Relating Justification Logic Modality and Type Theory in Curry–Howard fashion

Konstantinos Pouliasis

December 3, 2017

Abstract

This thesis is a work in the intersection of *Justification Logic* (JL) and *Curry–Howard Isomorphism* (CHI). Justification logic is an umbrella of modal logics of knowledge with explicit evidence. Justification logics have been used to tackle traditional problems in proof theory (in relation to Godel's provability), philosophy (Gettier examples, Russel's barn paradox). The Curry–Howard Isomorphism or *proofs–as–programs* is an understanding of logic that places logical studies in conjunction with type theory and – in current developments – category theory. The point being that understanding a system as a logic, a typed calculus and, a language of a class of categories constitutes a useful discovery that can have many applications. The application we will be mainly concerned is type systems for useful programming language constructs. This work is structured in three parts: The first part (chapter 2, chapter 3, chapter 4) is a revision of my second examination paper and constitutes a bird's eye view into my research topics: *intuitionistic logic, justified modality* and *type theory.* The relevant systems are introduced syntactically together with main metatheoretic proof techniques which will be useful in the rest of the thesis. The second part (chapter 5, chapter 6, chapter 7) features my main contributions. I will propose a modal type system that extends simple type theory (or, isomorphically, intuitionistic propositional logic) with elements of justification logic and will argue about its computational significance.

More specifically, I will show that the obtained calculus characterizes certain computational phenomena related to linking (module systems, foreign function implementations) that abound in modern programming language semantics. I will present full metatheoretic results obtained for this logic/ calculus utilizing techniques from the first part and will provide proofs in the Appendix. Finally, I conclude this work with a small “outro”, Chapter 8, where I informally show that the ideas underlying my contributions can be extended in interesting ways.

This work is my thesis requirement for my PhD candidacy under the supervision of Distinguished Professor Sergei Artemov at the Department of Computer Science of the Graduate Center, CUNY.

Acknowledgements

This thesis would have been impossible without my supervisor Prof. Sergei Artemov. He introduced me to the beauty of epistemic logic and its deep relations to proof theory back at my first years as a graduate student. He inspired and helped shape my topic of research and has not stopped being a source of inspiration ever since. He has provided guidance, support and encouragement through our numerous meetings. I want to thank him deeply.

In addition, I want to thank my committee members: Prof. Fitting and Prof. Shankar. Prof. Fitting's classes at the Graduate Center have been a tremendous experience and source of knowledge. The clarity of both his writings and presentations helped me realize the level of academic quality a young student should be striving and aiming toward. Prof. Shankar has been a great teacher and an inspiring advisor. We spent hours discussing topics related to my thesis or to the formal methods area at large. He was one of the people that invigorated my interest in programming language theory that has proven very valuable for my research work and my post-graduate career developments.

Prof. Zachos, my external committee member, has been one of the most important chapters of academic life. It is thanks to him that I attended the PhD program at CUNY and his guidance in all walks of my academic and non-academic life has contributed the most to its successful completion. He has been an inspiring scientist, a great mentor and a friend for the last fifteen years that I have known him.

During my studies at the Graduate Center I had the chance to meet prominent scientists both from CUNY and worldwide. These people helped me expand my scientific horizon and improve the quality of my work. I would like to thank especially Rohit Parikh, Thomas Wies, Mark Tuttle. I want to give special thanks to Giuseppe Primiero who believed in this work since its early stages and co-authored the first related publication with me. The late Kristoffer Rose inspired me with his vigor in programming languages, as well as, his passion for open source projects and teaching math and computer science to underrepresented students. In addition, he introduced me to the possibility of high-quality (and fun) research in the industry. He left us early and he is dearly missed.

Moreover, I want to thank my wonderful friends both in NYC and back home. Danai Dragonea, Dafni Anesti, Antonis Bouchagiar and Michalis Agathos took great care of me every summer after my teaching obligations had turned me to a jaded new yorker. All our beautiful excursions under the greek sun helped me stay happy and healthy throughout eight intense academic years. Krystal Raydo and George Rabanca became family and showed me immense support through the thick and thin in New York. Cagil Tasdemir has been the best friend and a great source of knowledge and wisdom. I want to thank fellow graduate students (and fellow surfers): Marios Georgiou, Nikos Melissaris and Ivo Vigan for our great time inside and outside school. My friend Antonis Stamboulis has been a major support throughout

my PhD. I had the privilege to discuss with him type theory and modality and having him in New York for the last couple of years has been the greatest gift.

Lots of CUNY staff have worked seamlessly with me to advance my progress. I want to thank Lina Garcia, and Dilvania Rodriguez. I am deeply indebted to Joe Driscoll for his great help and friendship. He accommodated my schedule to facilitate my teaching which was my main means of survival for many years. I want to thank him dearly.

Finally, I want to thank my kinfolk. Wilson Sherwin's love and understanding has been one of my most important motivations for completing this work and I am looking forward to our future together. My family Aliki, Doria, Vaso and Spyros deserve most of the credit for all my achievements including this work. I had the luck to have two amazing sisters and two loving parents and that's all the wealth a man could ask for.

This dissertation is dedicated to the memory of my late grandparents and grandaunt: Aliki Giannioti, a tuberculosis survivor, Theodoros (Lolos) Gianniotis, a fisherman, and Lambro Mitsiali, a factory worker, for raising me and my elder sister at their home in Anemomylos, Corfu and showing us how to build beauty with as little as a small house, a fishing boat and a well-kept garden painted fresh white every spring.

