \langle g\rangle\\ \bot\end{matrix}\qquad [\bot]^1}{\bot}\,\vee\mathcal{E}{:}1
$$

This translation preserves $\delta_\Box$ too, by means of applications of rewritings in Ξ concerning $\vee$-detours and permutations.

Thus, since deductions in NJp are strongly normalising w.r.t. Ξ, we have the desired strong normalisation for IEL-deductions.

⊠

3.3. Analyticity and other properties

This section rephrases all the results for IEL$^-$ shown in Section 2.3. The proof of the subformula property of normal IEL-deductions follows the same pattern as that for IEL$^-$, but definitions and preliminary lemmas need to pay attention to the additional rewriting $\delta_\Box$ introduced for $\Box$-detours that cannot occur in IEL$^-$.

The careful structural analysis of normal IEL-deductions allows, as in Section 2.3, to establish by proof-theoretic means all the properties of intuitionistic knowledge that in [AP16] are proven my model-theoretic arguments. In particular, the fact that the logic of intuitionistic knowledge is a proper extension of the logic of intuitionistic belief is here proven by using Lemma 2.3.13 and the subformula principle, without recurring to semantic considerations.

3.3.1. Proof of the subformula property

Definition 3.3.1 (After [Pra71]). A path in an IEL-deduction f is a sequence of formula occurrences $A_1,\cdots,A_n$ where:

1. A_1 is either an open assumption of f, or an assumption discharged by $\to \mathcal{I}$;
2. if $1 \leq i < k$, then
    - if A_i is the major premise of $\vee\mathcal{E}$, or a minor premise of $\Box\mathcal{I}$, then A_{i+1} is the corresponding assumption discharged by that rule;
    - if A_i
        a. a premise of $\bot_J$;
        b. a premise of a propositional $\mathcal{I}$-rule;
        c. a major premise of $\Box\mathcal{I}$;
        d. a major premise of an $\mathcal{E}$-rule other than $\vee\mathcal{E}$;
        e. a minor premise of $\vee\mathcal{E}$;
        f. a minor premise of $\Box\mathcal{E}$;

      then A_{i+1} is the conclusion of that rule;
3. A_k is either the end formula of the deduction, or the minor premise of $\to \mathcal{E}$.

We will call any sequence of formula occurrences in a path, in which one is a minor premise of an $\vee\mathcal{E}$, or of an $\Box\mathcal{E}$ and the following one is its conclusion, a segment.

The definition just given generalises Def. 2.3.2 to consider also $\Box\mathcal{E}$ occurrences in a deduction.

We can now show that any path in a normal IEL-deduction has the expected structure.

LEMMA 3.3.2. *Any path in a normal* IEL*-deduction f is made of segments ordered as follows:*

- *first, there are segments where the last formula occurrence is a major premise of an $\mathcal{E}$-rule, or a minor premise of $\Box\mathcal{I}$;*
- *then, there are segments whose last formula occurrence is a minor premise of an $\Box\mathcal{E}$-rule;*
- *afterwards, there are segments whose last formula occurrence is the premise of a $\bot_J$;*
- *finally, there are segments where the last formula occurrence is a (major) premise of an $\mathcal{I}$-rule.*[2]

[2] This means that the structure of normal IEL is still similar to the one in Figure 1.5, even though $\Box\mathcal{E}$ rules follow all the propositional $\mathcal{E}$-rules.

We call the first segment ending in the premise of a $\bot_J$, or in the (major) premise of an $\mathcal{I}$-rule the minimum segment *of f, and the formula repeating in it is called the* minimum formula.

Proof. Proceed by contradiction, as in the proof of Lemma 2.3.3.

⊠

Now, it is easy to prove the analogous of Prop. 2.3.4.

PROPOSITION 3.3.3. *In a path $A_1, \cdots, A_n$ of a normal* IEL*-deduction, every formula A in it is either a subformula of A_1, or of A_n.*

Proof. As for the proof of Prop: 2.3.4, a straightforward analysis of the structure of normal IEL-deductions justifies the claim.

⊠

To prove that every formula in an IEL-deduction is a subformula of the starting assumption of the path it is on, or of its ending formula, we can use the following

PROPOSITION 3.3.4. *Every formula A in an* IEL*-deduction f lies in at least one path of f.*

Proof. Straightforward induction on the height $\mathfrak{h}(f)$ of f, as for Prop. 2.3.5.

⊠

We can maintain Def. 2.3.6 and proceed proving the analogous of Lemmas 2.3.7 and 2.3.8. Those complete the ingredients to prove the main result, namely

THEOREM 3.3.5 (**Subformula principle**). *Every formula B occurring in a normal* IEL*-deduction f of A from assumptions Γ is a subformula of A or of some formula in Γ.*

Proof. By induction on the order $\mathfrak{o}$ of a path an occurrence of B is on, following the lines of reasoning in the proof of Theorem 2.3.9.

⊠

3.3.2. FURTHER PROPERTIES OF INTUITIONISTIC KNOWLEDGE

For IEL, a more traditional definition of neutral proofs can be given: We say that a IEL-deduction is neutral if it consists of an assumption, or its last inference is an $\mathcal{E}$-rule.

PROPOSITION 3.3.6. *In any normal and neutral* IEL*-deduction of A from Γ, $\Gamma \neq \varnothing$.*

Proof. By induction on the height $\mathfrak{h}$ of the given deduction, as in the proof of Prop. 2.3.12.

If $\mathfrak{h} > 0$, we need to consider the further case:

e. if the last rule is $\Box\mathcal{E}$, then the inductive hypothesis can be applied to the subdeduction ending in the major premise of that rule.

☒

All the other properties for intuitionistic belief follow in the same manner:

LEMMA 3.3.7. *The following hold:*

(i) *In any normal* IEL*-deduction of A, the last rule applied is the introduction rule for the main connective of A.*

(ii) *The reflection rule is admissible in* IEL.

(iii) IEL *satisfies the disjunction property.*

(iv) IEL *is $\Box$-prime.*

(v) *If* $\mathsf{IEL} \vdash \Box(A \vee B)$*, then* $\mathsf{IEL} \vdash \Box A$ *or* $\mathsf{IEL} \vdash \Box B$.

(vi) IEL *is consistent.*

(vii) IEL *is decidable.*

Proof. Each claim is proven in the same way as the analogous for IEL^-, discussed in Section 2.3.2. ☒

By a pure structural analysis of IEL and IEL^- we can also show that the the logic of intuitionistic knowledge is a proper extension of the logic of intuitionistic belief, as shown in [AP16] by a semantic argument.

LEMMA 3.3.8. $\mathsf{IEL}^- \subsetneq \mathsf{IEL}$.

Proof. Consider the formula $\neg\Box\bot$.

That formula is clearly provable in IEL:

$$\dfrac{\dfrac{[\Box\bot]^2 \qquad [\bot]^1}{\bot}\,\Box\mathcal{E}{:}1}{\neg\Box\bot}\,{\rightarrow}\mathcal{I}{:}2 .$$

Now suppose that $\mathsf{IEL}^- \vdash \neg\Box\bot$ too. Then by the normalisation theorem (Theorem 2.2.4) there exists a normal IEL^--deduction of $\neg\Box\bot$. By Lemma 2.3.13, the last rule of that deduction must be an $\rightarrow\mathcal{I}$, so that $\bot$ is derived from the hypothesis $\Box\bot$ in a *normal* IEL^--deduction. By the structure of normal deduction (Prop. 2.3.4), and the subformula principle (Theorem 2.3.9), that deduction must consist of $\mathcal{I}$-rules, and all formulas occurring in it can be among $\{\Box\bot, \bot\}$ only. But these two requirements cannot be simultaneously satisfied, so that $\mathsf{IEL}^- \nvdash \neg\Box\bot$ after all.

☒

A similar structural and combinatorial reasoning establishes also the following result of [AP16]:

LEMMA 3.3.9. *Intuitionistic verifications do not have the disjunction property, i.e. for* $\mathsf{L} \in \{\mathsf{IEL}^-, \mathsf{IEL}\}$, $\mathsf{L} \nvdash \Box(A \vee B) \to \Box A \vee \Box B$.

RELATED WORK

In the previous pages it has been shown that all the structural and proof-theoretic properties for intuitionistic belief presented in [PB19, PB21a, PB21b] and discussed in Chapter 2 hold also for intuitionistic knowledge.

As previous recalled, proof theory for IEL in terms of standard Gentzen-style sequent calculi is proposed in [KY16, SS19]. Those works are also relevant for complexity related investigations into the (first-order) logic of intuitionistic knowledge.

Similar remarks could be made for [Fio21], where sequent calculi for intuitionistic knowledge and belief are by introducing semantic notions in the structure of sequents according to the paradigm of implicit internalisation for sequent calculi. The resulting system is very efficient from the computational view-point, and resembles nested sequent calculi for intuitionistic logics of [GS20].

In [Rog21], a proof-irrelevant algebraic-categorical semantics for first-order IEL is given. It might be interesting to consider in future how the results discussed in the present chapter relate to that interpretation, and see if the proof-relevant categorical models for intuitionistic belief recalled in Section 2.4 can be extended to cover knowledge too.[3]

On the more traditional proof-theoretic side, it might be relevant to define translations of the natural deduction systems discussed here with the Gentzen-style sequent calculi of [KY16] and [SS19], by recurring to the general framework established in [NvP08, Ch. 8]. Those intriguing aspects are left to future work.

[3] The interpretation of [Rog21] is based on cover systems, which were originally developed by Robert Goldblatt [Gol11] for a topos-theoretic analysis of Heyting algebras with a nucleus operator. For epistemic modalities, the original perspective is generalised to consider a more general class of cover systems, but the view-point is still proof-irrelevant. Actually, it is not trivial at all to find the right insight to connect those algebraic considerations with the proof-theoretic semantics given in Sect. 2.4, and a reasonable categorical interpretation of IEL-deduction is still under development by the author.

4

Natural deduction for intuitionistic strong Löb logic

In the previous chapters, we have interpreted the co-reflection schema

$$A \rightarrow \Box A$$

in epistemic terms, by relying on the provability account of intuitionistic truth embodied in BHK informal semantics.

But what happen if we seriously take $\Box A$ as a provability statement over base (arithmetical) theory? It is clear that the provability condition ($\tau 5$) share the same "abstract" structure of co-reflection. That arithmetical formula states Σ_1-completeness of an underlying theory T which is based on *classical logic*: If the Σ_1 sentence A^* is true, then it is also provable.

We know that any theorem of Gödel-Löb provability logic can be arithmetically interpreted by a realisation function $*$ w.r.t. any "reasonable" T[1] over a classical basis. What about intuitionistic theories of arithmetic?

At present time, there is no uniform solution to the task of finding an adequate intuitionistic logic for provability w.r.t. those arithmetical systems. This is partially due to the highly classical object-level reasoning involved in the proof of Solovay's theorem. For sure, the intuitionistic counterpart of GL *is not* the provability logic for Heyting arithmetic: For the letter, a proper extension of IGL is needed.[2] Nevertheless, there are some modal systems that are known to capture the structural properties of formal provability in very specific intuitionistic theories of arithmetic. We discuss one of them, namely intuitionistic strong Löb logic, denoted by ISL.

This logic was identified first by Dick de Jongh and Alber Visser as a modal version of algebraic presentations of arithmetic over intuitionistic bases [dJV95]. Those authors describe it as a 'Kindergarten Theory', since most of

[1] Refer to Section 1.2.2.

[2] A most promising candidate is presented in [Moj22], in which a provability logic for HA is axiomatised by adding to IGL very specific conservativity principles, rephrased in modal terms. It is worth noticing that the definition of that final modal system goes through intermediate axiomatisations involving restricted co-reflection schemas: The reader is referred to that very recent preprint for the details.

the proof-theoretic results about GL have strikingly simple versions. This is true even for the fixpoint theorem, that, for GL, we recalled in Theorem 1.2.6. Since that theorem can be interpreted as a very abstract version of self-referential phenomena, it is not surprising that the interest in ISL went beyond the arithmetical realm, towards the areas of computer science dealing with fixpoint operators and (co)recursive definitions.

In particular, the type-theoretic community has become in recent years very attracted by intuitionistic strong Löb logic – as well as by intuitionistic Gödel-Löb logic – so that the provability smack of those modalities has acquired very different interpretations: For instance, [Nak00] proposes IGL as a logic for recursion, while [BM13] reads ISL and IGL as systems for studying later operators and guarded (co)recursion, and identifies the categorical structure underlying the modality in terms of the topos of trees. Similar considerations are made in [ML17] for a fragment of ISL using only $\Box, \top, \wedge$ as logical connectives; in that paper, a correspondence between the strong Löb schema

$$(\Box A \to A) \to A$$

and contraction endofunctors in cartesian categories is proposed.

In this chapter, we shall restrict our attention to the design of a natural deduction system for ISL – denoted by ISL – having good structural properties. In particular, we cared to define the calculus as a natural extension of IEL^- by means of an elimination rule for the modal operator without injuring the introduction rule characterising our natural deduction system for intuitionistic belief.

Section 4.1 recalls the original axiomatisation of intuitionistic strong Löb logic, and its relational semantics. In Section 4.2 we introduce our natural deduction calculus ISL and the modal λ-calculus obtained by decorating ISL-deductions with proof-terms; moreover, we discuss the design style of our system, which has been led by the intent of defining a calculus enjoying good structural properties. In particular, we prove that ISL strongly normalises w.r.t. a system of rewritings involving $\Box\mathcal{I}/\Box\mathcal{E}$ detours. Finally, we discuss strong normalisation w.r.t. further simplifications of deductions and its relevance for assuring analyticity of our calculus.

Writing and Revision Notes. This chapter exclusively contains unpublished original material on the intuitionistic strong Löb logic. It can be thought as a collection of preliminary results that the author has autonomously obtained in his own investigations of proof theory for (intuitionistic) provability logic. From another perspective, the calculus ISL described here can been seen as an evidence for the flexibility of system obtained by adding the $\Box\mathcal{I}$ corresponding to the schema $A \to \Box A$: In a sense, the results for ISL presented here suggest that IEL^- can be really used as a kernel calculus for developing natural deduction systems for intuitionistic modal logic based on co-reflection.

4.1. Axiomatic calculus for intuitionistic strong Löb logic

A first axiomatisation of intuitionistic strong Löb logic dates back to [dJV95], where an algebraic semantics is also discussed. We start by recalling the axiomatic calculus $\mathbb{ISL}$ together with its relational models, defined in [Lit14].

4.1.1. System $\mathbb{ISL}$

Definition 4.1.1. $\mathbb{ISL}$ is the axiomatic calculus given by:

- Axiom schemas for intuitionistic propositional logic;
- Axiom schema $\mathsf{K} : \Box(A \to B) \to \Box A \to \Box B$;
- Axiom schema of co-reflection $A \to \Box A$;
- Axiom schema $\mathsf{GL} : \Box(\Box A \to A) \to \Box A$;
- Modus Pones $\dfrac{A \to B \qquad A}{B}\ \mathit{MP}$ as the only inference rule.

As usual, we write $\Gamma \vdash_{\mathbb{ISL}} A$ when A is derivable in $\mathbb{ISL}$ assuming the set of hypotheses Γ, and we write $\mathbb{ISL} \vdash A$ when $\Gamma = \varnothing$.

It is clear that ISL defines a normal intuitionistic modal logic.

The notion of arithmetical realisation of Section 1.2.2 is still valid for this system when considering intuitionistic arithmetical theories. As stated at the beginning of this chapter, ISL is indeed special among the provability logics over an intuitionistic basis, for identifying all the relevant properties of formal provability in an intuitionistic arithmetical system: Only few other modal logics for intuitionistic arithmetic are known, so that it provides an important insight into constructive provability by means of a simple propositional calculus.[3]

4.1.2. Kripke Semantics for $\mathbb{ISL}$

Even though $\mathbb{ISL}$ was originally defined on the basis of algebraic considerations about (subsystems of) Heyting arithmetic, it is possible to prove that intuitionistic strong Löb logic is sound and complete w.r.t. a relatively simple possible world semantics.

[3] Notice that a less interesting arithmetical completeness result relates ISL with the provability logic of Σ_1-sentences, since the co-reflection schema corresponds to Σ_1-completeness, i.e. the provability condition $(\tau 5)$ of Section 1.2.

Definition 4.1.2. A model for $\mathbb{ISL}$ is a quadruple $\langle W, \leq, v, R\rangle$ where

- $\langle W, \leq, v\rangle$ is a standard model for intuitionistic propositional logic;
- R is a binary relation on W such that:
  - if xRy, then $x \leq y$; and
  - if $x \leq y$ and yRz, then xRz; graphically we have

$$\begin{tikzcd}
 & y \arrow[r, dotted, "R"] & z \\
x \arrow[ur, "\leq"] \arrow[urr, dotted, "R"'] & &
\end{tikzcd}$$

  - R is transitive;
  - R is Noetherian;
- v extends to a forcing relation $\Vdash$ such that
  - $x \Vdash \Box A$ iff $y \Vdash A$ for all y such that xRy.

Validity of a formula in a model and in a class of frames are defined as usual, and denoted by the relation $\vDash$, as for intuitionistic epistemic logics of the previous chapters.

Theorem 4.1.3. *The following hold:*

(**Soundness**) *If* $\mathbb{ISL} \vdash A$*, then* $\mathbb{ISL} \vDash A$*.*

(**Completeness**) *If* $\mathbb{ISL} \vDash A$*, then* $\mathbb{ISL} \vdash A$*.*

Proof. Soundness is proven by induction on the derivation of A.

For a proof of completeness, refer to [AM18].

⊠

4.2. Natural deduction for intuitionistic strong Löb logic

As for IEL, we are going to present a modular extension of IEL^- that formalises in the natural deduction paradigm ISL by means of a rule for eliminating modal formulas.

For this calculus, a strong normalisation theorem is proven w.r.t. an extended system of proof-rewritings that also considers $\Box\mathcal{I}/\Box\mathcal{E}$ detours. A modal λ-calculus is naturally defined by adding a destructor for $\Box$-types, on the lines of what is proven in the previous chapters about epistemic logics.

4.2.1. System ISL

The natural deduction calculus for intuitionistic strong Löb logic we want to define is obtained by considering an elimination rule for modal *assumptions*. That move – i.e. considering an elimination operating of hypotheses instead of the root of the subderivation immediately above the rule itself – fits the original philosophical motivations of Gentzen's paradigm only partially. Nevertheless, it has some technical advantages that need to be considered.

Definition 4.2.1. Let ISL be the system extending the natural deduction calculus IEL^- of Def. 2.2.1 by the following elimination rule:

$$\dfrac{\begin{matrix}\Gamma, [\Box A]^\star \\ \vdots \\ A\end{matrix} \qquad \begin{matrix}[A]^\star, \Delta \\ \vdots \\ C\end{matrix}}{C}\ \Box\mathcal{E}{:}\star$$

where Γ and Δ are *multisets of formulas*, and $\Box\mathcal{E}$ allows both multiple and vacuous discharge.

The rule corresponds to the **strong Löb schema**:

$$(\Box A \to A) \to A,$$

which can be used for an alternative axiomatisation of ISL [Lit14]. However, for structural reasons that we explain below, we have designed it as a G3-rule for natural deduction, on the lines of the formulation of propositional rules in [NvP08, Ch. 1 and 8].

As usual, we write $\Gamma \vdash_{\mathsf{ISL}} A$ when A is derivable in ISL assuming the set of hypotheses Γ, and we write $\mathsf{ISL} \vdash A$ when $\Gamma = \varnothing$.

It is easy to show that the axiomatic system $\mathbb{ISL}$ and the new natural deduction calculus ISL are logically equivalent.

Lemma 4.2.2. $\Gamma \vdash_{\mathsf{ISL}} A$ *iff* $\Gamma \vdash_{\mathbb{ISL}} A$.

Proof. Both directions are proven by induction on the height of the respective formal proof. By Lemma 2.2.2, it suffices to consider the additional schema GL for the axiomatic calculus, and the $\Box\mathcal{E}$ for the natural deduction system.

The schema $\Box(\Box A \to A) \to \Box A$ is deduced in ISL as follows:

$$\dfrac{\dfrac{[\Box(\Box A \to A)]^3 \qquad \dfrac{\dfrac{[\Box A \to A]^2 \qquad [\Box A]^1}{A}\to\mathcal{E} \qquad [A]^1}{A}\ \Box\mathcal{E}{:}1}{\Box A}\ \Box\mathcal{I}{:}2}{\Box(\Box A \to A) \to \Box A}\ \to\mathcal{I}{:}3$$

Conversely, consider the $\Box\mathcal{E}$ rule: By inductive hypothesis $\Gamma, \Box A \vdash_{\mathbb{ISL}} A$ and $\Delta, A \vdash_{\mathbb{ISL}} C$. By the deduction theorem, $\Gamma \vdash_{\mathbb{ISL}} \Box A \to A$ and $\Delta \vdash_{\mathbb{ISL}} A \to C$. Then, by normality, $\Box\Gamma \vdash_{\mathbb{ISL}} \Box(\Box A \to A)$, and by co-reflection $\Gamma \vdash_{\mathbb{ISL}} \Box(\Box A \to A)$. An application of MP with GL gives $\Gamma \vdash_{\mathbb{ISL}} \Box A$. A further application of MP with $\Gamma \vdash_{\mathbb{ISL}} \Box A \to A$ and applying again MP to the result with $\Delta \vdash_{\mathbb{ISL}} A \to C$ gives the desired conclusion.

⊠

As for IEL we are able to extend the modal λ-calculus for IEL$^-$ by decorating $\Box\mathcal{E}$ with proof names.

The λ-term corresponding to $\Box\mathcal{E}$ gives the eliminator for modal *variables*:

$$\frac{x : \Box A, \Gamma \vdash f : A \qquad y : A, \Delta \vdash g : C}{\Gamma, \Delta \vdash \ (\textsf{löb}\, x.f \ \textsf{with}\, y.g) \ \ : C}$$

4.2.2. Normalisation

For the modal operator we now need to consider potential detours made of the modal introduction rule followed by the corresponding elimination rule. That is the point where the efficiency of the G3-style formulation of $\Box\mathcal{E}$ for natural deduction plays its key role.

Suppose, indeed, that we had defined the eliminator for $\Box$ in a most natural way as

$$\begin{array}{c} \Gamma, [\Box A]^\star \\ \vdots \\ \dfrac{A}{A}\,\Box\mathcal{E}'{:}\star \end{array}$$

Then, the $\Box\mathcal{I}/\Box\mathcal{E}'$ detour would be

$$\dfrac{\dfrac{\begin{array}{c}\Gamma_1\\ \vdots\, f_1 \\ \Box A_1\end{array} \quad \begin{array}{c}\\ \vdots \\ \cdots\end{array} \quad \begin{array}{c}\Gamma_n \\ \vdots\, f_n \\ \Box A_n\end{array} \qquad \begin{array}{c}[A_1,\cdots,A_n]^1, \Delta \\ \vdots\, g \\ B\end{array}}{\Box B}\,\Box\mathcal{I}{:}1}{\Box B}\,\Box\mathcal{E}{:}\star$$

where $\star$ labels occurrences of $\Box\Box B$ among the assumptions. If $\Box\Box B$ occurs only among the assumptions of the subdeduction g, no issue arises, since we can easily rewrite the whole deduction as follows:

$$\dfrac{\begin{matrix}\Gamma_1\\ \vdots\, f_1\\ \Box A_1\end{matrix}\quad \begin{matrix}\vdots\\ \cdots\end{matrix}\quad \begin{matrix}\Gamma_n\\ \vdots\, f_n\\ \Box A_n\end{matrix}\qquad \begin{matrix}\dfrac{[\Box B]^1}{\Box\Box B}\,\Box\mathcal{I}\quad [A_1,\cdots,A_n]^2,\Delta\\ \vdots\, g\\ \dfrac{B}{B}\,\Box\mathcal{E}{:}1\end{matrix}}{\Box B}\,\Box\mathcal{I}{:}1$$

However, a little thought shows that no simplification can be performed when $\Box\Box B$ does not lie in the subdeduction g.

On the contrary, by using the G3-style elimination rule for the $\Box$, we can consider the following rewritings, that we denote by $o^1_\Box$ and $o^2_\Box$, respectively:

$$\dfrac{\dfrac{\begin{matrix}\Gamma_1\\ \vdots\\ \Box A_1\end{matrix}\quad\begin{matrix}\vdots\\ \cdots\end{matrix}\quad\begin{matrix}\Gamma_n\\ \vdots\\ \Box A_n\end{matrix}\quad\begin{matrix}[A_1,\cdots,A_n]^1,\Delta,[\Box\Box B]^2\\ \vdots\\ B\end{matrix}}{\Box B}\,\Box\mathcal{I}{:}1\qquad\begin{matrix}[\Box B]^2,\Theta\\ \vdots\\ C\end{matrix}}{C}\,\Box\mathcal{E}{:}2 \qquad \overset{o^1_\Box}{\rightsquigarrow}$$

$$\overset{o^1_\Box}{\rightsquigarrow}\qquad \begin{matrix}\dfrac{\begin{matrix}\Gamma_1\\ \vdots\\ \Box A_1\end{matrix}\quad\begin{matrix}\vdots\\ \cdots\end{matrix}\quad\begin{matrix}\Gamma_n\\ \vdots\\ \Box A_n\end{matrix}\quad \dfrac{\begin{matrix}\dfrac{[\Box B]^1}{\Box\Box B}\,\Box\mathcal{I}\quad[A_1,\cdots,A_n]^2,\Delta\\ \vdots\\ B\end{matrix}\qquad [B]^1}{B}\,\Box\mathcal{E}{:}1}{\Box B}\,\Box\mathcal{I}{:}2\\ \vdots\\ C\end{matrix}$$

$$\dfrac{\dfrac{\begin{matrix}\Gamma_1\\ \vdots\\ \Box A_1\end{matrix}\quad\begin{matrix}[\Box\Box B]^2,\Gamma_i\\ \vdots\\ \Box A_i\end{matrix}\quad\begin{matrix}\Gamma_n\\ \vdots\\ \Box A_n\end{matrix}\quad\begin{matrix}[A_1,\cdots,A_n]^1,\Delta\\ \vdots\\ B\end{matrix}}{\Box B}\,\Box\mathcal{I}{:}1\qquad\begin{matrix}[\Box B]^2,\Theta\\ \vdots\\ C\end{matrix}}{\Box B}\,\Box\mathcal{E}{:}2\qquad\overset{o^2_\Box}{\rightsquigarrow}$$

$$\overset{o^2_\Box}{\rightsquigarrow}\qquad\dfrac{\dfrac{\begin{matrix}[\Box\Box B]^3,\Gamma_i\\ \vdots\\ \Box A_i\end{matrix}\quad \dfrac{\begin{matrix}\Gamma_1\\ \vdots\\ \Box A_1\end{matrix}\quad [\Box A_i]^2\quad\begin{matrix}\Gamma_n\\ \vdots\\ \Box A_n\end{matrix}\quad\begin{matrix}[A_1,\cdots,A_n]^1,\Delta\\ \vdots\\ B\end{matrix}}{\Box B}\,\Box\mathcal{I}{:}1}{\Box B}\,\Box\mathcal{E}{:}2\qquad\begin{matrix}[\Box B]^3\\ \vdots\\ C\end{matrix}}{C}\,\Box\mathcal{E}{:}3$$

We are now ready to prove that ISL-deductions strongly normalise w.r.t. the system of rewritings $\Xi + \rho_\Box + \pi_\Box + o^1_\Box + o^2_\Box$. As for IEL, we can rely on the general proof strategy for Theorem 2.2.4.

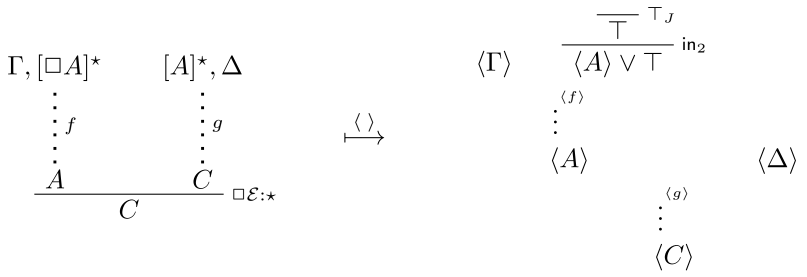
THEOREM 4.2.3. *Deductions in* ISL *strongly normalise w.r.t. the rewriting system* $\Xi + \rho_\Box + \pi_\Box + o^1_\Box + o^2_\Box$.

Proof. Tweak the translation $\langle - \rangle$ in the proof of Theorem 2.2.4 as follows:

$$\begin{array}{lcl}
\langle \bot \rangle & := & \bot \\
\langle \top \rangle & := & \top \\
\langle p \rangle & := & p \\
\langle A \to B \rangle & := & \langle A \rangle \to \langle B \rangle \\
\langle A \wedge B \rangle & := & \langle A \rangle \wedge \langle B \rangle \\
\langle A \vee B \rangle & := & \langle A \rangle \vee \langle B \rangle \\
\langle \Box A \rangle & := & \langle A \rangle \vee \top
\end{array}$$

$$\begin{array}{l}
\langle x \rangle := x \\
\langle * \rangle := * \\
\langle \bot_J(f) \rangle := \bot_J(\langle f \rangle) \\
\langle \lambda x\, f \rangle := \lambda x.\langle f \rangle \\
\langle f g \rangle := \langle f \rangle \langle g \rangle \\
\langle (f, g) \rangle := (\langle f \rangle, \langle g \rangle) \\
\langle \pi_i(f) \rangle := \pi_i(\langle f \rangle) \\
\langle \mathsf{C}(f, x.f_1, y.f_2) \rangle := \mathsf{C}(\langle f \rangle, x.\langle f_1 \rangle, y.\langle f_2 \rangle) \\
\langle \mathsf{in}_i(f) \rangle := \mathsf{in}_i(\langle f \rangle) \\
\langle \mathsf{box}[x_1, \cdots, x_n].\, g\, \mathsf{with}\, f_1, \cdots, f_n \rangle := \mathsf{C}(\langle f_n \rangle, x_n. \cdots \mathsf{C}(\langle f_2 \rangle, x_2.\mathsf{C}(\langle f_1 \rangle, x_1.\mathsf{in}_1(\langle g \rangle), y_1.\mathsf{in}_2(y_1)), y_2.\mathsf{in}_2(y_2)) \cdots, y_n.\mathsf{in}_2(y_n)) \\
\langle \mathsf{l\ddot{o}b}\, x.f\, \mathsf{with}\, y.g \rangle := \langle g[y := \langle f[x := \mathsf{in}_2(\lambda z.(z : \bot))] \rangle] \rangle.
\end{array}$$

As for the previous proofs, it is convenient to rephrase the translation of $\Box\mathcal{E}$ in the natural deduction formalism:



It is straightforward to see that all the rewritings in $\Xi + \rho_\Box + \pi_\Box + o^1_\Box + o^2_\Box$ are preserved by the translation and turned into rewritings in Ξ. Thus, since deductions in NJp are strongly normalising w.r.t. Ξ, we have the desired strong normalisation for ISL-deductions.

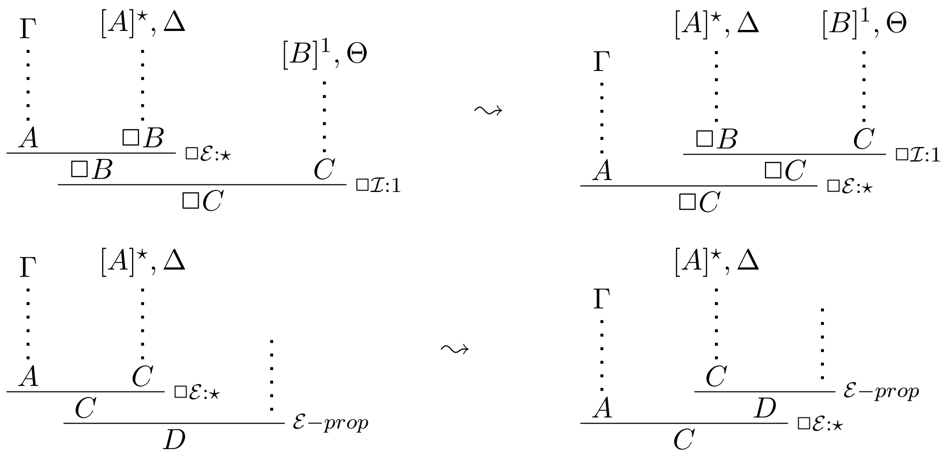
⊠

4.2.3. ON ANALYTICITY OF NORMAL ISL-DEDUCTIONS

As stated in Chapter 2, there might be several reasons to prove a normalisation theorem for a given natural deduction, but the main proof-theoretic aim of it is establishing the subformula property for the system under investigation.

We have also seen that that principle follows from the precise structural properties of normal deductions: In particular, for $\mathsf{NJ(p)}^=$, the subformula property follows from the canonical structure of normal deductions made of an analytic part followed by a synthetic part.[4] Similar considerations can be made for $\mathsf{IEL}^{(-)}$, since an analogous structural analysis can be performed on normal deductions for those systems.

However, for establishing the subformula property of $\mathsf{IEL}^{(-)}$ we needed to consider further conversions characterising the interaction of rules for the $\Box$ operator and for the disjunction. Similar considerations could be made for ISL: In the present setting, it is pretty natural to consider the following additional proof-rewritings $\kappa_{\Box}$[5]



Notice that the translation used in proving Theorem 4.2.3 preserves the rewritings of $\kappa_{\Box}$ too, so that ISL strongly normalises w.r.t. the extended system $\Xi + \rho_{\Box} + \pi_{\Box} + o^1_{\Box} + o^2_{\Box} + \kappa_{\Box}$.

Moreover, these simplifications should suffice for establishing analyticity of normal ISL-deductions. Actually, we are pretty confident that the subformula principle does hold for our calculus when we consider the whole systems of rewritings under discussion. Nevertheless, we are not currently in the position of proving a sufficiently detailed proof of that claim: The main issue here is identifying the precise canonical structure of normal deductions, since $\Box\mathcal{E}$ seems to be able to "ramble on" deductions and, at the same time, it has a peculiar non-local nature. Therefore, in full intellectual earnestness, we prefer to formulate the subformula property for ISL as

[4] See Section 1.4.2.

[5] It order to enhance readability, we only consider special cases of $\Box\mathcal{I}$, and use the notation $\star$ for discharging in $\Box\mathcal{E}$ without making explicit the position of the boxed assumption discharged by the rule: No confusion should arise, since the general setting is dealt with the same way as the special one we are considering here.

CONJECTURE 4.2.4 (**Subformula principle**). *Every formula B occurring in a normal – w.r.t. $\Xi + \rho_\Box + \pi_\Box + o^1_\Box + o^2_\Box + \kappa_\Box$ – ISL-deduction f of A from assumptions Γ is a subformula of A or of some formula in Γ.*

RELATED WORK

In this chapter we have seen some preliminary results on structural proof theory for intuitionistic strong Löb logic ISL, investigated first in [dJV95]. A natural deduction system ISL has been presented, and a strong normalisation theorem has been proven for it w.r.t. a specific set of proof-rewritings which are designed for establishing analyticity – hence consistency and decidability – of our calculus.

A different proof-theoretic analysis of ISL is presented in [vdGI20]. They develop $\mathsf{G3}$ and $\mathsf{G4}$ sequent calculi for intuitionistic strong Löb logic on the basis of their previous results for intuitionistic GL [vdGI21]. Their systems enjoy syntactic cut-elimination and Craig interpolation, and are designed so that a relational countermodel can be extracted from a failed proof search of a given sequent. Cut-elimination for those calculi does not suffice for establishing their analyticity, but it might be relevant to consider potential translations between their formalism and the one presented here.

Another structural investigation has been carried out by Albert Visser and Tadeusz Litak with a focus on interpolation – Craig's as well as uniform versions – results for ISL, based on their previous work on Lewisian implication [LV18, LV19].

Outside structural proof theory, it might be interesting to confront our system ISL with the categorical account of (guarded co-)recursion operators given in [ML17]. As stated at the end of the previous section, a structural analysis of ISL is made a bit more convoluted because of the peculiar "non-local and erratic" nature of $\Box\mathcal{E}$, which does not allow a direct derivation of the subformula principle from normalisation of deductions. At the risk of oversimplifying, one might say that the correspondence between normalisation and local completeness of natural deduction rules is not clear for ISL. From that perspective, a careful comparison with Conway's conditions [ÉB93] for recursion categories – which do not determine a purely equational system of rewritings – might be worth investigating, for the relevance that the Löb axiom has acquired in the study of fixpoint operators and in an abstract account of recursion.[6]

[6] Refer to e.g. [Nak00] and, more recently, [BM13].

5

Modular sequent calculi for interpretability logics

Interpretations arise in several areas of (meta)mathematics and there exist many variations on them.[1] For instance, it is possible to interpret propositional intuitionistic logic into classical Gödel-Löb logic, by establishing an equivalence between $\mathbb{GL}$ and the axiomatic calculus $\mathbb{IPC}$; from that, one could also interpret $\mathbb{IPC}$ into an arithmetical theory T that is adequate to $\mathbb{GL}$.

An even simpler example is given by Gödel's numbering, which interprets (a model for) meta-mathematical reasoning into the standard model for arithmetic by defining an injective function that maps finite strings of arithmetical symbols into $\mathbb{N}$, and a further function mapping each meta-predicate into its arithmetical counterpart.

In the present chapter, we will assume a very general version of it: An **interpretation** of a theory T into a theory T' is just a structure preserving translation t such that if $T \vdash A$ then $T' \vdash t(A)$. More precisely, we will consider arithmetical theories satisfying the Hilbert-Bernays-Löb provability conditions of Section 1.2.[2]

Modal logics for interpretability arise as an extension of the language of provability logic by means of a binary modal operator $\rhd$ capturing the relation of (relative) interpretability between two arithmetical theories: The propositional formula $A \rhd B$ is then intended as the modal counterpart of the arithmetical formula $Int_{\mathsf{T}}(\ulcorner A^* \urcorner, \ulcorner B^* \urcorner)$ – where $Int_{\mathsf{T}}(x, y)$ is the formal predicate for relative interpretability over T – expressing the fact that the arithmetical theory T extended by A^* interprets the arithmetical theory T extended by B^*.

The origins of interpretability logics date back to the work by Albert Visser [Vis90], who axiomatised the basic modal framework by extending the language of $\mathbb{GL}$ with the following schemas:[3]

- Axiom $\mathsf{IL1}$: $\Box(A \to B) \to A \rhd B$

[1] A first methodological treatment of the notion was presented first in [TMR53], where the basic properties of this concept are introduced.

[2] Here we recover the notation introduced in that section, to which the reader is referred.

[3] We assume that $\rhd$ binds stronger than $\to$, but weaker than the other connectives.

- Axiom $\mathsf{IL2}: A \rhd B \to B \rhd C \to A \rhd C$
- Axiom $\mathsf{IL3}: A \rhd C \to B \rhd C \to A \vee B \rhd C$
- Axiom $\mathsf{IL4}: A \rhd B \to \Diamond A \to \Diamond B$
- Axiom $\mathsf{IL5}: \Diamond A \rhd A$

These constitute the minimal logic for interpretability $\mathbb{IL}$, on top of which several systems can be constructed.

Completeness results w.r.t. a Kripke-style relational semantics were presented first by Dick de Jongh and Frank Veltman in [dJV90] for $\mathbb{IL}$ and some extensions by using a canonical model construction. More complex proofs were developed by the same authors in 1999, and further techniques were introduced to achieve subsequent completeness results since the beginning of 2000s by Joost Joosten and collaborators [GJ08]. Recent results have been obtained also for subsystems of $\mathbb{IL}$ by Taishi Kurahashi and Yuya Okawa [KO21].

On the arithmetical side, by tweaking Solovay's proof strategy for $\mathbb{GL}$, it is possible to prove also the *arithmetical* completeness of some extensions of $\mathbb{IL}$. The interpretability logic for T is, as expected,

$$\mathsf{IL}(\mathsf{T}) := \{A \,|\, \forall *, \mathsf{T} \vdash A^*\}.$$

In [Sha97], it is proved that this notion is Σ^0_3-complete.