Contents

Chapter 1

Introduction

The *Curry–Howard Isomorphism* (CHI) [17, 30] was first established as a deep connection between explicit proofs in intuitionistic logic and programs of a simple programming language that includes pairs, functions and union types [43, 50]. This relation has been a central topic of study in the field of type theory and has turned into the standard foundational approach to studying and designing programming languages especially of the functional paradigm. Since this relation has been established, the isomorphism has been extended to more complex logics and correspondingly to more complex programming language constructs. In the following I will be using *Curry–Howard Isomorphism* (CHI) and *proofs-as-programs* interchangeably.

There are great benefits both for a logician and the programming language designer in viewing things through the lenses of such a relation. From the programming language perspective certain linguistic phenomena are given categorical characterizations that do not depend on implementation specifics. For example the designer of the next hot programming language knows that adding pairs would have to adhere to the corresponding constructs of logical conjunction. In addition, adding more complex design features (e.g. state, concurrency, exceptions etc) can be done in a structured, orthogonal, and modular way by enriching the underlying logic and, correspondingly, the type system(see e.g. [24, 16, 37, 23]).

It should not be a surprise that languages of the typed functional paradigm have been gaining traction and more functional design principles are being added to languages of the object oriented paradigm. There are two main reasons for this, an old and a new. The older reason is mathematical correctness which is strongly related to the fact that reasoning about programs (of the lambda calculus and its extensions) can be done in an *equational way*, a property that is heavily connected to their underlying foundational principles as we will see. Features such as side effects or concurrency in the language are reflected via typing. For example, a program that changes global state has a type that says so explicitly. Similarly, a program that uses goto mechanisms would also say register its behavior in its type. All in all, even "unpure", non-functional constructs (state, mutable references) are added in a mathematical/ algebraic fashion under the CHI disciple. As a result, reasoning about properties of such programs is significantly simpler. Moreover, under a strongly typed doctrine, important properties of programs are checked statically by the type-checker and prior to their execution. Henceforth, the need for testing is reduced to verify only non-trivial properties.

The renewed interest in functional programming owes a lot to the difficulties of scaling concurrent programs in traditional programming paradigms. It is very hard to scale programs that make unlimited use of side effects (such as state change) in an implicit way (i.e. without leaving any trace in their typing) from sequential to multithreaded style of computation. Programming freedoms in traditional languages (paired with the easiness and textbook familiarity of the Von Neumann model) come at a large cost if one takes into account the need for proving correctness. The "purity" of programs in the lambda calculus – and the delimited "impurity" in its extensions – makes writing high-quality concurrent code an easier task. It is exciting to see that important metatheoretic results in the area of combinatory logic as e.g. the Church–Rosser property are the backbone of models for concurrent computation in modern functional languages.

On the other hand, the logician has good reasons to study logics as rules of program formation and reduction. First of all, such designs make logics implementable "for free" in modern theorem provers using the programmistic side of the correspondence. Secondly the study of logic in such a way has put upfront a Gentzen-style treatment of logical connectives where emphasis is given to the notion of proof, proof structure and proof reduction. This has sparked studies for more refined versions of proof relevant deduction than the ones discovered under the standard "axiomatic" approach (e.g. linear logics, substructural logics etc). As we will see, the Gentzen– Brouwer initiative in logic does not merely call for change of axiomatization but for a "proof relevant" interpretation of connectives that comes with a computational taste. Metatheory is also standardized once one studies logic this way; scalable techniques have been developed within the area of "proof-theoretic" semantics that make the passing from natural deduction of a logic to its cut free calculus pretty standard [48, 38]. In other words, treating logics within a Curry – Howard environment enriches logic as a discipline with good organizing principles. Finally, proof relevant treatments of logic – pushed further by ideas of *Martin-Löf Type Theory* [36] have sparked a renewed interest in a foundation of mathematics that stems from a treatment of proofs as *the* primitive objects of mathematics.

In this work, we are interested in the study of extending *Curry–Howard Isomorphism* (CHI) with basic constructive necessity of justification logic. There is a good reason to believe that this should be doable. Justification logic is a logic that relates logical necessity with the existence of a proof construct and that is exactly what working in realm of proof relevancy and CHI calls for. There are challenges to this task, both syntactical and semantical. First of all, there is a resemblance of the justification logic syntax with that of simple type theory (e.g. the use of the semicolon $a : A$) that initially might call for an antagonistic relation between the two systems. Of course, this is not a substantial issue since the two typing relations can be "colored" in a syntactical way. But resolving the syntactical overload would still leave a "meaning" question open; namely, how can one read the need of having two proofs of the "same thing" in a system. The main contribution of this work [44] it that it shows how such a relation of binding two kinds of proof systems is quite natural and gives a basic reading of validity and necessity on first, proof-theoretic principles. We will treat justification logic as a logic of *proof relevant validity.* by tracing justification logic back to its origin as an explicit, classical semantics to *Brower-Heyting-Kolmogorov* (BHK) proof constructs. We will present a modal logic that is based on this relation and we will argue that such phenomena of binding two kinds of constructions abound both in the realm of mathematical proofs but also in the realm of programming languages with constructs such as modules, foreign function calls and dynamic linkers.