However, if we assume that T is Σ_1-sound and proves full induction, by adding the schema

$$\mathsf{M} := A \rhd B \to A \wedge \Box C \rhd B \wedge \Box C,$$

we have that

$$\mathsf{IL}(\mathsf{T}) = \mathbb{ILM} := \mathbb{IL} + \mathsf{M}.$$

Similarly if we add to $\mathbb{IL}$ the schema

$$\mathsf{P} := A \rhd B \to \Box(A \rhd B),$$

then

$$\mathbb{IL} + \mathsf{P} =: \mathbb{ILP} = \mathsf{IL}(\mathsf{T})$$

for any T that is Σ_1-sound, finitely axiomatised, and such that it proves the totality of $\mathtt{supexp}$.

The most intriguing aspect of interpretability logics is exactly such a sensitivity to the base arithmetical theory. The main open question in the field is indeed establishing the interpretability logic of all reasonable arithmetical theories, i.e.,

$$\mathsf{IL}(\mathit{All}) := \{A \,|\, \forall \mathsf{T} \supseteq \mathsf{I}\Delta_0 + \mathsf{exp}, \forall *, \mathsf{T} \vdash A^*\}.$$

What we know is that $\mathbb{IL} \subseteq \mathsf{IL}(All) \subseteq \mathbb{ILM} \cap \mathbb{ILP}$, but a modal characterisation of $\mathsf{IL}(All)$ is still unknown.

There are many further open questions in the field. It is worth noticing that very few is known about proof theory for interpretability logics: Katsumi Sasaki gives in [Sas02] a Gentzen-style sequent calculus for $\mathbb{IL}$ only; while Tuomas Hakonemi and Joost Joosten present in [HJ16] a labelled tableaux system for some extensions of $\mathbb{IL}$ based on standard Veltman semantics.[4]

In the present chapter, this gap in the proof-theoretic analysis of interpretability logics is partially filled by introducing a family $\mathsf{G3IL}^\star$ of labelled sequent calculi which covers in a natural way a wide range of modal systems for interpretability. Their design is based on the methodology of [Neg05, Neg17] but, instead of working with formal relational semantics or formal neighbourhood semantics, these original calculi internalise the hybrid models by Rineke Verbrugge, usually called generalised Veltman structures [Ver92].

The main contribution of the work presented here consists then of the design of modular sequent systems satisfying the main structural desiderata, namely: admissibility of contraction and weakening, invertibility of logical rules, and a cut-elimination algorithm.

The chapter is organised as follows: In Section 5.1 we recall the axiomatic calculi for the main interpretability logics under investigations; in Section 5.2 the basic definitions and results in the model theory for interpretability logics are recalled, and the generalised Veltman semantics (GVS) is defined according to the most recent literature on the topic. Our family of sequent calculi $\mathsf{G3IL}^\star$ is then presented in Section 5.3; Section 5.4 is committed to the structural analysis of those calculi, and includes a constructive proof of admissibility of the cut rule for all the extensions. Finally, in Section 5.5 we prove soundness and completeness of our systems w.r.t. the standard axiomatic and semantic presentations.

Writing and Revision Notes. This chapter exclusively contains unpublished original material on modal logics for interpretability. To the best of our knowledge, $\mathsf{G3IL}^\star$ is the most comprehensive class of analytic calculi for interpretability logics that is currently available from the literature. The main result presented here is the cut-elimination theorem for these calculi, based on the complexity measure introduced in [Neg05], that we modified according to our more general setting.

5.1. Axiomatic calculi

Let's start by extending the language $\mathcal{L}_\Box$ with a binary modal operator $\rhd$; the resulting formal language will be denoted by $\mathcal{L}_{\Box,\rhd}$.

[4] See Section 5.2.1 below for a definition of the latter.

The basic axiomatic calculus for interpretability logics is given by the following

DEFINITION 5.1.1. Let $\mathbb{IL}$ denote the axiomatic system determined by

- the axiom schemas of $\mathbb{CPC}$;
- schema K : $\Box(A \to B) \to \Box A \to \Box B$;
- schema GL : $\Box(\Box A \to A) \to \Box A$;
- interpretability schemas
  - $\mathsf{IL1}$: $\Box(A \to B) \to A \rhd B$;
  - $\mathsf{IL2}$: $A \rhd B \to B \rhd C \to A \rhd C$
  - $\mathsf{IL3}$: $A \rhd C \to B \rhd C \to A \vee B \rhd C$
  - $\mathsf{IL4}$: $A \rhd B \to \Diamond A \to \Diamond B$
  - $\mathsf{IL5}$: $\Diamond A \rhd A$
- MP rule $\dfrac{A \to B \qquad A}{B}$
- Necessitation rule $\dfrac{A}{\Box A}$

As usual, we write $\Gamma \vdash_{\mathbb{IL}} A$ when A is derivable in $\mathbb{IL}$ assuming the set of hypotheses Γ, and we write $\mathbb{IL} \vdash A$ when $\Gamma = \varnothing$.

For this calculus, some interesting lemmas are provable:

LEMMA 5.1.2. *The following hold*

(i) $\mathbb{IL} \vdash \Box\neg A \to (A \rhd B)$;

(ii) $\mathbb{IL} \vdash A \vee \Diamond A \rhd A$;

(iii) $\mathbb{IL} \vdash A \rhd A \wedge \Box\neg A$;

(iv) $A \rhd B$*,* $A \wedge \Box\neg A \rhd B$ *and* $A \rhd B \wedge \Box\neg B$ *are interderivable over* $\mathbb{IL}$*;*

(v) $A \rhd \bot$ *and* $\Box\neg A$ *are interderivable over* $\mathbb{IL}$*;*

(vi) $\mathbb{IL} \vdash \Diamond A \rhd \neg(A \rhd \Diamond A)$.

Proof. Refer to e.g. [Vis90].

⊠

Item (v) of the previous lemma shows that we could dismiss the $\Box$ modality, since we could define $\Box A$ as $\neg A \rhd \bot$. This would simplify our base language. However, we will see in Sections 5.2.1 and 5.2.2 that it is possible to define $\Box$ via $\rhd$ as well as $\rhd$ via $\Box$ by semantics considerations. Those will lead to the design of our family of sequent calculi in Section 5.3.[5]

[5] Similar considerations underlie the definition of the tableaux systems in [HJ16], which deal, in any case, with a more limited number of interpretability logics.

5.1.1. Axiomatic extensions

On top of $\mathbb{IL}$, it is possible to add further modal principles that have specific relevance for arithmetical realisations. Here we will consider the following calculi:

Definition 5.1.3. Let us define as proper extensions of $\mathbb{IL}$

- $\mathbb{ILM} := \mathbb{IL} + \mathsf{M}$, where
$$\mathsf{M} := A \rhd B \to A \wedge \Box C \rhd B \wedge \Box C$$
is called the Montagna schema;
- $\mathbb{ILP} := \mathbb{IL} + \mathsf{P}$, where
$$\mathsf{P} := A \rhd B \to \Box(A \rhd B)$$
is called the persistence schema;
- $\mathbb{ILW} := \mathbb{IL} + \mathsf{W}$, where
$$\mathsf{W} := A \rhd B \to A \rhd B \wedge \Box\neg A$$
is called the de Jongh-Visser schema;
- $\mathbb{ILKM1} := \mathbb{IL} + \mathsf{KM1}$, where
$$\mathsf{KM1} := A \rhd \Diamond\top \to \top \rhd \neg A;$$
- $\mathbb{ILM}_0 := \mathbb{IL} + \mathsf{M}_0$, where
$$\mathsf{M}_0 := A \rhd B \to \Diamond A \wedge \Box C \rhd B \wedge \Box C;$$

5.2. Semantics for interpretability logics

As stated in the introduction of the chapter, we know – after [Sha88, Ber90], and [Vis90, Zam92], respectively – that $\mathbb{ILM}$ is the interpretability logic for arithmetical theories proving full induction, and $\mathbb{ILP}$ captures in an adequate the properties of formal interpretability over finitely axiomatised theories proving the totality of the superexponential function.[6]

Their proofs are based on Solovay's strategy for arithmetical completeness of Gödel-Löb logic, therefore they use in an essential way the characterisation of those axiomatic systems in terms of relational models. We now turn our presentation to these semantic aspects of interpretabilty logics.

[6] This means, for instance, that Montagna's principle holds for Peano arithmetic, but does not hold for Gödel-Bernays set theory; on the contrary, the persistence principle holds for Gödel-Bernays set theory, but does not hold for Peano arithmetic. Nevertheless, they share the same provability logic, namely GL.

5.2.1. Veltman semantics

A standard relational semantics for interpretability logics is obtained from possible world semantics by "decorating" frames with additional indexed accessibility relations.

Definition 5.2.1. A Veltman frame $\mathcal{F}$ consists of:

- a non-empty set W of possible worlds;
- a binary relation R on W which is transitive and Noetherian;
- a collection $\{S_x \mid x \in W\}$ of binary relations which are reflexive, transitive and such that
  - each S_x is defined on $R[x]$, where $R[x] := \{y \in W \mid xRy\}$; and
  - if $xRyRz$, then yS_xz.

A **Veltman model** $\mathcal{M}$ is obtained by adding an evaluation function v to a given Veltman frame, as usual. A **forcing relation** is then obtained by a standard definition for propositional connectives and $\Box$-modality, while for $\rhd$-modality we stipulate that

$x \Vdash A \rhd B$ iff for all y, if xRy and $y \Vdash A$, then there exists a z such that yS_xz and $z \Vdash B$.

According to the notation of Section 1.1, we write $\vDash_{\mathcal{M}} A$ when A is forced by any world in $\mathcal{M}$; similarly, we write $\vDash_{\mathcal{F}} A$ when $\vDash_{\mathcal{M}} A$ for any model $\mathcal{M}$ based on $\mathcal{F}$.

For extensions, some frame conditions are needed. A **frame condition** for a modal schema A is a (first or higher order) formula (A) in the language $\{R, \{S_x\}\}$ such that the structure $\mathcal{F}$ satisfies the property (A) iff, for any Veltman model $\mathcal{M}$ based on $\mathcal{F}$, $\vDash_{\mathcal{M}} A$.

In [Vis88, Vis90] many principles for interpretability were proposed first, together with their semantic characterisations. Nowadays, we know that

$$(\mathsf{W}) = (\mathsf{KW1}) = (\mathsf{F}) = \text{“}R \circ S_x \text{ is Noetherian” for any } x \in W,$$

where

$$\begin{array}{lll} \mathsf{KW1} & := & A \rhd \Diamond\top \to \top \rhd \neg A \\ \mathsf{F} & := & A \rhd \Diamond A \to \Box\neg A. \end{array}$$

Moreover we have that

$$(\mathsf{M}) = (\mathsf{KM1}) = (\mathsf{KM2}) = \text{ if } yS_xzRu, \text{ then } yRu,$$

where

$$\mathsf{KM2} \quad := \quad A \rhd B \to (\Box(B \to \Diamond C) \to \Box(A \to \Diamond C)),$$

and we know that $\mathsf{KM1}$ and $\mathsf{KM2}$ are interderivable over $\mathbb{IL}$.

By using Veltman semantics, it is also possible to show that

$$\mathbb{IL}\{\mathsf{F}, \mathsf{KW1}\} \nvdash \mathsf{KW1}^\circ,$$

where

$$\mathsf{KW1}^\circ := A \wedge B \rhd \Diamond A \rightarrow A \rhd (A \wedge \neg B).$$

This means that $\mathbb{ILF}, \mathbb{ILKW}1, \mathbb{ILKW}1^\circ$ are *incomplete* w.r.t. the standard relational semantics.

5.2.2. Generalised Veltman semantics

Generalised Veltman semantics (GVS) comes to rescue the situation. It has been developed by Rineke Verbrugge in [Ver92] by considering interpretations which are reminiscent of neighbourhood semantics for non-classical logics.

To be more precise, each S_x is now a relation between worlds and *sets of worlds*, satisfying specific properties which are identified by the schemas for $\rhd$.

Definition 5.2.2. A generalised Veltman frame $\mathcal{F}$ consists of

- a finite set $W \neq \varnothing$;
- a binary relation $R \subseteq W \times W$ which is irreflexive and transitive;
- a W-indexed set of relations $S_x \subseteq R[x] \times (\wp(R[x]) \smallsetminus \{\varnothing\})$;

satisfying the following conditions:

- Quasi-reflexivity: if xRy then $yS_x\{y\}$;
- Definiteness: if $xRyRz$ then $yS_x\{z\}$;
- Monotonicity: if yS_xa and $a \subseteq b \subseteq R[x]$ then yS_xb;
- Quasi-transitivity: if yS_xa and vS_xb_v for all $v \in a$, then $yS_x(\bigcup_{v\in a} b_v)$.

A **generalised Veltman model** is obtained by considering an usual evaluation function, which can be extended to a **forcing relation** defined as for standard semantics, with only one difference:

$x \Vdash A \rhd B$ iff for all y if xRy and $y \Vdash A$, then there exists an a such that yS_xa and $a \Vdash^\forall B$,

where $a \Vdash^\forall B$ abbreviates the expression "for any $z \in a, z \Vdash B$".

As for relational semantics, extensions for $\mathbb{IL}$ need ***generalised* frame conditions**: We denote by $(A)_{gen}$ the frame condition w.r.t. generalised Veltman semantics corresponding to the modal schema A.

We know that the following hold[7]

$$\begin{array}{lcl}
(\mathsf{M})_{gen} & = & yS_xa \Rightarrow \exists b \subseteq a, yS_xb \,\&\, R[b] \subseteq R[y] \\
(\mathsf{KM1})_{gen} & = & yS_xa \Rightarrow \exists i \in a, \forall z(iRz \Rightarrow yRz) \\
(\mathsf{P})_{gen} & = & xRx'RyS_xa \Rightarrow \exists b \subseteq a, yS_{x'}b \\
(\mathsf{M}_0)_{gen} & = & wRuRxS_wa \Rightarrow \exists b \subseteq a, uS_wb \,\&\, R[b] \subseteq R[u] \\
(\mathsf{P}_0)_{gen} & = & wRxRuS_wa \,\&\forall v \in a(R[v] \cap b \neq \varnothing) \Rightarrow \exists c \subseteq b, uS_xc \\
(\mathsf{W})_{gen} & = & yS_xa \Rightarrow \exists b \subseteq a, yS_xb \,\&\, R[b] \cap S_x^{-1}b = \varnothing \\
(\mathsf{R})_{gen} & = & wRxRuS_wa \Rightarrow \forall c \in \mathcal{C}(x,u), \exists b \subseteq a, xS_Wb \,\&\, R[b] \subseteq c \\
(\mathsf{W}^*)_{gen} = (\mathsf{M}_0)_{gen} + (\mathsf{W})_{gen}, & &
\end{array}$$

where

- $i \in R[b]$ iff there is an $x \in b$ such that xRi;
- $i \in S_x^{-1}b$ iff iS_xb;
- $\mathcal{C}(x,u) := \{c \subseteq R[x] \mid \forall d, uS_xd \Rightarrow d \cap c \neq \varnothing\}$;
- $\mathsf{P}_0 := A \rhd \Diamond B \to \Box(A \rhd B)$;
- $\mathsf{R} := A \rhd B \to \neg(A \rhd \neg C) \rhd B \wedge \Box C)$;
- $\mathsf{W}^* := A \rhd B \to B \wedge \Box C \rhd B \wedge \Box C \wedge \Box\neg A$.

For the basic system $\mathbb{IL}$ completeness results are known w.r.t. both standard Veltman semantics and GVS. Moreover, the techniques used to prove the completeness theorem for that system w.r.t. GVS can be easily extended to consider analogous results for $\mathbb{ILM}$, $\mathbb{ILP}$, and $\mathbb{ILW}$ [dJV90, dJV99], [Ver92].

However, for the other extensions the proof of modal completeness can be quite convoluted and very sensitive to the logic under consideration, so that proving that an extension of a given system is complete may need a very different proof strategy w.r.t. the one used for the completeness of the original subsystem. Some promising advances have been made recently by Joost Joosten and collaborators in a series of works aiming at developing a modular and uniform methodology to deal with GVS completeness of interpretability logic: The most recent literature on the topic includes [BGJ04, GBJM20, JRMV20, RMJ20].[8]

In any case, investigations on GVS suffice to establish that for the interpretability logics we have mentioned the interdependencies rendered in Figure 5.1 hold.

[7] We need to use symbolic connectives for the meta-level in order to enhance readability.

[8] It is worth noticing that there exist flourishing research in finding even more general interpretability principles, whose semantics is still under investigation: See e.g. [MJV20a, MJV20b].

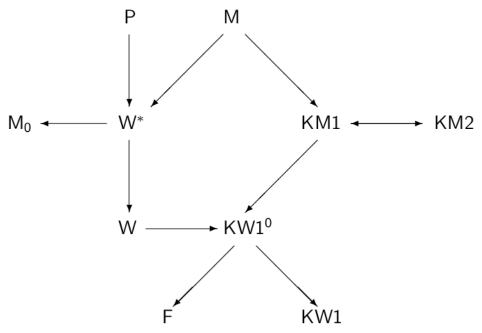


Figure 5.1: Interdependencies among interpretability logics, after [Vuk99]. An arrow from S to S' is interpreted as $\mathbb{IL} + \mathsf{S} \subsetneq \mathbb{IL} + \mathsf{S}'$.

Finally, we can summarise the current model-theoretic knowledge on interpretability logic by the following glossary:[9]

Modal principle	Veltman completeness	GVS completeness	Veltman FMP	GVS FMP
M	✓	✓	✓	✓
P	✓	✓	✓	✓
W	✓	✓	✓	✓
W*	✓	✓	?	✓
M$_0$	✓	✓	?	✓
P$_0$	×	✓	?	✓
R	?	✓	?	✓
F	×	?	✓	✓

5.3. Design of the labelled sequent calculi

We have seen that the language $\mathcal{L}_{\Box,\rhd}$ is somehow redundant: After Lemma 5.1.2(v) we know that $\Box A$ is equivalent to $\neg A \rhd \bot$. This invites to minimise the formal language for interpretability by considering the $\rhd$-modality as primitive. The resulting language will be denoted by $\mathcal{L}_{\rhd}$.

We need now to rephrase the inductive definition of well-formed formulas of $\mathbb{IL}$ – and its extensions – as follows:

Definition 5.3.1. The set of well-formed formulas of $\mathbb{IL}$ and its extensions w.r.t. $\mathcal{L}_{\rhd}$ is given by the following grammar

$$\mathtt{Form}_{\rhd} := \ p \mid A \wedge B \mid A \vee B \mid A \to B \mid \bot \mid A \rhd B \ ,$$

where $p \in \mathtt{Atm}$ and $A, B \in \mathtt{Form}_{\rhd}$.

[9] This information is collected from the results in [Ver92], [dJV90, dJV99], [GJ08], [MPV17, MV20], [PV16]. Recall from Section 1.1 that FMP abbreviates “finite model property”.

As the reader might expect, we define $\neg A := A \to \bot$, $A \leftrightarrow B := (A \to B) \wedge (B \to A)$, $\Box A := \neg A \rhd \bot$, and $\Diamond A := \neg\Box\neg A$.

On the axiomatic side, such a minimalist choice about the basic language is reflected by a minimalist axiomatisation of the basic system for interpretability:

DEFINITION 5.3.2. Let $\mathbb{IL}_{\rhd}$ denote the axiomatic calculus defined by

- Axiom schemas of $\mathbb{CPC}$;
- schema IL2 : $A \rhd B \to B \rhd C \to A \rhd C$;
- schema IL3 : $A \rhd C \to B \rhd C \to A \vee B \rhd C$;
- schema IL-Löb: $A \rhd (A \wedge (A \rhd \bot))$;
- MP Rule $\dfrac{A \to B \qquad A}{B}$;
- $\rhd$Rule $\dfrac{A \to B}{A \rhd B}$.

The extensions of $\mathbb{IL}_{\rhd}$ are obtained by adding the axiom schemas that we discussed in Sections 5.1.1 and 5.2; they will be denoted analogously to the systems based on multimodal $\mathbb{IL}$.

The calculi we are now going to present are obtained by labelling the formulas in $\texttt{Form}_{\rhd}$. The classes of models we are using for the definition of these calculi are based on generalised Veltman semantics: This is in line with the procedure we recalled in Section 1.3.2, which has been initiated by [Neg05] for relational semantics, and generalised further to neighbourhood semantics [Neg17, GNO21].[10]

5.3.1. CORE SYSTEM

By G3IL we denote the labelled sequent calculus for $\mathbb{IL}_{\rhd}$. The design of G3IL is based on an explicit internalisation of GVS by means of labels which allow the formalisation of the semantic information into a proof system.

According to Def. 5.2.2, the forcing condition for the $\rhd$-modality is

$$(\sharp) \quad x \Vdash A \rhd B \quad \text{iff} \quad \begin{array}{l}\text{for all } y\text{, if } xRy \text{ and } y \Vdash A,\\ \text{then there exists an } a \text{ such that } yS_x a \text{ and } a \Vdash^{\forall} B,\end{array}$$

where $a \Vdash^{\forall} B$ abbreviates the expression "for any $z \in a$, $z \Vdash B$".

As it comes, that forcing condition cannot be directly translated into a single sequent calculus rule because of the presence on alternating nested quantifiers on the right hand side of ($\sharp$).

[10] Such a move is clearly possible in virtue of the adequacy results for the systems under investigations w.r.t. GVS that we briefly recalled in the glossary at the end of Section 5.2.2.

Therefore it is necessary to introduce an intermediate indexed modality, which obeys the following forcing condition

$$(\flat)\quad y \Vdash \langle]_x B \quad \text{iff} \quad \text{there exists an } a \text{ such that } yS_x a \text{ and } a \Vdash^{\forall} B,$$

where $a \Vdash^{\forall} B$ abbreviates the expression "for any $z \in a$, $z \Vdash B$", as in ($\natural$). The forcing condition for $\rhd$ can then be rephrased as

$$(\sharp\flat)\quad x \Vdash A \rhd B \quad \text{iff} \quad \begin{array}{l}\text{for all } y, \text{ if } xRy \text{ and } y \Vdash A,\\ \text{then } y \Vdash \langle]_x B.\end{array}$$

Unfortunately, this is not enough yet. As a matter of fact, models for $\mathbb{IL}$ are based on frames which are irreflexive, transitive and *finite* – or, equivalently, transitive and Noetherian: Neither finiteness nor Noetherianess can be expressed by a semantic rule in line with the methodology of explicit internalisation, for first-order formulations need to be considered only.[11]

However, the treatment of Gödel-Löb logic described in [Neg14b] gives the right hint for proceeding with the design. Notice first that condition ($\sharp\flat$) establishes the logical equivalence

$$x \Vdash A \rhd B \qquad \text{iff} \qquad x \Vdash \Box(A \to \langle]_x B).$$

Moreover, we know that in any model based on *ITF*[12]

$$x \Vdash \Box A \qquad \text{iff} \qquad \text{for any } y, \text{ if } xRy \text{ and } y \Vdash \Box A, \text{ then } y \Vdash A.$$

This suggests to index with worlds the $\rhd$-modality, and to take the following forcing condition for it, whenever we are reasoning in models based on GVS:

$$(\natural)\quad x \Vdash A \rhd_i B \quad \text{iff} \quad \begin{array}{l}\text{for all } y, \text{ if } xRy \text{ and } y \Vdash A \rhd_i B,\\ \text{then, if } y \Vdash A,\ y \Vdash \langle]_i B.\end{array}$$

Let $\texttt{Form}^{i}_{\rhd}$ denote the formulas allowing indexed $\rhd$-modalities as well as the intermediate indexed modalities $\langle]_x$.

We are now ready to define the labelled sequent calculus $\mathsf{G3IL}$.

DEFINITION 5.3.3. Let $i, j, k, \cdots, x, y, z, \cdots$ be variables for worlds in a generalised Veltman model, and $a, b, c, \cdots$ variables for sets of worlds. **Relational atoms** are formulas of the following form and meaning:

- $y \in R[x]$, "world y is accessible to world x";
- $yS_x a$, "set a is S_x-accessible to world y";

[11] It is worth noticing, however, that in some cases this limitation can be partially circumvented by adopting the *systems of rules* described in [Neg14a].

[12] In Section 1.2 we defined *ITF* as the class of relational frames that are irreflexive transitive and finite.

- $y \in a$, “world y is a member of set a”;
- $a \subseteq b$, “set a is included into set b”.

Labelled formulas are defined as follows, for $A \in \mathtt{Form}^{i}_{\rhd}$:

- Relational atoms are labelled formulas;
- $x : A$, “world x forces formula A”;
- $a \Vdash^{\forall} A$, “formula A is forced by any world belonging to set a”.

We will use $\{x\}$ to denote the singleton set consisting of exactly the world x.

DEFINITION 5.3.4. Sequents of G3IL are expressions $\Gamma \Rightarrow \Delta$, where Γ and Δ are multisets of relational atoms and labelled formulas, and relational atoms may occur *only* in Γ.

The rules defining G3IL are given in Figure 5.2.

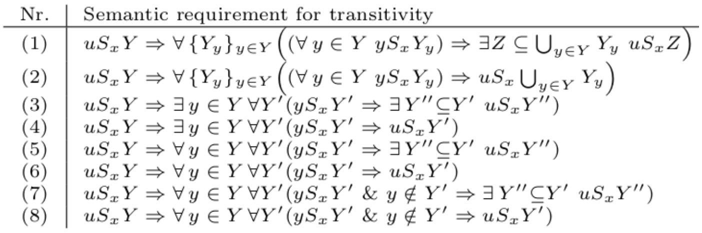
Some of those rules might deserve a little explanation:

- Side condition $(x!)$ in $\mathcal{R} \Vdash^{\forall}$ expresses the fact that x is a ‘fresh varibale’, i.e. it does not occur in the conclusion of the rule; similarly for $(y!)$ in $\mathcal{R}\rhd_i$; the meaning of $(a!)$ in $\mathcal{L}\langle]$ is analogous.
- The rules for $\langle]_x$ are defined according to the forcing condition ($\flat$).
- The rules for $\rhd_i$ are defined according to the forcing condition ($\natural$).
- The rules for GVS are defined as geometric rules [NvP11]; in particular we opted for an alternative definition of quasi-transitivity of the indexed S-relation. The condition imposed by Def. 5.2.2 is

  $$\text{if } yS_x a \text{ and } vS_x b_v \text{ for all } v \in a, \text{ then } yS_x(\textstyle\bigcup_{v \in a} b_v).$$

  As it is clear, this condition cannot be directly translated as a geometric rule.

  Nevertheless, in the literature it is possible to find several different conditions for quasi-transitivity:[13]



Nr.	Semantic requirement for transitivity
(1)	$uS_xY \Rightarrow \forall \{Y_y\}_{y \in Y} \Big((\forall\, y \in Y\ yS_xY_y) \Rightarrow \exists Z \subseteq \bigcup_{y \in Y} Y_y\ uS_xZ\Big)$
(2)	$uS_xY \Rightarrow \forall \{Y_y\}_{y \in Y} \Big((\forall\, y \in Y\ yS_xY_y) \Rightarrow uS_x \bigcup_{y \in Y} Y_y\Big)$
(3)	$uS_xY \Rightarrow \exists\, y \in Y\, \forall Y'(yS_xY' \Rightarrow \exists\, Y'' \subseteq Y'\ uS_xY'')$
(4)	$uS_xY \Rightarrow \exists\, y \in Y\, \forall Y'(yS_xY' \Rightarrow uS_xY')$
(5)	$uS_xY \Rightarrow \forall\, y \in Y\, \forall Y'(yS_xY' \Rightarrow \exists\, Y'' \subseteq Y'\ uS_xY'')$
(6)	$uS_xY \Rightarrow \forall\, y \in Y\, \forall Y'(yS_xY' \Rightarrow uS_xY')$
(7)	$uS_xY \Rightarrow \forall\, y \in Y\, \forall Y'(yS_xY'\ \&\ y \notin Y' \Rightarrow \exists\, Y'' \subseteq Y'\ uS_xY'')$
(8)	$uS_xY \Rightarrow \forall\, y \in Y\, \forall Y'(yS_xY'\ \&\ y \notin Y' \Rightarrow uS_xY')$

[13] The table is taken from [JRMV20].

Initial sequents

$x : p, \Gamma \Rightarrow \Delta, x : p$

$x : A \rhd_i B, \Gamma \Rightarrow \Delta, x : A \rhd_i B$

Classical propositional rules: the usual ones, refer to Figure 1.2

Local forcing rules

$$\frac{x : A, x \in A, a \Vdash^{\forall} A, \Gamma \Rightarrow \Delta}{x \in A, a \Vdash^{\forall} A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\Vdash^{\forall}$$

$$\frac{x \in a, \Gamma \Rightarrow \Delta, x : A}{\Gamma \Rightarrow \Delta, a \Vdash^{\forall} A}\ \mathcal{R}\Vdash^{\forall}_{(x!)}$$

Intermediate modality rules

$$\frac{yS_xa, a \Vdash^{\forall} A, \Gamma \Rightarrow \Delta}{y : \langle]_x A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\langle]_{(a!)}$$

$$\frac{yS_xa, \Gamma \Rightarrow \Delta, y : \langle]_x A, a \Vdash^{\forall} A}{yS_xa, \Gamma \Rightarrow \Delta, y : \langle]_x A}\ \mathcal{R}\langle]$$

Interpretability modality rules

$$\frac{y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A \qquad y : \langle]_i B, y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta \qquad y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A \rhd_i B}{y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\rhd_i$$

$$\frac{y \in R[x], y : A, \Gamma, y : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B}{\Gamma \Rightarrow \Delta, x : A \rhd_i B}\ \mathcal{R}\rhd_{i\,(y!)}$$

Rules for GVS

$$\frac{a \subseteq a, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ \mathit{Refl}\subseteq \qquad \frac{a \subseteq c, a \subseteq b, b \subseteq c, \Gamma \Rightarrow \Delta}{a \subseteq b, b \subseteq c, \Gamma \Rightarrow \Delta}\ \mathit{Trans}\subseteq$$

$$\frac{x \in b, x \in a, a \subseteq b, \Gamma \Rightarrow \Delta}{x \in a, a \subseteq b, \Gamma \Rightarrow \Delta}\ \mathcal{L}\subseteq$$

$$\frac{x \in \{x\}, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ \mathit{Sing}$$

$$\frac{Atm(y), Atm(x), y \in \{x\}, \Gamma \Rightarrow \Delta}{Atm(x), y \in \{x\}, \Gamma \Rightarrow \Delta}\ \mathit{Repl}_1 \qquad \frac{Atm(x), Atm(y), y \in \{x\}, \Gamma \Rightarrow \Delta}{Atm(y), y \in \{x\}, \Gamma \Rightarrow \Delta}\ \mathit{Repl}_2$$

where $Atm(x)$ has one of the following forms: $x : p, x \in a, x \in \{z\}$,$x \in R[z]$, $z \in R[x], xS_za, zS_xa$.

$$\frac{}{x \in R[x], \Gamma \Rightarrow \Delta}\ \mathit{Irrefl} \qquad \frac{z \in R[x], y \in R[x], z \in R[y], \Gamma \Rightarrow \Delta}{y \in R[x], z \in R[y], \Gamma \Rightarrow \Delta}\ \mathit{Trans}$$

$$\frac{z \in a, yS_xa, \Gamma \Rightarrow \Delta}{yS_xa, \Gamma \Rightarrow \Delta}\ \mathit{NE}_{(z!)} \qquad \frac{y \in R[x], a \subseteq R[x], yS_xa, \Gamma \Rightarrow \Delta}{yS_xa, \Gamma \Rightarrow \Delta}\ \mathit{DefS1}$$

$$\frac{yS_x\{z\}, y \in R[x], z \in R[y], \Gamma \Rightarrow \Delta}{y \in R[x], z \in R[y], \Gamma \Rightarrow \Delta}\ \mathit{DefS2} \qquad \frac{yS_xb, yS_xa, a \subseteq b, b \subseteq R[x], \Gamma \Rightarrow \Delta}{yS_xa, a \subseteq b, b \subseteq R[x], \Gamma \Rightarrow \Delta}\ \mathit{Mono}$$

$$\frac{yS_x\{y\}, y \in R[x], \Gamma \Rightarrow \Delta}{y \in R[x], \Gamma \Rightarrow \Delta}\ \mathit{Qrefl} \qquad \frac{yS_xb, yS_xa, z \in a, zS_xb, \Gamma \Rightarrow \Delta}{yS_xa, z \in a, zS_xb, \Gamma \Rightarrow \Delta}\ \mathit{Qtrans6}$$

Figure 5.2: Rules for G3IL

Condition 2 is in a sense the more natural one: The monotone closure of any S_x satisfying any of conditions 1-8 satisfies condition 2, and allows us to define an equivalent model However, for conditions 2-6 it is always possible to obtain a related standard model from a GVS-model, as it is shown in [MV20]. The rule *Qtrans6* is just the natural formalisation of condition 6 in the previous table, which is the simplest one.

- Rule *Sing* assures that the singleton contains at least one element; rules *Repl*$_1$ and *Repl*$_2$ that it contains at most one element, for indiscernibility of identicals.

- From Figure 5.2 the rules obtained by the closure condition [Neg05] of the system are omitted. For some rules dealing with GVS – for instance, *Trans* – there might be a duplication of a relational atom in the conclusion. Structural considerations – namely, the desideratum of admissibility of contraction – require then that a new rule is added to the system, in which the duplicated formulas are contracted into one.[14] However, the rules added to a system in order to satisfy such a closure condition play no role in the proof of semantic completeness of the calculi we are considering here and they can be shown to me admissible: This justifies our omission in favour of a better readability of the figure.

Here we see that the generalised Veltman semantics for $\mathbb{IL}$ can be considered as a geometric theory, and thus it can be formalised by a sequent calculus based on purely geometric rules.

5.3.2. Extensions

Calculi for extensions of $\mathbb{IL}$ are denoted by adding to $\mathsf{G3IL}$ the name of a modal schema as apex: Thus, for instance, $\mathsf{G3IL}^{\mathsf{M}}$ is the labelled calculus for $\mathbb{ILM}$, and $\mathsf{G3IL}^{\mathsf{P}}$ is the labelled calculus for $\mathbb{ILP}$. We denote by $\mathsf{G3IL}^{\star}$ the whole family of calculi for the interpretability logics considered in Def. 5.1.3.

Figure 5.3 shows the rules for the extensions of $\mathbb{IL}$ we are are interested in.

As you see, these rules are obtained by considering the generalised frame conditions characterising each extension of $\mathbb{IL}$. Moreover, we need to consider an extension of the language of labelled formulas:

Definition 5.3.5. Extend the Def. 5.3.3 by considering among **relational atoms**

[14] It is relevant to notice here that for each semantic characterisation, there is only a bounded number of additional rules generated by the closure condition. Moreover, that number is generally made smaller, since many cases of contracted rules are shown to be admissible in the base calculus. Refer to [NvP11, Ch. 6 and 11] for an exhaustive description of the procedure and its relevance for labelled calculi for modal logics. Similar considerations hold for the rules in Figure 5.3.

Additional rules for GVS

$$\frac{x \in a, y \in R[x], y \in R[a], \Gamma \Rightarrow \Delta}{y \in R[a], \Gamma \Rightarrow \Delta}\ \mathit{Rset1}_{(x!)} \qquad \frac{y \in R[a], x \in a, y \in R[x], \Gamma \Rightarrow \Delta}{x \in a, y \in R[x], \Gamma \Rightarrow \Delta}\ \mathit{Rset2}$$

$$\frac{yS_x a, y \in S_x^{-1}a, \Gamma \Rightarrow \Delta}{y \in S_x^{-1}a, \Gamma \Rightarrow \Delta}\ \mathit{Sset1} \qquad \frac{y \in S_x^{-1}a, yS_x a, \Gamma \Rightarrow \Delta}{yS_x a, \Gamma \Rightarrow \Delta}\ \mathit{Sset2}$$

$$\frac{c \subseteq a, c \subseteq b, c \subseteq a \cap b, \Gamma \Rightarrow \Delta}{c \subseteq a \cap b, \Gamma \Rightarrow \Delta}\ \cap_1 \qquad \frac{c \subseteq a \cap b, c \subseteq a, c \subseteq b, \Gamma \Rightarrow \Delta}{c \subseteq a, c \subseteq b, \Gamma \Rightarrow \Delta}\ \cap_2$$

$$\frac{}{x \in \varnothing, \Gamma \Rightarrow \Delta}\ \mathcal{L}\varnothing$$

Rules for interpretability principles

$$\frac{b \subseteq a, yS_x b, R[b] \subseteq R[y], yS_x a, \Gamma \Rightarrow \Delta}{yS_x a, \Gamma \Rightarrow \Delta}\ M_{(b!)} \qquad \frac{z \in a, R_z \subseteq R[y], yS_x a, \Gamma \Rightarrow \Delta}{yS_x a, \Gamma \Rightarrow \Delta}\ KM1_{(z!)}$$

$$\frac{b \subseteq a, zS_y b, y \in R[x], z \in R[y], zS_x a, \Gamma \Rightarrow \Delta}{y \in R[x], z \in R[y], zS_x a, \Gamma \Rightarrow \Delta}\ P_{(b!)}$$

$$\frac{b \subseteq a, yS_x b, R[b] \cap S_x^{-1}a \subseteq \varnothing, yS_x a, \Gamma \Rightarrow \Delta}{yS_x a, \Gamma \Rightarrow \Delta}\ W_{(b!)}$$

$$\frac{b \subseteq a, yS_x b, R[b] \subseteq R[y], y \in R[x], z \in R[y], zS_x a, \Gamma \Rightarrow \Delta}{y \in R[x], z \in R[y], zS_x a, \Gamma \Rightarrow \Delta}\ M_{0\,(b!)}$$

Figure 5.3: Rules for G3IL*

- $a \cap b$, “set a intersects set b”.
- $y \in R[a]$, “world y is accessible to a world x belonging to a”;
- $y \in S_x^{-1}a$, “set a is S_x-accessible to a world y”;
- $x \in \varnothing$, “world x is a member of the set $\varnothing$” – we take the latter as the only **constant for sets** of our language.

Labelled formulas are defined as as in Def. 5.3.3 w.r.t. this extended set of relation atoms.

5.4. Structural properties

We now want to study the structural properties of the whole family of calculi G3IL*. In order to proceed, we need first some preliminary definitions.

By the **height** of a derivation, we mean the number of nodes occurring in the longest derivation branch, minus one. In particular, the height of a derivation consisting only of an initial sequent is 0. We write $\overset{n}{\vdash} \Gamma \Rightarrow \Delta$ whenever there is a derivation of the sequent $\Gamma \Rightarrow \Delta$ in G3IL* with height bounded by n.

Next, we need the notion of **weight** of labelled formulas.

DEFINITION 5.4.1. The weight of relational atoms is 0. As for the other labelled formulas, let us say that the label of $x : A$ is x; and the label of $a \Vdash^{\forall} A$ is a. The label of a formula φ is denoted by $l(\varphi)$, and $p(\varphi)$ denotes the pure part of the formula, i.e. the part of the formula without the label and without the forcing condition.

The weight $\mathfrak{w}(\varphi)$ of a labelled formula φ is given by the ordered pair $\langle \mathfrak{w}(p(\varphi)), \mathfrak{w}(l(\varphi)) \rangle$ where

- For all world labels x and all set labels a, $\mathfrak{w}(x) = 0$ and $\mathfrak{w}(a) = 1 + n(\cap)$, where $n(\cap)$ denotes the number of formal intersections in a;
- $\mathfrak{w}(p) = \mathfrak{w}(\bot) = 1$;
- $\mathfrak{w}(A \circ B) = \mathfrak{w}(A) + \mathfrak{w}(B) + 1$, for $\circ$ conjunction, disjunction or implication;
- $\mathfrak{w}(\langle]_i A) = \mathfrak{w}(A) + 1$
- $\mathfrak{w}(A \rhd_i B) = \mathfrak{w}(A) + \mathfrak{w}(B) + 2$.

For substitution of labels, we can rely on the definitions given in [NvP11, Neg17]. We borrow notation from those works, and write e.g. $a \Vdash^{\forall} A[b/a]$ to mean the result of simultaneously substituting b for a, this way obtaining $b \Vdash^{\forall} A$; similarly for world label substitution.