This work is structured in three parts: The first part (chapter 2, chapter 3, chapter 4) is a revision of my second examination paper and constitutes a bird's eye view into my research topics: *intuitionistic logic, justified modality* and *type theory.* The relevant systems are introduced syntactically together with main metatheoretic proof techniques which will be useful in the rest of the thesis. The second part (chapter 5, **??**, chapter 7) constitutes my main contributions. I will propose a modal type system that extends simple type theory (or, speaking from the logical side of CHI, intuitionistic propositional logic) with elements of justification logic and will argue about its computational significance. More specifically, I will show that the obtained calculus characterizes certain computational phenomena that abound in modern programming language semantics. I will present full metatheoretic results obtained for this logic/ calculus utilizing techniques from the first part and will provide proofs in the Appendix. In the Appendix, the interested reader can find links to the active Github repo that contains an implementation of this calculus (its term and type systems) in the metaprogramming framework "Makam" as an additional "proof of concept" result, Finally, I conclude this work with a small "outro", Chapter 8, where I informally show that the ideas underlying my contributions can be extended in interesting ways.

Chapter 2

Intuitionistic Logic

2.1 Intuitionism

In this and the subsequent chapter, I will be presenting foundational work in the intersection of *Intuitionistic Logic* and *Type Theory.* The presentation is scaffolding following Robert Harper's lecture videos in *Homotopy Type Theory* [25] and the accompanying notes by students of the class [28]. I will often deviate to standard textbooks in the field [10], [22], [43] to present further important results.

2.1.1 A bird's eye view

In a nutshell, *Intuitionistic mathematics* is a program in foundations of mathematics that extends *Brouwer's program* [15]. Brouwer, in an almost Kantian fashion, viewed mathematical reasoning as a human faculty and mathematics as a language of the “creative subject” aiming to communicate mathematical concepts. The concept of *algorithm* as a step–by–step constructive process is brought in the foreground in Brouwer’s program. As a result, intuitionistic theories are amenable to computational interpretations. We are using the terms intutionistic and *constructive* interchangeably.

For the purposes of this paper, the main diverging point of Brouwer’s program, later explicated by Heyting [29] and Kolmogorov [32] [9], lies in the treatment of proofs. In contrast to classical approaches to foundations that treat proof objects as external to theories, the constructive approach treats proofs as the fundamental forms of construction and hence, as first class citizens. As a result, the constructive view of logic draws heavily from proof theory and Gentzen’s developments [21]. For the reader interested also in the philosophical implications of constructive foundations and *antirealism*, Dummet’s treatment is a classic in the field [18].

It has to be emphasized that proofs in the intuitionistic approach are treated as stand–alone and are not bound to formal systems (i.e the notion of proof *precedes* that of a formal system). It is necessary, hence, to draw a distinction between the notion of *proof as construction* and the notion of *proof in a formal system* [27, 26].

A *formal proof* is a proof given in a fixed formal system and it is constructed by the application of the rules in that system, recursively. Formal proofs can, thus, be viewed as strings or *gödelizations* of textual derivation in some fixed system.

As Harper puts it “Although every formal proof (in a specific system) is also a proof (assuming soundness of the system) the converse is not true”. This conforms with Gödel’s Incompleteness Theorem, which precisely states that there exist true propositions (with a proof in *some* formal system), but for which there is no formal proof within the system in question. This *openness* of the nature of proofs is necessary for a foundational treatment of proofs that respects Gödelian phenomena.

Following the same line of thought, and adopting the doctrine of *proof relevance* for obtaining true judgments, leads to another main difference of the constructive approach and the classical one i.e. the (default) absence of the *law of excluded middle*.

2.2 IPL

Intuitionistic Propositional Logic (IPL) can be viewed as “the logic of *proof relevance*” conforming with the intuitionistic view described in 2.1. To judge a fact as *true* one may provide a *proof* appropriate of the fact. *Proofs* can be synthesized to obtain proofs for more complex facts (*introduction rules*) and consumed to provide proofs relevant for other facts (*elimination rules*). The importance of the interplay between introduction and elimination rules was developed by Gentzen. A discussion on the meaning of the logical connectives that is prevalent in *MLTT* can be found in [35]. Following the presentation style by Martin-Löf we split the notions of *judgment* and *proposition*. We have two main kinds of judgments:

- *Judgments* that are logical arguments about the truth (or, equivalently, proof) of a *proposition.* They might, optionally, involve assumptions about the truth (or, equivalently, proof) of other propositions. We might call these *logical judgments.*
- Judgments on *propositionality* or typeability. *Propositions* are the *subjects* of *logical judgments.* If something is judged to be a proposition then it belongs to the universe of discourse and can be mentioned in *logical judgments.*

In addition, since a *logical judgment* might involve a set Γ of assumptions (or a *context*), it is convenient to add a third kind of judgment of the form $\Gamma\ \mathsf{ctx}$ Thus, in IPL, we get the judgments $\phi\ \in\ \mathsf{Prop}$, $\phi\,\mathsf{true}$ and $\Gamma\,\mathsf{ctx}$:

$\phi \in \mathsf{Prop}$	ϕ is a (well-formed) proposition
$\phi\ \ \mathsf{true}$	Proposition ϕ is true i.e., has a proof.
$\Gamma\,\mathsf{ctx}$	Γ is a (well-formed) context of assumptions