It is now routine to show that G3IL* enjoys height preserving substitution for world and set labels:

PROPOSITION 5.4.2. *The following hold:*

(i) If $\overset{n}{\vdash} \Gamma \Rightarrow \Delta$, *then* $\overset{n}{\vdash} \Gamma[y/x] \Rightarrow \Delta[y/x]$;

(ii) If $\overset{n}{\vdash} \Gamma \Rightarrow \Delta$, *then* $\overset{n}{\vdash} \Gamma[b/a] \Rightarrow \Delta[b/a]$.

Proof. Straightforward induction on n. If $n = 0$, then $\Gamma \Rightarrow \Delta$ is an initial sequent, or a conclusion of $\mathcal{L}\bot$, $\mathcal{L}\varnothing$, or *Irrefl*. The same is true for $\Gamma[y/x] \Rightarrow \Delta[y/x]$ and for $\Gamma[b/a] \Rightarrow \Delta[b/a]$.[15] If $n > 0$, we consider the last rule applied. If the latter has no variable conditions, then we apply the inductive hypothesis to the premise(s), followed by that very rule. Otherwise, the rule needs some care in case the substituted variable coincides with the fresh variable of the premise: In that case, we need to apply twice the inductive hypothesis to the premise, first to replace the fresh variable with another fresh variable – different from the one we wish to substitute – and secondly to perform the desired substitution.

⊠

[15] It is important to notice that $\varnothing$ is a *constant* of our language, and therefore it cannot be subject to substitution.

5.4.1. General initial sequents, weakening, contraction, invertibility

For the remaining sections, let φ denote either a relational atom or a (proper) labelled formula.

We start with a rather simple result.

Lemma 5.4.3. *The following sequents are derivable in* G3IL**:*

1. $a \Vdash^\forall A, \Gamma \Rightarrow \Delta, a \Vdash^\forall A$*;*

2. $x : A, \Gamma \Rightarrow \Delta, x : A$*.*

Proof. The two cases are proven by mutual induction on the weight of the labelled formulas. The general strategy is to apply the left and right rule to treat the two formula occurrences, until two formula occurrences of smaller weight are reached.

By means of example, we prove case 2, subcase $x : \langle]_i A$:

$$\cfrac{\cfrac{\begin{array}{c}\vdots\,\text{IH}\\ xS_ia, a \Vdash^\forall A, \Gamma \Rightarrow \Delta, x : \langle]_i A, a \Vdash^\forall A\end{array}}{xS_ia, a \Vdash^\forall A, \Gamma \Rightarrow \Delta, x : \langle]_i A}\,\mathcal{R}\langle]}{x : \langle]_i A, \Gamma \Rightarrow \Delta, x : \langle]_i A}\,\mathcal{L}\langle]$$

where we can apply the inductive hypothesis to the top sequent since $\mathfrak{w}(a \Vdash^\forall A) < \mathfrak{w}(x : \langle]_i A)$.

Notice that the subcase $x : A \rhd_i B$ is easily managed since sequents $x : A \rhd_i B, \Gamma \Rightarrow \Delta, x : A \rhd_i B$ are initial, and hence derivable by design.

$\boxtimes$

We want now to establish admissibility of weakening in G3IL*.

Lemma 5.4.4. *The rules of weakening*

$$\frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}\,\mathcal{L}Wk \qquad\qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}\,\mathcal{R}Wk$$

are height-preserving admissible in G3IL**.*

Proof. We need to show that if $\overset{n}{\vdash} \Gamma \Rightarrow \Delta$, then $\overset{n}{\vdash} \varphi, \Gamma \Rightarrow \Delta$ and $\overset{n}{\vdash} \Gamma \Rightarrow \Delta, \varphi$. The proof consists of a straightforward induction on n, following the lines of the analogous results in [NvP11, Neg17], which the reader is referred to for the details.

$\boxtimes$

Next we can prove that all the rules of G3IL* are invertible.[16]

[16] Recall from Section 1.3.1 that a rule is invertible when if its conclusion is derivable, so are its premise(s). This is a key feature of G3-style sequent calculi, whose main consequence is the dismiss of backtracking in a root-first proof search.

LEMMA 5.4.5. *All the rules of* G3IL* *are invertible.*

Proof. We proceed by induction on the height of the derivation, distinguishing cases on the basis of the rule under consideration. Notice first that the rules for extensions are clearly (height-preserving) invertible by (height-preserving) admissibility of weakening; the same remark applies to the rules for GVS, as well as to $\mathcal{L}\rhd$, $\mathcal{L}\Vdash^{\forall}$, and $\mathcal{R}\langle]$. Propositional cases are dealt with as in [Neg05]. We can thus limit our to proof to the invertibility of $\mathcal{R}\rhd$. Assume then that $\overset{n}{\vdash} \Gamma \Rightarrow \Delta, x : A \rhd_i B$. If $n = 0$ and $x : A \rhd_i B$ is not principal, then also $\overset{0}{\vdash} y \in R[x], y : A, y : A \rhd_i B, \Gamma \Rightarrow \Delta, y : \langle]_i B$. If it is principal, then $\Gamma = \Gamma', x : A \rhd_i B$ and we need to prove that

$$y \in R[x], y : A, y : A \rhd_i B, \Gamma', x : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B$$

is derivable. But this is provable by application of $\mathcal{L}\rhd$ to the initial sequent $y \in R[x], y : A, y : A \rhd_i B, \Gamma', x : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B, y : A \rhd_i B$ with the derivable sequents

$$y \in R[x], y : A, y : A \rhd_i B, \Gamma', x : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B, y : A$$

and

$$y : \langle]_i B, y \in R[x], y : A, y : A \rhd_i B, \Gamma', x : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B.$$

If $n > 0$ and $x : A \rhd_i B$ is principal in the last rule, then we have the desired result. Otherwise, it suffices to apply the inductive hypothesis to the premise(s).

$\boxtimes$

Notice that Lemma 5.4.5 cannot be strengthened into an height-preserving invertibility of the rules just because of the case we discussed in its proof: This is analogous to what happens for G3GL in [Neg05], which we used as a model for the design of G3IL*.

We now want to prove admissibility of contraction. Before we proceed with the proof, it is appropriate to introduce a notion that will be also used in the proof of cut-elimination.

DEFINITION 5.4.6 (After [NvP11]). The range $\mathfrak{r}(x)$ of a world label x in a derivation $\mathcal{D}$ in G3IL* is the set of world labels y such that either $y \in R[x]$ or for some $n \geq 1$ and for some $x_1, \cdots, x_n$ the relational atoms $x_1 \in R[x]$, $x_2 \in R[x_1]$,$\cdots$, $y \in R[x_n]$ appear in the antecedent of sequents of $\mathcal{D}$. The range $\mathfrak{r}(a)$ of a set label a in $\mathcal{D}$ is defined as $\mathfrak{r}(a) := max\{\mathfrak{r}(x) \,|\, x \in a\} \cup \{*\}$ ordered w.r.t. set inclusion. We set $\mathfrak{r}(\{x\}) = \mathfrak{r}(x)$ and $\mathfrak{r}(\varnothing) = \varnothing$.

Finally, we say that a rule is range-preserving admissible if the elimination of the rule does not increase the ranges of labels in the derivation.

THEOREM 5.4.7. *The rules of contraction*

$$\frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}\ \mathcal{L}Ctr \qquad\qquad \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}\ \mathcal{R}Ctr$$

are range-preserving admissible in G3IL*.

Proof. By simultaneous induction for left and right contraction, with primary induction on the weight of the contracted formula, and secondary induction on the height of the derivation. The only case requiring some care is when the contracted formula is $x : A \rhd_i B$ in the consequent, of weight n. If the contracted formula is not principal for the last rule in the derivation – i.e. the latter is not $\mathcal{R}\rhd$ on $x : A \rhd_i B$ – then we apply the secondary inductive hypothesis to the premise(s), followed by the rule. Otherwise, we invert the premise of $\mathcal{R}\rhd$ to obtain

$$y \in R[x], y : A, y : A \rhd_i B, y \in R[x], y : A, y : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B, y : \langle]_i B.$$

We can now apply the inductive hypotheses to obtain the derivation of

$$y \in R[x], y : A, y : A \rhd_i B \Rightarrow \Delta, y : \langle]_i B$$

from which the conclusion of $\mathcal{R}Ctr$ follows by applying $\mathcal{R}\rhd$. Range is preserved since in inverting $\mathcal{R}\rhd$ we use a label that is already present in the derivation tree.

For the other cases, refer to e.g. [NvP11, Neg17].

⊠

5.4.2. CUT-ELIMINATION THEOREM

We have finally collected all the material required to prove the main result of the present chapter, namely cut-elimination for G3IL*.

THEOREM 5.4.8 (Cut admissibility). *The rule of cut*

$$\frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\ Cut$$

is admissible in G3IL*.

Proof. The proof proceeds by primary induction on the weight of the cut formula, secondary induction on the range of the label of the cut formula, and tertiary induction on the sum of heights of the premises of cut.

Notice first that if y is in the range of x then $\mathfrak{r}(y) < \mathfrak{r}(x)$, because of the proof system design.

Following [NvP08, NvP11], we start by distinguishing cases according to the rules applied to derive the premises of cut:

1. At least one of the premises of *Cut* is an initial sequent;
2. The cut formula is not principal in the derivation of a least one premise;
3. The cut formula is the principal formula of both derivations of the premises.

Case 1. Assume the leftmost premise of *Cut* is an initial sequent. Then the cut formula is $x : p$ or $x : A \rhd_i B$, and the conclusion of the cut can be obtained from the rightmost premise by weakening. If the sequent is initial in virtue of some labelled formula ψ occurring in both Γ and Δ, then the conclusion of *Cut* is an initial sequent too. A similar argument works if we assume that the rightmost premise of *Cut* is an initial sequent. If $x : \bot$ is the cut formula φ and the leftmost premise of *Cut* is not initial, it has been derived by some rule R. If R is $\mathcal{L}\bot$, then $x : \bot$ occurs in the conclusion of the cut, and therefore we can obtain that sequent from $\mathcal{L}\bot$. Similarly if R is $\mathcal{L}\varnothing$ or *Irrefl*. Otherwise, if R is different from $\mathcal{L}\bot$, $\mathcal{L}\varnothing$ and *Irrefl*, we can permute the cut up on the left premise and eliminate it by inductive hypotheses.

Case 2. Assume the cut formula is not principal in the last rule leading to the leftmost premise of *Cut*. The general situation is the following:

$$\cfrac{\cfrac{\begin{matrix}\vdots\,\mathcal{D}_1\\ \Gamma^* \Rightarrow \Delta^*, \varphi\end{matrix}}{\Gamma \Rightarrow \Delta, \varphi}\,R \qquad \begin{matrix}\vdots\,\mathcal{D}_2\\ \varphi, \Gamma' \Rightarrow \Delta'\end{matrix}}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\mathit{Cut}$$

Then we perform the following lifting of the cut and rely on the inductive hypotheses:

$$\cfrac{\cfrac{\begin{matrix}\vdots\,\mathcal{D}_1\\ \Gamma^* \Rightarrow \Delta^*, \varphi\end{matrix} \qquad \begin{matrix}\vdots\,\mathcal{D}_2\\ \varphi, \Gamma' \Rightarrow \Delta'\end{matrix}}{\Gamma^*, \Gamma' \Rightarrow \Delta^*, \Delta'}\,\mathit{Cut}}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,R$$

A bit of care is needed for the subcases of *Repl*$_1$ and *Repl*$_2$. We describe the situation with the right premise of *Cut* obtained by *Repl*$_1$, since the general setting follows the same line of reasoning. Assume then we have

$$\dfrac{\overset{\vdots\,\mathcal{D}_1}{\Gamma \Rightarrow \Delta, x : p} \qquad \dfrac{\overset{\vdots\,\mathcal{D}_2}{y \in \{x\}, x : p, y : p, \Gamma' \Rightarrow \Delta'}}{y \in \{x\}, x : p, \Gamma' \Rightarrow \Delta'}\,\mathit{Repl}_1}{y \in \{x\}, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\mathit{Cut}$$

Now, if $\mathcal{D}_1$ is an initial sequent, or if $x : \bot$, $x : \varnothing$, or $x \in R[x]$ occurs in Γ we can weaken the right premise of *Cut*, or recur to the appropriate 0-ary $\mathcal{L}$-rule. Otherwise, the left premise of the cut must have been derived by some rule, and $x : p$ cannot be its principal formula. Lifting the cut does the job.

Case 3. We omit the propositional case, referring to [NvP11, Thm. 11.9]. Notice first that we need not to consider the rules for extensions of $\mathbb{IL}$, since by design relational atoms only occur in the antecedent of sequents; similar considerations hold for the rules for GVS. For the local forcing rules we have

$$\dfrac{\dfrac{\overset{\vdots\,\mathcal{D}_1}{y \in a, \Gamma \Rightarrow \Delta, y : A}}{\Gamma \Rightarrow \Delta, a \Vdash^{\forall} A}\,\mathcal{R}{\Vdash^{\forall}} \qquad \dfrac{\overset{\vdots\,\mathcal{D}_2}{x : A, x \in a, a \Vdash^{\forall} A, \Gamma' \Rightarrow \Delta'}}{x \in a, a \Vdash^{\forall} A, \Gamma' \Rightarrow \Delta'}\,\mathcal{L}{\Vdash^{\forall}}}{\Gamma, x \in a, \Gamma' \Rightarrow \Delta, \Delta'}\,\mathit{Cut}$$

which is solved by

$$\dfrac{\overset{\vdots\,\mathcal{D}_1[x/y]}{x \in a, \Gamma \Rightarrow \Delta, x : A} \qquad \dfrac{\Gamma \Rightarrow \Delta, a \Vdash^{\forall} A \qquad \overset{\vdots\,\mathcal{D}_2}{x : A, x \in a, a \Vdash^{\forall} A, \Gamma' \Rightarrow \Delta'}}{x \in a, x : A, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\mathit{Cut}}{x \in a, x \in a, \Gamma, \Gamma, \Gamma' \Rightarrow \Delta, \Delta', \Delta}\,\mathit{Cut}$$

where the upper cut is on derivations of smaller height, and on a label of same range; the lower one is on formulas of smaller weight. Application(s) of contraction to the conclusion gives the conclusion of the cut.

For the intermediate modality we have

$$\dfrac{\dfrac{\overset{\vdots\,\mathcal{D}_1}{yS_xa, \Gamma \Rightarrow \Delta, y : \langle]_x A, a \Vdash^{\forall} A}}{yS_xa, \Gamma \Rightarrow \Delta, y : \langle]_x A}\,\mathcal{R}\langle] \qquad \dfrac{\overset{\vdots\,\mathcal{D}_2}{yS_xb, b \Vdash^{\forall} A, \Gamma' \Rightarrow \Delta'}}{y : \langle]_x A, \Gamma' \Rightarrow \Delta'}\,\mathcal{L}\langle]}{yS_xa, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\mathit{Cut}$$

which is solved by

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\mathcal{D}_1\\ yS_xa, \Gamma \Rightarrow \Delta, y : \langle]_x A, a \Vdash^{\forall} A\end{array} \qquad y : \langle]_x A, \Gamma' \Rightarrow \Delta'}{yS_xa, \Gamma, \Gamma' \Rightarrow \Delta, \Delta', a \Vdash^{\forall} A}\,\textit{Cut} \qquad \begin{array}{c}\vdots\,\mathcal{D}_2[a/b]\\ yS_xa, a \Vdash^{\forall} A, \Gamma' \Rightarrow \Delta'\end{array}}{yS_xa, yS_xa, \Gamma, \Gamma', \Gamma' \Rightarrow \Delta, \Delta', \Delta'}\,\textit{Cut}$$

where the upper cut is on derivations of smaller height, and the lower one is on formulas of smaller weight. Application(s) of contraction to the conclusion gives the conclusion of the cut.

Finally we consider the case of rules for $\rhd$. The general setting is

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\mathcal{D}_1\\ z \in R[x], z : A, z : A \rhd_i B, \Gamma \Rightarrow \Delta, z : \langle]_i B\end{array}}{\Gamma \Rightarrow \Delta, x : A \rhd_i B}\,\mathcal{R}\rhd \qquad \dfrac{\mathcal{D}_2 \qquad \mathcal{D}_3 \qquad \mathcal{D}_4}{y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta'}\,\mathcal{L}\rhd}{y \in R[x], \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\textit{Cut}$$

where $\mathcal{D}_2$ is a derivation of $y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta', y : A$; $\mathcal{D}_3$ is a derivation of $y : \langle]_i B, y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta'$; and $\mathcal{D}_4$ is a derivation of $y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta', y : A \rhd_i B$.

Perform the following steps:

a.

$$\dfrac{\Gamma \Rightarrow \Delta, x : A \rhd_i B \qquad \begin{array}{c}\vdots\,\mathcal{D}_2\\ y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta', y : A\end{array}}{y \in R[x], \Gamma, \Gamma' \Rightarrow \Delta, \Delta', y : A}\,\textit{Cut}$$

where the cut is on derivations of smaller height.

b.

$$\dfrac{\Gamma \Rightarrow \Delta, x : A \rhd_i B \qquad \begin{array}{c}\vdots\,\mathcal{D}_3\\ y \in R[x], y : \langle]_i B, x : A \rhd_i B, \Gamma' \Rightarrow \Delta'\end{array}}{y \in R[x], y : \langle]_i B, \Gamma, \Gamma' \Rightarrow \Delta, \Delta'}\,\textit{Cut}$$

where the cut is on derivations of smaller height.

c.

$$\dfrac{\Gamma \Rightarrow \Delta, x : A \rhd_i B \qquad \begin{array}{c}\vdots\, \mathcal{D}_3\\ y \in R[x], x : A \rhd_i B, \Gamma' \Rightarrow \Delta', y : A \rhd_i B\end{array}}{y \in R[x], \Gamma, \Gamma' \Rightarrow \Delta, \Delta', y : A \rhd_i B}\,\textit{Cut}$$

where the cut is on derivations of smaller height.

d. Finally proceed as follows:

$$\dfrac{\begin{array}{c}\vdots\, a.\\ \cdots \Rightarrow \cdots, y : A\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\vdots\, \text{c.}\\ \cdots \Rightarrow \cdots, y : A \rhd_i B\end{array} \qquad \begin{array}{c}\vdots\, \mathcal{D}_1[y/z]\\ y \in R[x], y : A, y : A \rhd_i B, \Gamma \Rightarrow \Delta, y : \langle]_i B\end{array}}{y \in R[x]^2, \Gamma^2, \Gamma', y : A \Rightarrow \Delta^2, \Delta', y : \langle]_i B}\,\textit{Cut} \qquad \begin{array}{c}\vdots\, \text{b.}\\ y : \langle]_i B, \cdots \Rightarrow, \cdots\end{array}}{y \in R[x]^3, \Gamma^3, \Gamma'^2, y : A \Rightarrow \Delta^3, \Delta'^2}\,\textit{Cut}}{y \in R[x]^4, \Gamma^4, \Gamma'^3 \Rightarrow \Delta^4, \Delta'^3}\,\textit{Cut}$$

where the first cut from the top is on a label of smaller range, the second one and the third are on formulas of smaller weight.[17] The conclusion of the original cut can then be obtained by applying steps of contraction to the conclusion of the derivation described at point *d*.

⊠

Now that we know that the cut rule is admissible in G3IL* we can prove admissibility of generalised replacement rules.

LEMMA 5.4.9. *The rules Repl*$_1$ *and Repl*$_2$ *generalised to all formulas of the language are admissible in* G3IL*.

Proof. By simultaneous induction on the weight of formulas. Since contraction and cut are admissible for our calculi, it is enough to prove that the sequent $y \in \{x\}, \varphi(x) \Rightarrow \varphi(y)$ is derivable for any labelled formula φ, and then apply a *Cut*, followed by contraction steps, to the premise of generalised *Repl*$_1$. For *Repl*$_2$, we reason symmetrically.

The proof then proceed by a straightforward induction on the weight of $\varphi(x)$ on the lines of [NvP08, Lemma 6.5.2].

⊠

We opened Section 2.3 by endorsing a famous claim of Raymond Smullyan's about the real relevance of cut elimination theorems. For classical propositional logic, indeed, cut admissibility guarantees that the subformula principle holds in G3cp. For labelled calculi that principle needs to be somehow

[17] For reasons of space, we had to replace some unessential formulas in sequents by dots.

relaxed. In its original version, the subformula property holds whenever any sequent occurring in a derivation of a given sequent $\Gamma \Rightarrow \Delta$ contains only subformulas of the formulas composing the latter. In our systems, the rules for $\rhd$ introduce the intermediate modality $\langle]_i$, by decomposing $A \rhd_i B$ into A and $\langle]_i B$, and the latter is not, in strict terms, a subformula of $A \rhd_i B$. However, its weight is defined to be smaller than that of interpretability formulas: Therefore, we might say that it is less complex, and, in that flexible sense, loosely generated by a $\rhd$-formula. Moreover, it is not hard to prove a pureness condition on labels: In G3IL*, any derivation contains either eigenvariables in rules with freshness condition, or labels already present in the conclusion. In this sense, the family of calculi we have design can be considered analytic, in line with the standard G3 paradigm.

5.5. Completeness

In the present section, we prove that our family of labelled sequent calculi is sound and complete w.r.t. the semantic and axiomatic presentations of interpretability logics under investigation.

We give syntactic proofs of those results, and then discuss some aspects concerning a direct proof of semantic completeness for G3IL*.

5.5.1. Syntactic completeness

For a start we need to interpret the underlying language of the labelled formulas into $\mathcal{L}_{\rhd}$, in which the modality is not indexed. Therefore we agree to read $x : A \rhd B$ as $x : A \rhd_x B$ unless otherwise stated.

We immediately have

Theorem 5.5.1 (**Completeness**). *If a formula A is a lemma of $\mathbb{IL}$ or any of its extensions, then there is a derivation of the sequent $\Rightarrow x : A$ in the calculus* G3IL* *for the corresponding logic.*

Proof. We rely on the equivalence between $\mathbb{IL}$ and $\mathbb{IL}_{\rhd}$, and prove that all the axiom schemas and inference rules of the latter are derivable or proven to be admissible in G3IL. For the extensions, we only need to prove the axiom schema corresponding to the specific semantic rule of the family of labelled systems.

We recall the axiomatisation of $\mathbb{IL}_{\rhd}$ and its extensions from Def. 5.3.2 and Section 5.1.1:

$\mathbb{IL}_{\rhd}$	standard axiomatisation of $\mathbb{CPC}$
	schema IL2 : $A \rhd B \to B \rhd C \to A \rhd C$
	schema IL3 : $A \rhd C \to B \rhd C \to A \vee B \rhd C$
	schema IL-Löb: $A \rhd (A \wedge (A \rhd \bot))$
	$\rhd$Rule $\quad \dfrac{A \to B}{A \rhd B}$
Extensions	schema M : $\quad A \rhd B \to A \wedge (\neg C \rhd \bot) \rhd B \wedge (\neg C \rhd \bot)$
	schema P : $\quad A \rhd B \to \neg(A \rhd B) \rhd \bot$
	schema W : $\quad A \rhd B \to A \rhd B \wedge (A \rhd \bot)$
	schema KM1 : $\quad A \rhd \neg(\top \rhd \bot) \to \top \rhd \neg A$
	schema M_0 $\quad A \rhd B \to \neg(A \rhd \bot) \wedge (\neg C \rhd \bot) \rhd B \wedge (\neg C \rhd \bot)$

For propositional logic, the derivability of axiom schemas is straightforward; admissibility of cut assures the admissibility of modus ponens.

Next notice that the simplest $\mathcal{L}$-rule for $\rhd$

$$\dfrac{y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A \qquad y : \langle]_i B, y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta}{y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\rhd^{S}$$

is admissible in our systems, for any sequent of the form

$$y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A \rhd_i B$$

is derivable in G3IL*. The same holds for the rule

$$\dfrac{y \in R[x], y : A, \Gamma \Rightarrow \Delta, y : \langle]_i B}{\Gamma \Rightarrow \Delta, x : A \rhd_i B}\ \mathcal{R}\rhd^{S}{}_{(y!)}$$

Now, we proceed with the basic calculus:

IL2: The derivation is rendered in Figure 5.4.

IL3: The derivation is rendered in Figure 5.5.

IL-Löb: The derivation is rendered in Figure 5.6.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{
              \dfrac{
                \dfrac{
                  \dfrac{
                    \dfrac{
                      \dfrac{
                        \dfrac{
                          yS_xb, zS_xb, b \Vdash^{\forall} C, z : B, z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C, b \Vdash^{\forall} C
                        }{
                          yS_xb, zS_xb, b \Vdash^{\forall} C, z : B, z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
                        }\,\mathcal{R}\langle]
                      }{
                        zS_xb, b \Vdash^{\forall} C, z : B, z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
                      }\,\mathit{Qtrans6}
                    }{
                      z : \langle]_x C, z : B, z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
                    }\,\mathcal{L}\langle]
                    \qquad
                    z : B, \cdots \Rightarrow \cdots, z : B
                  }{
                    z : B, z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
                  }\,\mathcal{L}\rhd^{S}
                }{
                  z \in R[x], a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
                }\,\mathcal{L}\Vdash^{\forall}
              }{
                a \subseteq R[x], z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x]^2, y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
              }\,\mathcal{L}\subseteq
            }{
              z \in a, yS_xa, a \Vdash^{\forall} B, y \in R[x], y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, z : C
            }\,\mathit{DefS}_1
          }{
            yS_xa, a \Vdash^{\forall} B, y \in R[x], y : A, x : A \rhd_x B, x : B \rhd_x C \Rightarrow y : \langle]_x C, a \Vdash^{\forall} C
          }\,\mathcal{R}\Vdash^{\forall}
        }{
          yS_xa, a \Vdash^{\forall} B, y \in R[x], y : A, x : A \rhd_x B, x : B_{\rhd} C \Rightarrow y : \langle]_x C
        }\,\mathcal{R}\langle]
      }{
        y : \langle]_x B, y \in R[x], y : A, x : A \rhd_x B, x : B_{\rhd} C \Rightarrow y : \langle]_x C
      }\,\mathcal{L}\langle]
      \qquad
      y : A, \cdots \Rightarrow \cdots, y : A
    }{
      y \in R[x], y : A, x : A \rhd_x B, x : B_{\rhd} C \Rightarrow y : \langle]_x C
    }\,\mathcal{L}\rhd^{S}
  }{
    x : A \rhd_x B, x : B \rhd_x C \Rightarrow x : A \rhd_x C
  }\,\mathcal{R}\rhd^{S}
$$

Figure 5.4: Derivation of IL2

$$\dfrac{\dfrac{\dfrac{\dfrac{y : B, \cdots \Rightarrow \cdots, y : B \qquad y : A, \cdots \Rightarrow \cdots, y : B}{y \in R[x], y : A \vee B, x : A \rhd_x C, x : B \rhd_x C \Rightarrow y : \langle]_x C, y : A, y : B}\,\mathcal{L}\vee \qquad y : \langle]_x C, \cdots \Rightarrow y : \langle]_x C, y : A}{y \in R[x], y : A \vee B, x : A \rhd_x C, x : B \rhd_x C \Rightarrow y : \langle]_x C, y : A}\,\mathcal{L}\rhd^S \qquad y : \langle]_x C, y \in R[x], y : A \vee B, x : A \rhd_x C, x : B \rhd_x C \Rightarrow y : \langle]_x C}{y \in R[x], y : A \vee B, x : A \rhd_x C, x : B \rhd_x C \Rightarrow y : \langle]_x C}\,\mathcal{L}\rhd^S}{x : A \rhd_x C, x : B \rhd_x C \Rightarrow x : A \vee B \rhd_x C}\,\mathcal{R}\rhd^S$$

Figure 5.5: Derivation of IL3

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            \dfrac{
              \dfrac{
                \dfrac{
                  \dfrac{
                    yS_xb, yS_x\{z\}, z \in \{z\}, zS_xb, {\color{blue}b \Vdash^{\forall} (A \wedge (A \rhd \bot))}, z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot, {\color{blue}b \Vdash^{\forall} (A \wedge (A \rhd \bot))}
                  }{
                    yS_xb, yS_x\{z\}, z \in \{z\}, zS_xb, b \Vdash^{\forall} (A \wedge (A \rhd \bot)), z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
                  } \mathcal{R}\langle]
                }{
                  yS_x\{z\}, z \in \{z\}, zS_xb, b \Vdash^{\forall} (A \wedge (A \rhd \bot)), z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
                } \mathit{Qtrans6}
              }{
                z \in \{z\}, zS_xb, b \Vdash^{\forall} (A \wedge (A \rhd \bot)), z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
              } \mathit{DefS}_2
            }{
              zS_xb, b \Vdash^{\forall} (A \wedge (A \rhd \bot)), z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
            } \mathit{Sing}
          }{
            z : \langle]_x(A \wedge (A \rhd \bot)), z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
          } \mathcal{R}\langle]
          \qquad
          z : A, \cdots \Rightarrow \cdots , z : A
        }{
          z \in R[y], z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
        } \mathcal{L}\rhd^{S}
      }{
        z \in R[i], z : A, i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), z : \langle]_i\bot
      } \mathit{Repl}_2
    }{
      i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), i : A \rhd_i \bot
    } \mathcal{R}\rhd^{S}
    \qquad
    \dfrac{
      i : A, \cdots \Rightarrow \cdots , i : A
    }{
      i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), i : A
    } \mathit{Repl}_2
  }{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{
            i \in \{y\}, yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), i : (A \wedge (A \rhd \bot))
          }{
            yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot)), \{y\} \Vdash^{\forall} (A \wedge (A \rhd \bot))
          } \mathcal{R}\Vdash^{\forall}
        }{
          yS_x\{y\}, y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot))
        } \mathcal{R}\langle]
      }{
        y \in R[x], y : A, y : A \rhd_x (A \wedge (A \rhd \bot)) \Rightarrow y : \langle]_x(A \wedge (A \rhd \bot))
      } \mathit{Qrefl}
    }{
      \Rightarrow x : A \rhd_x (A \wedge (A \rhd \bot))
    } \mathcal{R}\rhd
  } \mathcal{R}\wedge
}{}
$$

Figure 5.6: Derivation of IL-Löb

Löb Rule:

$$
\begin{array}{c}
\vdots\ \text{By assumption } [i/x] \\
\dfrac{\Rightarrow i : A \to B}{i : A \Rightarrow i : B}\ \textit{Inversion } \mathcal{R}{\to} \\
\hline
yS_x\{y\}, y \in R[x], y : A \rhd_x B, y : A, i \in \{y\}, i : A \Rightarrow i : B \quad \mathcal{L}\textit{Wk} \\
\hline
yS_x\{y\}, y \in R[x], y : A \rhd_x B, y : A, i \in \{y\} \Rightarrow i : B \quad \textit{Repl}_1 \\
\hline
yS_x\{y\}, y \in R[x], y : A \rhd_x B, y : A, i \in \{y\} \Rightarrow i : B, y : \langle]_x B \quad \mathcal{R}\textit{Wk} \\
\hline
yS_x\{y\}, y \in R[x], y : A \rhd_x B, y : A \Rightarrow y : \langle]_x B, \{y\} \Vdash^{\forall} B \quad \mathcal{R}{\Vdash^{\forall}} \\
\hline
yS_x\{y\}, y \in R[x], y : A \rhd_x B, y : A \Rightarrow y : \langle]_x B \quad \mathcal{R}\langle] \\
\hline
y \in R[x], y : A \rhd_x B, y : A \Rightarrow y : \langle]_x B \quad \textit{Qrefl} \\
\hline
\Rightarrow x : A \rhd_x B \quad \mathcal{R}{\rhd}
\end{array}
$$

For reasons of space, we are forced to omit the derivations of the interpretability principles for extensions of $\mathbb{IL}_{\rhd}$, since they could not be readable in their rendering on screen. In any case, the reader is warmly invited to check by pencil and paper that each of the modal schemas characterising those extensions is indeed derivable in the calculus defined by the appropriate semantic rule.

⊠

5.5.2. Soundness

For proving the converse direction, we need to interpret relational atoms and labelled formulas in generalised Veltman models. The calculi per se do not have a direct formula interpretation in the language of $\mathtt{Form}_{\rhd}^{(i)}$ but we can define an interpretation for them in GVS: We need a function that interprets the labels in generalised Veltman frames, thus connecting the syntactic elements of the calculus with the semantic notions of Section 5.2.2.

Definition 5.5.2. Let $\mathcal{M} = \langle W, R, \{S_x\}_{x \in W}, v \rangle$ be a generalised Veltman model for $\mathbb{IL}$ or its extensions, $\mathcal{W}$ a set of world labels, and $\mathcal{A}$ a set of set labels. A **$\mathcal{WA}$-interpretation** over $\mathcal{M}$ consists of a pair of functions (ρ, σ) such that:

- $\rho : \mathcal{W} \to W$ maps each $i \in \mathcal{W}$ into a world $\rho(i) \in W$;

- $\sigma : \mathcal{A} \to (\wp(W) \setminus \{\varnothing\})$ maps each $a \in \mathcal{A}$ into a non-empty set of worlds $\sigma(a) \in \wp(R[x])$, for $x \in W$.

The notion of **satisfiability** of a labelled formula under a $\mathcal{WA}$-interpretation is defined by cases on the form of that formula

- $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x]$ if $\rho(x) R \rho(y)$;

- $\rho, \sigma \vDash_{\mathcal{M}} y \in a$ if $\rho(y) \in \sigma(a)$;
- $\rho, \sigma \vDash_{\mathcal{M}} yS_x a$ iff $y \in S_x^{-1}a$: if $\rho(y)S_{\rho(x)}\sigma(a)$;
- $\rho, \sigma \vDash_{\mathcal{M}} y \in R[a]$ if, for some $x \in \sigma(a)$, $xR\rho(y)$;
- $\rho, \sigma \vDash_{\mathcal{M}} a \subseteq b$ if $\sigma(a) \subseteq \sigma(b)$;
- $\rho, \sigma \vDash_{\mathcal{M}} a \cap b$ if $\sigma(a) \cap \sigma(b)$;
- $\rho, \sigma \vDash_{\mathcal{M}} y \in \{x\}$ if $\rho(y) \in \{\rho(x)\}$;
- $\rho, \sigma \vDash_{\mathcal{M}} yS_i\{x\}$ if $\rho(y)S_{\rho(i)}\{\rho(x)\}$;
- $\rho, \sigma \vDash_{\mathcal{M}} x : p$ if $\rho(x) \Vdash_{\mathcal{M}} p$, and similarly for formulas obtained by classical propositional connectives;
- $\rho, \sigma \vDash_{\mathcal{M}} a \Vdash^{\forall} A$ if $\rho(a) \Vdash^{\forall} A$;
- $\rho, \sigma \vDash_{\mathcal{M}} x : \langle]_i A$ if, for some $a \in (\wp(R[\rho(i)]) \smallsetminus \{\varnothing\})$, $\rho(x)S_{\rho(i)}a$ and $a \Vdash^{\forall} A$;
- $\rho, \sigma \vDash_{\mathcal{M}} x : A \rhd_i B$ if, for all $y \in R[\rho(x)]$, if $y \Vdash_{\mathcal{M}} A$, then $y \Vdash_{\mathcal{M}} \langle]_{\rho(i)} B$.

Given a sequent $\Gamma \Rightarrow \Delta$, let $\mathcal{W}, \mathcal{A}$ be the sets of world and set labels occurring in $\Gamma \cup \Delta$, and let (ρ, σ) be a $\mathcal{W}\mathcal{A}$-interpretation.
Define $\rho, \sigma \vDash_{\mathcal{M}} \Gamma \Rightarrow \Delta$ if either $\rho, \sigma \nvDash_{\mathcal{M}} \varphi$ for some $\varphi \in \Gamma$ or $\rho, \sigma \vDash_{\mathcal{M}} \psi$ for some $\psi \in \Delta$. Define then validity under all interpretations by $\vDash_{\mathcal{M}} \Gamma \Rightarrow \Delta$ iff $\rho, \sigma \vDash_{\mathcal{M}} \Gamma \Rightarrow \Delta$ for all (ρ, σ). Finally, let us say that a sequent is valid in all generalised Veltman models if $\rho, \sigma \vDash_{\mathcal{M}} \Gamma \Rightarrow \Delta$ for all models based on GVS appropriate to a specific interpretability logic.

THEOREM 5.5.3 (Soundness). *If a sequent $\Rightarrow x : A$ is derivable in* G3IL**, then A is a lemma of $\mathbb{IL}$ or of the corresponding extension.*

Proof. We prove something stronger, namely:

(♠) If a sequent $\Gamma \Rightarrow \Delta$ is derivable in G3IL*, then it is valid in the corresponding class of generalised Veltman models.

By modal completeness of the axiomatic calculi for interpretability we are dealing with, the main theorem follows.

Now, the proof of (♠) is by straightforward induction on the height of the derivation, by recurring to the notion of interpretation defined before. By means of example we show soundness of the left and right rule for the $\rhd$-operator.

$\mathcal{L}\rhd$ By inductive hypothesis we have $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A$; $\rho, \sigma \vDash_{\mathcal{M}} y : \langle]_i B, y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta$; and $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta, y : A \rhd_i B$. $\rho(y) \in R[\rho(x)]$. There are two relevant cases to consider.

One has $\rho, \sigma \vDash_{\mathcal{M}} y : A$, $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x]$ and $\rho, \sigma \nvDash_{\mathcal{M}} \langle]_i B$. From the former we have that $\rho(y) \Vdash A$; from the latter that $\rho(y) \nVdash \langle]_i B$. By definition, this means that $\rho, \sigma \nvDash_{\mathcal{M}} x : A \rhd_i B$, so that $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x], x : A \rhd_i B, \Gamma \Rightarrow \Delta$.

The second one is $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x]$ and $\rho, \sigma \nvDash_{\mathcal{M}} y : A \rhd_i B$. From the latter we have that $\rho(y) \nVdash A \rhd_i B$, from which we know that there exists a $z \in R[y]$ such that $z \Vdash A$ and $z \nVdash \langle]_i B$. By transitivity we see that we can reduce the situation to the first one.

$\mathcal{R}\rhd$ Assume by inductive hypothesis that $\rho, \sigma \vDash_{\mathcal{M}} y \in R[x], y : A \rhd_i B, y : A, \Gamma \Rightarrow \Delta, y : \langle]_i B$. We reason by contradiction. If none of Δ and $x : A \rhd_i B$ is valid under this interpretation, then there exists a $w \in R[\rho(x)]$ such that $w \Vdash A$ and $w \nVdash \langle]_i B$. Then we can define a new ρ' which is identical to ρ except possibly on y, for which we set $\rho'(y) = w$. But by assumption on validity of the premise of the rule, since $w \Vdash A$ and $w \nVdash \langle]_i B$, we have $w \nVdash A \rhd_i B$. It is now straightforward to see that an ascending R-chain can be built on the model, contrary to the assumption that $\mathcal{M}$ is finite – or, equivalently, Noetherian.

⊠

5.5.3. On termination

Labelled sequent calculi have a peculiar characteristic: From a failed proof search, it is generally possible to extract a countermodel to the sequent at the root of the derivation tree. This way, a direct and constructive proof of modal completeness is obtained.[18]

For our family G3IL*, we could claim that the same holds: It is clearly possible to define a Tait-Schütte-Takeuti procedure to construct a refutation of a sequent from a failed proof-search; however, in the present setting, it is essential to define that saturation process along with a terminating strategy for performing proof search in G3IL*.

The reader familiar with this kind of tasks will see at once that our systems are rather complex to handle from a combinatorial perspective, and this imposes several cases to check when proving that root-first proof search in G3IL* does terminate.[19]

[18] Refer [Neg14b] for an extensive discussion.

[19] In particular, it is rather difficult to handle the interaction of labels, because of the presence of the indexed relation S, as well as the need to label the interpretability modality to express Noetherian induction by the $\mathcal{R}\rhd$ rule.

For sure, it is first essential to present an equivalent basic proof system for IL, that – following [Neg05, Sect. 5] – might call G3KIL. This basic calculus is obtained by replacing $\mathcal{L}\rhd$ with the simplified rule $\mathcal{L}\rhd^{S}$ that we used in the proof of Theorem 5.5.1; extensions are built on top of it, according to the rules in Figure 5.3.

We are confident that G3KIL – as well as its extensions $\mathsf{G3KIL}^{\mathsf{M}}$ and $\mathsf{G3KIL}^{\mathsf{P}}$ – can be shown to satisfy terminating proof search, by adapting the proof strategy of [GNO21] to our much more intricate setting. A complete proof of termination for those systems should be then easily extended to the other interpretability logic constituting our uniform family. A further step in difficulty would probably materialise when dealing with the labelled sequent calculus for $\mathbb{ILP}_0$, since –in order to define the generalised frame condition corresponding to the schema P_0 by means of a geometric rule – we would need further rules for set theoretic operations on world sets, and thus further cases of interaction between world and set labels.

Since at present time, we can only provide an informal argument supporting our claims, in complete earnestness, we prefer to propose termination of proof search in our family of labelled sequent calculi as the following

Conjecture 5.5.4 (**Termination**). *There exists a strategy making proof search in* $\mathsf{G3KIL}^{\star}$ *for a sequent of the form* $\Rightarrow x : A$ *always terminate in a finite number of steps. Moreover, from a failed proof search, it is possible to extract a countermodel to* A *belonging to appropriate class of generalised Veltman frames.*

Related work

In this chapter we have presented a family $\mathsf{G3IL}^{\star}$ of labelled sequent calculi for interpretability logics based on the generalised Veltman semantics by [Ver92], which subsumes the standard relational semantics for those modal logic discussed in [dJV90].

Our systems are modularly obtained from a basic calculus G3IL according to the methodology of [NvP08, Ch. 6] for formalising geometric theories in the G3 paradigm for sequent calculi. We have proved that all these calculi do satisfy the main desiderata for good G3-sequent systems; in particular, we have given a detailed proof of the cut-elimination theorem for $\mathsf{G3IL}^{\star}$ (Theorem 5.4.8), for which we used a ternary measure of complexity of the cut instance.

To the best of our knowledge, there are no proof-theoretic treatments of interpretability logics as extensive as the one we have presented here.

A labelled system for IL and some extensions is defined in [HJ16] as prefixed tableaux internalising standard Veltman semantics. Our approach can be considered its direct dual – in virtue of the duality subsisting among labelled tableaux and labelled sequent calculi – but it differs from that work by Joost Joosten and Tuomas Hakoniemi for our focus on structural properties of the calculi, and for being based on *generalised* Veltman semantics: This has allowed us to cover a wider range of interpretability principles – notably, the system $\mathbb{ILW}$ based on de Jongh-Visser schema, that cannot be characterised by a first-order formula of the language of standard Veltman semantics.

On the more traditional side, Katsumi Sasaki gives in [Sas02] a Gentzen-style sequent calculus for the basic interpretability logic IL. No extension is considered in that work, nor – as it is common for standard sequent calculi – it is easy to see how to extend the calculus discussed in that paper to cover further interpretability logics.

We care to stress that what we have discussed in the present chapter can be also seen as a case study for investigating the potentiality of explicit internalisation of semantics in sequent calculi: Indeed, it is far from obvious how to design a sequent system for the logics we have considered according to the methodology of implicit internalisation. Actually, to the best of our knowledge, no attempt has been made yet to do so by the scientific community.

As future work, we would be pleased to solve Conjecture 5.5.4 by providing a fully detailed proof of termination of proof search in our sequent calculi. This would be far from a purely technical result, since although decidability of all the interpretability logics under consideration is known, their analysis from the complexity theory point of view is still at its very beginning, and it is limited to the results in [BJ11] and [MPV19].

Finally, it might be relevant to apply the labelling technique we adopted here to provide constructive proof of modal completeness for interpretability logics: The latter is indeed a flourishing research field on the topic, and several model-theoretic techniques have been developed in recent years, as summarised by [JRMV20]. In spite of progressively achieving a modular character, parts of those proofs are still slightly obscure, and sensitive to the logic under investigation, since they rely on (variations of) Henkin's method.[20] A "reverse engineering" approach relying on the design methodology we used in the present chapter might then been useful to develop alternative modular proofs of completeness, and even solve some open problems on model theoretic matters for specific interpretability principles.

[20] The proofs by Joosten and collaborators are based on a 'labelling technique' that has no relation to labelled sequent calculi: It is just a coincidence that their approach and ours are both talking about labels for interpretability logics.

Part II

Automated Reasoning

Introduction to Part II

Civilisation advances by extending the number of important operations which can be performed without thinking about them.
Alfred North Whitehead, *An introduction to mathematics*, 1911

In 2023, 110 years since the publication of Bertrand Russell and Alfred Whitehead's *Principia Mathematica* will be celebrated.

That work should be correctly considered a first mammoth achievement in rigorous formalisation of mathematics and mathematical reasoning on the basis of logical primitives. Their success was mainly technical – even though it was led by a very principled philosophical position, namely logicism – since the development of formal proofs from their basic laws was painstaking, but, rephrasing Russell's own words, required a most exhausting intellectual strain.

From a philosophical perspective, however, Russell and Whitehead's masterpiece can also be thought of as a concrete embodiment in *type theory* of Gottfried Leibniz's *calculus ratiocinator* and *characteristica universalis*.

From the practical point of view, the advent of computers made a revolution in the field, since they are designed to handle formal objects without incurring into error, so that formalising proofs is made easier, and it is also possible to check formal derivations. Thanks to computers, the correctness of a mathematical proof becomes an algorithmically verifiable question – at least, when the right research tool is chosen.

There exist several kinds of computer programs – generally categorised as *proof assistants*, or *theorem provers* – that are capable of helping the mathematician in developing or verifying a proof of a statement, concerning a mathematical theory as well as the expected behaviour of a hardware or software component, as witnessed by the survey [HUW14].

The design of those proof assistants is the exact place where structural proof theory acquires a most relevant applicative nature. Because of the combinatorial and procedural nature of its results and methodology, structural proof theory plays a key role in the development and theory of modern proof assistants and theorem provers [Avi18, Avi19].

The level of automation varies from each proof assistant to another, but their general relevance for mathematical practice has been largely recognised in the last decades. In particular, impressive developments in connecting geometric ideas with the formal notions underlying the logical engine of several proof assistants – namely, the advent of *homotopy type theory* and *univalent reasoning*, patronised by the Fields Medalist Vladimir Voevodsky [The13]– have shown that there exist mutual benefits in thinking about and practising mathematics from a computerised – hence, applied proof-theoretic – perspective [Gra18]. The intuitionistic version of type theory by Per Martin-Löf can be consistently extended by logical principles capturing the informal practice of mathematicians of identifying isomorphic structures, so that the computer formalisation of common everyday mathematics is even more natural for the user. On the other hand, introducing topological and geometrical ideas in the design of computer programs is capable of making easier the computer verification of mathematical models for hardware and software, by allowing more flexible – sometimes referred to as 'synthetic' – coding procedures. On paper, univalence should also be able to provide a completely new approach to theoretical computer science by coding techniques that work uniformly on isomorphic data structures when formalised in different ways.

Nevertheless, even before the advent of homotopy type theory, proof-theoretic techniques and results have played a very active role in computerised reasoning: The correctness of formal proofs developed by a proof assistant depends in an essential way on good computational and structural properties of the very proof assistant. And those properties can be thought of as the type theoretic counterparts of very well-known proof-theoretic results, like normalisation and cut-elimination for calculi in Gentzen's paradigm [Avi21].

In this second part of the thesis, we will make an extensive use of two proof assistants, namely HOL Light – based on simple type theory – and UniMath – based on univalent type theory. We have chosen these two tools since each of them has a peculiar characteristic.

HOL Light [Har17] is designed according to the LCF paradigm, therefore it has a minimalist logical engine that can be safely extended by constructing within the system new deductive rules for performing specific tasks. We have used this intrinsic flexibility of the proof assistant to define within HOL Light itself a theorem prover and countermodel constructor for Gödel-Löb logic: This has been possible since higher order reasoning facilitates the formalisation of modal completeness results, and the proof development mechanism of HOL Light can be extended to efficiently perform a proof search in labelled sequent calculi in complete autonomy.

UniMath [VAG$^+$22], on the contrary, is a univalent version of (a subsystem of) a highly structured and engineered proof assistant, namely Coq [BC13]. It has an intuitionstic and algorithmic nature, and it is therefore a most natural environment to experiment on formalisation of mathematical structures with an eye towards computational aspects of coding. In particular, we have used the normalisation algorithm of Coq to perform automated evaluations of known constructions in universal algebra, which we have implemented according to the univalent point of view.

The structure of the present part of the thesis is as follows:

▷ Chapter 6 presents a complete formalisation in HOL Light of modal completeness of Gödel-Löb logic. We start with a standard deep embedding in HOL Light of the axiomatic calculus $\mathbb{GL}$ and its relational semantics, and then proceed with the proof of soundness and completeness of the former w.r.t. the latter. Our formal proof for completeness uses automated higher order reasoning of the proof assistant to simplify some convoluted passages in Henkin's method adapted for dealing with $\mathbb{GL}$, which are only sketched in the literature – e.g. in [Boo95]. To the best of our knowledge, this is the only formalisation of modal completeness for that logic currently available.

  Furthermore, and more importantly, we use our proof of completeness to proceed with a *shallow embedding* of the labelled sequent calculus G3KGL in HOL Light, by defining general proof development rules that are capable of mimicking a proof search in G3KGL without the need to define the calculus at the object level. That proof search strategy is completely automated as a tactic of HOL Light which, in case of failure, is capable of providing the user with a countermodel to the formula she has wished to prove in Gödel-Löb logic.

▷ Chapter 7 surveys the code we have developed for the UniMath library on universal algebra. We recall the basic notions of the field – namely, of algebraic multisorted signatures, algebras over a signature, and equational algebras – and then proceed with an implementation of a stack machine that, thanks to the normalisation procedure of Coq, we use to define the terms of an algebra over a signature in such a way that the computer is capable of performing autonomously several operations on them – without recurring to Coq mechanisms for inductive types, which are not allowed in the UniMath coding style. Moreover, we show that the algebras over a signature and equational systems define two univalent categories – that we construct by recurring to the formalism of displayed categories. Besides the formalisation, some concrete examples are proposed of known algebraic structures rephrased in our formalism; these can be thought as an invitation for the reader to check our claims about the computability of our constructions in univalent universal algebra, which has been the main aim of our implementation, largely inspired by the so-called Poincaré principle [Bar92].

6

A THEOREM PROVER AND COUNTERMODEL CONSTRUCTOR FOR PROVABILITY LOGIC IN HOL LIGHT

This chapter deals with two different, but closely related, properties of Gödel-Löb provability logic (GL): Its modal completeness, and its decidability.

We present first a formalised proof in HOL Light proof assistant of modal completeness of GL, that we then use to implement within that very proof assistant a theorem prover for the same logic, which, in case of failure of the proof search, produces a countermodel for the formula that is not a theorem of GL.

Our work starts with a deep embedding of the syntax of propositional modal logic together with the corresponding relational semantics. Next, we introduce the traditional axiomatic calculus $\mathbb{GL}$ and prove the validity of the system w.r.t. irreflexive transitive finite frames.

A more interesting part then follows, consisting of the proof of a number of lemmas *in* $\mathbb{GL}$ which are necessary for proving

THEOREM 6.0.1. *For any formula A, $\mathbb{GL} \vdash A$ iff A is true in any irreflexive transitive finite frame.*

In order to achieve that, we have to formally verify a series of preliminary lemmas and constructions involving the behaviour of syntactical objects used in the standard proof of the completeness theorem. These unavoidable steps are very often only proof-sketched in wide-adopted textbooks in logic, for they mainly involve "standard" reasoning *within* the proof system we are dealing with. But when we are working in a formal setting like we did with HOL Light, we need to split down the main goal into several subgoals, dealing with both the object- and the meta-level. Sometimes, the HOL Light infrastructure does play a very active role in checking the details of the informal reasoning; in other occasions, we have to make a smart use of our formal tools, and develop alternative (or simpler) proof-strategies, still totally verified by the computer.

As it is known, for any logical calculus, a completeness result w.r.t. finite models –aka finite model property– implies the decidability of that very logic. Therefore, the formal proof for Theorem 6.0.1 we discuss in the first part of the present work could be used, in principle, to develop a decision algorithm for $\mathbb{GL}$. That, in turn, would be a useful tool for automating in HOL Light the proof of theorems of GL.

As a matter of fact, developing proofs in axiomatic calculi is never an easy task. In our formalisation we had to develop several proof in the axiomatic calculus for $\mathbb{GL}$, and only in few cases it has been possible to let the proof search to the automation mechanism of HOL Light.

By having a formal proof of the finite model property, one could hope to solve the goal of checking whether a formula is a theorem of GL by shifting from the syntactic problem of finding a proof of that formula in the axiomatic calculus to the semantic problem of checking the validity of that formula in any finite model by applying automated first-order reasoning as implemented in the proof assistant.

Such a strategy has many shortcomings, unfortunately. From the complexity theory view-point, the syntax-to-semantic shift is far from being optimal, since testing the validity of a formula A of size n requires to consider all models with cardinality k, for any $k \leq 2^n$. Less technically, from the practical view-point, nothing assures that the semantic problem is always easier to handle by the proof assistant than the original syntactic goal.

A more promising strategy is suggested by structural proof theory for modal logics.

After Section 1.3, we know that many sequent calculi for non-classical logics are based on an "internalisation" of possible world semantics in Gentzen's original formalism.

This internalisation can be achieved by extending the syntax of formulas constituting standard sequents; or by extending the structure of the very sequents, with the aim of mimicking the structure of models the logic is adequate for (e.g., nested sequents or hypersequents).

We opted for the first option, since our formalisation of Kripke semantics for GL is *per se* a labelling technique in disguise.

Therefore, in the second part of this chapter we introduce what might be considered a shallow embedding in HOL Light of Negri's labelled sequent calculus G3KGL.

To state it clearly: While in the first part we defined the axiomatic system $\mathbb{GL}$ within our proof assistant by means of an inductive definition of the derivability relation for $\mathbb{GL}$, in the second part we define *new tactics* of HOL Light in order to perform a proof search in G3KGL by using the automation infrastructure provided by HOL Light itself.

By relying on the meta-theory for G3KGL developed in [Neg14b], we can safely claim that such an embedding provides a decision algorithm for GL: If our automated proof search positively terminates on a modal formula given as input, HOL Light produces a new theorem, stating that the input formula is a theorem of GL; otherwise, our algorithm prints all the information necessary to construct an appropriate countermodel for the input formula.

Our code is integrated into the current HOL Light distribution, and it is freely available from [Har22]. In the next pages, we will refer to the files constituting the directory `GL` of the most recent distribution.

We can briefly summarise the contents of the chapter as follows:

- In Section 6.1, we introduce the basic ingredients of our development, namely the formal counterparts of the syntax and relational semantics for provability logic, along with some lemmas and general definitions which are useful to handle the implementation of these objects in a uniform way, i.e. without the restriction to the specific modal system we are interested in. The formalisation constitutes large part of the file `modal.ml`;

- In Section 6.2, we formally define the axiomatic calculus $\mathbb{GL}$, and prove in a neat way the validity lemma for this system. Moreover, we give formal proofs of several lemmas *in* $\mathbb{GL}$ ($\mathbb{GL}$-lemmas, for short), whose majority is in fact common to all normal modal logics, so that our proofs might be re-used in subsequent implementations of different systems. This corresponds to contents of our code in `gl.ml`;

- In Section 6.3 we give our formal proof of modal completeness of $\mathbb{GL}$, starting with the definition of maximal consistent *lists* of formulas. In order to prove their syntactic properties – and, in particular, the extension lemma for consistent lists of formulas to maximal consistent lists – we use the $\mathbb{GL}$-lemmas and, at the same time, we adapt an already known general proof-strategy to maximise the gain from the formal tools provided by HOL Light – or, informally, from higher-order reasoning. At the end of the Section, we give the formal definition of bisimilarity for our setup and we prove the associated bisimulation theorem [Pop94, Ch. 11]. Our notion of bisimilarity is polymorphic, in the sense that it can relate classes of frames sitting on different types. With this tool at hand, we can correctly state our completeness theorem in its natural generality (`COMPLETENESS_THEOREM_GEN`) – i.e. for irreflexive, transitive finite frames over any (infinite) type – this way obtaining the finite model property for GL w.r.t. frames having finite *sets* of formulas as possible worlds, as in the standard Henkin's construction. These results are gathered in the file `completeness.ml`.

- Section 6.4 opens the description of our original theorem prover for GL. We collect some basic notions and techniques for proof-theoretic investigations on modal logic. In particular, we recall the labelled sequent

> calculus G3KGL and its main properties. That provides the meta-theory for our decision algorithm. Finally, we describe our implementation of G3KGL – documented by the file `decid.ml` – to give a decision procedure for GL. We recover the formalisation of Kripke semantics for GL presented in Section 6.1 and use it to define new tactics mimicking the rules for G3KGL. Then, we properly define our decision algorithm on the lines of a specific terminating proof search strategy in the labelled sequent calculus. This way, we succeed in extending the HOL Light automation toolbox with an "internal" theorem prover for GL that is also capable of producing a countermodel to any formula for which proof search fails. We propose some hands-on examples of use by considering modal principles that have a certain relevance for meta-mathematical investigations; the interested reader will find further examples in the file `tests.ml`.

We could say that our implementation fluidly moves between two different levels of abstraction: The deep embedding of the axiomatic calculus is adequately reflected in the deep embedding of the possible world semantics by means of the completeness theorem formalised at the meta-level of HOL Light. Such a formal proof, in turn, assures that, at the meta-level, we can safely proceed with a shallow embedding of the labelled sequent calculus by extending the tactics of HOL Light in order to mirror the rules of that calculus. Because of the proof-theoretic property of the "informal" sequent calculus, its formalised counterpart is in fact a real theorem prover and countermodel constructor for the modal logic, built in HOL Light.

Nothing, in the methodology we have just sketched, is specific to GL, so that the procedure we present here can be easily adapted to implement any labelled sequent calculus known in the literature.

We care to stress that our formalisation does not tweak any original HOL Light tools, and it is therefore "foundationally safe". Moreover, since we only used that original formal infrastructure, our results can be easily translated into another theorem prover belonging to the HOL family – or, more generally, endowed with the same automation toolbox.

Writing and Revision Notes. The code we are surveying here is freely available from the official HOL Light distribution, constituting the directory `GL`. The formalisation of modal completeness for $\mathbb{GL}$ has been presented first in [MPB21] at Interactive Theorem Proving (ITP) Conference 2021. The description of our theorem prover and countermodel constructor is given in [MPB22], which the present chapter is based on.

HOL Light Notation. The HOL Light proof assistant is based on *classical* higher-order logic with polymorphic type variables and where equality is the only primitive notion. From a logical viewpoint, the formal engine defined by the *term-conversions* and *inference rules* underlying HOL Light is the same as that described in [LS88], extended by an infinity axiom and the classical characterization of Hilbert's choice operator. A survey of its logical engine and implementation is given in Appendix A.

From a practical perspective, it is a theorem prover privileging a procedural proof style development – i.e. when using it, we have to solve goals by applying *tactics* that reduce them to (eventually) simpler subgoals, so that the interactive aspect of proving is highlighted. Proof-terms can then be constructed by means of *tacticals* that compact the proof into a few lines of code evaluated by the machine.

Logical operators – defined in terms of equality – and λ-abstraction are denoted by specific symbols in ASCII: For the reader's sake, we give a partial glossary in the next table. In the third column of the table, we also report the notation used for the object logic GL (introduced at the beginning of Section 6.1.1).

Informal notation	HOL notation	GL notation	Description
$\bot$	`F`	`False`	Falsity
$\top$	`T`	`True`	Truth
$\neg p$	`~ p`	`Not p`	Negation
$p \wedge q$	`/\`	`&&`	Conjunction
$p \vee q$	`\/`	`||`	Disjunction
$p \Longrightarrow q$	`==>`	`-->`	Implication
$p \Longleftrightarrow q$	`<=>`	`<->`	Biconditional
$\Box p$		`Box p`	Modal Operator
$p_1, \ldots p_N \vdash p$	`p1 ... pN |- p`		HOL theorem
$\vdash p$		`|-- p`	Derivability in $\mathbb{GL}$
$\forall x.\, P(x)$	`!x. P(x)`		Universal quantification
$\exists x.\, P(x)$	`?x. P(x)`		Existential quantification
$\lambda x.\, M(x)$	`\x. M(x)`		Lambda abstraction
$x \in s$	`x IN s`		Set membership

We recall that a Boolean function `s :` α `-> bool` is also called a *set on* α in the HOL parlance. The notation `x IN s` is equivalent to `s x` and must not be confused with a type annotation `x :` α.

In the following sections, we will directly state our results as theorems and definitions in the HOL Light syntax. Note that theorems are prefixed by the turnstile symbol, as in `|- 2 + 2 = 4`. We often report a theorem with its associated name, that is, the name of its associated OCaml constant, e.g.

```
ADD_SYM
  |- !m n. m + n = n + m
```

As expository style, we omit formal proofs at all, but the meaning of definitions, lemmas, and theorems in natural language is hopefully clear after the table we have just given.

We warn the reader that the HOL Light printing mechanism omits type information completely. However, here we manually add type annotations when they might be useful, or even indispensable, in order to avoid ambiguity – including the case of our main results, COMPLETENESS_THEOREM and COMPLETENESS_THEOREM_GEN.

As already told in the introduction, our contribution is now part of the HOL Light distribution. The reader interested in performing these results on her machine – and perhaps building further formalisation on top of it – can run our code with the command

```
loadt "GL/make.ml";;
```

at the HOL Light prompt.

6.1. Basics of modal logic

Following Section 1.2, we deal with a logic that extends classical propositional reasoning by means of a single modal operator which is intended to capture the abstract properties of the provability predicate for arithmetic.

To reason about and within this logic, we have to "teach" HOL Light – our meta-language – how to identify it, starting with its syntax – the object-language – and semantics – the interpretation of this very object-language.

We want to keep everything neat and clean from a foundational perspective, therefore we will define *both* the object-language and its interpretation with no relation to the HOL Light environment. In other terms: Our formulas and operators are *real* syntactic objects which we keep distinct from their semantic counterparts – and from the logical operators of the theorem prover too.

6.1.1. Language and semantics defined

Let us start by fixing the propositional modal language $\mathcal{L}$ we will use throughout the present work. We consider *all* classical propositional operators – conjunction, disjunction, implication, equivalence, negation, along with the 0-ary symbols $\top$ and $\bot$ – and we add a modal unary connective $\Box$. The starting point is, as usual, a denumerable infinite set of propositional atoms $a_0, a_1, \cdots$. Accordingly, formulas of this language will have one of the following forms

$$a \mid A \land B \mid A \lor B \mid A \to B \mid A \leftrightarrow B \mid \neg A \mid \top \mid \bot \mid \Box A \; .$$

The following code extends the HOL system with an the **inductive type of formulas** up to the **atoms** – which we identify with the denumerable type of strings – by using the above **connectives**:

```
let form_INDUCT,form_RECURSION = define_type
  "form = False
        | True
        | Atom string
        | Not form
        | && form form
        | || form form
        | --> form form
        | <-> form form
        | Box form";;
```

Next, we turn to the semantics for our modal language. We use relational models defined in Section 1.1. We recall that a relational frame is made of a non-empty set ‘of possible worlds’ `W`, together with a binary relation `R` on `W`. To this, we add an evaluation function `V` which assigns to each atom of our language and each world `w` in `W` a Boolean value. This is extended to a forcing relation `holds`, defined recursively on the structure of the input formula `p`, that computes the truth-value of `p` in a specific world `w`, on the lines of Def. 1.1.9.

```
let holds = new_recursive_definition form_RECURSION
  ‘(holds f V False (w:W) <=> F) /\
   (holds f V True w <=> T) /\
   (holds f V (Atom s) w <=> V s w) /\
   (holds f V (Not p) w <=> ~(holds f V p w)) /\
   (holds f V (p && q) w <=> holds f V p w /\ holds f V q w) /\
   (holds f V (p || q) w <=> holds f V p w \/ holds f V q w) /\
   (holds f V (p --> q) w <=> holds f V p w ==> holds f V q w) /\
   (holds f V (p <-> q) w <=> holds f V p w <=> holds f V q w) /\
   (holds f V (Box p) w <=> !u. u IN FST f /\ SND f w u ==> holds f V p u)‘;;
```

In the previous lines of code, `f` stands for a generic relational frame – i.e. a pair `(W,R)` of a set of worlds and an accessibility relation – and `V` is an evaluation of propositional variables. Then, the validity of a formula `p` with respect to a frame `(W,R)`, and a class of frames `L`, denoted respectively `holds_in (W,R) p` and `L |= p`, are

```
let holds_in = new_definition
  ‘holds_in (W,R) p <=> !V w. w IN W ==> holds (W,R) V p w‘;;

let valid = new_definition
  ‘L |= p <=> !f. L f ==> holds_in f p‘;;
```

The above formalisation is essentially the one presented in John Harrison’s HOL Light Tutorial [Har17, Sect. 20]. Notice that the usual notion of relational frame requires that the set of possible worlds is non-empty: That condition could be imposed by adapting the `valid` relation. We have preferred to stick to Harrison’s original definitions in our code, but in the next section, when we define the classes of frames we are dealing with, the requirement on `W` is correctly integrated in the corresponding types.

6.1.2. Frames for GL

For carrying out our formalisation, we are interested in the logic of the (non-empty) frames whose underlying relation R is transitive and conversely well-founded – aka Noetherian – on the corresponding set of possible worlds; in other terms, we want to study the modal tautologies in models based on an accessibility relation R on W such that

- if xRy and yRz, then xRz; and
- for no $X \subseteq W$ there are infinite R-chains $x_0 R x_1 R x_2 \cdots$.

In HOL Light, `WF R` states that `R` is a well-founded relation: Then, we express the latter condition as `WF(\x y. R y x)`. Here we see a recurrent motif in logic: Defining a system from the semantic perspective requires non-trivial tools from the foundational point of view, for, to express the second condition, a first-order language is not enough. However, that is not an issue here, since our underlying system is natively higher order:[1]

```
let TRANSNT = new_definition
  ‘TRANSNT (W:W->bool,R:W->W->bool) <=>
   ~(W = {}) /\
   (!x y:W. R x y ==> x IN W /\ y IN W) /\
   (!x y z:W. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z) /\
   WF(\x y. R y x)‘;;
```

From a theoretical point of view, moreover, the question has no deep consequences as we can characterize this class of frames by using a *propositional* language extended by a modal operator $\Box$ that satisfies the *Gödel-Löb axiom schema* (GL) : $\Box(\Box A \to A) \to \Box A$. Here is the formal version of our claim:

```
TRANSNT_EQ_LOB
  |- !W:W->bool R:W->W->bool.
       (!x y:W. R x y ==> x IN W /\ y IN W)
       ==> ((!x y z. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z) /\
            WF (\x y. R y x) <=>
            (!p. holds_in (W,R) (Box(Box p --> p) --> Box p)))
```

The informal proof of the above result is standard and can be found in [Pop94, Theorem 5.7]. The computer implementation of the proof is made easy thanks to Harrison's tactic `MODAL_SCHEMA_TAC` for semantic reasoning in modal logic, documented in [Har17, Sect. 20.3].

By using this preliminary result, we could say that the frame property of being transitive and Noetherian can be captured by Gödel-Löb modal axiom, without recurring to a higher-order language. Nevertheless, that class of frames is not particularly informative from a logical point of view: A frame

[1] We warn the reader that in the previous statement there occur two interrelated mathematical objects both denoted `W` (for convenience): One is the type `W` and the other is the set `W` on the former (in the sense explained in the introduction about the HOL syntax).

in TRANSNT can be too huge to be used in practice – for instance, for checking whether a formula is indeed a theorem of our logic. In particular, when aiming at a completeness theorem, one wants to consider models that are useful for further investigations on the properties of the very logic under consideration – in the present case, decidability of GL, which, as for any other normal modal logic, is an easy corollary of the *finite model property* [Pop94, Ch. 13].

To this aim, we note that, by definition of Noetherianness, our R cannot be reflexive – otherwise $xRxRx\cdots$ would give us an infinite R-chain. This is not enough: The frames we want to investigate are precisely those whose W is finite, and whose R is both irreflexive and transitive:

```
let ITF = new_definition
  `ITF (W:W->bool,R:W->W->bool) <=>
   ~(W = {}) /\
   (!x y:W. R x y ==> x IN W /\ y IN W) /\
   FINITE W /\
   (!x. x IN W ==> ~R x x) /\
   (!x y z. x IN W /\ y IN W /\ z IN W /\ R x y /\ R y z ==> R x z)`;;
```

Now it is easy to see that ITF is a subclass of TRANSNT:

```
ITF_NT
  |- !W R:W->W->bool. ITF(W,R) ==> TRANSNT(W,R)
```

That will be the class of frames whose logic we are now going to define syntactically, on the lines of Section 1.2.

6.2. Axiomatizing GL

We want to identify the logical system generating all the modal tautologies for transitive Noetherian frames; more precisely, we want to isolate the *generators* of the modal tautologies in the subclass of transitive Noetherian frames which are finite, transitive, and irreflexive. Notice that the lemma ITF_NT allows us to derive the former result as a corollary of the latter.

When dealing with the very notion of tautology – or *theoremhood*, discarding the complexity or structural aspects of *derivability* in a formal system – it is convenient to focus on axiomatic calculi. The calculus we are dealing with here is precisely $\mathbb{GL}$.

It is clear from the definition of the forcing relation that for classical operators any axiomatization of propositional classical logic will do the job. Here, we adopt a basic system in which only $\to$ and $\bot$ are primitive – from the axiomatic perspective – and all the remaining classical connectives are defined by axiom schemas and by the inference rule of Modus Ponens imposing their standard behaviour.

Following Section 1.2, to this classical engine we add

- the axiom schema K: $\Box(A \to B) \to \Box A \to \Box B$;

- the axiom schema GL: $\Box(\Box A \to A) \to \Box A$;
- the necessitation rule NR: $\dfrac{A}{\Box A}\ \mathsf{NR}$,

where A, B are generic formulas (not simply atoms).

Then, here is the complete definition of the axiom system $\mathbb{GL}$. The set of axioms is encoded via the inductive predicate `GLaxiom`:

```
let GLaxiom_RULES,GLaxiom_INDUCT,GLaxiom_CASES = new_inductive_definition
  `(!p q. GLaxiom (p --> (q --> p))) /\
   (!p q r. GLaxiom ((p --> q --> r) --> (p --> q) --> (p --> r))) /\
   (!p. GLaxiom (((p --> False) --> False) --> p)) /\
   (!p q. GLaxiom ((p <-> q) --> p --> q)) /\
   (!p q. GLaxiom ((p <-> q) --> q --> p)) /\
   (!p q. GLaxiom ((p --> q) --> (q --> p) --> (p <-> q))) /\
   GLaxiom (True <-> False --> False) /\
   (!p. GLaxiom (Not p <-> p --> False)) /\
   (!p q. GLaxiom (p && q <-> (p --> q --> False) --> False)) /\
   (!p q. GLaxiom (p || q <-> Not(Not p && Not q))) /\
   (!p q. GLaxiom (Box (p --> q) --> Box p --> Box q)) /\
   (!p. GLaxiom (Box (Box p --> p) --> Box p))`;;
```

The judgment $\mathbb{GL} \vdash A$, denoted `|-- A` in the machine code (not to be confused with the symbol for HOL theorems `|-`), is also inductively defined in the expected way:

```
let GLproves_RULES,GLproves_INDUCT,GLproves_CASES = new_inductive_definition
  `(!p. GLaxiom p ==> |-- p) /\
   (!p q. |-- (p --> q) /\ |-- p ==> |-- q) /\
   (!p. |-- p ==> |-- (Box p))`;;
```

6.2.1. $\mathbb{GL}$-LEMMAS

As usual, $\mathbb{GL} \vdash A$ denotes the existence of a derivation of A from the axioms of $\mathbb{GL}$; we could also define a notion of derivability from a set of assumptions just by tweaking the previous definitions in order to handle the specific limitations on NR – so that the deduction theorem would hold [HN12] – but this would be inessential to our intents.

Proving some lemmas in the axiomatic calculus $\mathbb{GL}$ is a technical interlude necessary for obtaining the completeness result.

In accordance with this aim, we denoted the classical axioms and rules of the system as the propositional schemas used by Harrison in the file `Arithmetic/derived.ml` of the HOL Light standard distribution [Har22] – where, in fact, many of our lemmas relying only on the propositional calculus are already proven there w.r.t. an axiomatic system for first-order classical logic; our further lemmas involving modal reasoning are denoted by names that are commonly used in informal presentations.

Therefore, the code in `gl.ml` mainly consists of the formalised proofs of those lemmas *in* $\mathbb{GL}$ that are useful for the formalised results we present in the next section. This file might be thought of as a "kernel" for further experiments in reasoning about axiomatic calculi by using HOL Light. The lemmas we proved are, indeed, standard tautologies of classical propositional logic, along with specific theorems of minimal modal logic and its extension for transitive frames – i.e. of the systems $\mathbb{K}$ and $\mathbb{K}4$ [Pop94] –, so that by applying minor changes in basic definitions, they are – so to speak – take-away proof-terms for extensions of that very minimal system within the realm of normal modal logics.

More precisely, we have given, whenever it was useful, a "sequent-style natural deduction" characterization of classical operators both in terms of an implicit (or internal) deduction – and in that case we named the lemma with the suffix `_th` –, such as

```
GL_modusponens_th
  |- !p q. |-- ((p --> q) && p --> q)
```

and as a derived rule of the axiomatic system mimicking the behaviour of the connective in Gentzen's formalism, e.g.,

```
GL_and_elim
  |- !p q r. |-- (r --> p && q) ==> |-- (r --> q)  /\ |-- (r --> p)
```

We had to prove about 120 such results of varying degrees of difficulty. We believe that this file is well worth the effort of its development, for two main reasons to be considered – along with the just mentioned fact that they provide a (not so) minimal set of internal lemmas which can be moved to different axiomatic calculi at, basically, no cost.

Indeed, on the one hand, these lemmas simplify the subsequent formal proofs involving consistent lists of formulas since they let us work formally within the scope of $\vdash$, so that we can rearrange subgoals according to their most useful equivalent form by applying the appropriate $\mathbb{GL}$-lemma(s).

On the other hand, the endeavour of giving formal proofs of these lemmas of the calculus $\mathbb{GL}$ has been important for checking how much our proof-assistant is "friendly" and efficient in performing this specific task.

As it is known, any axiomatic system fits very well an investigation involving the notion of *theoremhood* for a specific logic, but its lack of naturalness w.r.t. the practice of developing informal proofs makes it an unsatisfactory model for the notion of *deducibility*. In more practical terms: Developing a formal proof of a theorem in an axiomatic system *by pencil and paper* can be a dull and uninformative task.

We therefore left the proof search to the HOL Light toolbox as much as possible. Unfortunately, we have to express mixed feelings about the general experience. In most cases, relying on the automation tools of this specific proof assistant did save our time and resources when trying to give a formal proof in $\mathbb{GL}$. Nevertheless, there has been a number of $\mathbb{GL}$-lemmas for proving which those automation tools did not reveal useful at all. In those cases, actually, we had to perform a tentative search of the specific instances of axioms from which deriving the lemmas,[2] so that interactive proving them had advantages as well as traditional instruments of everyday mathematicians.

Just to stress the general point: It is clearly possible – and actually useful in general – to rely on the resources of HOL Light to develop formal proofs both *about* and *within* an axiomatic calculus for a specific logic, in particular when the lemmas of the object system have relevance or practical utility for mechanizing (meta-)results on it; however, these very resources – and, as far as we can see, the tools of any other general proof assistant – do not look peculiarly satisfactory for pursuing investigations on derivability within axiomatic systems.

6.2.2. Soundness lemma

At this point, we can prove that $\mathbb{GL}$ is sound – i.e. every formula derivable in the calculus is a tautology in the class of irreflexive transitive finite frames. This is obtained by simply unfolding the relevant definitions and applying theorems `TRANSNT_EQ_LOB` and `ITF_NT` of Section 6.1.2:

```
GL_TRANSNT_VALID
  |- !p. (|-- p) ==> TRANSNT:(W->bool)#(W->W->bool)->bool |= p

GL_ITF_VALID
  |- !p. |-- p ==> ITF:(W->bool)#(W->W->bool)->bool |= p
```

From this, we get a model-theoretic proof of consistency for the calculus

```
GL_consistent
  ~ |-- False
```

Having exhausted the contents of file `gl.ml`, we shall move to consider the the mechanized proof of completeness for the calculus w.r.t. this very same class of frames.

6.3. Modal completeness

When dealing with normal modal logics, it is common to develop a proof of completeness w.r.t. relational semantics by using the so-called ‘canonical model method’. We sketched the main lines of that procedure in Section 1.1.4. For $\mathbb{GL}$, we cannot directly pursue this strategy, since the logic is not compact: Maximal consistent sets are (in general) infinite objects, though the notion of derivability involves only a finite set of formulas. We cannot therefore reduce the semantic notion of (in)coherent set of formulas to the syntactic one of (in)consistent set of formulas: When extending a consistent set of formulas

[2] The HOL Light tactics for first-order reasoning `MESON` and `METIS` were unable, for example, to instantiate autonomously the obvious middle formula for the transitivity of an implication, or even the specific formulas of a schema to apply to the goal in order to rewrite it.

to a maximal consistent one, we might end up with a *syntactically* consistent set that nevertheless cannot be *semantically* satisfied.

In spite of this, it is possible to achieve a completeness result by

1. identifying the relevant properties of maximal consistent sets of formulas; and

2. tweaking the definitions so that those properties hold for specific consistent sets of formulas related to the formula we want to find a countermodel to.

That is, basically, the key idea behind the proof given in [Boo95, Ch. 5]. In that monograph, however, the construction of a maximal consistent set from a simply consistent one is only proof-sketched and relies on a syntactic manipulation of formulas. By using HOL Light we do succeed in giving a detailed proof of completeness as direct as that by Boolos. But, as a matter of fact, we can claim something more: We can do that by carrying out in a *very natural way* a tweaked Lindenbaum construction to extend consistent *lists* to maximal consistent ones. This way, we succeed in preserving the standard Henkin-style completeness proofs; and, at the same time, we avoid the symbolic subtleties sketched in [Boo95] that have no real relevance for the argument, but have the unpleasant consequence of making the formalised proof unnecessarily long, so that the implementation would sound rather pedantic, or even dull.

Furthermore, the proof of the main lemma `EXTEND_MAXIMAL_CONSISTENT` is rather general and does not rely on any specific property of $\mathbb{GL}$: Our strategy suits all the other normal (mono)modal logics – we only need to modify the subsequent definition of `STANDARD_RELATION` according to the specific system under consideration. Thus, we provide a way for establishing completeness à la Henkin *and* the finite model property without recurring to filtrations of canonical models for those systems.

6.3.1. Maximal consistent lists

Following the standard practice, we need to consider consistent finite sets of formulas for our proof of completeness. In principle, we can employ general sets of formulas in the formalisation, but, from the practical viewpoint, lists without repetitions are better suited, since they are automatically finite and we can easily manipulate them by structural recursion. We define first the operation of finite conjunction of formulas in a list:[3]

[3] Notice that in this definition we perform a case analysis where the singleton list is treated separately (i.e., we have `CONJLIST [p] = p`). This is slightly uncomfortable in certain formal proof steps: In retrospect, we might have used a simpler version of this function. However, since this is a minor detail, we preferred not to change our code.

```
let CONJLIST = new_recursive_definition list_RECURSION
  `CONJLIST [] = True /\
   (!p X. CONJLIST (CONS p X) = if X = [] then p else p && CONJLIST X)`;;
```

We proceed by proving some properties on lists of formulas and some $\mathbb{GL}$-lemmas involving `CONJLIST`. In particular, since $\mathbb{GL}$ is a normal modal logic – i.e. its modal operator distributes over implication and preserves theoremhood – we have that our $\Box$ distributes over the conjunction of X so that we have `CONJLIST_MAP_BOX`:

$$\mathbb{GL} \vdash \Box \bigwedge X \leftrightarrow \bigwedge \Box X,$$

where $\Box X$ is an abuse of notation for the list obtained by "boxing" each formula in X.