The natural deduction system of IPL is given below:

Prop Formation

$$\frac{}{P_i \in \mathsf{Prop}}\ \textsc{Atom} \qquad \frac{}{\top \in \mathsf{Prop}}\ \textsc{Top} \qquad \frac{}{\bot \in \mathsf{Prop}}\ \textsc{Bottom}$$

$$\frac{\phi_1 \in \mathsf{Prop} \qquad \phi_2 \in \mathsf{Prop}}{\phi_1 \supset \phi_2 \in \mathsf{Prop}}\ \textsc{Arr} \qquad \frac{\phi_1 \in \mathsf{Prop} \qquad \phi_2 \in \mathsf{Prop}}{\phi_1 \wedge \phi_2 \in \mathsf{Prop}}\ \textsc{Conj}$$

$$\frac{\phi_1 \in \mathsf{Prop} \qquad \phi_2 \in \mathsf{Prop}}{\phi_1 \vee \phi_2 \in \mathsf{Prop}}\ \textsc{Disj}$$

Context Formation

$$\frac{}{\circ\ \ \mathsf{ctx}}\ \textsc{Nil} \qquad \frac{\Gamma\ \ \mathsf{ctx} \qquad \phi \in \mathsf{Prop}}{\Gamma, \phi\ \mathsf{true}\ \ \mathsf{ctx}}\ \Gamma\text{-}\textsc{Ext}$$

Context Reflection

$$\frac{\Gamma\ \ \mathsf{ctx} \qquad \phi\ \mathsf{true} \in \Gamma}{\Gamma \vdash \phi\ \mathsf{true}}\ \Gamma\text{-}\textsc{Refl}$$

Top Introduction – Bottom Elimination

$$\frac{}{\Gamma \vdash \top\ \mathsf{true}}\ \top\mathrm{I} \qquad \frac{\Gamma \vdash \bot\ \mathsf{true}}{\Gamma \vdash \phi\ \mathsf{true}}\ \bot\mathrm{E}$$

Implication Introduction and Elimination

$$\frac{\Gamma, \phi_1\ \mathsf{true} \vdash \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1 \supset \phi_2\ \mathsf{true}}\supset\mathrm{I} \qquad \frac{\Gamma \vdash \phi_1 \supset \phi_2\ \mathsf{true} \qquad \Gamma \vdash \phi_1\ \mathsf{true}}{\Gamma \vdash \phi_2\ \mathsf{true}}\supset\mathrm{E}$$

Conjunction Introduction and Elimination

$$\frac{\Gamma \vdash \phi_1\ \mathsf{true} \qquad \Gamma \vdash \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{true}}\wedge\mathrm{I}$$

$$\frac{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1\ \mathsf{true}}\wedge\mathrm{E}\textsc{l} \qquad \frac{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_2\ \mathsf{true}}\wedge\mathrm{E}\textsc{r}$$

Disjunction Introduction and Elimination

$$\frac{\Gamma \vdash \phi_1\ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2\ \mathsf{true}}\vee\mathrm{I}\textsc{l} \qquad \frac{\Gamma \vdash \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2\ \mathsf{true}}\vee\mathrm{I}\textsc{r}$$

$$\frac{\Gamma \vdash \phi_1 \vee \phi_2\ \mathsf{true} \qquad \Gamma, \phi_1\ \mathsf{true} \vdash \phi\ \mathsf{true} \qquad \Gamma, \phi_2\ \mathsf{true} \vdash \phi\ \mathsf{true}}{\Gamma \vdash \phi\ \mathsf{true}}\vee\mathrm{E}$$

2.2.1 Basic Properties of Intuitionistic Entailment

Reflexivity

$$\frac{}{\Gamma, \phi\ \mathsf{true} \vdash \phi\ \mathsf{true}}$$

Transitivity

$$\frac{\Gamma \vdash \psi\ \mathsf{true} \qquad \Gamma, \psi\ \mathsf{true} \vdash \phi\ \mathsf{true}}{\Gamma, \phi\ \mathsf{true} \vdash \phi\ \mathsf{true}}$$

Contraction

$$\frac{\Gamma, \phi\ \mathsf{true}, \phi\ \mathsf{true} \vdash \psi\ \mathsf{true}}{\Gamma, \phi\ \mathsf{true} \vdash \psi\ \mathsf{true}}$$

Exchange

$$\frac{\Gamma \vdash \phi\ \mathsf{true}}{\pi(\Gamma) \vdash \phi\ \mathsf{true}}$$

Where $\pi(\Gamma)$ is a meta-symbol standing for any permutation of Γ.

2.3 Order Theoretic Semantics: *Heyting Algebras*

IPL viewed order theoretically gives rise to a *Heyting Algebra(HA)*. To define *HA* we need the notion of a *lattice*. For our purposes we define it as follows[1]:

Definition: A *lattice* is a non-empty *pre–order* with finite meets and joins.

In addition, we define *bounded lattice* as follows:

Definition: A *bounded lattice* $(L, \leq)$ is a lattice that additionally has a greatest element 1 and a least element 0, which satisfy

$0 \leq x \leq 1$ for every x in L

Finally, we can define *HA*:

Definition: A *HA* is a bounded lattice $(L, \leq, 0, 1)$ s.t. for every $a, b \in L$ there exists an x (we name it $a \to b$) with the properties:

1. $a \wedge x \leq b$
2. x is the greatest such element

Axiomatization of HAs

We can axiomatize the meet (i.e. greatest lower bound)($\wedge$) of ϕ, ψ for any lower bound χ.