We are now able to define the notion of **consistent list of formulas** and prove the main properties of this kind of objects:

```
let CONSISTENT = new_definition
  `CONSISTENT (l:form list) <=> ~ (|-- (Not (CONJLIST l)))`;;
```

In particular, we prove that:

- a consistent list cannot contain both A and $\neg A$ for any formula A, nor $\bot$ (see theorems `CONSISTENT_LEMMA`, `CONSISTENT_NC`, and `FALSE_IMP_NOT_CONSISTENT`, respectively);
- for any consistent list X and formula A, either $X + A$ is consistent, or $X + \neg A$ is consistent (`CONSISTENT_EM`), where $+$ denotes the usual operation of appending an element to a list.

Our **maximal consistent lists** w.r.t. a given formula A will be consistent lists that do not contain repetitions and that contain, for any subformula of A, that very subformula or its negation:[4]

```
let MAXIMAL_CONSISTENT = new_definition
  `MAXIMAL_CONSISTENT p X <=>
   CONSISTENT X /\ NOREPETITION X /\
   (!q. q SUBFORMULA p ==> MEM q X \/ MEM (Not q) X)`;;
```

where `X` is a list of formulas and `MEM q X` is the membership relation for lists. We then establish the main closure property of maximal consistent lists:

```
MAXIMAL_CONSISTENT_LEMMA
  |- !p X A b. MAXIMAL_CONSISTENT p X /\
               (!q. MEM q A ==> MEM q X) /\
               b SUBFORMULA p /\
               |-- (CONJLIST A --> b)
               ==> MEM b X
```

[4] Here we define the set of subformulas of A as the reflexive transitive closure of the set of formulas on which the main connective of A operates: This way, the definition is simplified and it is easier to establish standard properties of the set of subformulas by means of general higher-order lemmas in HOL Light for the closure of a given relation.

After proving some further lemmas with practical utility – in particular, the fact that any maximal consistent list behaves like a restricted bivalent evaluation for classical connectives (MAXIMAL_CONSISTENT_MEM_NOT and MAXIMAL_CONSISTENT_MEM_CASES) – we can finally define the ideal (type of counter)model we are interested in. The type STANDARD_MODEL consists, for a given formula p, of:

1. the set of maximal consistent lists w.r.t. p made of *subsentences* of p – i.e. its subformulas or their negations – as possible worlds;
2. an irreflexive transitive accessibility relation R such that for any subformula Box q of p and any world w, Box q is in w iff, for any x R-accessible from w, q is in x;
3. an atomic evaluation that gives value T (true) to a in w iff a is a subformula of p.

After defining the relation of *subsentences* as

```
let SUBSENTENCE = new_definition
  `!p q. q SUBSENTENCE p <=>
         q SUBFORMULA p \/ (?q'. q = Not q' /\ q' SUBFORMULA p)`;;
```

we can introduce the corresponding code:

```
let GL_STANDARD_FRAME = new_definition
  `GL_STANDARD_FRAME p (W,R) <=>
   W = {w | MAXIMAL_CONSISTENT p w /\ (!q. MEM q w ==> q SUBSENTENCE p)} /\
   ITF (W,R) /\
   (!q w. Box q SUBFORMULA p /\ w IN W
          ==> (MEM (Box q) w <=> !x. R w x ==> MEM q x))`;;

let GL_STANDARD_MODEL = new_definition
  `GL_STANDARD_MODEL p (W,R) V <=>
   GL_STANDARD_FRAME p (W,R) /\
   (!a w. w IN W ==> (V a w <=> MEM (Atom a) w /\ Atom a SUBFORMULA p))`;;
```

6.3.2. Maximal extensions

What we have to do now is to show that the type GL_STANDARD_MODEL is non-empty. We achieve this by constructing suitable maximal consistent lists of formulas from specific consistent ones.

Our original strategy differs from the presentation given in e.g. [Boo95] for being closer to the standard Lindenbaum construction commonly used in proving completeness results. By doing so, we have been able to circumvent both the pure technicalities in formalizing the combinatorial argument sketched in [Boo95, p.79] *and* the problem – apparently inherent to the Lindenbaum extension – due to the non-compactness of the system, as we mentioned before.

The main lemma states then that, from any consistent list X of subsentences of a formula A, we can construct a maximal consistent list of subsentences of A by extending (if necessary) X:

```
EXTEND_MAXIMAL_CONSISTENT
  |- !p X.
       CONSISTENT X /\
       (!q. MEM q X ==> q SUBSENTENCE p)
       ==> ?M. MAXIMAL_CONSISTENT p M /\
               (!q. MEM q M ==> q SUBSENTENCE p) /\
               X SUBLIST M
```

The proof-sketch is basically that one described in Section 1.1.4: Given a formula A, we proceed in a step-by-step construction by iterating over the subformulas B of A not contained in X. At each step, we append to the list X the subformula B – if the resulting list is consistent – or its negation $\neg B$ – otherwise.

This way, we are in the pleasant condition of carrying out the construction by using the HOL Light device efficiently, and, at the same time, we do not have to worry about the non-compactness of $\mathbb{GL}$, since we are working with finite objects – the type `list` – from the very beginning.

Henceforth, we see that – under the assumption that A is not a $\mathbb{GL}$-lemma – the set of possible worlds in `STANDARD_FRAME` w.r.t. A is non-empty, as required by the definition of relational structures:

```
NONEMPTY_MAXIMAL_CONSISTENT
  |- !p. ~ |-- p
         ==> ?M. MAXIMAL_CONSISTENT p M /\
                 MEM (Not p) M /\
                 (!q. MEM q M ==> q SUBSENTENCE p)
```

Next, we have to define an R satisfying the condition 2 for a `STANDARD_FRAME`; the following does the job:

```
let GL_STANDARD_REL = new_definition
  `GL_STANDARD_REL p w x <=>
   MAXIMAL_CONSISTENT p w /\
   (!q. MEM q w ==> q SUBSENTENCE p) /\
   MAXIMAL_CONSISTENT p x /\
   (!q. MEM q x ==> q SUBSENTENCE p) /\
   (!B. MEM (Box B) w ==> MEM (Box B) x /\ MEM B x) /\
   (?E. MEM (Box E) x /\ MEM (Not (Box E)) w)`;;
```

Such an accessibility relation, together with the set of the specific maximal consistent lists we are dealing with, defines a structure in `ITF` with the required properties:

```
ITF_MAXIMAL_CONSISTENT
  |- !p. ~ |-- p
         ==> ITF ({M | MAXIMAL_CONSISTENT p M /\
                       (!q. MEM q M ==> q SUBSENTENCE p)},
                  GL_STANDARD_REL p),

ACCESSIBILITY_LEMMA
  |- !p M w q.
       ~ |-- p /\
       MAXIMAL_CONSISTENT p M /\
       (!q. MEM q M ==> q SUBSENTENCE p) /\
          MAXIMAL_CONSISTENT p w /\
          (!q. MEM q w ==> q SUBSENTENCE p) /\
          MEM (Not p) M /\
          Box q SUBFORMULA p /\
          (!x. GL_STANDARD_REL p w x ==> MEM q x)
          ==> MEM (Box q) w,
```

6.3.3. Truth lemma and completeness

For our ideal model, it remains to reduce the semantic relation of forcing to the more tractable one of membership to the specific world. More formally, we prove – by induction on the complexity of the subformula B of A – that if $\mathbb{GL} \nvdash A$, then for any world w of the standard model, B holds in w iff B is member of w:

```
  GL_truth_lemma
    |- !W R p V q.
          ~ |-- p /\
          GL_STANDARD_MODEL p (W,R) V /\
          q SUBFORMULA p
          ==> !w. w IN W ==> (MEM q w <=> holds (W,R) V q w),
```

Finally, we are able to prove the main result: If $\mathbb{GL} \nvdash A$, then the list $[\neg A]$ is consistent, and by applying `EXTEND_MAXIMAL_CONSISTENT`, we obtain a maximal consistent list X w.r.t. A that extends it, so that, by applying `GL_truth_lemma`, we have that $X \nVdash A$ in our standard model. The corresponding formal proof reduces to the application of those previous results and the appropriate instantiations:

```
  COMPLETENESS_THEOREM
    |- !p. ITF:(form list->bool)#(form list->form list->bool)->bool |= p
            ==> |-- p,
```

Notice that the family of frames `ITF` is polymorphic, but, at this stage, our result holds only for frames on the domain `form list`, as indicated by the type annotation. This is not an intrinsic limitation: The next section is devoted indeed to generalise this theorem to frames on an arbitrary domain.

6.3.4. Generalizing via bisimulation

As we stated before, our theorem `COMPLETENESS_THEOREM` provides the modal completeness for $\mathbb{GL}$ with respect to a semantics defined using models built on the type `:form list`. It is obvious that the same result must hold whenever we consider models built on any infinite type. To obtain a formal proof of this fact, we need to establish a *correspondence* between models built on different types. It is well-known that a good way to make rigorous such a correspondence is by means of the notion of *bisimulation* recalled in Section 1.1.4.

In our context, given two models `(W1,R1)` and `(W2,R2)` sitting respectively on types `:A` and `:B`, each with an evaluation function `V1` and `V2`, a bisimulation is a binary relation `Z:A->B->bool` that relates two worlds `w1:A` and `w2:B` when they can *simulate* each other. The formal definition is as follows:

```
BISIMIMULATION
  |- BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z <=>
     (!w1:A w2:B.
        Z w1 w2
        ==> w1 IN W1 /\ w2 IN W2 /\
            (!a:string. V1 a w1 <=> V2 a w2) /\
            (!w1'. R1 w1 w1'  ==> ?w2'. w2' IN W2 /\ Z w1' w2' /\ R2 w2 w2') /\
            (!w2'. R2 w2 w2' ==> ?w1'. w1' IN W1 /\ Z w1' w2' /\ R1 w1 w1'))
```

Then, we say that two worlds are *bisimilar* if there exists a bisimulation between them:

```
let BISIMILAR = new_definition
  `BISIMILAR (W1,R1,V1) (W2,R2,V2) (w1:A) (w2:B) <=>
   ?Z. BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z /\ Z w1 w2`;;
```

The key fact is that the semantic predicate `holds` respects bisimilarity:

```
BISIMILAR_HOLDS
  |- !W1 R1 V1 W2 R2 V2 w1:A w2:B.
       BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2
       ==> (!p. holds (W1,R1) V1 p w1 <=> holds (W2,R2) V2 p w2)
```

From this, we can prove that validity is preserved by bisimilarity. The precise statements are the following:

```
BISIMILAR_HOLDS_IN
  |- !W1 R1 W2 R2.
       (!V1 w1:A. ?V2 w2:B. BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2)
       ==> (!p. holds_in (W2,R2) p ==> holds_in (W1,R1) p)

BISIMILAR_VALID
  |- !L1 L2.
       (!W1 R1 V1 w1:A.
          L1 (W1,R1) /\ w1 IN W1
          ==> ?W2 R2 V2 w2:B. L2 (W2,R2) /\
                              BISIMILAR (W1,R1,V1) (W2,R2,V2) w1 w2)
       ==> (!p. L2 |= p ==> L1 |= p)
```

In the last theorem, recall that the statement `L(W,R)` means that `(W R)` is a frame in the class of frames `L`.

Finally, we can explicitly define a bisimulation between `ITF`-models on the type `:form list` and on any infinite type `:A`. From this, it follows at once the desired generalization of completeness for $\mathbb{GL}$:

```
COMPLETENESS_THEOREM_GEN
  |- !p. INFINITE (:A) /\ ITF:(A->bool)#(A->A->bool)->bool |= p ==> |-- p
```

Furthermore, from the proof that the relation

```
\w1 w2. MAXIMAL_CONSISTENT p w1 /\ (!q. MEM q w1 ==> q SUBSENTENCE p) /\
        w2 IN GL_STDWORLDS p /\
        set_of_list w1 = w2
```

defines a bisimulation between the ITF-standard model based on maximal consistent *lists* of formulae and the model based on corresponding *sets* of formulas we obtain the traditional version of modal completeness, corresponding to theorem GL_COUNTERMODEL_FINITE_SETS in our code.

6.4. Implementing G3KGL

By using our EXTEND_MAXIMAL_CONSISTENT lemma, we succeeded in giving a rather neat proof of both completeness and the finite model property for $\mathbb{GL}$.

As an immediate corollary, we have that the system $\mathbb{GL}$ is decidable and, in principle, we could implement a decision procedure for it in OCaml. A naive approach would proceed as follows.

We define the tactic NAIVE_GL_TAC and its associated rule NAIVE_GL_RULE that perform the following steps:

1. Apply the completeness theorem;
2. Unfold some definitions;
3. Try to solve the resulting *semantic* problem using first-order reasoning.

Here is the corresponding code.

```
let NAIVE_GL_TAC : tactic =
  MATCH_MP_TAC COMPLETENESS_THEOREM THEN
  REWRITE_TAC[valid; FORALL_PAIR_THM; holds_in; holds;
              ITF; GSYM MEMBER_NOT_EMPTY] THEN
  MESON_TAC[];;

let NAIVE_GL_RULE tm = prove(tm, REPEAT GEN_TAC THEN GL_TAC);;
```

The above strategy is able to prove automatically some lemmas which are common to normal modal logics, but require some effort when derived in an axiomatic system. As an example, consider the following $\mathbb{GL}$-lemma:

```
GL_box_iff_th
  |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

When developing a proof of it within the axiomatic calculus, we need to “help” HOL Light by instantiating several further $\mathbb{GL}$-lemmas, so that the resulting proof-term consists of ten lines of code. On the contrary, our rule is able to check it in few steps:

```
# NAIVE_GL_RULE `!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`;;
  0..0..1..6..11..19..32..solved at 39
  0..0..1..6..11..19..32..solved at 39
val it : thm = |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

In spite of this, the automation offered by MESON tactic is unexpectedly disappointing for some lemmas. For instance, the previous procedure is not even able to prove the basic instance Gödel-Löb scheme

$$\mathbb{GL} \vdash \Box(\Box\bot \longrightarrow \bot) \longrightarrow \Box\bot\,.$$

This suggests that NAIVE_GL_TAC is based on a very inadequate strategy. And that is where structural proof theory comes at rescue.

6.4.1. The calculus G3KGL

The frame conditions characterising GL – i.e. Noetherianity and transitivity, or, equivalently, irreflexivity, transitivity and finiteness – cannot be expressed by a (co)geometric implication, for being finiteness and Noetherianity intrinsically second order.

Therefore it is not possible to define a labelled sequent calculus for GL by simply extending G3K with further semantic rules, as described in Section 1.3.

Nevertheless, it is still possible to internalise the possible world semantics by relying on a modified definition of the forcing relation – valid for ITF models – in which the standard condition for □ is substituted by the following:[5]

$$x \Vdash \Box A \qquad \text{iff} \qquad \text{for all } y, \text{ if } xRy \text{ and } y \Vdash \Box A, \text{ then } y \Vdash A.$$

In [Neg05], this suggests to modify the $R\Box$ rule according to the right-to-left direction of that forcing condition.

The resulting labelled sequent calculus G3KGL is summarised in Figure 6.1.

The rule $R\Box^{Löb}$ obeys to the side condition of not being y free in Γ, Δ, on the line with the universal quantifier used in the forcing condition.

In [Neg14b], it is proven that G3KGL has good structural properties. Moreover, it is easily shown that the proof search in this calculus is terminating, and its failure allows to construct a countermodel to the formula at the root of the derivation tree: It suffices to consider the top sequent of an open branch, and assume that all the labelled formulas and relational atoms in its antecedent are true, while all the labelled formulas in its consequent are false. This way, a syntactic proof of decidability for GL is obtained.

It is not hard to see how to use both our formalisation of modal completeness and the already known proof theory for G3KGL to the aim of implementing a decision algorithm in HOL Light for GL: Our predicate `holds (W,R) V A x` corresponds exactly to the labelled formula $x : A$.

Thus we have three different ways of expressing the fact that a world x forces A in a given model $\langle W, R, v\rangle$:

[5] This idea was exploited in the definition of the sequent calculi G3IL* in Chapter 5.

Initial sequents:

$$x : p, \Gamma \Rightarrow \Delta, x : p$$

Propositional rules:

$$\dfrac{}{x : \bot, \Gamma \Rightarrow \Delta}\ L\bot$$

$$\dfrac{x : A, x : B, \Gamma \Rightarrow \Delta}{x : A \wedge B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\wedge \qquad \dfrac{\Gamma \Rightarrow \Delta, x : A \qquad \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \wedge B}\ \mathcal{R}\wedge$$

$$\dfrac{x : A, \Gamma \Rightarrow \Delta \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \vee B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\vee \qquad \dfrac{\Gamma \Rightarrow \Delta, x : A, x : B}{\Gamma \Rightarrow \Delta, x : A \vee B}\ \mathcal{R}\vee$$

$$\dfrac{\Gamma \Rightarrow \Delta, x : A}{x : \neg A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\neg \qquad \dfrac{x : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \neg A}\ \mathcal{R}\neg$$

$$\dfrac{\Gamma \Rightarrow \Delta, x : A \qquad x : B, \Gamma \Rightarrow \Delta}{x : A \to B, \Gamma \Rightarrow \Delta}\ \mathcal{L}\!\to \qquad \dfrac{x : A, \Gamma \Rightarrow \Delta, x : B}{\Gamma \Rightarrow \Delta, x : A \to B}\ \mathcal{R}\!\to$$

Modal rules:

$$\dfrac{y : A, xRy, x : \Box A, \Gamma \Rightarrow \Delta}{xRy, x : \Box A, \Gamma \Rightarrow \Delta}\ \mathcal{L}\Box \qquad \dfrac{xRy, y : \Box A, \Gamma \Rightarrow \Delta, y : A}{\Gamma \Rightarrow \Delta, x : \Box A}\ \mathcal{R}\Box^{Löb}_{(y!)}$$

Semantic rules:

$$\dfrac{}{xRx, \Gamma \Rightarrow \Delta}\ Irref \qquad \dfrac{xRz, xRy, yRz, \Gamma \Rightarrow \Delta}{xRy, yRz, \Gamma \Rightarrow \Delta}\ Trans$$

Figure 6.1: Rules of the calculus G3KGL

SEMANTIC NOTATION	$x \Vdash A$
LABELLED SEQUENT CALCULUS NOTATION	$x : A$
HOL LIGHT NOTATION	`holds (W,R) V A x`

Let us call any expression of forcing in HOL Light notation a `holds`-proposition.

This suggests that a *deep* embedding of G3KGL is not necessary at all. Since internalising possible world semantics in sequent calculi is, in fact, a syntactic formalisation of that very semantics, we can use *our own* formalisation in HOL Light of validity in relational frames and *adapt* the goalstack mechanism of the theorem prover to develop G3KGL proofs by relying on that very mechanism.

That adaptation starts with a generalisation of the standard tactics for classical propositional logic which are already defined in HOL Light.

As an abstract deductive system, the logical engine underlying the proof development in HOL Light consists of a single-consequent sequent calculus for higher-order logic. We need to work on a multi-consequent sequent calculus made of multisets of `holds`-propositions and (formalised) relational atoms. To handle commas, we recur to the logical operators of HOL Light: A comma in the antecedent is formalised by the connective `/\`, while in the consequent it is formalised by the connective `\/`.

Since we cannot directly define multisets, we need to formalise the sequent calculus rules to operate on lists, and to handle permutations by means of standard conversions of a goal that has the general shape of an n-ary disjunction of `holds`-propositions, for $n \geq 0$.

The intermediate tactics we now need to define operate

- on the components of a general goal-term consisting of a finite disjunction of `holds`-propositions: These correspond to the labelled formulas that appear on the right-hand side of a sequent of G3KGL, so that some our tactics can behave as the appropriate right rules of that calculus;
- on the list of hypotheses of the goal-stack, which correspond to the labelled formulas and possible relational atoms that occur on the left-hand side of a sequent of G3KGL, so that some our tactics can behave as the appropriate left rules of that calculus.

In order to make the automation of the process easier, among the hypotheses of each goal-stack we make explicit the assumptions about the transitivity of the accessibility relation `(trans)`, the left-to-right direction of the standard forcing condition for the $\Box$ `(boxl1)`, `(boxl2)`, and the right-to-left direction of the forcing condition for the same modality in `ITF` models `(boxr1),(boxr2)`. This way, the formal counterparts of *Trans*, $\mathcal{L}\Box$ and $\mathcal{R}\Box^{Löb}$ can be properly executed by combining standard tactics and applying the results to those meta-hypotheses made explicit.

Our classical propositional tactics implement, for purely practical reasons, left and right rules for the (bi)implication free fragment of $\mathcal{L}$: This implies that the only intermediate tactics determining a branching in the derivation tree – i.e., the generation of subgoals in HOL Light goal-stack – are those corresponding to $\mathcal{L}\vee$ and $\mathcal{R}\wedge$. The translation of a formula of the labelled language, given as a goal, to its equivalent formula in the implemented fragment is produced automatically by means of the conversions for the negation and conjunctive normal forms already present in the theorem prover:[6]

[6] Here `HOLDS_NNFC_UNFOLD` is the formal theorem stating that the definition of the predicate `holds` can be rephrased, for purely propositional formulas, by using only conjunction, disjunction and negation as operators at the meta-level.

```
let HOLDS_NNFC_UNFOLD_CONV : conv =
    GEN_REWRITE_CONV TOP_DEPTH_CONV
      [HOLDS_NNFC_UNFOLD; OR_CLAUSES; AND_CLAUSES] THENC
    NNFC_CONV
```

A similar conversion is defined for the labelled formulas that appear among the hypotheses, i.e. on the left-hand side of our formal sequents.

Left rules for propositional connectives are handled by specific tactics: Each of them is defined by using HOL Light theorem tactic(al)s, which can be thought of as operators on a given goal taking theorems as input to apply a resulting tactic to the latter. For instance, the left rule for negation $\mathcal{L}\neg$ is defined by

```
let NEG_LEFT_TAC : thm_tactic =
    let pth = MESON [] ‘~P ==> (P \/ Q) ==> Q‘ in
    MATCH_MP_TAC o MATCH_MP pth
```

which uses the propositional tautology `~P ==> (P \/ Q) ==> Q` in an MP rule instantiated with a negated `holds`-proposition occurring among the hypotheses; then it adds the `holds`-proposition among the disjuncts of the new goal, as expected. $\mathcal{R}\neg$ is defined analogously by `NEG_RIGHT_TAC`.

On the contrary $\mathcal{L}\vee$ and $\mathcal{R}\wedge$ are implemented by combining theorem tactic(al)s based on the basic operators `CONJ_TAC` and `DISJ_CASES`.

Modal rules are handled by means of the hypotheses `boxr1`, `boxr2`, `boxl1` and `boxl2` that are made explicit in each goal stack. $\mathcal{L}\Box$ is implemented via theorem tactic(al)s instantiating `boxl1` or `boxl2` with the appropriate `holds`-proposition and relational atom; similarly, $\mathcal{R}\Box^{Löb}$ is obtained by instantiating `boxr1` or `boxr2`, to change the current goal term. The same approach works also for the rule *Trans*, implemented by a theorem tactical `ACC_TCL` that instantiates and applies the meta-hypothesis `trans` to each appropriate relational atom among the hypotheses of a current goal stack.

This basically completes the formalisation – or shallow embedding – of G3KGL in HOL Light.

6.4.2. Design of the proof search

Our efforts turn now to run in HOL Light an automated proof search w.r.t. the implementation of G3KGL we have sketched in the previous section. We can again rely on theorem tactic(al)s to build the main algorithm, but this time we need to define them recursively.

First, we have to apply recursively the left rules for propositional connectives, as well as the $\mathcal{L}\Box$ rule: This is made possible by the theorem tactic `HOLDS_TAC`. Furthermore, we need to saturate the sequents w.r.t. the latter modal rule, by considering all the possible relational atoms and applications of the rule for transitivity, and, eventually, further left rules: That is the job of the theorem tactic `SATURATE_ACC_TAC`.

After that, it is possible to proceed with the $\mathcal{R}\Box^{Löb}$: That is trigged by the BOX_RIGHT_TAC, which operates by applying (the implementation of) $\mathcal{R}\Box^{Löb}$ after SATURATE_ACC_TAC and HOLDS_TAC.

At this point, it is possible to optimise the application of BOX_RIGHT_TAC by applying the latter tactic after a "sorting tactic" SORT_BOX_TAC: That tactic performs a conversion of the goal term and orders it so that priority is given to negated holds-propositions, followed by those holds-propositions formalising the forcing of a boxed formula. Each of these types of holds-propositions are sorted furthermore as follows:

> holds WR V p w precedes holds WR V q w if p occurs free in q and q does not occur free in p; or if p is "less than" q w.r.t. the structural ordering of types provided by the OCaml module Pervasives.

The complete proof search is performed by the definitive tactic GL_TAC, from which we define the expected GL_RULE, defined as follows:

```
let GL_RIGHT_TAC : tactic =
    CONV_TAC (HOLDS_NNFC_UNFOLD_CONV) THEN
    PURE_ASM_REWRITE_TAC[AND_CLAUSES; OR_CLAUSES; NOT_CLAUSES] THEN
    CONV_TAC CNF_CONV THEN
    REPEAT CONJ_TAC THEN
    TRY (NEG_RIGHT_TAC HOLDS_TAC)

  let GL_STEP_TAC : tactic =
    (FIRST o map CHANGED_TAC)
      [GL_RIGHT_TAC;
       SORT_BOX_TAC THEN BOX_RIGHT_TAC]

  let INNER_GL_TAC : tactic = REPEAT GL_STEP_TAC

let GL_TAC : tactic =
  REPEAT GEN_TAC THEN REPEAT (CONV_TAC let_CONV) THEN REPEAT GEN_TAC THEN
  REWRITE_TAC[diam_DEF; dotbox_DEF] THEN MATCH_MP_TAC COMPLETENESS_NUM THEN
  REPEAT GEN_TAC THEN INTRO_TAC "trans boxr1 boxr2 boxl1 boxl2 w" THEN
  REPEAT GEN_TAC THEN Rule_gl.INNER_GL_TAC THEN GL_BUILD_COUNTERMODEL;;

let GL_RULE (tm:term) : thm =
  prove(tm,GL_TAC);;
```

Our tactic works as expected:

1. Given a formula A of $\mathcal{L}$, OCaml let-terms are rewritten together with definable modal operators, and the goal is set to |-- A;

2. A model $\langle W, R, v\rangle$ and a world $w \in W$ – where W sits on the type num – are introduced. The main goal is now holds (W,R) V A w;

3. Meta-hypotheses trans boxr1 boxr2 boxl1 boxl2 w are introduced to be able to handle modal and relational rules;[7]

[7] The meta-hypotheses w states that $w \in W$ so that we the labelling at the previous step is justified indeed.

4. All possible propositional rules are applied after unfolding the modified definition of the predicate holds given by HOLDS_NNFC_UNFOLD. This assures that, at each step of the proof search, the goal term is a finite conjunction of disjunctions of positive and negative holds-propositions. As usual, priority is given to non-branching rules, i.e. to those that do not generate subgoals. Furthermore, the hypothesis list is checked, and trans is applied whenever possible; the same holds for $\mathcal{L}\Box$, which is applied to any appropriate hypothesis after the tactic triggering trans. Each new goal term is reordered by SORT_BOX_TAC, which always precedes the implementation of $\mathcal{R}\Box^{Löb}$.

The procedure is repeated starting from step 2. The tactic ruling it is

```
FIRST o map CHANGED_TAC,
```

which triggers the correct non-failing tactic – corresponding to a specific step of the very procedure – in GL_STEP_TAC.

At each step, moreover, the following condition is checked by calling ASM_REWRITE_TAC:

Closing The same holds-proposition occurs both among the current hypotheses *and* the disjuncts of the (sub)goal; or holds (W,R) V False x occurs in the current hypothesis list for some label x.

This condition states that the current branch is closed, i.e. an initial sequent has been reached, or the sequent currently analysed has a labelled formula $x : \bot$ in the antecedent.

Termination of the proof search is assured by the results presented in [Neg14b]. Therefore, we have been authorised to conclude our GL_TAC by a FAIL_TAC that, when none of the steps 2–4 can be repeated during a proof search, our algorithm terminates, informing us that a countermodel to the input formula can be built.

That is exactly the job of our GL_BUILD_COUNTERMODEL tactic: It considers the goal state which the previous tactics of GL_TAC stopped at, collects all the hypotheses, discarding the meta-hypotheses, and negates all the disjuncts constituting the goal term. Again, by referring to the results in [NvP11, Neg14b], we know that this information suffices to the user to construct a relational countermodel for the formula A given as input to our theorem prover for GL.

6.4.3. Some examples

Because of its adequate arithmetical semantics, Gödel-Löb logic reveals an exceptionally simple instrument to study arithmetical phenomenon of self-reference, as well as Gödel's results concerning (in)completeness and (un)provability of consistency.

Recall from Section 1.2 that an arithmetical realisation $*$ in Peano arithmetic (PA)[8] of our modal language consists of a function commuting with propositional connectives and such that $(\Box A)^* := Bew(\ulcorner A^* \urcorner)$, where $Bew(x)$ is the formal provability predicate for PA.

Under this interpretation, we will read modal formulas as follows:

$\Box A$	A is provable in PA
$\neg\Box\neg A$	A is consistent with PA
$\neg\Box A$	A is unprovable in PA
$\Box\neg A$	A is refutable in PA
$(\Box A) \vee (\Box\neg A)$	A is decidable in PA
$(\neg\Box A) \wedge (\neg\Box\neg A)$	A is undecidable in PA
$\Box(A \leftrightarrow B)$	A and B are equivalent over PA
$\Box\bot$	PA is inconsistent
$\neg\Box\bot,\ \Diamond\top$	PA is consistent

We are now going to test our theorem prover on some modal formulas that have a meta-mathematical relevance.

FORMALISED GÖDEL'S SECOND INCOMPLETENESS THEOREM. In PA, the following is provable: If PA is consistent, it cannot prove its own consistency. The corresponding modal formula is

$$\neg\Box\bot \ \rightarrow \ \neg\Box\Diamond\top$$

Here is the running of our prover:

```
# let GL_second_incompleteness_theorem = GL_RULE
  ‘|-- (Not (Box False) --> Not (Box (Diam True)))‘;;

val GL_second_incompleteness_theorem : thm =
  |- |-- (Not Box False --> Not Box Diam True)
```

UNDECIDABILITY OF CONSISTENCY. If PA is does not prove its inconsistency, then its consistency is undecidable. The corresponding modal formula is

$$\neg(\Box\Box\bot) \rightarrow \neg(\Box\neg\Box\bot) \wedge \neg(\Box\neg\neg\Box\bot)$$

Here is the running of our prover:

```
# let GL_PA_undecidability_of_consistency = GL_RULE
  ‘|-- ( Not (Box (Box False))
        --> Not (Box (Not (Box False))) &&
            Not (Box (Not (Not (Box False)))) )‘;;

val GL_PA_undecidability_of_consistency : thm =
  |- |--
     (Not Box Box False -->
      Not Box Not Box False && Not Box Not Not Box False)
```

[8] Actually, we could consider any Σ_1-sound arithmetical theory T extending IΣ_1 [Ver17].

UNDECIDABILITY OF GÖDEL’S FORMULA. The formula stating its own unprovability is undecidable in PA, if the latter does not prove its inconsistency. The corresponding modal formula is

$$\Box(A \leftrightarrow \neg\Box A) \wedge \neg\Box\Box\bot \to \neg\Box A \wedge \neg\Box\neg A$$

Here is the running of our prover:

```
# let GL_undecidability_of_Godels_formula = GL_RULE
  ‘!p. |-- ( Box (p <-> Not (Box p)) && Not (Box (Box False))
            --> Not (Box p) && Not (Box (Not p)) )‘;;

val GL_undecidability_of_Godels_formula : thm =
  |- !p. |--
         (Box (p <-> Not Box p) && Not Box Box False -->
          Not Box p && Not Box Not p)
```

CONSTRUCTION OF A COUNTERMODEL. This is an example of countermodel construction from a failed proof search of the following reflection principle:

$$\Box(\Box p \vee \Box\neg p) \to (\Box p \vee \neg\Box\neg p)$$

Since such a formula is not a theorem of $\mathbb{GL}$, we know that there exists an arithmetical sentence ϕ such that it is consistent with PA that both ϕ is undecidable and it is provable that ϕ is decidable.

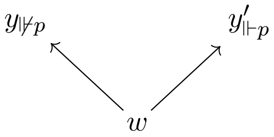
Here is the interactive running of our prover:

```
# g ‘!p . |-- (Box (Box p || Box (Not p)) --> (Box p || Box (Not p)))‘;;
val it : goalstack = 1 subgoal (1 total)

‘!p. |-- (Box (Box p || Box Not p) --> Box p || Box Not p)‘

# e GL_TAC;;
Exception: Failure "Contermodel stored in reference the_gl_countermodel.".
```

The tactic notifies that a countermodel as been stored in the memory: All we have to do in order to read it is to run

```
# !the_gl_countermodel;;
val it : term =
  ‘holds (W,R) V p y’ /\
   holds (W,R) V (Box Not p) y’ /\
   R w y’ /\
   holds (W,R) V (Box p) y’ /\
   y’ IN W /\
   R w y /\
   holds (W,R) V (Box p) y /\
   y IN W /\
   holds (W,R) V (Box (Box p || Box Not p)) w /\
   w IN W /\
   ~holds (W,R) V p y‘
```

As expected, the structure that the countermodel constructor returns can be graphically rendered as



Related work

Our formalisation gives a mechanical proof of completeness for $\mathbb{GL}$ in HOL Light which sticks to original Henkin's method for classical logic. In its standard version, its nature is synthetic and intrinsically semantic [FM12], and, as we stated before, it is the core of the canonical model construction for most of normal modal logics.

That very approach does not work for $\mathbb{GL}$. Nevertheless, the modified extension lemma we proved in our mechanization introduces an analytic flavour to the strategy – for building maximal consistent lists in terms of components of a given non-provable formula in the calculus – and shows that Henkin's idea can be applied to $\mathbb{GL}$ too modulo appropriate changes.

As far as we know, *no other mechanized proof of modal completeness for* $\mathbb{GL}$ has been given before, despite there exist formalisations of similar results for several other logics, manly propositional and first-order classical and intuitionistic logic.

Formal proof of semantic completeness for *classical logic* has defined an established trend in interactive theorem proving since [Sha85], where a Hintikka-style strategy is used to define a theoremhood checker for formulas built up by negation and disjunction only.

In fact, a very general treatment of systems for classical propositional logic is given in [MN18]. There, an axiomatic calculus is investigated along with natural deduction, sequent calculus, and resolution system in Isabelle/HOL, and completeness is proven by Hintikka-style method for sequent calculus first, to be lifted then to the other formalisms by means of translations of each system into the others. Their formalisation is more ambitious than ours, but, at the same time, it is focused on a very different aim. A similar overview of meta-theoretical results for several calculi formalised in Isabelle/HOL is given in [Bla19], where, again, a more general investigation – unrelated to modal logics – is provided.

Concerning the area of intuitionistic modalities, [Bak17] gives a constructive proof of completeness of IS4 w.r.t. a specific relational semantics verified in Agda, but it uses natural deduction and applies modal completeness to obtain a normalization result for the terms of the associated λ-calculus.

A Henkin-style completeness proof for $\mathbb{S}5$ formalised in Lean is presented in [Ben19]. That work applies the standard method of canonical models – since $\mathbb{S}5$ is compact.

More recently, [XN20] used the HOL4 theorem prover for a general treatment of model theory of modal systems. For future work, it might be interesting to make use of the formalisation therein along with the main lines of our implementation of axiomatic calculi to merge the two presentations – syntactic and semantic – in an exhaustive way.

Our formalisation, however, has been led by the aim of developing a (prototypical) theorem prover in HOL Light for normal modal logics. The results concerning GL that we have presented here can be thought of as a case study of our original underlying methodology.

Automated deduction for modal logic has become a relevant scientific activity in recent years, and exhaustive comparison of our prover with other implementations of modal systems is beyond our purposes. In spite of this, we care to mention at least three different development lines on that trend.

The work of [GK21] consists of an extremely efficient hybridism of SAT-solvers, modal clause-learning, and tableaux methods for modal systems. That prover deals with minimal modal logic K, and its extensions T and S4. The current version of their implementation does not produce a proof, nor a countermodel, for the input formula; however, the code is publicly available, and minor tweaks should make it do so.

The conference paper [GS20] presents a theorem prover for intuitionistic modal logics implementing proof search for Tait-style nested sequent calculi in Prolog. Because of the structural properties of those calculi, that prover returns, for each input formula, a proof in the appropriate calculus, or a countermodel extracted from the failed proof search in the system. Similar remarks could be formulated for the implementation described in [GLO$^+$22] concerning several logics for counterfactuals.

The latter formalisations are just two examples of an established modus operandi in implementing proof search in extended sequent calculi for non-classical logics by using the mere depth-first search mechanism of Prolog. Other instances of that line are e.g. [AOP10, GGOP05, GGP07], [OP03, OP05, OP08, OP14], and [DNOP21]. None of those provers deals explicitly with GL, but that development approach would find no issue in formalising G3KGL too.

Notice, in any case, that adequacy (soundness and completeness) of all those implementations rely on results that are *informally* proven at the meta-level. Our theorem prover for GL, on the contrary, is *certified* to be sound and complete by our *theorems* in HOL Light `GL_ITF_VALID` and `COMPLETENESS_THEOREM_GEN`, whose correctness, in turn, depends on the LCF approach. Nothing, in principle, prevents the HOL Light user from implementing certified theorem provers for the logics of the Prolog-based development line by proving the appropriate adequacy theorems for the logic under investigation. That proof search is well-behaved would rest, in those cases as well as ours, on the properties of the general proof techniques of HOL Light.

Similar remarks could be made about [PF21], where a general framework for object-level reasoning with multiset-based sequent calculi in HOL Light is proposed. They present a *deep* embedding of those systems by defining in HOL Light an appropriate derivability relation between multisets and encode two G1 calculi – namely, a fragment of propositional intuitionistic logic and its Curry-Howard analogous type theory. Specific tactics are then defined in order to perform an interactive proof search of a given sequent. For our purposes, it might be interesting to check whether their implementation may be of some help to enhance the performance and functionality of our prototypical theorem prover.[9]

Indeed, moving from the experiment about GL proposed in the present work, we plan to develop in future a more general (and more refined) mechanism – still based on the methodology discussed here – to deal with (ideally) the whole cube of normal modal logic within HOL Light. An immediate step in that direction would be to enhance the implementation of formal proofs in G3KGL, so that a positive answer to a given input formula would produce also a real derivation tree in the labelled sequent calculus of that very formula.

[9] One might say that the framework of [PF21] is similar to the works in Prolog for aiming a direct deep embedding of a sequent calculus, but it is also close to our implementation for adopting the LCF approach and for choosing HOL Light as environment.

7

Universal algebra in UniMath

This chapter consists of a report on an ongoing implementation of universal algebra within the formal environment of UniMath [VAG+22].

The leading motivation has been to provide a general framework for formalizing and studying algebraic structures as presented in the field of universal algebra [ARV10] *within* a proof assistant. By providing a formal system for isolating the invariants of the theory we are interested in, univalent mathematics has seemed to provide a suitable environment to carry out our endeavour since the very beginning. In particular, since it is natural to study algebraic structures up to isomorphism, univalent mathematics seems to be especially suited for this kind of task.

The choice of working within the UniMath environment has appeared natural, since it provides a minimalist implementation of univalent type theory. At the same time, the system comes with a large repository of mechanised results covering several fields of mathematics, so that it opens a wide range of possibilities for future development of our formalisation.

By the code surveyed here, the main notions concerning multi-sorted signatures are introduced. Developing them in a formal environment has required some expedients in the definitions of the basics, and, accordingly, of some subsequent constructions too. In particular, we had to introduce heterogeneous vectors and generalise types involving signatures by introducing (what we called) "sorted sets".

Having signatures, we then have given the related formalisation of the category of algebras using the notion of a displayed category [AL19] over the category of *sorted* hSets, whose univalence is proven by adapting the strategy used for the univalence of functor categories. The resulting construction is still a modular one and the resulting proof-term is more concise, for sure, than the one obtained by checking that algebras and homomorphisms satisfy the axioms for standard categories.

Defining terms is made complex by the fact that, by precise choice, UniMath does not make use of the theory of (Co)Inductive Constructions.[1] There-

[1] See Appendix B for a brief overview of the deductive system underlying UniMath, and further discussions on formalised univalent reasoning.

fore, we encode terms as lists of operation symbols, to be thought of as instructions for a stack-based machine. Terms are those lists of symbols that may be virtually executed without generating type errors or stack underflows.

Moreover, we prove that the term algebra over a signature is the initial object in the corresponding category, and that, more generally, an algebra of terms over a signature and a set of variables has the desired universal mapping property.

Our formalisation also includes the notion of equations and of algebras modelling an equation system associated with a signature; as for the category of algebras, we use the displayed category formalism to construct the univalent category of equational algebras over a given signature σ as the full subcategory of algebras over σ satisfying an equation system.

In the main body of the present chapter, we will discuss all the notions we introduced in our implementation, so that the overall material is structured as a presentation of the code, followed by the discussion of some examples, a quick anticipation of future work and a brief comparison with different formalisations of similar topics as coda. In details, we can summarise it as follows:

- In Section 7.1.1, we introduce the very basics of universal algebra along with some auxiliary definitions involving the new types we need to handle multi-sorted signatures, their algebras, and homomorphisms;

- In Sections 7.1.2, 7.1.3, and 7.1.4, we present the main details of our implementation of terms, prove that term algebras and free algebras do have the required universal property – stated as the contractibility of the type of out-going homomorphisms – and discuss the practical and methodological relevance of our induction principle on terms;

- In Section 7.1.5, we introduce systems of equations and equational algebras over a signature;

- In Section 7.1.6, we sketch the main lines of our constructions of the categories of algebras and equational algebras over a signature in terms of displayed categories over a base category of indexed hSets whose univalence is proven by a proof-strategy very close to that one adopted for functor categories;

- Finally, Section 7.2 is devoted to three applications of our implementation, namely: lists (Section 7.2.1), monoids (Section 7.2.2), and Tarski's semantics of propositional boolean formulas (Section 7.2.3).

GOALS AND METHODOLOGY. What we have mechanised is not, clearly, a mathematical novelty, but our endeavour has some payoffs.

Firstly, the code introduces in the UniMath library a minimal set of definitions and results that is open to the community of developers for future achievements and formal investigations on the relation between pre-categorical research in general algebraic structures and its subsequent development in e.g. Lawvere theories [ARV10].

Secondly, a peculiar feature of our code is the original implementation of term algebras over a signature, which fits within the original approach to mechanisation of univalent mathematics, but highlights the computational relevance of the constructions avoiding the restrictions imposed on the Coq engine in UniMath.

As a matter of fact, terms for a signature constitute a family inductively defined over the operation symbols together with an additional set of variables. As it is known, however, UniMath makes use of just *some* features of the Coq proof assistant. To be precise, both `record` and `inductive` types are avoided in order to keep the system sound from a foundational/philosophical viewpoint.

On the other hand, one of our main goals has been to make *all* our constructions about terms *evaluable* – as far as possible – by the built-in automation mechanisms of the proof assistant. More precisely, we represent each term using a sequence of function symbols. This sequence is thought to be executed by a stack machine: Each symbol of arity n pops n elements from the stack and pushes a new element at the top. A term is denoted by a sequence of function symbols that a stack-like machine can execute without type errors and stack underflow, returning a stack with a single element.

This approach led us to prove both a recursion and induction principle on terms that is evaluable as a functional term of the formal system. This is somehow mandatory when sticking to a form of (small scale) reflection: with our formalised stack-machine we have written in UniMath an implicit algorithm to compute terms over a signature; by means of our induction principle we can run it – so to speak – *within* the very formal system of UniMath, and use it to reason about terms in a safe way.

Moreover, our methodology sympathizes (in a sense) with what Henk Barendregt has called Poincaré principle: Our implementation of terms allows us to rely on the very core engine of UniMath when dealing with these formal objects, so that whenever we want to handle them we can focus on the real demonstrative contents of the formalisation, leaving to the automation behind the computer proof assistant the trivial computational steps involved in the very proof-term.

Generally speaking, we find standard categorical presentations, though perspicaciously elegant in their abstractness, lacking a certain suitability for computerized mathematics. By contrast, our goal is justified by a specific need of methodological coherence – we just sketched it few line above – when approaching a work in formalisation. Having proof-terms that the computational machinery of UniMath practically evaluates as a correctly typed function seems to fit the philosophy and aims of the mechanisation of mathematics better than just giving a formal counterpart of traditional mathematical notions that the computer cannot handle feasibly.

FORMALISATION. Definitions and results we discuss in the present work are labelled with their corresponding proof-term identifiers in the formalisation files. Notice that we make an extensive and exclusive use of the proof-as-programs paradigm here, so that e.g. existential statements are proven by inhabiting the corresponding Σ-type, and similarly for the other kinds of propositions.

To improve readability, in what follows, most proofs and technicalities are omitted, even though they are available in our repository. Our code is freely accessible from `https://github.com/amato-gianluca/UniMath`. The revision discussed in this chapter is tagged as `ITP2021`. That version has been already integrated in the official UniMath repository [VAG$^+$22]. In our own repository, the reader might find further experiments about the encoding of W-types according to our formalism.

The implementation discussed in the present chapter consists of the files in the directories

- `UniMath/Algebra/Universal_Algebra` for the basics of universal algebra (together with auxiliary definitions and results), and
- `UniMath/CategoryTheory/categories/Universal_Algebra` for the categories of algebras and equational algebras over a signature.

In order to help readers to browse our library, we summarise the dependencies between the files by the diagram in Figure 7.1 – where an arrow pointing to a node indicate the dependency of the target from the source.

WRITING AND REVISION NOTES. This chapter is based on the conference paper [AMPPB20] presented at Workshop on Homotopy Type Theory/Univalent Foundations (HoTT/UF) 2020, and the preprint [AMPB21]. Changes include an extended discussion of our constructions of displayed categories of (equational) algebras, and some minor local revisions.

7.1. SURVEYING THE CODE

In the following subsections we present and comment on the main constructions constituting our library. As anticipated, in the present work we want to make the code contained in the formalisation files easily readable, so that we privilege a certain clarity of exposition over its exhaustiveness. The interested reader can fill in the details by browsing the related files we mention during the discussion.

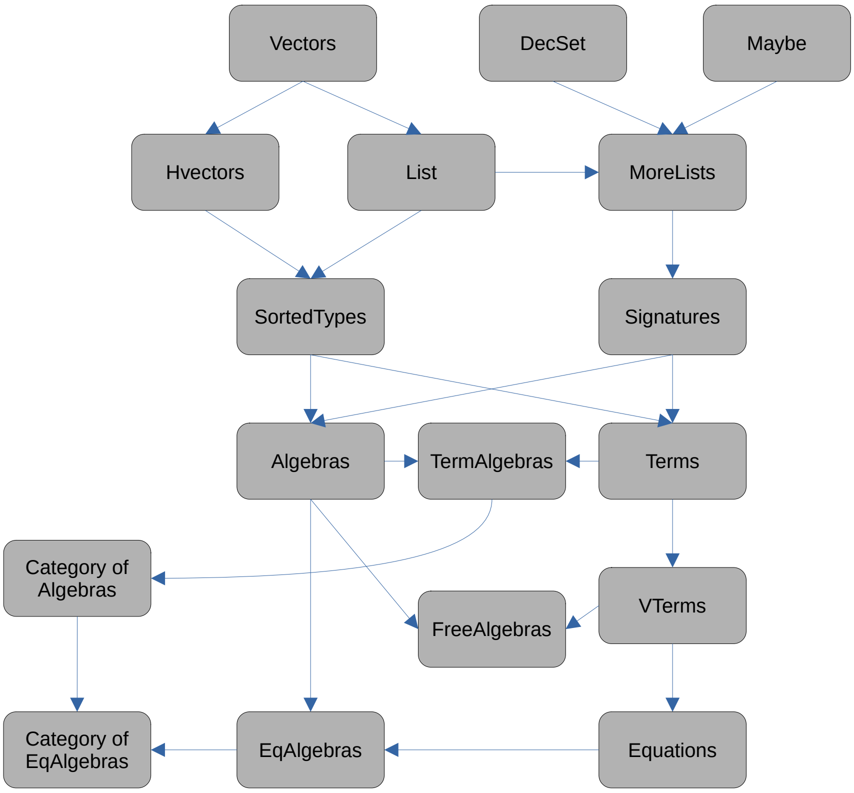


Figure 7.1: Intermodule dependencies of the universal algebra formalisation in UniMath.

7.1.1. SIGNATURES AND ALGEBRAS

We start by defining a **multi-sorted signature** to be made of a *decidable set* of sorts along with operations classified by arities and result sorts, as in standard practice

```
Definition signature : UU := ∑ (S: decSet) (O: hSet), O → list S × S.
```

The three natural projections associated to signatures are named `sort`, `names`, and `ar`

```
Definition sorts (σ: signature) := pr1 σ.
Definition names (σ: signature) := pr12 σ.
Definition ar    (σ: signature) := pr22 σ.
```

We also add specific projections for arities

```
Definition sort  {σ: signature} (nm: names σ) : sorts σ        := pr2 (ar σ nm).
Definition arity {σ: signature} (nm: names σ) : list (sorts σ) := pr1 (ar σ nm).
```

These simple definitions rely on two files in `UniMath/Combinatorics`, namely `Vectors.v` and `Lists.v`.

We changed these libraries in two aspects. First of all, we redefined lists in terms of the new datatype `vec` instead of using the ad-hoc type `iterprod` in the standard version of the file. Moreover, we changed a couple of theorems from opaque (`Qed.` conclusion) to transparent (`Defined.` conclusion). The latter changes are needed to make terms *compute* correctly.

Note that, in a signature, the set of sorts should be a `decSet`: This is a type whose equality is decidable, as defined in the file `DecSet.v`. We need this extra property because – as we previously stated – we want to *evaluate* terms in the UniMath engine: We can achieve that by pushing sorts into a stack, and we need to check that the very stack contains certain sequences of sorts before applying an operator symbol. Note also that a decSet also enjoys the defining property of an hSet. Operators are only required to be in `hSet`.

A signature may be alternatively specified through the type `signature_simple`. In a simple signature, the types for sorts and operation symbols are standard finite sets, and the map from operations symbols to domain and range is replaced by a list. In this way, the definition of a new signature is made simpler.

```
Definition signature_simple : UU := ∑ (ns: nat), list (list (⟦ ns ⟧) × ⟦ ns ⟧).

Definition make_signature_simple {ns: nat} (ar: list (list (⟦ ns ⟧) × ⟦ ns ⟧))
  : signature_simple := ns ,, ar.

Coercion signature_simple_compile (σ: signature_simple) : signature
  := make_signature (⟦ pr1 σ ⟧ ,, isdeceqstn _)
                    (stnset (length (pr2 σ))) (nth (pr2 σ)).
```

Single-sorted signatures are then defined as special cases of `signature_simple`.

```
Definition signature_simple_single_sorted : UU := list nat.
Definition make_signature_simple_single_sorted (ar: list nat) :
 signature_simple_single_sorted := ar.

Coercion signature_simple_single_sorted_compile
    (σ: signature_simple_single_sorted)
  : signature
  := make_signature_single_sorted (stnset (length σ)) (nth σ).
```

Moving to the file `Algebras.v`, we define an **algebra** over a given signature σ to be, as usual, support sets indexed by sorts together with operations with appropriate sorts:

```
Definition algebra (σ: signature): UU
  := ∑ A: shSet (sorts σ), ∏ nm: names σ, A⋆ (arity nm) → A (sort nm).

Definition supportset {σ: signature} (A: algebra σ) := pr1 A.

Definition support {σ: signature} (A: algebra σ): sUU (sorts σ) := pr1 A.

Definition ops {σ: signature} (A: algebra σ) := pr2 A.

Definition dom {σ: signature} (A: algebra σ) (nm: names σ): UU := A⋆ (arity nm).

Definition rng {σ: signature} (A: algebra σ) (nm: names σ): UU
  := support A (sort nm).
```

We declare the projections `supportset`, `support`, and `ops` as type coercions. Moreover, as for signatures, we simplify the building term for algebras when starting from a simple signature:

```
Definition make_algebra_simple
    (σ: signature_simple) (A: Vector hSet (pr1 σ))
    (ops: (λ a, (el A)⋆ (dirprod_pr1 a) → el A (dirprod_pr2 a))⋆ (pr2 σ))
  : algebra σ.
```

A similar proof-term (`make_algebra_simple_single_sorted`) is given for single-sorted signatures.

As the reader can see, these definitions rely on many different notions and notations. These are introduced in `MoreLists.v`, `SortedTypes.v` and `HVectors.v` files.

The file `MoreLists.v` contains notations for lists, such as `[v1; ...; vn]` for list literals and `::` for *cons*, together with additional properties which cannot be found in the standard library.

The type `hvec` in `HVectors.v` denotes heterogeneous vectors:[2] If `v` is a vector of types `U1`, `U2`, ..., `Un`, then `hvec v` is the product type `U1 × (U2 × ... × (Un × unit))`. We introduce several basic operations on heterogeneous vectors: Often they have the same syntax as the corresponding operations on plain vectors, and a name which begins with the prefix `h`. We also introduce notations for heterogeneous vectors, such as `[(v1; ...; vn)]` for a literal and `:::` for prefixing.

[2] We need this type to handle operations taking inputs of different sorts.

Sorted types are types indexed by elements of another type (the index type), so that an element of `sUU S` is an `S`-sorted type, i.e. an `S`-indexed family of types. Similarly, the type `shSet S` is an `S`-indexed family of hSets.

For functions, `X s→ Y` denotes the type of `S`-sorted mapping between `X` and `Y`, i.e. of `S`-indexed families of functions `X s → Y s`.

More prominently, for any `S`-sorted type `X`, its lifting to `list S` is denoted by `X⋆`, and is ruled by the identity `X [s1; s2; ...; sn] = [X s1 ; X s2 ; ... ; X sn]`. Accordingly, if `f` is an indexed mapping between `S`-indexed types `X` and `Y`, then `f⋆⋆` is the lifting of `f` to a `list S`-indexed mapping between `X⋆` and `Y⋆`. This operation ⋆⋆ is indeed functorial, and we prove that in a form which does not require function extensionality, since resorting to axioms would break computability of terms.

All of these notions allow us to define **algebra homomorphisms**:

```
Definition ishom {σ: signature} {A1 A2: algebra σ} (h: A1 s→ A2) : UU
  := ∏ (nm: names σ) (x: dom A1 nm), h _ (A1 nm x) = A2 nm (h⋆⋆ _ x).

Definition hom {σ: signature} (A1 A2: algebra σ): UU := ∑ (h: A1 s→ A2), ishom h.
```

As expected, the property of being an homomorphism belongs to `hProp`, so that the type `A1 ↷ A2` of homomorphisms between `A1` and `A2` is an `hSet`:

```
Theorem isapropishom {σ: signature} {A1 A2: algebra σ} (f: sfun A1 A2)
  : isaprop (ishom f).

Theorem isasethom {σ: signature} (A1 A2: algebra σ): isaset (A1 ↷ A2).
```

Next, we prove – by lemmas `ishomid` and `ishomcomp` – that the identity function determines an identity homomorphism, and that the property `ishom` is closed under composition.

7.1.2. Terms and free algebras

The file `Algebras.v` is closed by the construction of the unit algebra as the final algebra among those defined over a given signature:

```
Definition unitalgebra (σ: signature): algebra σ
  := make_algebra (sunitset (sorts σ)) tosunit.

Theorem iscontrhomstounit {σ: signature} (A: algebra σ)
  : iscontr (hom A (unitalgebra σ)).
```

However, we are mostly interested in the initial object of the category of algebras, namely the algebra of terms over a given signature. In standard textbooks, the set of terms over a signature σ and a (disjoint) set V of variables is defined as the least set including V and closed under application of symbols of σ.

For being inductive types unavailable in our formal system, we have developed a *peculiar device* to implement that notion.

In our formalisation we start with the special case where the set of variables V is empty. The rough and general idea can be sketched as follows:

1. A sequence of function symbols is thought of as a series of commands to be executed by a *stack machine* whose stack is made of sorts, and which we define by means of a maybe monad we construct from raw in `Monad.v`:

   ```
   Local Definition oplist (σ: signature):= list (names σ).
   Local Definition stack (σ: signature): UU := maybe (list (sorts σ)).
   ```

2. When an operation symbol is executed, its arity is popped out from the stack and replaced by its range. When a stack underflow occurs, or when the sorts present in the stack are not the ones expected by the operator, the stack goes into an error condition which is propagated by successive operations. We implement this process by means of two functions: `opexec` and `oplistexec`:

   ```
   Local Definition opexec (nm: names σ): stack σ → stack σ
     := flatmap (λ ss, just (sort nm :: ss)) ∘
        flatmap (λ ss, prefix_remove (arity nm) ss).

   Local Definition oplistexec (l: oplist σ): stack σ := foldr opexec (just []) l.
   ```

   The former is the stack transformation corresponding to the execution of the operation symbol `nm`. The latter returns the stack corresponding to the execution of the entire `oplist l` starting from the empty stack. The list is executed from the last to the first operation symbol.

   Several additional lemmas are required in order to make us able to handle stacks – by concatenating, splitting, etc. – without incurring failures breaking down the whole process, as defined in `Terms.v`.[3]

3. Finally, we define a term to be just a list of operation symbols that, after being executed by `oplistexec`, returns a list of length one with appropriate sort:[4]

   ```
   Local Definition isaterm (s: sorts σ) (l: oplist σ): UU
     := oplistexec l = just ([s]).


   Local Definition term (σ: signature) (s: sorts σ): UU
     := ∑ t: oplist σ, isaterm s t.
   ```

Terms may be built using the `build_term` constructor and decomposed trough the `princop` and `subterms` accessors:

[3] In particular, since we need to decide when a stack is correctly executed and when an underflow occurs, we see the reasons for choosing sorts to constitute a decidable set.

[4] From a purely HoTT-perspective, we can easily see also that the type of stacks over σ is an hSet, so that the property of being a term is not proof-relevant (`isapropisaterm`).

```
Local Definition build_term (nm: names σ) (v: (term σ)⋆ (arity nm)):
  term σ (sort nm).
Definition princop {s: sorts σ} (t: term σ s): names σ.
Definition subterms {s: sorts σ} (t: term σ s): (term σ)⋆.
```

The implementation function build_term is quite straightforward: It concatenates nm and the oplists underlying the terms in v, and builds a proof that the resulting oplist is a term from the proofs that the elements of v are terms. The princop and subterms accessors are projections of a more complex operation called term_decompose which breaks a term in principal operation symbols nm and subterms v, and, at the same time, provides the proof-terms that characterize their behaviour.

7.1.3. Induction on terms

At this point, we proceed in proving induction over terms. The inductive hypothesis, being quite complex, is stated in the term_ind_HP type.

```
Definition term_ind_HP (P: ∏ (s: sorts σ), term σ s → UU) : UU
  := ∏ (nm: names σ) (v: (term σ)⋆ (arity nm)) (IH: hvec (h1map_vector P v))
     , P (sort nm) (build_term nm v).
```

Given a family P of types, indexed by a sort s and a term over s, the inductive hypothesis is a function that, given an operation symbol nm, a sequence of terms v, and a sequence of proofs of P for all terms in v, is able to build a proof of P for the term build_term nm v, i.e. $nm(v_1, \ldots, v_n)$. The identifier h1map_vector simply denotes the variant of map for heterogeneous vectors. Given this auxiliary definition, the **induction principle for terms** may be easily stated as follows:

```
Theorem term_ind (P: ∏ (s: sorts σ), term σ s → UU) (R: term_ind_HP P)
                 {s: sorts σ} (t: term σ s)
  : P s t.
```

The proof proceeds by induction on the length of the oplist underlying t, using the term_ind_onlength auxiliary function.

Simple examples of use of the induction principle on terms are the depth and fromterm functions. The former computes the depth of a term, and the latter is essentially the evaluation map for ground terms in an algebra.[5]

```
Local Definition fromterm {A: sUU (sorts σ)}
                          (op : ∏ (nm : names σ), A⋆ (arity nm) → A (sort nm))
                          {s: sorts σ}
    : term σ s → A s
    := term_ind (λ s _, A s) (λ nm v rec, op nm (h2lower rec)).
```

[5] The h2lower proof-term which appears in the definition of fromterm is just a technicality needed to convert between types which are provably equal but not convertible. This might be replaced by a transport, if we were not interested in computability. The same can be said for the proof term h1lift, later in the definition of term_ind.

In order to reason effectively on inductive definitions, we need an induction unfolding property. For natural numbers, it is

```
nat_rect P a IH (S n) = IH n (nat_rect P a IH n),
```

which means that the result of applying the recursive definition to `S n` may be obtained by applying the recursive definition to `n` and then the inductive hypothesis. While this induction unfolding properties are provable just by `reflexivity` for many inductive types, this does not hold for terms, and a quite complex proof is needed:

```
Lemma term_ind_step (P: ∏ (s: sorts σ), term σ s → UU) (R: term_ind_HP P)
                    (nm: names σ) (v: (term σ)⋆ (arity nm))
  : term_ind P R (build_term nm v)
    = R nm v (h2map (λ s t q, term_ind P R t) (h1lift v)).
```

Notice that many of the definitions which appear in `Terms.v` are declared as `Local`. This is so because they are considered internal implementation details and should not be used unless explicitly needed. In particular, this holds for a set of identifiers that will be redefined in `VTerms.v` to work on terms with variables. Since sometimes it may be convenient to have specialized functions that only work with ground terms, they are exported through a series of notations, such as:

```
Notation gterm := term.
Notation build_gterm := build_term.
```

7.1.4. TERMS WITH VARIABLES AND FREE ALGEBRAS

Considering **terms with variables** is what we do in file `VTerms.v`. The idea is that a term with variables in V over a signature σ is a ground term in a new signature where constant symbols are enlarged with the variables in V. Variables and corresponding sorts are declared in a `varspec` (*variable specification*), while `vsignature` builds the new signature.

```
Definition varspec (σ: signature) := ∑ V: hSet, V → sorts σ.

Definition vsignature (σ : signature) (V: varspec σ): signature
  := make_signature (sorts σ) (setcoprod (names σ) V)
                              (sumofmaps (ar σ) (λ v, nil ,, varsort v)).
```

The proof-terms `namelift` and `varname` are the injections of, respectively, operation sysmbols and variables in the extended signature.

```
Definition namelift (V: varspec σ) (nm: names σ): names (vsignature σ V)
  := inl nm.

Definition varname {V: varspec σ} (v: V): names (vsignature σ V) := inr v.
```

Then, a list of definitions comes: They essentially introduce terms with variables by resorting to ground terms.

```
Definition term (σ: signature) (V: varspec σ)
  : sUU (sorts σ) := gterm (vsignature σ V).

Definition build_term {V: varspec σ} (nm: names σ) (v: (term σ V)⋆ (arity nm))
  : term σ V (sort nm) := build_gterm (namelift V nm) v.

Definition varterm {V: varspec σ} (v: V)
  : term σ V (varsort v) := build_gterm (varname v) [()].
```

Finally, in `FreeAlgebras.v` we pack terms and the `build_term` operation into the algebra $T_\sigma(V)$ of terms over a given signature σ and set of variables V. For this algebra, we prove the expected universal property:

```
Definition free_algebra (σ: signature) (V: varspec σ): algebra σ :=
  @make_algebra σ (termset σ V) build_term.

Definition universalmap
  : ∑ h: free_algebra σ V ↷ a, ∏ v: V, h _ (varterm v) = α v.

Definition iscontr_universalmap
  : iscontr (∑ h:free_algebra σ V ↷ a, ∏ v:V, h (varsort v) (varterm v) = α v).
```

In `TermAlgebras.v` we just consider the special case of `FreeAlgebras.v` for the empty set of variables, i.e. for ground terms. In this case, the universal mapping property is replaced by the initiality of the ground term algebra.

7.1.5. Equations and equational algebras

Equations and their associated structures are key notions in universal algebra. Although an extensive treatment of equational algebras and varieties is out of the scope of the present work, the basic definitions are already present in our implementation in file `EqAlgebras.v`.

In our setting, an *equation* is a pair of terms (with variables) of the same sort. Their intended meaning is to specify *identities law* where variables are implicitly universally quantified.

```
Definition equation (σ : signature) (V: varspec σ): UU
  := ∑ s: sorts σ, term σ V s × term σ V s.
```

The associated projections are denoted `eqsort`, `lhs`, and `rhs` respectively. An *equation system* is just a family of equations.

```
Definition eqsystem (σ : signature) (V: varspec σ): UU
  := ∑ E : UU, E → equation σ V.
```

Then, we pack all the above data into an *equational specification*, that is a signature endowed with an equation system (and the necessary variable specification).

```
Definition eqspec: UU  := ∑ (σ : signature) (V: varspec σ), eqsystem σ V.
```

The interpretation of an equation is easily defined using function `fromterm` introduced in Sect. 7.1.3. More precisely, the predicate `holds` that checks if the universal closure of an equation `e` holds in an algebra `a` is given as follows:

```
Definition holds {σ: signature} {V: varspec σ}
                 (a: algebra σ) (e: equation σ V) : UU
  := ∏ α, fromterm a α (eqsort e) (lhs e) = fromterm a α (eqsort e) (rhs e).
```

From this, it is immediate to define the type `eqalgebra` of *equational algebras* as those algebras in which all the equations of a given equational specification hold.

7.1.6. Categorical structures

Universal algebra has a natural and fruitful interplay with category theory [HP07]. As claimed in the introduction, our mechanisation includes basic categorical constructions for organizing and reasoning about universal algebra structures. In agreement with the general philosophy of univalent mathematics,[6] we can prove that the categories we are interested in – of algebras and equational algebras – are univalent indeed.

In order to develop formal proofs of that property, two possible strategies are available.

A simplest one consists of building the desired category from scratch, and then prove that univalence holds between any pair of isomorphic objects. However, experience has shown that this strategy often lacks a certain naturalness, and it makes the steps involved in the construction hard.

The second available strategy has revealed practicable in a more efficient way: We define the desired category in a step-by-step construction by adding *layers* to a *base category* already given. Such a notion of layer corresponds precisely to a *displayed category* [AL19]: Displayed categories can be thought of as the type-theoretic counterpart of fibrations, and constitute a widely adopted instrument to reason about categories even at higher dimensions [AFMvdW19] in the UniMath library.

To this end, a simple approach would be to proceed in two separate steps, first build the desired categories, then write the proofs that they are univalent. After defining a displayed category over a base category, we can then build a *total category* whose univalence is proven by checking univalence for the base category *and* a displayed version of univalence for the category displayed over the base. This is a generalised version of the so-called *structure identity principle*, introduced first by Peter Aczel as invariance of all structural properties of isomorphic structures (broadly considered).

To build our category of algebras, we apply that very principle: The structure of algebras and homomorphisms is displayed over a base category of shSets that we construct from raw.

[6] See the remarks in [AKS15], where category theory was introduced first in a HoTT-setting.

In a bit more detailed manner, when building the main category of algebras over a given signature σ,

- We associate to each sorted-hSet the property of being an algebra – the fibration in `algebra`;
- To each sorted-function, we associate the property `ishom`;
- We then use the fact that the identity sorted-function defines an algebra homomorphism, and that `ishom` is closed under composition of sorted-functions, as stated by `ishomid` and `ishomcomp`, respectively;[7]
- Finally, we use the UniMath Lemma `is_univalent_disp_from_SIP_data` to prove displayed univalence by showing that the property of being an algebra is an hSet indeed, and that any two interpretations of symbols of σ are equal whenever the identity sorted-function is an homomorphism w.r.t. these given assignments.

At this point, proving that the base category of shSets is univalent revealed already non-trivial. Nevertheless, we managed on the issue by tweaking the proof-terms already constructed for functor categories in UniMath. The resulting total category of algebras is therefore univalent in the usual sense.

Turning now to equational algebras, we do not have to start the construction again from scratch: Within the displayed category formalism we can identify the "substructure" of algebras over shSets satisfying a system of equations. In other terms, we can take for equational algebras the layer over the category of shSets made of the *full displayed subcategory* of the displayed category of algebras identified by the type `is_eqalgebra`.

Again, proving displayed univalence for this layer is not difficult, so that the total category of equational algebras over a system of equations is univalent, as required.

Finally, we rephrase the universal property of the term algebra shown in Section 7.1.2: We can state its initiality in the category of algebras over a given σ by means of the proof-term made of the of the algebra itself and the contractibility of out-going homomorphisms, previously constructed.

The reader interested in the details of these categorical results is referred to our code located in the subdirectory
`UniMath/CategoryTheory/categories/UniversalAlgebra`.

7.2. Some successful experiments

In this section, we want to illustrate by simple examples how to use our framework in three different settings.

[7] See the end of Section 7.1.1.

7.2.1. List algebras

We start with a very simple multi-sorted example, the signature of the list datatype and its algebras.

We will show how to specify a signature in our framework and how to interpret a list datatype as an algebra.[8]

We will need two sorts, one for elements and the other for lists. Correspondingly, we name the two elements `●0` and `●1` of the standard finite set with two elements `⟦ 2 ⟧`.

```
Definition elem_sort_idx: ⟦ 2 ⟧ := ●0.
Definition list_sort_idx: ⟦ 2 ⟧ := ●1.
```

Our signature for the language of lists will consist of two operation symbols for the usual constructors *nil* and *cons* respectively.

Such a signature is encoded with a list of pairs. Each pair describe the input (a list of sorts) and the output (a sort) for the corresponding constructor.

```
Definition list_signature: signature_simple
  := make_signature_simple
       [ ( nil ,, list_sort_idx ) ;
         ( [elem_sort_idx ; list_sort_idx] ,, list_sort_idx ) ]%list.
```

For enhanced readability, we assign explicit names to the operator symbols.

```
Definition  nil_idx: names list_signature := ●0.
Definition cons_idx: names list_signature := ●1.
```

Now, we can endow the list datatype of UniMath (`listset`) with the structure of an algebra over `list_signature` by using the list constructors `nil` and `const`.

We fix a type A for our elements.

```
Variable A : hSet.
```

Then, the class of algebras over `list_signature` is given by

```
Definition list_algebra := make_algebra_simple list_signature
  [( A ; listset A )]
  [( λ _, nil ; λ p, cons (pr1 p) (pr12 p) )].
```

From now on in this section, lemmas are just simple verification of convertibility. They are all proven by reflexivity and the proof scripts are omitted.

To begin with, we check that the sort of elements is A and the sort of lists is given by the associated list datatype:

```
Lemma elem_sort_id : supportset list_algebra elem_sort_idx = A.

Lemma list_sort_id : supportset list_algebra list_sort_idx = listset A.
```

[8] The code for this example can be found in the module `UniMath.Algebra.Universal.Examples.ListDataType`.

Next, we define the associated algebra constructors.
First, let's consider the empty list constructor.

```
Definition list_nil : listset A := ops list_algebra nil_idx tt.
```

As expected, it reduces to the usual *nil* constructor.

```
Lemma list_nil_id : list_nil = @nil A.
```

For the list *cons* constructor, the situation is more complicated. The domain of the constructor is the product `A × listset A × unit`, meaning that the constructor has two (uncurried) arguments

```
Lemma list_cons_dom_id : dom list_algebra cons_idx = A × listset A × unit.
```

Thus, the operation extracted by the `ops` projection has type with the following form

```
Definition list_cons : A × listset A × unit → listset A
  := ops list_algebra cons_idx.
```

That said, our `list_cons` operation reduces to the usual list cons.

```
Lemma list_cons_id (x: A) (l: listset A) : list_cons (x,, (l,, tt)) = cons x l.
```

7.2.2. Equational algebras of monoids

From now on, we will consider single sorted examples for the sake of simplicity.

In this Section, we will discuss the `eqalgebra` of monoids.[9]

To define single sorted signatures, our function `make_signature_simple_single_sorted` is a handy shorthand – introduced in Section 7.1.1 – taking only a list of natural numbers.

```
Definition monoid_signature := make_signature_simple_single_sorted [2; 0].
```

Monoids are already defined in UniMath.

Similarly to what we did in the previous section with lists, we endow monoids with the structure of a monoid algebra.

```
Definition monoid_algebra (M: monoid) : algebra monoid_signature
  := make_algebra_simple_single_sorted monoid_signature M
       [( λ p, op (pr1 p) (pr12 p) ;
          λ _, unel M )].
```

Next, we provide a variable specification, i.e. an hSet of variables together with a map from variables to sorts. Since `monoid_signature` is single-sorted, the only available sort is `tt`.

Then, we build the associated algebra of open terms that will be used to specify the equations of the theory of monoids.

[9] The code for this example can be found in the module `UniMath.Algebra.Universal.Examples.Monoid`.

```
Definition monoid_varspec : varspec monoid_signature
  := make_varspec monoid_signature natset (λ _, tt).

Definition Mon : UU := term monoid_signature monoid_varspec tt.
Definition mul : Mon → Mon → Mon := build_term_curried (●0: names monoid_signature).
Definition id  : Mon := build_term_curried (●1: names monoid_signature).
```

Term variables are associated to natural numebers. In this case, three variables `x`, `y`, `z` will suffice for our needs:

```
Definition x : Mon := varterm (0: monoid_varspec).
Definition y : Mon := varterm (1: monoid_varspec).
Definition z : Mon := varterm (2: monoid_varspec).
```

Now, we have all the ingredients to specify our equations: The monoid axioms of associativity, left identity, and right identity.

```
Definition monoid_equation : UU := equation monoid_signature monoid_varspec.
Definition monoid_mul_lid : monoid_equation := tt,, make_dirprod (mul id x) x.
Definition monoid_mul_rid : monoid_equation := tt,, make_dirprod (mul x id) x.
Definition monoid_mul_assoc : monoid_equation
  := tt,, make_dirprod (mul (mul x y) z) (mul x (mul y z)).
```

We pack the above equations together into an equation system (`monoid_axioms`) and its associated equational specification (`monoid_eqspec`); finally, we define the class of equational algebras of monoids `monoid_eqalgebra`.[10]

Next, we want to show that every "classical" monoid M has a natural structure of equational algebra.

We have two show that M is a model for our equation system. Let's begin then with the left-identity axiom

```
Lemma holds_monoid_mul_lid : holds (monoid_algebra M) monoid_mul_lid.
Proof.
  intro α. cbn in α.
  change (fromterm (monoid_algebra M) α tt (mul id x) = α 0).
  change (op (unel M) (α 0) = α 0).
  apply lunax.
Qed.
```

As you see, we fix the variable evaluation α, then we observe that our goal reduces to the same law expressed in the usual language of monoids – `op` for the product, `unel M` for the identity, α `0` for the first variable `x` – and then the goal is solved at once by applying the corresponding monoid axiom `lunax`.

The other two laws – for right identity and associativity – are proven in the same way.

Thus, we can now pack everything into a monoid eqalgebra by definitions `is_eqalgebra_monoid` and `make_monoid_eqalgebra`.

[10] We omit the formal construction which is uncomplicated and essentially reduces to uninteresting bookkeeping.

7.2.3. Algebra of booleans and Tarski's semantics

We conclude the code survey with a further example based on a simple single sorted algebraic language: the algebra of booleans, and its connectives.[11]

The language considered has the usual boolean operators: truth, falsity, negation, conjunction, disjunction, and implication. Arities can be simply specified by naturals (the number of arguments).

We use the function `make_signature_simple_single_sorted` to build a signature from the list of arities:

```
Definition bool_signature :=
  make_signature_simple_single_sorted [0; 0; 1; 2; 2; 2].
```

Obviously, the type of booleans is already defined in UniMath, together with its usual constants and operations: `false`, `true`, `negb`, `andb`, `orb`, `implb`.

Now, booleans form an hSet, which is denoted `boolset`. It is easy to organize all of those constituents into an algebra for our signature by specifying the translation:

```
Definition bool_algebra :=
  make_algebra_simple_single_sorted bool_signature boolset
  [( λ _, false ;
     λ _, true ;
     λ x, negb (pr1 x) ;
     λ x, andb (pr1 x) (pr12 x) ;
     λ x, orb (pr1 x) (pr12 x) ;
     λ x, implb (pr1 x) (pr12 x) )].
```

Next, we build the algebra of (open) terms, that is, boolean formulas.

This is done in two steps. First, we give a variable specification, i.e. a set of type variables:

```
Definition bool_varspec := make_varspec bool_signature natset (λ _, tt).
```

Then, we define the algebra of terms and the associated constructors.

```
Definition T := term bool_signature bool_varspec tt.
Definition bot  : T         := build_term_curried (•0 : names bool_signature).
Definition top  : T         := build_term_curried (•1 : names bool_signature).
Definition neg  : T → T     := build_term_curried (•2 : names bool_signature).
Definition conj : T → T → T := build_term_curried (•3 : names bool_signature).
Definition disj : T → T → T := build_term_curried (•4 : names bool_signature).
Definition impl : T → T → T := build_term_curried (•5 : names bool_signature).
```

Finally, we use the universal property of the term algebra to define the interpretation of boolean formulas:

```
Definition interp (α: assignment bool_algebra bool_varspec) (t: T) : bool :=
  fromterm (ops bool_algebra) α tt t.
```

[11] The code for this example can be found in the module `UniMath.Algebra.Universal.Examples.Bool`.

At this point, we can check the effectiveness of our definitions with some applications.

To set-up our tests, we introduce three variables `x`, `y`, `z` and a simple evaluation function `v` for variables that assigns true to the variable `x` and `y` (the variable of index 0 and 1) and false otherwise.

Now, we can run the interpretation function by using the Coq internal evaluation mechanism by using the vernacular command `Eval` *strategy* `in` *term*.

This way, we see that the evaluation of the formula $x \wedge (z \to \neg y)$ becomes:

```
Eval lazy in
    interp (λ n, match n with 0 => true | 1 => true | _ => false end)
          (conj x (impl z (not y))).
```

The reader is invited to notice that the choice of the lazy strategy is not accidental. Computations required to evaluate such a proof term are pretty heavy and the standard call by value strategy does not seem able to produce a result in reasonable time.

A few other examples are available in our code as, for instance, a proof of Dummett's tautology:

```
Lemma Dummett : ∏ i, interp i (disj (impl x y) (impl y x)) = true.
Proof.
  intro i. lazy.
  induction (i 0); induction (i 1); apply idpath.
Qed.
```

Notice that this formal proof is just a case analysis for truth-tables in disguise: We instantiate the values of `x` and `y` by applying `induction` twice, but the remaining job is left to the computing mechanism of Coq, which is able to autonomously verify that the evaluation does yield the value `true` in all cases – we only need to apply `idpath`.

Related work

By the code just surveyed, we covered most of fundamental concepts in universal algebra. We plan to enhance our implementation along three directions:

1. First of all, we wish to streamline the interface provided by the library. With the current state of implementation, the user is exposed to many technical details which have no theoretical relevance; these include: the internal signatures generated by `vsignature` for dealing with variables in terms; the existence of two term algebras, one for ground terms, the other for general terms, while the former should only be a particular case of the latter.

   We plan to redesign the interface in order to hide the internal details as much as possible. Furthermore, the interface for heterogeneous vectors might be generalized to make the `HVectors` module more useful outside of the scope of our library.

2. Next, to include more advanced results. On the one hand, we plan to complete the treatment of equational algebras by defining the initial algebra of terms modulo equational congruence. On the other hand, we want to include some relevant theorems, such as the homomorphism theorems and Birkhoff's theorem for varieties.