[1] One can take a lattice being a partial order. The same results hold with slight modifications.

$$\overline{\phi \wedge \psi \leq \phi} \qquad\qquad \overline{\phi \wedge \psi \leq \psi}$$

$$\frac{\chi \leq \phi \quad \chi \leq \psi}{\chi \leq \phi \wedge \psi}$$

We can axiomatize the join ($\vee$)(i.e. the least upper bound) of ϕ, ψ for any upper bound χ as follows .

$$\overline{\phi \leq \phi \vee \psi} \qquad\qquad \overline{\psi \leq \phi \vee \psi}$$

$$\frac{\phi \leq \chi \quad \psi \leq \chi}{\phi \vee \psi \leq \chi}$$

We can axiomatize the existence of a greatest element as follows:

$$\overline{\chi \leq 1}$$

which says that 1 is the greatest element.

We can axiomatize the existence of a least element as follows:

$$\overline{0 \leq \chi}$$

which says that 0 is the least element.

Finally, to axiomatize *HAs* we require the existence of exponentials for every ϕ, ψ as follows:

$$\frac{}{\phi \wedge (\phi \supset \psi) \leq \psi} \qquad \frac{\phi \wedge \chi \leq \psi}{\chi \leq \phi \supset \psi}$$

Soundness and Completeness

Theorem. $\Gamma \vdash_{IPL} \phi\, \mathsf{true}$ iff for any *Heyting Algebra* H we have $\Gamma^{+} \leq \phi^{*}$ where $*$ is defined as the lifting of any map of Props to elements of H and $(+)$ is defined inductively on the length of Γ as follows

$$\begin{aligned} \circ^{+} &= \top \\ (\Gamma, \phi)^{+} &= \Gamma^{+} \wedge \phi * \end{aligned}$$

Chapter 3

Typed lambda calculus

3.1 From intuitionistic provability to proof trees

IPL can be viewed as a declarative axiomatization of proof constructs. Take the introduction rule for conjunction as an example:

$$\frac{\Gamma \vdash A \;\mathsf{true} \qquad \Gamma \vdash B \;\mathsf{true}}{\Gamma \vdash A \wedge B \;\mathsf{true}} \wedge\mathrm{I}$$

The rule says, “given the existence a proof of A and a proof of B from assumptions Γ, there exists a proof of $A \wedge B$ from assumptions Γ at hand ”.

We used the description “declarative” because in this format *IPL* sequents $\Gamma \vdash \;\mathsf{true}$ do not describe how such existentials are realized. It is in essence a logic of “proof relevant truth” but it does not involve the proofs themselves as first class objects.

An alternative presentation is to explicate proof constructs by directly providing a system of “proof trees”. Such an approach was actually championed in Gentzen’s natural deduction systems and is the necessary move to obtain proof calculi. Once we have explicit proof objects (either as trees, or as we will see, as terms) the system is enriched with equality principles involving such objects. Such rules give computational value (“proof dynamics”) to the constructs and drive the idea of “Curry–Howard Isomorphism” and its extensions. We will provide such a formulation in proof trees of judgments together with the equality rules on trees, essentially following Gentzen.

Proof trees of judgments have the following shape:

$$\begin{array}{c} J_1, \ldots, J_i \\ \vdots \\ \\ J \end{array}$$

We focus on judgments J of the form $A\,\mathsf{true}$. Here are the the rules for constructing proof trees with labeled assumptions[1].

$$\begin{array}{c} x_1 : A_1\ \mathsf{true}, \ldots, x_i : A_i\ \mathsf{true} \\ \vdots \\ A_{j\in 1\ldots i}\ \mathsf{true} \end{array} \qquad\qquad \begin{array}{c} x_1 : A_1\ \mathsf{true}, \ldots, x_i : A_i\ \mathsf{true} \\ \vdots \\ \top\ \mathsf{true} \end{array}$$

$$\dfrac{\begin{array}{c} \mathcal{D} \\ A\ \mathsf{true} \end{array} \qquad \begin{array}{c} \mathcal{E} \\ B\ \mathsf{true} \end{array}}{A \wedge B\ \mathsf{true}}$$

$$\dfrac{\begin{array}{c} \mathcal{D} \\ A \wedge B\ \mathsf{true} \end{array}}{A\ \mathsf{true}} \qquad\qquad \dfrac{\begin{array}{c} \mathcal{D} \\ A \wedge B\ \mathsf{true} \end{array}}{B\ \mathsf{true}}$$

$$\dfrac{\begin{array}{c} \overline{x : A} \\ \mathcal{D} \\ B\ \mathsf{true} \end{array}}{A \supset B\ \mathsf{true}} \qquad\qquad \dfrac{\begin{array}{c} \mathcal{D} \\ A \supset B\ \mathsf{true} \end{array} \qquad \begin{array}{c} \mathcal{E} \\ A\ \mathsf{true} \end{array}}{B\ \mathsf{true}}$$

[1] Strictly speaking the constructs are directed acyclic graphs and not *trees* since assumptions with the same label are bound and substitutable together but we will be cavalier with such a detail