3. Finally, to extend the library with refined applications and examples of univalent reasoning. This would give evidence that even the minimalist environment of UniMath does allow its user to approach mechanised mathematics with the advantages of both univalent reasoning – to handle equivalent objects as naturally as in informal mathematics – and the automation process of the proof assistant – to be smartly used for performing "internal" implementations in order to leave all computations with no demonstrative significance to the machine.

The code that the present chapter has surveyed is not the only existing mechanisation of universal algebra. In fact, different approaches to the field in computerized mathematics are already known.

A classical work on implementing universal algebra in dependent type theory is [Cap99], where he systematically uses setoids in Coq to handle equality on structures. Another attempt, still based on setoids, has been recently carried on in Agda [GGP18].

In even more recent times, the works [DeM21a, DeM21b, DC21] draw on the multi-sorted version of [Abe21] to develop an extensive and setoid-based Agda library on single-sorted universal algebra that strives to be as powerful as Abel's formalisation but a bit more sensitive to foundational aspects.

On the categorical side, initial semantics furnishes elegant techniques for studying induction and recursion principles in a general setting encompassing applications in programming languages and logic. Assuming univalence, steady research activity has produced over the time a number of contributions to the UniMath library, see e.g. [AHLM18].

However, in the HOTT/UF perspective, the results in [Lyn17] – further developed in [LS19] – seem to settle in a framework that more closely compares with ours. Despite both our formalisation and Lynge's one assume univalent mathematics as formal environment, the study we are proposing here differs from his one by adopting a more foundational perspective. This point of view materialises in our choice of UniMath over CoqHoTT, which is the system adopted by Lynge for his encoding. Moreover, our focus makes the implementation we are proposing different also from categorical treatments mentioned above because of the care we have taken about making the constructions easily evaluable by the very normalisation procedure of proof-terms.

Appendices

A

HOL Light logical engine

HOL Light is a fully programmable proof assistant based on classical higher-order logic, using the programming language OCaml at both the implementation and interaction levels [Har22]. It has been developed and maintained by John Harrison, and belongs to the family of theorem provers originated by Michael Gordon's HOL system [GM93]: It differs from its ancestor because of a minimalist design and a very small logical kernel.

From a mathematical point of view, HOL Light engine consists of simple type theory with polymorphic variables. This logical system is, at the same time, an instantiation of the theory introduced first in [Chu40] and the internal logic of an elementary topos.[1]

Let us start by making explicit this underlying deductive calculus.

The type theory of HOL Light is based on two primitive types: The type `bool` of Booleans and the type `ind` of individuals. Given two types A, B, a function type $A \to B$ can be constructed.

Each typed term has a single well-defined type, but each constant with polymorphic type determines an infinite family of constant terms.

The only primitive logical constant is polymorphic equality

$$=:\ A \to A \to \texttt{bool}.$$

Functional application of $=$ is rendered by infixing it, so that we write $t_1 = t_2$ instead of $(= t_1)t_2$. Moreover, equality between Booleans is used as logical equivalence $\Leftrightarrow$, since formulas of our theory inhabit the type `bool`.

The proof system is based on ten primitive inference rules, rendered by sequent-style natural deduction. This means that we are dealing with judgements of the form $\Gamma \vdash P$ where Γ is a set of formulas – i.e. a context of terms sitting on `bool` – and P is a single formula – i.e. a term sitting on `bool`.[2] This allows us to omit type annotation from the rules, which are collected in Figure A.1.

[1] In [LS88] this system is called *higher order categorical logic*, and corresponds to *local set theory* of [Bel08]. Both these presentation were largely inspired by [BJ81].

[2] We need to change the sequent symbol in order to avoid a formalism clash between that symbol, the symbol for function types $\to$, and the symbol for implication between Booleans $\Rightarrow$.

$$\frac{}{t = t}\ \texttt{REFL} \qquad \frac{\Gamma \vdash s = t \qquad \Delta \vdash t = u}{\Gamma \cup \Delta \vdash s = u}\ \texttt{TRANS}$$

$$\frac{\Gamma \vdash s = t \qquad \Delta \vdash u = v}{\Gamma \cup \Delta \vdash s(u) = t(v)}\ \texttt{MK\_COMB} \qquad \frac{}{\{P\} \vdash P}\ \texttt{ASSUME}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash \lambda x.s = \lambda x.t}\ \texttt{ABS} \qquad \frac{}{\vdash (\lambda x.t)x = t}\ \texttt{BETA}_{(x!)}$$

$$\frac{\Gamma \vdash P \Leftrightarrow Q \qquad \Delta \vdash P}{\Gamma \cup \Delta \vdash Q}\ \texttt{EQ\_MP} \qquad \frac{\Gamma \vdash P \qquad \Delta \vdash Q}{(\Gamma - \{Q\}) \cup (\Delta - \{P\} \vdash P \Leftrightarrow Q}\ \texttt{DEDUCT\_ANTISYM\_RULE}$$

$$\frac{\Gamma[x_1, \cdots, x_n] \vdash P[x_1, \cdots, x_n]}{\Gamma[t_1, \cdots, t_n] \vdash P[t_1, \cdots, t_n]}\ \texttt{INST} \qquad \frac{\Gamma[A_1, \cdots, A_n] \vdash P[A_1, \cdots, A_n]}{\Gamma[B_1, \cdots, B_n] \vdash P[B_1, \cdots, B_n]}\ \texttt{INST\_TYPE}$$

Figure A.1: Primitive rules of HOL Light

The side condition $(x!)$ of `ABS` imposes that x does not occur free in Γ, while `INST` and `INST_TYPE` assume capture-avoiding substitution. `MK_COMB` requires that the composite terms are well-typed, therefore the types need to agree.

Logical operators are apparently lacking. This is not the case, since after [Hen63] we know how to define them in such a setting:

$$\begin{array}{rcc}
\top & := & (\lambda P.P) = (\lambda P.P) \\
\forall & := & \lambda P.(P = \lambda x.\top) \\
\bot & := & \forall P.P \\
\wedge & := & \lambda P.\lambda Q.(\lambda f.fPQ) = (\lambda f.f\top\top) \\
\Rightarrow & := & \lambda P.\lambda Q.P \wedge Q \ \Leftrightarrow P \\
\vee & := & \lambda P.\lambda Q.\forall R.(P \Rightarrow R) \Rightarrow (Q \Rightarrow R) \Rightarrow R \\
\exists & := & \lambda P.\forall Q.(\forall x.P(x) \Rightarrow Q) \Rightarrow Q \\
\neg & := & \lambda P.P \Rightarrow \bot \\
\exists! & := & \lambda P.\exists P \wedge \forall x.\forall y.Px \wedge Py \Rightarrow x = y
\end{array}$$

Then we see that the rules of Figure A.1 suffice to define an intuitionistic higher-order logical kernel, which is further extended by three “mathematical axioms” characterising HOL Light:

1. EXTENSIONALITY: This is given by the rule $\dfrac{}{\vdash (\lambda x.tx) = t}\ \texttt{ETA\_AX}$;
2. CHOICE: This is given by the rule $\dfrac{}{\vdash \forall P.\forall x.Px \Rightarrow P\varepsilon(P)}\ \texttt{SELECT\_AX}$, which requires the introduction of a further polymorphic logical constant in the base language, namely the Hilbert choice operator

$$\varepsilon : (A \to \texttt{bool}) \to A;$$

3. INFINITY: This is given by the rule

$$\frac{}{\vdash \exists f : \mathtt{ind} \to \mathtt{ind}.\mathtt{ONE\_ONE}\, f \land \neg \mathtt{ONTO}\, f}\ \mathtt{INFINITY\_AX}$$

and imposes that the type `ind` is infinite by postulating the existence of an injective endofunction on `ind` that is not surjective.

The `ETA_AX` makes pointwise equal functions formally equal, so that it does work as a function extensionality axiom.
It is known, after [Dia75] and [MG78],[3] that the axiom of choice implies that the logic of HOL Light is classical.
The axiom of infinity postulates that the type of individuals is Dedekind-infinite, so that one could select a subset of `ind` that would behave like a natural numbers type with f as successor function.
Overall, the proof-theoretic strength of the system obtained by adding these three axioms equals bounded Zermelo set theory BZC [Mac12], so that a huge amount of mathematical constructions can be carried out in it, after the mathematical basic concepts one is interested in are defined.

New constants and types are introduced indeed by two definitory principles: The rule for defining a constant allows one to introduce a new constant c and an axiom $\vdash c = t$ – subject to some side conditions on free variables and polymorphic types in t – unless c has been previously defined; the rule for defining a new type behaves similarly. Both definitory principles operates at the object level, but are designed to produce only **conservative extensions** of the basic system: That is on the lines of the design principles of the theorem provers in the HOL family [GM93], which require that new mathematical entities can be defined only by exhibiting a model of them in the existing theory, so that new constants are only definitional extensions.

That is the whole logical kernel at the foundations of HOL Light. Its implementation is based on the LCF programming paradigm [GMW79], whose main principles are:

- The deductive engine behind any proof must be built on top of a *minimal number of primitive rules*, so that its consistency should assure the correctness of the proof themselves;

- The entire system is *embedded inside a programming language* that is then used to implement new inference rules, whose soundness relies on the primitive rules and the type discipline of the ambient programming language.

[3] Refer however to the discussion in [Ten20a, Ten20b] about the base hypotheses leading to that claim.

For HOL Light, these principles are made factual by an OCaml embedding that represents logical notions by defining **three basic datatypes**: `hol_type` for types; `term` for terms; `thm` for theorems.

This means that the rules of the abstract proof system of Figure A.1 becomes programs returning type `thm`: Proving a statement turns into exhibiting an inhabitant of that very type. The validity of such a statement relies then on the correctness of the (program implementing the) rules of the proof system.[4]

Nevertheless, the user is not forced to apply directly those rules, since HOL Light comes with several higher-level derived rules that are easier to interact with. These are designed by combining the primitive rules, and even though no user is expected to concern with their definitions, any user can write her own special purpose proof-rules on the basis of the primitives and the "standard" higher-level ones.

Furthermore, proof development in HOL Light is highly procedural, and human-machine interaction is privileged. The most common way of proving a theorem is via the interactive discovery of proofs by the tactics and the goal stack mechanism of HOL Light.

Tactics break down a goal – i.e. a theorem the user wishes to prove – into more manageable subgoals. Moreover, if a subgoal is solved, the tactics are also able to tackle the development of the proof of the main goal by an appropriate OCaml function. This is, in a sense, a computerised version of what happens in a root-first proof search in a sequent calculus, but here the user has only to keep applying the tactics, and the computer automatically reverses the proof into the top-down construction of the derivation tree built suing the standard primitive rules.

The **goal stack** allows the user to perform tactics and, if necessary, to retract a step of the proof development or correct an application of a tactic. When a tactic generates more than one subgoal, then the latter are presented to the user one at a time, together with the related **lists of hypotheses**. This is similar to considering a branch of a derivation tree at a time, since the goal stack keeps track of the current subgoals to be solved like the root-first proof seeker does when dealing with branching rules.

Some tactics are very simple and very natural to use in the goal stack, since they correspond closely to the rules of natural deduction. But in HOL Light tactics can be combined by means of the so-called **tacticals**. For instance, a most basic tactical is `REPEAT`, that repeats the application of an input tactic until it fails. Another tactical that appears everywhere in the code discussed in Chapter 6 is `THEN`: It simply executes two tactics in sequence, so that in principle any proof development can be compressed into a single large tactic – this way, the user may dispense with the goal stack at all when presenting a formal proof in the theorem prover.

[4] Clearly, the correctness of those programs rests in turn on the OCaml type discipline.

HOL Light comes equipped with much more complex tactic(al)s,[5] some of which are rather domain specific – e.g. real arithmetic or cardinality – so that the construction of proofs is more user friendly. Moreover, a declarative interface has been developed by Harrison, in order to make proof scripts even more easier to read and adapt, as well as more portable, than the standard procedural approach [Har96b].

It is worth noticing that HOL Light has been applied over the years to very different contexts for academic research as well as industrial research, as witnessed by the very HOL Light standard distribution, that includes code ranging from modules of the Flyspeck project [HAB+17, HAB+15] to verification of floating-point algorithms [Har06a, Har12, FJBE+17].

For further information about HOL Light, refer to [Har16]; for a hands-on introduction to the theorem prover see [Har17] and consider surfing the official distribution [Har22]. A key feature that we stress to point at is the really minimalistic design approach of HOL Light: The core that the user needs to trust for developing correct proofs – even when defining new domain specific rules – is so small that it only consists of about 400 l.o.c. This is what makes peculiarly simple a formal verification of that very core, as witnessed by the experiment in self-verification of [Har06b].

[5] Among them `ASM_MESON_TAC` is able to semi-decide statements in first-order theories with identity via model elimination of [Lov68] here based on a backward search introduced in [Har96a].

B

Univalent type theory and UniMath

Per Martin-Löf's intuitionistic type theory – here called dependent type theory, [ML87, MLS84] – dates back to 1971, and, after various revisions during subsequent years, consists of an intuitionistic theory of iterated inductive definitions which had widespread conceptual influence on logically oriented programming languages

The basic syntax of that theory is much more expressive than that of simple type theory. Indeed, most of the informal definitions given in Section 1.5 can be formally defined from scratch in the sequent-style calculus for this extended theory.

Let's start by recalling that formal calculus.
Our objects and types consist of terms generated by the following grammar:

$$t \quad ::= \quad x \quad | \quad \lambda x.\, t \quad | \quad t(t^{'}) \quad | \quad c \quad | \quad f$$

where x indicates a variable, c a primitive constant, and f a defined constant.

We will work with three kinds of judgements:

$$\Gamma \,\mathsf{ctx} \qquad \Gamma \Rightarrow a : A \qquad \Gamma \Rightarrow a \equiv a^{'} : A.$$

As in Section 1.5, Γ (and, generally, capitalized greeks) indicates a type-theoretic context, i.e. a list $x_1 : A_1, \ldots, x_n : A_n$ informing that the distinct variables $x_1, \ldots, x_n$ have type, respectively, $A_1, \ldots, A_n$.

Furthermore, in the present setting, we assume a hierarchy of type universes, denoted by primitive constants

$$\mathcal{U}_0 \quad , \quad \ldots \quad , \quad \mathcal{U}_n \quad , \quad \ldots \qquad .$$

Judgement $\Gamma\,\mathsf{ctx}$ expresses formally the fact that Γ is a consistent context, and it is defined in Figure B.1.

The hierarchy of type universes is defined in Figure B.2. Less formally, we assume given a cumulative hierarchy of universes where

$$\mathcal{U}_m : \mathcal{U}_n \quad \text{for} \quad m < n, \quad \text{and}$$

$$\frac{}{\cdot\ \mathsf{ctx}}\ \textsf{ctx-}\textsc{emp} \qquad \frac{x_1 : A_1, \ldots, x_{n-1} : A_{n-1} \Rightarrow A_n : \mathcal{U}_i}{(x_1 : A_1, \ldots, x_{n-1} : A_{n-1}, x_n : A_n)\ \mathsf{ctx}}\ \textsf{ctx-}\textsc{ext}\ ,$$

where $x_n \notin \{x_1, \ldots, x_{n-1}\}$.

Figure B.1: Rules for well-formed contexts

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\textsc{-intro} \qquad \frac{\Gamma\ \Rightarrow\ A : \mathcal{U}_i}{\Gamma\ \Rightarrow\ A : \mathcal{U}_{i+1}}\ \mathcal{U}\textsc{-cumul}\ .$$

Figure B.2: Rules for universes

$$\text{if} \quad A : \mathcal{U}_m \quad \text{and} \quad m \leq n, \quad \text{then} \quad A : \mathcal{U}_n.$$

The only basic structural rule is

$$\frac{(x_1 : A_1, \ldots, x_n : A_n)\ \mathsf{ctx}}{x_1 : A_1, \ldots, x_n : A_n \Rightarrow x_i : A_i}\ \mathsf{Vble}$$

where $i \in \{i, \ldots, n\}$.[1]

Finally, we assume $\equiv$ is an equivalence relation preserved by typing and constructors of each type.[2]

Type-formers are defined by the standard methodology, i.e. we define each type-former by:

- a formation rule, stating when the type-former can be applied;
- introduction rules, stating how to inhabit the type;
- elimination rules, stating how to use an inhabitant of the type;
- computation rules, consisting in judgemental equalities which define the result of the application of elimination rules to results of introduction rules;

[1] Actually, the following rules are admissible:

$$\frac{\Gamma \Rightarrow a : A \qquad \Gamma, x : A, \Delta \Rightarrow b : B}{\Gamma, \Delta[a/x] \Rightarrow b[a/x]}\ \mathsf{Subst}_1 \qquad \frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma, \Delta \Rightarrow b : B}{\Gamma, x : A, \Delta \Rightarrow b : B}\ \mathsf{Wkg}_1$$

$$\frac{\Gamma \Rightarrow a : A \qquad \Gamma, x : A, \Delta \Rightarrow b \equiv c : B}{\Gamma, \Delta[a/x] \Rightarrow b[a/x] \equiv c[a/x]}\ \mathsf{Subst}_2 \qquad \frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma, \Delta \Rightarrow b \equiv c : B}{\Gamma, x : A, \Delta \Rightarrow b \equiv c : B}\ \mathsf{Wkg}_2\ .$$

[2] We will not formalize the corresponding rules defining these conditions, since they should be easily identified.

- (optional) uniqueness principles, consisting in judgemental equalities determining uniquely an inhabitant of the type by means of the result of an elimination rule applied to it.

Function types correspond to $\prod$-types and are defined in Figure B.3. We see that computation rule for $\prod$-types corresponds to β-reduction for dependent functions, and the uniqueness principle consists of η-reduction. Note also that the arrow type is just a special case of dependent function type with constant $B : A \to \mathcal{U}_i$.

In the proof-as-programs paradigm, an inhabitant of a general $\prod$-type corresponds to a deduction in NJ of a universal formula $\forall x.A(x) \to B(x)$.

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma, x : A \Rightarrow B : \mathcal{U}_i}{\Gamma \Rightarrow \prod_{x:A} B : \mathcal{U}_i}\ \textstyle\prod\text{-FORM}$$

$$\frac{\Gamma, x : A \Rightarrow b : B}{\Gamma \Rightarrow (\lambda x : A.\, b) : \prod_{x:A} B}\ \textstyle\prod-\text{INTRO}$$

$$\frac{\Gamma \Rightarrow f : \prod_{x:A} B \qquad \Gamma \Rightarrow a : A}{\Gamma \Rightarrow f(a) : B[a/x]}\ \textstyle\prod-\text{ELIM}$$

$$\frac{\Gamma, x : A \Rightarrow b : B \qquad \Gamma \Rightarrow a : A}{\Gamma \Rightarrow (\lambda x : A.\, b)a \equiv b[a/x] : B[a/x]}\ \textstyle\prod-\text{COMP}$$

$$\frac{\Gamma \Rightarrow f : \prod_{x:A} B}{\Gamma \Rightarrow (\lambda x : A. fx) \equiv f}\ \textstyle\prod-\text{UNIQ}$$

Figure B.3: Rules for dependent function types

For $\sum$-types we have the rules in Figure B.4. Informally, elimination and computation rules state that to construct a dependent function from a $\sum$-type to a family $C : \sum_{x:A} B \to \mathcal{U}_i$, we only need for a function $g : \prod_{x:A} \prod_{y:B[a/x]} C(a, b)$ from which we can derive a function $f : \prod_{p:\sum_{x:A} B} C(p)$ such that $f((a, b)) :\equiv g(a)(b)$.

Note also that when B does not depend from A, we have the product type $A \times B$; moreover, canonical projections π_1 and π_2 can be derived from the induction rule in both dependent and non-dependent cases.

In the proof-as-programs paradigm, an inhabitant of a general $\sum$-type corresponds to a deduction in NJ of an existential formula $\exists x.A(x) \land B(x)$.

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma, x : A \Rightarrow B : \mathcal{U}_i}{\Gamma \Rightarrow \sum_{x:A} B : \mathcal{U}_i} \textstyle\sum\text{-FORM}$$

$$\frac{\Gamma, x : A \Rightarrow B : \mathcal{U}_i \qquad \Gamma \Rightarrow a : A \qquad \Gamma \Rightarrow b : B[a/x]}{\Gamma \Rightarrow (a, b) : \sum_{x:A} B} \textstyle\sum\text{-INTRO}$$

$$\frac{\Gamma, z : \sum_{x:A} B \Rightarrow C : \mathcal{U}_i \qquad \Gamma, x : A, y : B \Rightarrow g : C[(x,y)/z] \qquad \Gamma \Rightarrow p : \sum_{x:A} B}{\Gamma \Rightarrow \mathsf{ind}_{\sum_{x:A} B}(z.C, x.y.g, p) : C[p/z]} \textstyle\sum\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, z : \sum_{x:A} B \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow a : A \qquad \Gamma, x : A, y : B \Rightarrow g : C[(x,y)/z] \\ \Gamma \Rightarrow b : B[a/x]\end{array}}{\Gamma \Rightarrow \mathsf{ind}_{\sum_{x:A} B}(z.C, x.y.g, (a, b)) \equiv g[(a,b)/(x,y)]} \textstyle\sum\text{-COMP}$$

Here punctuations indicate that $\sum_{x:A} B$ binds (free occurrences of) x in B, $\mathsf{ind}_{\sum_{x:A} B}$ binds (free occurrences of) z in C, and (free occurrences of) x and y in g.

Figure B.4: Rules for dependent pair types

Sum types are defined according to their behaviour in simple type theory by the rules in Figure B.5. Informally, elimination and computation rules state that a function $f : \prod_{x:A+B} C$ for $C : A+B \to \mathcal{U}_i$ can be defined by case analysis from $c : \prod_{x:A} C(\mathsf{in}_1(x))$ and $d : \prod_{y:B} C(\mathsf{in}_2(y))$, and generalised the proof-term of simple type theory $\mathsf{C}(t, x.t_1, y.t_2)$ to handle dependent types.

For the empty type $\bot$ we use the rules in Figure B.6. They state that there are no elements with type $\bot$, and that $\bot$-elimination consists of the *ex falso quodlibet*: from an inhabitant of $\bot$, we obtain an inhabitant of C for any type C, as expected in the proofs-as-programs paradigm. Notice also that those rules make explicit that we could consider $\bot$ as a nullary version of sum types.

The unit type $\top$ is defined in Figure B.7. Here we see how to consider $\top$ as a nullary version of product types with $\star$ as unique inhabitant: uniqueness principle for this type is in fact easily derivable from $\top$-ELIM.

The logical engine of dependent type theory is thus that of first-order NJ extended by extensionality principles – the uniqueness rules in the sequent-style presentation – for each canonical proof-term.

To this, the *intensional* version of dependent type theory adds identity

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma \Rightarrow B : \mathcal{U}_i}{\Gamma \Rightarrow A + B : \mathcal{U}_i}\ +\text{-FORM}$$

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma \Rightarrow B : \mathcal{U}_i \qquad \Gamma \Rightarrow a : A}{\Gamma \Rightarrow \mathsf{in}_1(a) : A + B}\ +\text{-INTRO}_1$$

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma \Rightarrow B : \mathcal{U}_i \qquad \Gamma \Rightarrow b : B}{\Gamma \Rightarrow \mathsf{in}_2(b) : A + B}\ +\text{-INTRO}_2$$

$$\frac{\begin{array}{c}\Gamma, z : A + B \Rightarrow C : \mathcal{U}_i \qquad \Gamma, x : A \Rightarrow c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y : B \Rightarrow d : C[\mathsf{inr}(y)/z] \\ \Gamma \Rightarrow e : A + B\end{array}}{\Gamma \Rightarrow \mathsf{ind}_{A+B}(z.C, x.c, y.d, e) : C[e/z]}\ +\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, z : A + B \Rightarrow C : \mathcal{U}_i \qquad \Gamma, x : A \Rightarrow c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y : B \Rightarrow d : C[\mathsf{inr}(y)/z] \\ \Gamma \Rightarrow a : A\end{array}}{\Gamma \Rightarrow \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{in}_1(a)) \equiv c[a/x] : C[\mathsf{in}_1(a)/z]}\ +\text{-COMP}_1$$

$$\frac{\begin{array}{c}\Gamma, z : A + B \Rightarrow C : \mathcal{U}_i \qquad \Gamma, x : A \Rightarrow c : C[\mathsf{inl}(x)/z] \qquad \Gamma, y : B \Rightarrow d : C[\mathsf{inr}(y)/z] \\ \Gamma \Rightarrow b : B\end{array}}{\Gamma \Rightarrow \mathsf{ind}_{A+B}(z.C, x.c, y.d, \mathsf{in}_2(b)) \equiv d[b/y] : C[\mathsf{in}_2(b)/z]}\ +\text{-COMP}_2$$

Here ind_{A+B} binds z in C, x in c and y in d.

Figure B.5: Rules for sum types

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow \bot : \mathcal{U}_i}\ \bot\text{-FORM}$$

$$\frac{\Gamma, x : \bot \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow a : \bot}{\mathsf{ind}_{\bot}(x.C, a) : C[a/x]}\ \bot\text{-ELIM}$$

Figure B.6: Rules for the empty type

types, so that the logical system is in fact first-order NJ *with equality*, in the proofs-as-programs paradigm.

For identity types, the rules are collected in Figure B.8. Inhabitants of the identity type in a type A are called **paths** in A, and we will refer to the= -ELIM rule as **path induction**: Informally, this rule states that to prove that a property holds for every $x, y : A$ and every path $p : x =_A y$, it suffices to consider the case when x and y are equal and p is refl_x; from a computational point of view, given $p : a =_A b$, path induction permits to substitute both a and b by a variable x, and to define an element of C, given a couple of elements of A,

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow \top : \mathcal{U}_1}\ \top\text{-}\textsc{form}$$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow \star : \top}\ \top\text{-}\textsc{intro}$$

$$\frac{\Gamma, x : \top \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow c : C[\star/x] \qquad \Gamma \Rightarrow a : \top}{\Gamma \Rightarrow \mathsf{ind}_\top(x.C, c, a) : C[a/x]}\ \top\text{-}\textsc{elim}$$

$$\frac{\Gamma, x : \top \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow c : C[\star/x]}{\Gamma \Rightarrow \mathsf{ind}_\top(x.C, c, \star) \equiv c : C[\star/x]}\ \top\text{-}\textsc{comp}$$

As usual punctuations define binding of free occurrences.

Figure B.7: Rules for unit type

by defining it on the diagonal (x, x, refl_x).

Notice that the distinction between **judgemental equality** ($\equiv$) and **propositional equality** ($=$) is what really makes the version of dependent type theory we use here an intensional calculus. This roughly means that in Martin-Löf's type theory one may have $a =_A b$ and a statement $\varphi(a)$, and yet may not be able to claim $\varphi(b)$, though whenever $a \equiv b : A$ and $\varphi(a)$ holds, one *always* has $\varphi(b)$.[3]

For identity types, elimination and computation rules are defined to capture *Leibniz's principle of indiscernibility*:

> Identical elements are those that satisfy the same properties.

In this system, however, this is achieved without any (impredicative) quantification over all properties, but defining the whole identity-type family as

[3] One could reasonably consider judgemental equality a stricter notion of sameness than its propositional counterpart, and introduce a specific rule designed to collapse the two versions into the sole judgemental equality:

$$\frac{p : a =_A b}{a \equiv b : A}\ =\text{-}\textsc{Ref}$$

This rule, known as *identity reflection*, turns dependent type theory into a purely extensional calculus with dependent types which is, in some sense, simpler to be conceptually grasped; here substitution of equals for equals is always permitted in all contexts. That move has, however, a quite unpleasant consequence: In the extensional theory, type-checking is *undecidable*, since equality itself is so. Vice versa, intensional dependent type theory is strongly normalising, thus every term has a unique normal form and every computational process of the calculus always terminates.

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma \Rightarrow a : A \qquad \Gamma \Rightarrow b : A}{\Gamma \Rightarrow a =_A b : \mathcal{U}_i} = \text{-FORM}$$

$$\frac{\Gamma \Rightarrow A : \mathcal{U}_i \qquad \Gamma \Rightarrow a : A}{\Gamma \Rightarrow \mathsf{refl}_a : a =_A a} = \text{-INTRO}$$

$$\frac{\begin{array}{cc} \Gamma, x : A, y : A, p : x =_A y \Rightarrow C : \mathcal{U}_i & \Gamma, z : A \Rightarrow c : C[(z, z, \mathsf{refl}_z)/(x, y, p)] \\ \Gamma \Rightarrow a : A & \Gamma \Rightarrow b : A \\ \multicolumn{2}{c}{\Gamma \Rightarrow p^{'} : a =_A b} \end{array}}{\Gamma \Rightarrow \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, b, p^{'}) : C[(a, b, p^{'})/(x, y, p))]} = \text{-ELIM}$$

$$\frac{\begin{array}{cc} \Gamma, x : A, y : A, p : x =_A y \Rightarrow C : \mathcal{U}_i & \Gamma, z : A \Rightarrow c : C[(z, z, \mathsf{refl}_z)/(x, y, p)] \\ \multicolumn{2}{c}{\Gamma \Rightarrow a : A} \end{array}}{\Gamma \Rightarrow \mathsf{ind}_{=_A}(x.y.p.C, z.c, a, a, \mathsf{refl}_a) : C[(a, a, \mathsf{refl}_a)/(x, y, p))]} = \text{-COMP}$$

Figure B.8: Rules for identity types

inductively generated by the sole reflexivity constructor refl_x.

Now, set-theoretic models and many recursion-theoretic models validate extensional Martin-Löf's type theory, making the calculus even more intuitive from a mathematical point of view than the intensional version in spite of its computational intractability.

The highly constructive philosophy beneath the latter actually invites to consider proofs of an equality between elements of a given type to be worth of further mathematical inquiries, eventually becoming objects of further proofs and constructions.

In such a situation we may have, say, a witness p of the equality between x and y in the type A,

$$p : x =_A y,$$

together with another term q witnessing the same equality. Clearly a question now arises whether p equals q. But in fact, one may give different proofs α, β of this very statement:

$$\begin{aligned} \alpha : &\quad p \underset{x =_A y}{=} q \\ \beta : &\quad p \underset{x =_A y}{=} q \end{aligned}$$

And we may ask whether α is equal to β, and so on.

This pattern of reasoning yields a "tower of equality proofs" stratified on levels which has been for a long time attempted to be "bombed down" using various *ad hoc* informal principles, or formally adding external axioms to the calculus.

In spite of this, in 1998, Martin Hofmann and Thomas Streicher observed that each type of dependent type theory is endowed by equality proofs *within* the calculus with a *groupoid structure* basically definable via $=$-elimination rule.[4]

This is the key insight beneath **homotopy type theory** as a whole, whose basic idea is to use this correspondence between types and ∞-groupoids to relate type theory, higher category theory, and, definitely, homotopy theory.

That is the reason why a proof p of an equality $x =_A y$ is called a path: It can interpreted as a continuous path from the point x to the point y in the space associated to A. Another proof q of the same equality can now be considered a *different* path from x to y in A; and a possible proof α of the higher equality $p = q$ as a *continuous deformation* of the path p into the path q, i.e. a *homotopy* between p and q. Accordingly, any dependent type $x : A \Rightarrow P(x) : \mathcal{U}_i$ can be regarded as a *fibration*, since we can always *transport* a term $t : P(x)$ along a given path $p : x =_A y$ to obtain an element of $P(y)$, defining the transport function by sole $=$-elimination.[5]

It is clear that in this perspective, the relevant notion of sameness for types *is not* equality, but rather **homotopy equivalence**.

It is then not surprising at all that the definitive idea for completing the horizon revealed by homotopy type theory has come from homotopy theorist Vladimir Voevodsky, who independently formulated a homotopical reading of the dependent type theory underlying the Coq proof assistant.
Voevodsky's **Univalence Axiom** states that the type of equivalences between two types is itself *equivalent* to the type of their equalities.

More formally, a function $f : X \to Y$ is an equivalence if any $y : Y$ is the image of a unique $x : X$. This way, any equivalence $f : X \to Y$ comes with a homotopic – i.e. punctual on type inhabitants – inverse $f^i : Y \to X$. Now it is straightforward to see that the identity map on any X is an equivalence, so that the type $X \simeq X$[6] is inhabited indeed. At this point, it is possible to construct an inhabitant $\mathsf{idtoeqv}$ of the type $(A =_{\mathcal{U}_i} A) \to (A \simeq A)$, proving that identical types are equivalent. Univalence imposes that $\mathsf{idtoeqv}$ is itself an equivalence.

Homotopically, this can be rephrased as follows:

> *In the space of all types, the continuous paths between any two types correspond to the equivalences between them.*[7]

[4] A groupoid is a category in which every arrow has an inverse; hence the types of intensional Martin-Löf's theory are actually weak ∞-groupoids, i.e. groupoids with arrows of higher dimensions, where higher equality proofs correspond to these higher arrows.

[5] These are the key features of the homotopical interpretation of dependent type theory as it was originally proposed in [AW09].

[6] The inhabitants of the type of equivalences $X \simeq Y$ between any two types X, Y are just pairs (f, p) where p is a proof-term assuring that $f : X \to Y$ is in fact an equivalence.

[7] A formal treatment of this principle is presented in [The13, Ch.3-4]. For an exceptionally clear analysis of the univalent principle, the reader is referred to [Esc18].

As a consequence, univalence permits to clarify the status of the equality relation on any type universe, and to work on equalities between types as we were handling homotopy equivalences between spaces.[8]

In type-theoretic terms, univalence generalises propositional extensionality: If the latter states that equivalent propositions are identical – as it happens in e.g. the local set theory described in Appendix A – univalence makes precise the meaning of that statement, and extends that identification to each type, independently of the complexity of the categorical structure defined by its associated identity type.[9]

At a more general level, univalence makes the whole (extended) intensional calculus of dependent type theory a **theory of invariants w.r.t. the relation of equivalence of types**.

The informal structuralist approach to mathematics has quested for a formal foundational theory for long, since the common practice of identifying isomorphic structures and investigating properties unaffected by concrete instantiations of a given structure has lacked a principled theoretical account. Univalent type theory does embody this account as a precise logical systems for the foundations of mathematics.[10] Moreover, by considering structured identity types, it is possible to define new kinds of inductive types, built by the parallel definition of the generators for inhabitants of the type and further generators for paths of the associated identity type.[11]

Thus, in homotopy type theory, structural and synthetic style of reasoning is encouraged, along with more frequent interactions between mathematicians and program developers, in order to analyse – and to make efficient use of – the computational character of proofs in this foundational meta-theory.[12]

The theory we use to develop universal algebra in Chapter 7 adds only one basic inductive type $\mathbb{N}$ to this underlying theory, which is defined according to the rules in Figure B.9. Less formally, elimination and computation rules states the possibility to define a function $f : \prod_{x:\mathbb{N}} C$, with $C : \mathbb{N} \to \mathcal{U}_i$ by primitive recursion from $c_0 : C(0)$ and a function $c_s : \prod_{x:\mathbb{N}} C(n) \to C(\mathsf{suc}(n))$.

[8] From this axiom we can derive a principle of function extensionality for dependent functions which homotopically states the equality of any two *sections* of the same fibration just whenever these are homotopic.

[9] In the homotopic perspective, propositional extensionality deals with contractible spaces only, set-theoretic extensionality deals with discrete spaces – that are here called hSets – while univalence deals with any kind of spaces.

[10] See [Esc19] and [AN19, ANST21] for clear discussions of that perspective.

[11] The reader is referred to [The13, Ch. 6] for an introduction.

[12] Recent advances are made concerning the full-constructivisation of univalent type theory by means of new categorical-geometric formal principles for unchaining the computational aspects of univalent reasoning, as witnessed by the new generation of proof assistants for cubical type theory [VMA21], [SHC22], [RDT22], [Jet22].

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-}\textsc{form}$$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \Rightarrow 0 : \mathbb{N}}\ \mathbb{N}\text{-}\textsc{intro}_1$$

$$\frac{\Gamma \Rightarrow n : \mathbb{N}}{\Gamma \Rightarrow \mathsf{suc}(n) : \mathbb{N}}\ \mathbb{N}\text{-}\textsc{intro}_2$$

$$\frac{\Gamma, x : \mathbb{N} \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow c_0 : C[0/x] \qquad \Gamma, x : \mathbb{N}, y : C \Rightarrow c_s : C[\mathsf{suc}(x)/x] \qquad \Gamma \Rightarrow n : \mathbb{N}}{\Gamma \Rightarrow \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n) : C[n/x]}\ \mathbb{N}\text{-}\textsc{elim}$$

$$\frac{\Gamma, x : \mathbb{N} \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow c_0 : C[0/x] \qquad \Gamma, x : \mathbb{N}, y : C \Rightarrow c_s : C[\mathsf{suc}(x)/x]}{\Gamma \Rightarrow \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, 0) \equiv c_0 : C[0/x]}\ \mathbb{N}\text{-}\textsc{comp}_1$$

$$\frac{\Gamma, x : \mathbb{N} \Rightarrow C : \mathcal{U}_i \qquad \Gamma \Rightarrow c_0 : C[0/x] \qquad \Gamma, x : \mathbb{N}, y : C \Rightarrow c_s : C[\mathsf{suc}(x)/x] \qquad \Gamma \Rightarrow n : \mathbb{N}}{\Gamma \Rightarrow \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, \mathsf{suc}(n)) \equiv c_s[(n, \mathsf{ind}_{\mathbb{N}}(x.C, c_0, x.y.c_s, n))/(x, y)] : C[\mathsf{suc}(n)/x]}\ \mathbb{N}\text{-}\textsc{comp}_2$$

Figure B.9: Rules for natural numbers type

This version of dependent type theory constitutes the engine of the UniMath proof development.
Therefore one could say that UniMath lies somehow in between the categories of proof assistants and of proof formalisation styles.

In very rough terms, it consists of a **proper subsystem** of the proof assistant Coq [Coq22]. That, in turn, is based on Thierry Coquand's Calculus of (Co)Inductive Constructions (Co(C)IC), which extends intensional dependent type theory by means of polymorphic variables, as well as several (co)inductive data types and related constructions.

Coquand's system – already mentioned at the end of Section 1.5.1 – is used as well as different proper extensions of dependent type theory to provide the basic logical engine of different implementations of homotopy type theory, e.g [BGL+17], [BHC+22], [VMA21]. However, it is relevant to notice that, at present time, no homotopical interpretation of those extensions is known. This means that the univalent/homotopic constructions carried out on the basis of those systems might necessitate revisions if some parts of the underlying base theory needed arrangement in future.[13]

[13] There is a further potential flaw in this pleroma: To the best of our knowledge, a complete description as abstract proof systems of these theories does not exists, so that it is rather hard to analyse them from a proof-theoretic perspective in order to establish neat and definitive results on them. In contrast with the minimalist approach of the system described in Appendix

This cannot happen for UniMath, since it uses only the skeleton of Coq necessary to implement the constructions presented in the previous pages of this appendix, i.e. intensional dependent type theory with natural numbers type.[14]

This is achieved since, by design, in UniMath it is made use of

- no `record` types;
- no `inductive` types (other than $\mathbb{N}$);
- no `match` constructs.

Univalent reasoning is implemented by adding to this core system a **univalence axiom** – from which function extensionality is derived – and a **propositional resizing principle**, that characterises the overall system as impredicative. The latter principle is simulated by the `type-in-type` Coq flag, which however exposes the general user to the risk of inconsistencies. Nevertheless, resizing is mandatory for a full account of univalent/homotopical reasoning, as it is the only method assuring that any set quotient (X, R) does live in the same universe as $X : \mathcal{U}_i$. This is required in the structural approach behind univalent foundations and the related "intuitive" computerisation of mathematics without recurring to ad hoc setoid-related constructions.

Since univalence is formalised by an axiom, the constructions involving univalent reasoning need to be carried out by the user of UniMath, for the normalisation procedures of Coq are broken by this non-constructive implementation.[15]

The consistency of propositional resizing with univalence was sketched in [Voe11]. Refinements, improvements, and better implementations of this minimalist version of univalent type theory are still under development by the maintainers of the main library [VAG$^+$22] and several collaborators, pointing towards an autonomous proof assistant for that underlying theory.

A, it is also quite hard to develop a description of formal proofs in those extensions that can be checked by independent verifiers. Partial results in this latter research line, considering only Coq, are presented in [BW97], and more recently in [SAB$^+$20].

[14] Refer to [Ber06] and [BC13] for an introduction to the full mechanisms behind Coq.

[15] The theoretical computability of univalence is in any case definitely established in [SA21] and [SHC22]. Its practicability is witnessed by [VMA21].

Bibliography

[Abe21] Andreas Abel. Birkhoff's Completeness Theorem for Multi-Sorted Algebras Formalized in Agda. *CoRR*, abs/2111.07936, 2021.

[AF98] Jeremy Avigad and Solomon Feferman. Gödel's functional ("Dialectica") interpretation. *Handbook of proof theory*, 137:337–405, 1998.

[AFMvdW19] Benedikt Ahrens, Dan Frumin, Marco Maggesi, and Niels van der Weide. Bicategories in Univalent Foundations. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[AHLM18] Benedikt Ahrens, André Hirschowitz, Ambroise Lafont, and Marco Maggesi. High-Level Signatures and Initial Semantics. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:22, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[AKS15] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent Categories and the Rezk Completion. In Maria del Mar González, Paul C. Yang, Nicola Gambino, and Joachim Kock, editors, *Extended Abstracts Fall 2013*, pages 75–76, Cham, 2015. Springer International Publishing.

[AL19] Benedikt Ahrens and Peter LeFanu Lumsdaine. Displayed Categories. *Logical Methods in Computer Science*, Volume 15, Issue 1, March 2019.

[AM18] Mohammad Ardeshir and Mojtaba Mojtahedi. The $\Sigma 1$-provability logic of HA. *Annals of Pure and Applied Logic*, 169(10):997–1043, 2018.

[AMPB21] Gianluca Amato, Marco Maggesi, and Cosimo Perini Brogi. Universal algebra in UniMath. *arXiv preprint arXiv:2102.05952*, 2021.

[AMPPB20] Gianluca Amato, Marco Maggesi, Maurizio Parton, and Cosimo Perini Brogi. Universal Algebra in UniMath. In *Workshop on Homotopy Type Theory/Univalent Foundations – HoTT/UF2020*, 2020.

[AN19] Benedikt Ahrens and Paige Randall North. Univalent foundations and the equivalence principle. In *Reflections on the Foundations of Mathematics*, pages 137–150. Springer, 2019.

[ANST21] Benedikt Ahrens, Paige Randall North, Michael Shulman, and Dimitris Tsementzis. The univalence principle. *arXiv preprint arXiv:2102.06275*, 2021.

[AOP10] Régis Alenda, Nicola Olivetti, and Gian Luca Pozzato. CSL-lean: A theorem-prover for the logic of comparative concept similarity. *Electronic Notes in Theoretical Computer Science*, 262:3–16, 2010.

[AP16] Sergei Artemov and Tudor Protopopescu. Intuitionistic epistemic logic. *The Review of Symbolic Logic*, 9.2:266–298, 2016.

[ARV10] Jiří Adámek, Jiří Rosickỳ, and Enrico Maria Vitale. *Algebraic theories: a categorical introduction to general algebra*, volume 184. Cambridge University Press, 2010.

[Avi18] Jeremy Avigad. Proof theory. In *Introduction to Formal Philosophy*, pages 177–190. Springer, 2018.

[Avi19] Jeremy Avigad. The mechanization of mathematics. In *The Best Writing on Mathematics 2019*, pages 150–170. Princeton University Press, 2019.

[Avi21] Jeremy Avigad. Foundations. *arXiv preprint arXiv:2009.09541*, 2021.

[Avr96] Arnon Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In *Logic: from foundations to applications: European logic colloquium*, pages 1–32, 1996.

[AW09] Steve Awodey and Michael A Warren. Homotopy theoretic models of identity types. In *Mathematical proceedings of the cambridge philosophical society*, volume 146, pages 45–55. Cambridge University Press, 2009.

[Bak17] Miëtek Bak. Introspective Kripke models and normalisation by evaluation for the $\lambda^{\Box}$-calculus. 7th Workshop on Intuitionistic Modal Logic and Applications (IMLA 2017). https://github.com/mietek/imla2017/blob/master/doc/imla2017.pdf, 2017.

[Bal21] Roberta Ballarin. Modern Origins of Modal Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[Bar92] Henk P. Barendregt. *Lambda calculi with types*. Oxford: Clarendon Press, 1992.

[BC13] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[Bel62] Nuel D. Belnap. Tonk, plonk and plink. *Analysis*, 22(6):130–134, 1962.

[Bel08] John L. Bell. *Toposes and local set theories: an introduction*. Dover, 2008.

[Bel12] John L. Bell. Types, Sets, and Categories. *Sets and Extensions in the Twentieth Century*, 6:633–687, 2012.

[Ben19] Bruno Bentzen. A Henkin-style completeness proof for the modal logic S5. *CoRR*, abs/1910.01697, 2019.

[Ber90] Alessandro Berarducci. The interpretability logic of peano arithmetic. *The Journal of Symbolic Logic*, 55(3):1059–1089, 1990.

[Ber06] Yves Bertot. Coq in a Hurry. *arXiv preprint cs/0603118*, 2006.

[BFS20] Kai Brünnler, Dandolo Flumini, and Thomas Studer. A logic of blockchain updates. *Journal of logic and computation*, 30(8):1469–1485, 2020.

[BGJ04] Marta Bilkova, Evan Goris, and Joost J. Joosten. Smart labels. *Liber Amicorum for Dick de Jongh. Institute for Logic, Language and Computation*, 2004.

[BGL^{+}17] Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, Matthieu Sozeau, and Bas Spitters. The HoTT library: a formalization of homotopy type theory in Coq. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, pages 164–172, 2017.

[BHC$^+$22] Guillaume Brunerie, Kuen-Bang Hou (Favonia), Evan Cavallo, Tim Baumann, Eric Finster, Jesper Cockx, Christian Sattler, Chris Jeris, Michael Shulman, et al. Homotopy Type Theory in Agda. Available at https://github.com/HoTT/HoTT-Agda, 2022.

[BJ81] André Boileau and André Joyal. La logique des topos. *The Journal of Symbolic Logic*, 46(1):6–16, 1981.

[BJ11] Félix Bou and Joost J. Joosten. The closed fragment of IL is PSPACE hard. *Electronic Notes in Theoretical Computer Science*, 278:47–54, 2011.

[Bla19] Jasmin Christian Blanchette. Formalizing the Metatheory of Logical Calculi and Automatic Provers in Isabelle/HOL (Invited Talk). In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, page 1–13, New York, NY, USA, 2019. Association for Computing Machinery.

[BM13] Lars Birkedal and Rasmus Ejlers Møgelberg. Intensional type theory with guarded recursive types qua fixed points on universes. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 213–222. IEEE, 2013.

[Boo95] George Boolos. *The logic of provability*. Cambridge university press, 1995.

[Brü09] Kai Brünnler. Deep sequent systems for modal logic. *Archive for Mathematical Logic*, 48(6):551–577, 2009.

[BvB07] Patrick Blackburn and Johan van Benthem. Modal logic: a semantic perspective. In *Handbook of modal logic*, volume 3, pages 1–84. Elsevier, 2007.

[BW97] Bruno Barras and Benjamin Werner. Coq in coq. 1997.

[Cap99] Venanzio Capretta. Universal algebra in type theory. In Yves Bertot, Gilles Dowek, André Hirschowits, Christine Paulin, and Laurent Théry, editors, *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs '99*, volume 1690 of *LNCS*, pages 131–148. Springer, 1999.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940.

[Cop02] B. Jack Copeland. The genesis of possible worlds semantics. *Journal of Philosophical logic*, 31(2):99–137, 2002.

[Coq22] The Coq proof assistant Development Team. The Coq proof assistant. Available at https://coq.inria.fr/, 2022.

[CZ97] Alexander V. Chagrov and Michael Zakharyaschev. *Modal Logic*, volume 35 of *Oxford logic guides*. Oxford University Press, 1997.

[DC21] William DeMeo and Jacques Carette. A Machine-checked proof of Birkhoff's Variety Theorem in Martin-Löf Type Theory. *arXiv e-prints*, pages arXiv–2101, 2021.

[DeM21a] William DeMeo. The Agda Universal Algebra Library, Part 1: Foundation. *arXiv preprint arXiv:2103.05581*, 2021.

[DeM21b] William DeMeo. The Agda Universal Algebra Library, Part 2: Structure. *arXiv preprint arXiv:2103.09092*, 2021.

[dG02] Philippe de Groote. On the strong normalisation of intuitionistic natural deduction with permutation-conversions. *Information and Computation*, 178.2:441–464, 2002.

[Dia75] Radu Diaconescu. Axiom of choice and complementation. *Proceedings of the American Mathematical Society*, 51(1):176–178, 1975.

[dJV90] Dick de Jongh and Frank Veltman. Provability logics for relative interpretability. In *Mathematical logic*, pages 31–42. Springer, 1990.

[dJV95] Dick de Jongh and Albert Visser. Embeddings of Heyting Algebras (revised version of ML-1993-14). 1995.

[dJV99] Dick de Jongh and Frank Veltman. Modal completeness of ILW. *Essays dedicated to Johan van Benthem on the occasion of his 50th birthday. Amsterdam University Press, Amsterdam*, 1999.

[DN03] René David and Karim Nour. A short proof of the strong normalization of classical natural deduction with disjunction. *The Journal of Symbolic Logic*, 68(4):1277–1288, 2003.

[DNOP21] Tiziano Dalmonte, Sara Negri, Nicola Olivetti, and Gian Luca Pozzato. Theorem Proving for Non-normal Modal Logics. In *OVERLAY 2020*, Udine, Italy, September 2021.

[DON18] Tiziano Dalmonte, Nicola Olivetti, and Sara Negri. Non-normal modal logics: Bi-neighbourhood semantics and its labelled calculi. In *Advances in Modal Logic 2018*, 2018.

[dPGM04] Valeria de Paiva, Rajeev Goré, and Michael Mendler. Modalities in constructive logics and type theories. *Journal of Logic and Computation*, 14(4):439–446, 2004.

[dPR11] Valeria de Paiva and Eike Ritter. Basic constructive modality. *Logic without Frontiers: Festschrift for Walter Alexandre Carnielli on the occasion of his 60th Birthday*, pages 411–428, 2011.

[Dum00] Michael Dummett. *Elements of intuitionism*. Oxford University Press, 2000.

[ÉB93] Zoltán Ésik and Stephen L Bloom. *Iteration theories: The equational logic of iterative processes*. Springer-Vlg, 1993.

[EM45] Samuel Eilenberg and Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294, 1945.

[Esc18] Martín Hötzel Escardó. A self-contained, brief and complete formulation of Voevodsky's Univalence Axiom. `https://arxiv.org/abs/1803.02294`, 2018.

[Esc19] Martın Hötzel Escardó. Equality of mathematical structures. *CCC 2019: Computability, Continuity, Constructivity-from Logic to Algorithms*, page 17, 2019.

[Fio21] Guido Fiorino. Linear depth deduction with subformula property for intuitionistic epistemic logic. *arXiv preprint arXiv:2103.03377*, 2021.

[Fit12] Melvin Fitting. Prefixed tableaus and nested sequents. *Annals of Pure and Applied Logic*, 163(3):291–313, 2012.

[Fit13] Melvin Fitting. *Proof methods for modal and intuitionistic logics*, volume 169. Springer Science & Business Media, 2013.

[FJBE$^+$17] Warren E Ferguson Jr, Jesse Bingham, Levent Erkök, John R Harrison, and Joe Leslie-Hurd. Digit serial methods with applications to division and square root (with mechanically checked correctness proofs). *arXiv preprint arXiv:1708.00140*, 2017.

[FKS20] Peng Fu, Kohei Kishida, and Peter Selinger. Linear dependent type theory for quantum programming languages: Extended abstract. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 440–453, New York, NY, USA, 2020. Association for Computing Machinery.

[Flo67] R.W. Floyd. Assigning meanings to program. In *Proc. Symposia in Applied Mathematics, 1967*, volume 19, pages 19–32, 1967.

[FM12] Melvin Fitting and Richard L. Mendelsohn. *First-order modal logic*, volume 277. Springer Science & Business Media, 2012.

[FS77] Gisèle Fischer Servi. On modal logic with an intuitionistic base. *Studia Logica*, 36(3):141–149, 1977.

[Gab96] Dov M. Gabbay. Labelled Deductive Systems. *Oxford Logic Guides 33*, 1, 1996.

[GBJM20] Evan Goris, Marta Bílková, Joost J Joosten, and Luka Mikec. Assuring and critical labels for relations between maximal consistent sets for interpretability logics. *arXiv preprint arXiv:2003.04623*, 2020.

[Gen35a] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische zeitschrift*, 39:176–210, 1935.

[Gen35b] Gerhard Gentzen. Untersuchungen über das logische Schließen II. *Mathematische zeitschrift*, 39:405–431, 1935.

[Get63] Edmund L. Gettier. Is Justified True Belief Knowledge? *Analysis*, 23(6):121–123, 1963.

[GGOP05] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. Analytic tableaux for KLM preferential and cumulative logics. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 666–681. Springer, 2005.

[GGP07] Laura Giordano, Valentina Gliozzi, and Gian Luca Pozzato. KLMLean 2.0: A theorem prover for KLM logics of nonmonotonic reasoning. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 238–244. Springer, 2007.

[GGP18] Emmanuel Gunther, Alejandro Gadea, and Miguel Pagano. Formalization of universal algebra in Agda. *Electronic Notes in Theoretical Computer Science*, 338:147–166, 2018.

[GJ08] Evan Goris and Joost Joosten. Modal Matters for Interpretability Logics. *Logic Journal of the IGPL*, 16(4):371–412, 2008.

[GK21] Rajeev Goré and Cormac Kikkert. CEGAR-Tableaux: Improved Modal Satisfiability via Modal Clause-Learning and SAT. In *International Conference on Automated Reasoning with*

Analytic Tableaux and Related Methods, pages 74–91. Springer, 2021.

[GLO^{+}17] Marianna Girlando, Bjoern Lellmann, Nicola Olivetti, Gian Luca Pozzato, and Quentin Vitalis. VINTE: an implementation of internal calculi for Lewis' logics of counterfactual reasoning. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 149–159. Springer, 2017.

[GLO^{+}22] Marianna Girlando, Björn Lellmann, Nicola Olivetti, Stefano Pesce, and Gian Luca Pozzato. Calculi, countermodel generation and theorem prover for strong logics of counterfactual reasoning. *Journal of Logic and Computation*, 01 2022. exab084.

[GM93] Michael JC Gordon and Tom F. Melham. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[GMW79] Michael Gordon, Robin Milner, and Christopher Wadsworth. *Edinburgh LCF*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[GNO18] Marianna Girlando, Sara Negri, and Nicola Olivetti. Counterfactual logic: Labelled and internal calculi, two sides of the same coin? In *Advances in Modal Logics 2018*, volume 12, pages 291–310, 2018.

[GNO21] Marianna Girlando, Sara Negri, and Nicola Olivetti. Uniform labelled calculi for preferential conditional logics based on neighbourhood semantics. *Journal of Logic and Computation*, 31(3):947–997, 2021.

[GNOR18] Marianna Girlando, Sara Negri, Nicola Olivetti, and Vincent Risch. Conditional beliefs: from neighbourhood semantics to sequent calculus. *The review of symbolic logic*, 11(4):736–779, 2018.

[GO21] Guido Gherardi and Eugenio Orlandelli. Super-strict implications. *Bulletin of the Section of Logic*, 50(1):1–34, Jan. 2021.

[Göd86] Kurt Gödel. Eine Interpretation des Intuitionistischen Aussagenkalküls. Ergebnisse eines Mathematischen Kolloquiums, 4: 39–40, 1933. english translation, with an introductory note by A.S. Troelstra. *Kurt Gödel, Collected Works*, 1:296–303, 1986.

[Gol06] Robert Goldblatt. Mathematical modal logic: A view of its evolution. In *Handbook of the History of Logic*, volume 7, pages 1–98. Elsevier, 2006.

[Gol11] Robert Goldblatt. Cover semantics for quantified lax logic. *Journal of Logic and Computation*, 21(6):1035–1063, 2011.

[GR12] Rajeev Goré and Revantha Ramanayake. Labelled tree sequents, tree hypersequents and nested (deep) sequents. *Advances in modal logic*, 9:279–299, 2012.

[Gra18] Daniel Grayson. An introduction to univalent foundations for mathematicians. *Bulletin of the American Mathematical Society*, 55(4):427–450, 2018.

[Gri89] Timothy G. Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–58, 1989.

[GRS21] Rajeev Goré, Revantha Ramanayake, and Ian Shillito. Cut-elimination for provability logic by terminating proof-search: formalised and deconstructed using Coq. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 299–313. Springer, 2021.

[GS20] Marianna Girlando and Lutz Straßburger. Moin: A nested sequent theorem prover for intuitionistic modal logics (system description). In *International Joint Conference on Automated Reasoning*, pages 398–407. Springer, 2020.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, 1989.

[HAB$^+$15] Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Le Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, et al. A formal proof of the kepler conjecture. *arXiv preprint arXiv:1501.02155*, 2015.

[HAB$^+$17] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Hoang Le Truong, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, et al. A formal proof of the kepler conjecture. In *Forum of mathematics, Pi*, volume 5. Cambridge University Press, 2017.

[Har96a] John Harrison. Optimizing proof search in model elimination. In *International Conference on Automated Deduction*, pages 313–327. Springer, 1996.

[Har96b] John Harrison. Proof style. In *International Workshop on Types for Proofs and Programs*, pages 154–172. Springer, 1996.

[Har06a] John Harrison. Floating-point verification using theorem proving. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 211–242. Springer, 2006.

[Har06b] John Harrison. Towards self-verification of hol light. In *International Joint Conference on Automated Reasoning*, pages 177–191. Springer, 2006.

[Har11] Robert Harper. The Holy Trinity. Post at `https://existentialtype.wordpress.com/2011/03/27/the-holy-trinity/`, 2011.

[Har12] John Harrison. *Theorem proving with the real numbers*. Springer Science & Business Media, 2012.

[Har16] John Harrison. The HOL Light System Reference. `https://www.cl.cam.ac.uk/~jrh13/hol-light/reference.pdf`, 2016.

[Har17] John Harrison. HOL Light tutorial. `http://www.cl.cam.ac.uk/~jrh13/hol-light/tutorial.pdf`, 2017.

[Har22] John Harrison. The HOL Light Theorem Prover. Available at `https://github.com/jrh13/hol-light`, 2022.

[Hen63] Leon Henkin. A theory of propositional types. *Fundamenta Mathematicae*, 52(3):323–344, 1963.

[Hin62] Kaarlo Jaakko Juhani Hintikka. *Knowledge and belief: An introduction to the logic of the two notions*. Cornell University Press, 1962.

[Hin97] J. Roger Hindley. *Basic simple type theory*. Number 42. Cambridge University Press, 1997.

[HJ16] Tuomas A Hakoniemi and Joost J Joosten. Labelled tableaux for interpretability logics. *arXiv preprint arXiv:1605.05612*, 2016.

[HN12] Raul Hakli and Sara Negri. Does the deduction theorem fail for modal logic? *Synthese*, 187(3):849–867, 2012.

[HP07] Martin Hyland and John Power. The category theoretic understanding of universal algebra: Lawvere theories and monads. *Electronic Notes in Theoretical Computer Science*, 172:437–458, 2007.

[HS08] J. Roger Hindley and Jonathan P. Seldin. *Lambda-calculus and Combinators, an Introduction*, volume 2. Cambridge University Press Cambridge, 2008.

[Hum15] Lloyd Humberstone. *Philosophical applications of modal logic*. College Publications, 2015.

[HUW14] John Harrison, Josef Urban, and Freek Wiedijk. History of Interactive Theorem Proving. In *Computational Logic*, volume 9, pages 135–214, 2014.

[Iem19] Rosalie Iemhoff. Uniform interpolation and sequent calculi in modal logic. *Archive for Mathematical Logic*, 58(1):155–181, 2019.

[Iem20] Rosalie Iemhoff. Intuitionism in the Philosophy of Mathematics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2020 edition, 2020.

[IS18] Jonathan Jenkins Ichikawa and Matthias Steup. The Analysis of Knowledge. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2018 edition, 2018.

[Jac99] Bart Jacobs. *Categorical logic and type theory*. Elsevier, 1999.

[Jet22] Contributors of JetBrains. Arend Theorem Prover. https://arend-lang.github.io/, 2022.

[JRMV20] Joost J. Joosten, Jan Mas Rovira, Luka Mikec, and Mladen Vuković. An overview of generalised Veltman semantics. *arXiv preprint arXiv:2007.04722*, 2020.

[Kak16] Yoshihiko Kakutani. Calculi for intuitionistic normal modal logic. *arXiv preprint arXiv:1606.03180*, 2016.

[Kan57] Stig Kanger. Provability in logic. *Stockholm studies in philosophy*, 1957.

[Kas94] Ryo Kashima. Cut-free sequent calculi for some tense logics. *Studia Logica*, pages 119–135, 1994.

[Ket45] Oiva Ketonen. Untersuchungen Zum Prädikatenkalkul. *Journal of Symbolic Logic*, 10(4):127–130, 1945.

[KL16] Roman Kuznets and Björn Lellmann. Grafting hypersequents onto nested sequents. *Logic Journal of the IGPL*, 24(3):375–423, 2016.

[Kle52] S.C. Kleene. Permutability of inferences in Gentzen's calculi LK and LJ. *Memoirs of the American Mathematical Society*, 1952.

[KO21] Taishi Kurahashi and Yuya Okawa. Modal completeness of sublogics of the interpretability logic IL. *Mathematical Logic Quarterly*, 67(2):164–185, 2021.

[KP11] Clemens Kupke and Dirk Pattinson. Coalgebraic semantics of modal logics: An overview. *Theoretical Computer Science*, 412(38):5070–5094, 2011.

[Kri59] Saul A Kripke. A completeness theorem in modal logic 1. *The Journal of Symbolic Logic*, 24(1):1–14, 1959.

[Kri65] Saul A. Kripke. Semantical analysis of intuitionistic logic I. In *Studies in Logic and the Foundations of Mathematics*, volume 40, pages 92–130. Elsevier, 1965.

[Kur13] Hidenori Kurokawa. Hypersequent calculi for modal logics extending S4. In *JSAI International Symposium on Artificial Intelligence*, pages 51–68. Springer, 2013.

[KY16] Vladimir N. Krupski and Alexey Yatmanov. Sequent calculus for intuitionistic epistemic logic IEL. In *International Symposium on Logical Foundations of Computer Science*, pages 187–201. Springer, 2016.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958.

[Law69] F William Lawvere. Adjointness in foundations. *Dialectica*, pages 281–296, 1969.

[Law71] F. William Lawvere. Quantifiers as sheaves. In *Proc Intern. Congress of Math.*, pages 1506–1511. Gauthier-Villars, 1971.

[Lew18] Clarence Irving Lewis. *A survey of symbolic logic*. University of California press, 1918.

[Lip00] Peter Lipton. Tracking Track Records: John Worrall: Relying on Meta-induction? In *Aristotelian Society Supplementary Volume*, volume 74, pages 179–205. Wiley Online Library, 2000.

[Lit14] Tadeusz Litak. Constructive modalities with provability smack. In *Leo Esakia on duality in modal and intuitionistic logics*, pages 187–216. Springer, 2014.

[Lon00] John R. Longley. Notions of computability at higher types I. In *Logic Colloquium*, volume 19, pages 32–142, 2000.

[Lov68] Donald W. Loveland. Mechanical theorem-proving by model elimination. In *Automation of Reasoning*, pages 117–134. Springer, 1968.

[LS88] Joachim Lambek and Philip J. Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

[LS19] Andreas Lynge and Bas Spitters. Universal algebra in HoTT. In *TYPES 2019, 25th International Conference on Types for Proofs and Programs*, 2019.

[Łu30] Jan Łukasiewicz. Philosophical remarks on many-valued systems of propositional logic. *Jan Łukasiewicz Selected Works*, 1930.

[LV18] Tadeusz Litak and Albert Visser. Lewis meets Brouwer: constructive strict implication. *Indagationes Mathematicae*, 29(1):36–90, 2018.

[LV19] Tadeusz Litak and Albert Visser. Lewisian fixed points I: two incomparable constructions. *arXiv preprint arXiv:1905.09450*, 2019.

[Lyn17] Andreas Lynge. Universal algebra in HoTT, 2017. Bachelor's thesis, Department of Mathematics, Aarhus University.

[Mac06] Hugh MacColl. *Symbolic Logic and its applications*. Longmans, Green, 1906.

[Mac12] Saunders MacLane. *Mathematics: form and function*. Springer Science & Business Media, 2012.

[Mac13] Saunders MacLane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[Mar08] Jean-Pierre Marquis. *From a geometrical point of view: A study of the history and philosophy of category theory*, volume 14. Springer Science & Business Media, 2008.

[MB06] Marino Miculan and Giorgio Bacci. Modal logics for brane calculus. In *International Conference on Computational Methods in Systems Biology*, pages 1–16. Springer, 2006.

[MF74] Grigori Mints and R. Feys. Sistemy Lyuisa i sistema T (Supplement to the Russian translation). *R. Feys, Modal Logic*, pages 422–509, 1974.

[MG78] John Myhill and Nelson Goodman. Choice implies excluded middle. *Zeit Math Log*, 23:461, 1978.

[Min92] G. Mints. *Short Introduction to Modal Logic.* Center for the Study of Language and Information Publication Lecture Notes. Cambridge University Press, 1992.

[MJV20a] Luka Mikec, Joost J. Joosten, and Mladen Vuković. A W-flavoured series of interpretability principles. *Short Papers, Advances in Modal Logic, AiML*, 2020:60–64, 2020.

[MJV20b] Luka Mikec, Joost J. Joosten, and Mladen Vukovic. On ILWR-frames. *Logic and Applications LAP 2020*, page 50, 2020.

[MKO95] David McAllester, Jakov Kucan, and D.F. Otth. A proof of strong normalization for F2, Fω, and beyond. *Information and Computation*, 121(2):193–200, 1995.

[ML87] Per Martin-Löf. Truth of a proposition, evidence of a judgement, validity of a proof. *Synthese*, pages 407–420, 1987.

[ML17] Stefan Milius and Tadeusz Litak. Guard your daggers and traces: Properties of guarded (co-) recursion. *Fundamenta Informaticae*, 150(3-4):407–449, 2017.

[MLS84] Per Martin-Löf and Giovanni Sambin. *Intuitionistic type theory*, volume 9. Bibliopolis Naples, 1984.

[MN18] Julius Michaelis and Tobias Nipkow. Formalized proof systems for propositional logic. In A. Abel, F. Nordvall Forsberg, and A. Kaposi, editors, *23rd Int. Conf. Types for Proofs and Programs (TYPES 2017)*, volume 104 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[Moj22] Mojtaba Mojtahedi. On Provability Logic of HA. *arXiv preprint arXiv:2206.00445*, 2022.

[Mos21] Joan Moschovakis. Intuitionistic Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[MP08] Conor McBride and R.A. Paterson. Applicative programming with effects. *Journal of functional programming*, 18(1):1–13, 2008.

[MPB21] Marco Maggesi and Cosimo Perini Brogi. A Formal Proof of Modal Completeness for Provability Logic. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[MPB22] Marco Maggesi and Cosimo Perini Brogi. A theorem prover and countermodel constructor for provability logic in HOL Light. *arXiv preprint 2205.03659*, 2022.

[MPV17] Luka Mikec, Tin Perkov, and Mladen Vuković. Decidability of interpretability logics ILM_0 and ILW^*. *Logic Journal of the IGPL*, 25(5):758–772, 2017.

[MPV19] Luka Mikec, Fedor Pakhomov, and Mladen Vuković. Complexity of the interpretability logic il. *Logic Journal of the IGPL*, 27(1):1–7, 2019.

[MS14] Sonia Marin and Lutz Straßburger. Label-free modular systems for classical and intuitionistic modal logics. In *Advances in Modal Logic 10*, 2014.

[MU21] Leonardo de Moura and Sebastian Ullrich. The Lean 4 Theorem Prover and Programming Language. In *International Conference on Automated Deduction*, pages 625–635. Springer, 2021.

[MV20] Luka Mikec and Mladen Vuković. Interpretability logics and generalised Veltman semantics. *The Journal of Symbolic Logic*, 85(2):749–772, 2020.

[Nak00] Hiroshi Nakano. A modality for recursion. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No.99CB36332)*, pages 255–266, 2000.

[Neg05] Sara Negri. Proof analysis in modal logic. *Journal of Philosophical Logic*, 34(5):507–544, 2005.

[Neg14a] Sara Negri. Proof analysis beyond geometric theories: from rule systems to systems of rules. *Journal of Logic and Computation*, 26(2):513–537, 2014.

[Neg14b] Sara Negri. Proofs and countermodels in non-classical logics. *Logica Universalis*, 8(1):25–60, 2014.

[Neg17] Sara Negri. Proof theory for non-normal modal logics: The neighbourhood formalism and basic results. *IfCoLog Journal of Logics and their Applications*, 4(4):1241–1286, 2017.

[NO19] Sara Negri and Eugenio Orlandelli. Proof theory for quantified monotone modal logics. *Logic Journal of the IGPL*, 27(4):478–506, 2019.

[Noz81] Robert Nozick. Philosophical explanations. *Ethics*, 94(2), 1981.

[NP21] Sara Negri and Edi Pavlović. Proof-theoretic analysis of the logics of agency: The deliberative stit. *Studia Logica*, 109(3):473–507, 2021.

[NvP08] Sara Negri and Jan von Plato. *Structural proof theory*. Cambridge university press, 2008.

[NvP11] Sara Negri and Jan von Plato. *Proof analysis: a contribution to Hilbert's last problem*. Cambridge University Press, 2011.

[OC18] Eugenio Orlandelli and Giovanna Corsi. Labelled calculi for quantified modal logics with non-rigid and non-denoting terms. *Automated Reasoning in Quantified Non-Classical Logics*, page 64, 2018.

[OP03] Nicola Olivetti and Gian Luca Pozzato. CondLean: A theorem prover for conditional logics. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 264–270. Springer, 2003.

[OP05] Nicola Olivetti and Gian Luca Pozzato. CondLean 3.0: Improving CondLean for stronger conditional logics. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 328–332. Springer, 2005.

[OP08] Nicola Olivetti and Gian Luca Pozzato. Theorem proving for conditional logics: CondLean and GoalDuck. *Journal of Applied Non-Classical Logics*, 18(4):427–473, 2008.

[OP14] Nicola Olivetti and Gian Luca Pozzato. NESCOND: an implementation of nested sequent calculi for conditional logics. In *International Joint Conference on Automated Reasoning*, pages 511–518. Springer, 2014.

[OP15] Nicola Olivetti and Gian Luca Pozzato. A standard internal calculus for Lewis' counterfactual logics. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 270–286. Springer, 2015.

[Orl28] Ivan E. Orlov. The calculus of compatibility of propositions. *Mathematics of the USSR, Sbornik*, 35:263–286, 1928.

[Orl21] Eugenio Orlandelli. Labelled calculi for quantified modal logics with definite descriptions. *Journal of Logic and Computation*, 31(3):923–946, 2021.

[PB19] Cosimo Perini Brogi. A Curry-Howard Correspondence for Intuitionistic Belief. *Poster session of The Proof Society Summer School 2019. Swansea University. Computational Foundry*, 2019.

[PB21a] Cosimo Perini Brogi. An analytic calculus for intuitionistic belief. *arXiv preprint arXiv:2103.01734*, 2021.

[PB21b] Cosimo Perini Brogi. Curry–Howard–Lambek correspondence for intuitionistic belief. *Studia Logica*, 109(6):1441–1461, 2021.

[Pel99] Francis Jeffry Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20(1):1–31, 1999.

[PF21] Petros Papapanagiotou and Jacques Fleuriot. Object-level reasoning with logics encoded in hol light. In *Fifteenth Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, pages 18–34. Open Publishing Association, 2021.

[Pog09] Francesca Poggiolesi. The method of tree-hypersequents for modal propositional logic. In *Towards mathematical philosophy*, pages 31–51. Springer, 2009.

[Pog10] Francesca Poggiolesi. *Gentzen calculi for modal propositional logic*, volume 32. Springer Science & Business Media, 2010.

[Pog16] Francesca Poggiolesi. Natural deduction calculi and sequent calculi for counterfactual logics. *Studia Logica*, 104(5):1003–1036, 2016.

[Pop94] Sally Popkorn. *First steps in modal logic*. Cambridge University Press, 1994.

[Pot83] Garrel Pottinger. Uniform, cut-free formulations of T, S4 and S5. *Journal of Symbolic Logic*, 48(3):900, 1983.

[Pra65] Dag Prawitz. *Natural deduction: A proof-theoretical study*. Courier Dover Publications, 1965.

[Pra71] Dag Prawitz. Ideas and results in proof theory. In *Studies in Logic and the Foundations of Mathematics*, volume 63, pages 235–307. Elsevier, 1971.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 109–121. IEEE, 1976.

[Pra19] Dag Prawitz. The fundamental problem of general proof theory. *Studia Logica*, 107(1):11–29, 2019.

[Pri57] Arthur N. Prior. *Time and modality*. Oxford University Press, 1957.

[Pri60] Arthur N. Prior. The Runabout Inference-Ticket. *Analysis*, 21(2):38–39, 12 1960.

[Pri08] Graham Priest. *An introduction to non-classical logic: From if to is*. Cambridge University Press, 2008.

[Pro12] Carlo Proietti. Intuitionistic epistemic logic, Kripke models and Fitch's paradox. *Journal of philosophical logic*, 41(5):877–900, 2012.

[PS86] Gordon Plotkin and Colin Stirling. A framework for intuitionistic modal logics. In *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*, pages 399–406, 1986.

[PV16] Tin Perkov and Mladen Vuković. Filtrations of generalized Veltman models. *Mathematical Logic Quarterly*, 62(4-5):412–419, 2016.

[RDT22] The RedPRL Development Team. The Red* family of proof assistants. https://redprl.org/, 2022.

[Res05] Greg Restall. Proofnets for S5: sequents and circuits for modal logic. In *Logic Colloquium*, volume 28, pages 3–1, 2005.

[RMJ20] J. Mas Rovira, Luka Mikec, and Joost J. Joosten. Generalised veltman semantics in agda. *Short Papers, Advances in Modal Logic, AiML*, 2020:86–90, 2020.

[Rog21] Daniel Rogozin. Categorical and algebraic aspects of the intuitionistic modal logic IEL-and its predicate extensions. *Journal of Logic and Computation*, 31(1):347–374, 2021.

[RS20] Michael Rathjen and Wilfried Sieg. Proof Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2020 edition, 2020.

[SA21] Jonathan Sterling and Carlo Angiuli. Normalization for Cubical Type Theory. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–15, 2021.

[SAB$^+$20] Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. The METACOQ project. *Journal of Automated Reasoning*, 64(5):947–999, 2020.

[Sas02] Katsumi Sasaki. A cut-free sequent system for the smallest interpretability logic. *Studia Logica*, 70(3):353–372, 2002.

[Sch93] Klaus Schild. Combining terminological logics with tense logic. In *Portuguese Conference on Artificial Intelligence*, pages 105–120. Springer, 1993.

[Sch21] Peter Scholze. Liquid tensor experiment. *Experimental Mathematics*, pages 1–6, 2021.

[Seg82] Krister Segerberg. A completeness theorem in the modal logic of programs. *Banach Center Publications*, 9:31–46, 1982.

[Sha85] Natarajan Shankar. Towards mechanical metamathematics. *Journal of Automated Reasoning*, 1(4):407–434, 1985.

[Sha88] Vladimir Yurievich Shavrukov. The logic of relative interpretability over Peano arithmetic. *Preprint*, 5, 1988.

[Sha97] Vladimir Yurievich Shavrukov. Interpreting reflexive theories in finitely many axioms. *Fundamenta Mathematicae*, 152(2):99–116, 1997.

[SHC22] Christian Sattler, Simon Huber, and Thierry Coquand. Canonicity and homotopy canonicity for cubical type theory. *Logical Methods in Computer Science*, 18, 2022.

[Sim94] Alex K. Simpson. The proof theory and semantics of intuitionistic modal logic. *PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics*, 1994.

[Smu68] Raymond M. Smullyan. Analytic cut. *The Journal of Symbolic Logic*, 33(4):560–564, 1968.

[Sol76] Robert M. Solovay. Provability interpretations of modal logic. *Israel journal of mathematics*, 25(3-4):287–304, 1976.

[SS19] Youan Su and Katsuhiko Sano. First-order intuitionistic epistemic logic. In *International Workshop on Logic, Rationality and Interaction*, pages 326–339. Springer, 2019.

[SU06] Morten H. Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2006.

[Ten20a] Neil Tennant. Does Choice Really Imply Excluded Middle? Part I: Regimentation of the Goodman–Myhill Result, and Its Immediate Reception. *Philosophia Mathematica*, 28(2):139–171, 2020.

[Ten20b] Neil Tennant. Does Choice Really Imply Excluded Middle? Part II: Historical, Philosophical, and Foundational Reflections on the Goodman–Myhill Result. *Philosophia Mathematica*, 29(1):28–63, 2020.

[The13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[TMR53] Alfred Tarski, Andrzej Mostowski, and Raphael Mitchel Robinson. *Undecidable theories*, volume 13. Elsevier, 1953.

[Tro69] Anne Sjerp Troelstra. *Principles of Intuitionism*. Springer, 1969.

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory*. Number 43. Cambridge University Press, 2000.

[VAG$^+$22] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. UniMath — A computer-checked library of univalent mathematics. Available at https://github.com/UniMath/UniMath, 2022.

[Val83] Silvio Valentini. The modal logic of provability: cut-elimination. *Journal of Philosophical logic*, pages 471–476, 1983.

[vB16] Johan van Benthem. Modal logic: A contemporary view. *Internet Encyclopedia of Philosophy, http://www.iep.utm.edu/modal-lo*, 2016.

[vdGI20] Iris van der Giessen and Rosalie Iemhoff. Proof theory for intuitionistic strong l\" ob logic. *arXiv preprint arXiv:2011.10383*, 2020.

[vdGI21] Iris van der Giessen and Rosalie Iemhoff. Sequent calculi for intuitionistic gödel–löb logic. *Notre Dame Journal of Formal Logic*, 62(2):221–246, 2021.

[vDT88] Dirk van Dalen and Anne Troelstra. *Constructivism in Mathematics. An Introduction I*, volume 121 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1988.

[Ver92] L.C. Verbrugge. Verzamelingen-Veltman frames en modellen (set Veltman frames and models). *Unpublished manuscript, Amsterdam*, 1992.

[Ver17] Rineke (L.C.) Verbrugge. Provability Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017.

[Vis88] Albert Visser. Preliminary notes on interpretability logic. *Logic group preprint series*, 29, 1988.

[Vis90] Albert Visser. Interpretability logic. In *Mathematical logic*, pages 175–209. Springer, 1990.

[VMA21] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming*, 31, 2021.

[Voe11] Vladimir Voevodsky. Resizing rules–their use and semantic justification. *Slides from a talk at TYPES, Bergen*, 2011.

[vP18] Jan von Plato. The Development of Proof Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2018 edition, 2018.

[Vuk99] Mladen Vuković. The principles of interpretability. *Notre Dame Journal of Formal Logic*, 40(2):227–235, 1999.

[vW51] Georg Henrik von Wright. Deontic logic. *Mind*, 60(237):1–15, 1951.

[VZ19] Albert Visser and Jetze Zoethout. Provability logic and the completeness principle. *Annals of Pure and Applied Logic*, 170(6):718–753, 2019.

[Wil92] Timothy Williamson. On intuitionistic modal epistemic logic. *Journal of Philosophical Logic*, 21.1:63–89, 1992.

[XN20] Yiming Xu and Michael Norrish. Mechanised modal model theory. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 518–533, Cham, 2020. Springer International Publishing.

[Zac19] Richard Zach. Hilbert's Program. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2019 edition, 2019.

[Zam92] Domenico Zambella. On the proofs of arithmetical completeness for interpretability logic. *Notre Dame journal of formal logic*, 33(4):542–551, 1992.