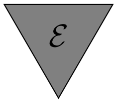
$$\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}}{A \lor B\ \mathsf{true}} \qquad\qquad \dfrac{\begin{array}{c}\mathcal{D}\\ B\ \mathsf{true}\end{array}}{A \lor B\ \mathsf{true}}$$

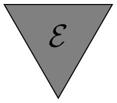
$$\dfrac{\begin{array}{c}\\ \mathcal{D}\\ A \lor B\ \mathsf{true}\end{array} \quad \begin{array}{c}A\ \mathsf{true}\\ \mathcal{E}\\ C\ \mathsf{true}\end{array} \quad \begin{array}{c}B\ \mathsf{true}\\ \mathcal{F}\\ C\ \mathsf{true}\end{array}}{C\ \mathsf{true}}$$

$$\dfrac{\begin{array}{c}\mathcal{D}\\ \bot\ \mathsf{true}\end{array}}{C\ \mathsf{true}}$$

3.1.1 Properties of Intuitionistic Entailment Redux

Proof trees by their nature satisfy the properties of entailment in 2.2.1. We will not bother with reflection and contraction. The first is trivial and the second can be shown by simple induction on the structure of trees; the proof is highlighting that reflection on hypothesis is order-irrelevant. Transitivity is established by *compositionality* of proof trees and reflects the essence of hypothetical reasoning: proof trees of the appropriate proposition can be "plugged in" for assumptions to create new, well-formed trees.

Theorem. If $\begin{array}{c} x : A \\ \mathcal{D} \\ B\ \textsf{true} \end{array}$ and 

$A\ \textsf{true}$ are valid proof trees their composition denoted as 

$\begin{array}{c} A\ \textsf{true} \\ \mathcal{D} \\ B\ \textsf{true} \end{array}$, defined by substituting all occurrences of $x : A$ for E in $\mathcal{D}$, is a valid proof tree for $B\ \textsf{true}$.

3.1.2 Equating Proof Trees

Having proof objects as first class citizens, permits for developing logics, essentially, as theories of (typed) equality among such objects. This idea stemmed from Gentzen's work on natural deduction and cut elimination and it is what gives proofs computational content. Following are the proposed equalities for the proof relevant *IPL* introduced initially by Gentzen as the driver of the proof cut elimination. We will be revisiting these very same equalities and reframing them as equalities among proof terms in the next section. Nevertheless, they originated in proof tree form. We show indicatively the equalities regarding the $\supset, \wedge$ connectives reserving the rest for the more concise notation.

$$\dfrac{\dfrac{\begin{array}{c}\overline{x : A}\\ \mathcal{D}\\ B\ \mathsf{true}\end{array}}{A \supset B\ \mathsf{true}} \qquad \begin{array}{c}\mathcal{E}\\ A\ \mathsf{true}\end{array}}{B\ \mathsf{true}} \quad =_\beta \quad \begin{array}{c}\mathcal{E}\\ A\ \mathsf{true}\\ \mathcal{D}\\ B\ \mathsf{true}\end{array}$$

$$\begin{array}{c}\mathcal{D}\\ A \supset B\ \mathsf{true}\end{array} \quad =_\eta \quad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A \supset B\ \mathsf{true}\end{array} \qquad \overline{x : A}}{B\ \mathsf{true}}}{A \supset B}$$

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array} \qquad \begin{array}{c}\mathcal{E}\\ B\ \mathsf{true}\end{array}}{A \wedge B\ \mathsf{true}}}{A\ \mathsf{true}} \quad =_\beta \quad \begin{array}{c}\mathcal{D}\\ A\ \mathsf{true}\end{array}$$

$$\dfrac{\dfrac{\overset{\mathcal{D}}{A\ \mathsf{true}} \qquad \overset{\mathcal{E}}{B\ \mathsf{true}}}{A \wedge B\ \mathsf{true}}}{B\ \mathsf{true}} \quad =_\beta \quad \overset{\mathcal{E}}{B\ \mathsf{true}}$$

$$\overset{\mathcal{D}}{A \wedge B\ \mathsf{true}} \quad =_\eta \quad \dfrac{\dfrac{\dfrac{\mathcal{D}}{A \wedge B\ \mathsf{true}}}{A\ \mathsf{true}} \qquad \dfrac{\dfrac{\mathcal{D}}{A \wedge B\ \mathsf{true}}}{B\ \mathsf{true}}}{A \wedge B}$$

3.2 Linear representation of trees with proof terms: λ calculus

Proof terms provide an alternative, linear representation for proof trees. The simply typed lambda calculus and its equational system can, thus, be viewed as a calculus for proof trees and proof reductions for intuitionistic logic. What's more, following the doctrine of proof relevance and of characterizing connectives by their proof reductions, i.e. working in the realm of Curry – Howard Isomorphism, we hit two birds with one stone: we both develop proof relevant logics and we get typed programming languages that reflect their computational content. The "simplest" language obtained within this program is the simply typed lambda calculus, but we will see that the same doctrine extends to different logics with different judgmental constructs.

Simply typed lambda calculus

Type Formation

$$\frac{}{P_i \in \mathsf{Type}}\ \textsc{Atom} \qquad \frac{}{\top \in \mathsf{Type}}\ \textsc{Top} \qquad \frac{}{\bot \in \mathsf{Type}}\ \textsc{Bottom}$$

$$\frac{A \in \mathsf{Type} \qquad B \in \mathsf{Type}}{A \to B \in \mathsf{Type}}\ \textsc{Arr} \qquad \frac{A \in \mathsf{Type} \qquad B \in \mathsf{Type}}{A \times B \in \mathsf{Type}}\ \textsc{Prod}$$

$$\frac{A \in \mathsf{Type} \qquad B \in \mathsf{Type}}{A + B \in \mathsf{Type}}\ \textsc{Union}$$

Context Formation

$$\frac{}{\mathsf{nil}\ \ \mathsf{ctx}}\ \textsc{Nil} \qquad \frac{\Gamma\ \mathsf{ctx} \qquad A \in \mathsf{Type} \qquad x \text{ fresh in } \Gamma}{\Gamma,\ x : A\ \ \mathsf{ctx}}\ \Gamma\text{-}\textsc{Add}$$

Context Reflection

$$\frac{\Gamma\ \mathsf{ctx} \qquad x : A \in \Gamma}{\Gamma \vdash x : A}\ \Gamma\text{-}\textsc{Refl}$$

Top Introduction – Bottom Elimination

$$\frac{}{\Gamma \vdash \langle\rangle : \top}\ \top\mathrm{I} \qquad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathrm{abort}[A](M) : A}\ \bot\mathrm{E}$$

Function Construction and Application

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.\ M : A \to B}\ \lambda\text{–}\textsc{Abs} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash M^{'} : A}{\Gamma \vdash (MM^{'}) : B}\ \textsc{App}$$

Tuple Construction and Projections

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash M^{'} : B}{\Gamma \vdash \langle M, M^{'} \rangle : A \times B}\ \textsc{Tup}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \mathrm{fst}(M) : A}\ \mathrm{L}\textsc{Prj} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \mathrm{snd}(M) : B}\ \mathrm{R}\textsc{Prj}$$

Union Construction and Elimination

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathrm{inj_l}[B](M) : A + B}\ \textsc{Inj}\mathrm{L} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathrm{inj_r}[A](M) : A + B}\ \textsc{Inj}\mathrm{R}$$

$$\frac{\Gamma \vdash M : A + B \qquad \Gamma, x : A \vdash N : C \qquad \Gamma, y : B \vdash O : C}{\Gamma \vdash \text{case } M \text{ of } \text{inj}_{\text{l}}[B](x) \longmapsto N| \ \text{inj}_{\text{r}}[A](y) \longmapsto O : C} \vee\text{E}$$

3.2.1 Definitional Equality: Proof tree equalities as term equalities

Gentzen's principles transliterate to an equational system for terms. In the following we are defining a congruence relation on proof terms which is usually coined as *definitional equality* and denoted $M = M' : A$. We want definitional equality $=$ to be the least congruence closed under the β, η rules that directly reflect Gentzen's principles in term form.

Definition A *congruence* is

- an equivalence relation (i.e. reflexive, symmetric and transitive)
- that commutes with operators E.g.

$$\frac{\Gamma \vdash M = M' : A}{\Gamma \vdash \text{fst}(M) = \text{fst}(M') : A}$$

Informally , we should be able to replace "equals with equals" everywhere in a term.

Local Soundness

Local soundness captures the idea that "elimination rules are not too strong", which is the informal notion that the elimination rules should cancel the introduction rules and give nothing new. The so-called β equality rules are as follows:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \mathrm{fst}(\langle M, N\rangle) = M : A}\,\beta\wedge_1 \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \mathrm{snd}(\langle M, N\rangle) = N : B}\,\beta\wedge_2$$

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x : A.\ M)(N) = [N/x]M : B}\,\beta \supset$$

$$\frac{\Gamma, x : A \vdash N : C \qquad \Gamma, y : B \vdash O : C \qquad \Gamma \vdash P : A}{\Gamma \vdash (\mathrm{case}\ \ \mathrm{inj_l}[B](P)\ \ \mathrm{of}\ \ \mathrm{inj_l}[B](x) \longmapsto N|\ \ \mathrm{inj_r}[A](y) \longmapsto O) = [P/x]N : C}$$

$$\frac{\Gamma, x : A \vdash N : C \qquad \Gamma, y : B \vdash O : C \qquad \Gamma \vdash Q : B}{\Gamma \vdash (\mathrm{case}\ \ \mathrm{inj_r}[A](Q)\ \ \mathrm{of}\ \ \mathrm{inj_l}[B](x) \longmapsto N|\ \ \mathrm{inj_r}[A](y) \longmapsto O) = [Q/y]O : C}$$

Local completeness

Local completeness on the other hand captures the idea that "elimination rules are not too weak". Given any proof of a specific type one should be able to reintroduce it by extracting its constituents via elimination rules. Another way to see this is that canonical forms of a certain type structure should have certain shape.

The η rules (a.k.a. *identity expansion*) are given below. We assume that in the $\eta \to$ case the variable x does not appear free in the term M:

$$\frac{\Gamma \vdash M : \top}{\Gamma \vdash M = \langle\ \rangle : \top}\ \eta\top \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash M = \langle \mathrm{fst}(M), \mathrm{snd}(M) \rangle : A \times B}\ \eta\times$$

$$\frac{\Gamma \vdash M : A \to B}{\Gamma \vdash M = \lambda x : A.Mx : A \to B}\ \eta \to$$

$$\frac{\Gamma \vdash M : A + B}{\Gamma \vdash M = \text{case } M \text{ of} \quad \begin{array}{l} \mid \mathrm{inj_l}[B](x) \mapsto \mathrm{inj_l}[B](x) \\ \mathrm{inj_r}[A](y) \mapsto \mathrm{inj_r}[A](y) : A + B \end{array}}\ \eta\vee$$

3.3 Operational (a.k.a "term") Semantics

It is obvious that the system is consistent in terms of provability. It's forgetful projection is exactly IPL for which we have provided order-theoretic models. We would like to show consistency for the proof relevant model. One way is operational semantics. We will be discussing only the →, ∧ fragment.

The first step toward operational semantics is to break the symmetry of the definitional equivalence and construct a one-way reduction relation on lambda terms. Towards this definition we first define the notion of a *redex*[2]:

Definition. • A β-redex is every term of the form:

$$(\lambda x : A.N)M|\,\mathrm{fst}(\langle M, N\rangle)|\,\mathrm{snd}(\langle M, N\rangle)$$

• An η-redex is every term of the form:

$$\lambda x : A.Mx|\langle \mathrm{fst}\, M, \mathrm{snd}\, M\rangle$$

A *normal form* is a term where no redex occurs. To make the term surrounding the redex explicit, we can use a *term context*, i.e. a term with a single term hole, such as $\lambda x : A.[]$, $(e[\bullet])$, $[\bullet]e$, where a hole can be substituted for a term to give a larger term. To be strict all single hole terms have the following diagram:

$$H := [\bullet]|\ (M[\bullet])|\ ([\bullet]M)|\ \lambda x : A.H|\ \langle H, M\rangle|\ \langle M, H\rangle$$

Now we have enough tools to define the *(one-step) $\beta\eta$-reduction* between two terms that include redexes as follows:

One-step $\mapsto_{\beta\eta}$ reduction

$$(\lambda x : A.N)M \mapsto [M/x]N\ (\beta)$$

[2] we only present it for the × → part of the calculus

$$\text{fst}\langle M, N\rangle \mapsto M\ (\beta) \qquad \text{snd}\langle M, N\rangle \mapsto N\ (\beta)$$

$$\lambda x : Mx \mapsto M\ (\eta) \qquad \langle \text{fst}\, M, \text{snd}\, M\rangle \mapsto M\ (\eta)$$

$$\frac{M \mapsto M^{'}}{H[M] \mapsto H[M^{'}]}\ (\textsc{subterm})$$

Now we can define the reflexive, transitive closure of the previous relation as $\mapsto^*_{\beta\eta}$ to denote zero or more reduction steps. The following facts – leading to a computational proof of consistency – hold:

Theorem. Church – Rosser for $\mapsto_{\beta\eta}$ For every term M, if $M \mapsto_{\beta\eta} N_1$ and $M \mapsto_{\beta\eta} N_2$ then there exists $N^{'}$ s.t. $N_1 \mapsto N^{'}$ and $N_2 \mapsto N^{'}$

Theorem. Church – Rosser for $\mapsto^*_{\beta\eta}$ For every term M, if $M \mapsto^*_{\beta\eta} N_1$ and $M \mapsto^*_{\beta\eta} N_2$ then there exists $N^{'}$ s.t. $N_1 \mapsto^*_{\beta\eta} N^{'}$ and $N_2 \mapsto^*_{\beta\eta} N^{'}$

The first consistency result for the equational system comes straight from the Church–Rosser properties. Since it is easy to show that for any terms M, N s.t. $\Gamma \vdash M = N : A$, based on the = axiomatization, there exists a finite sequence of terms $N_0, \ldots, N_i$ such that $M \mapsto^*_{\beta\eta} N_0 \leftarrow\!\!\dashv^*_{\beta\eta} N_1 \mapsto^*_{\beta\eta} N_2 \leftarrow\!\!\dashv^*_{\beta\eta} \ldots \leftarrow\!\!\dashv_{\beta\eta} N_i$. Now we can obtain:

Theorem. Definitional equality implies common contractum For any terms, M,N if $\Gamma \vdash M = N : A$ then there exists term L s.t. $M, N \mapsto^*_{\beta\eta} L$

And as a result:

Theorem. Consistency of definitional equality of terms The definitional equality $=$ is not trivial i.e. it won't equate any two terms.

Moving toward consistency of the whole system (i.e. there is not term of type $\perp$), we prove a theorem for the existence of normal forms.

Theorem. Weak normalization theorem For any term M, there exists a finite sequence of terms s.t. $M \mapsto_\beta N_0 \mapsto_\beta N_1 \mapsto_\beta N_2 \mapsto_\beta \ldots \mapsto_\beta N_i$ where N_i is a β normal form.

It is common place in metatheoretic proofs for such systems that induction on the structure of the term does not "go through". Intuitively, a reduction can be "enlarging" the term yet, it is doing progress based on a different kind of metric. The idea is that we can choose a reduction strategy such that the number of *redexes of a specific type* (to be defined soon) reduce. Here are the steps towards the proof. We omit redexes related to disjunction but the proof extends to such cases pretty easily.

Definition The *degree of a type* A is defined as follows:

- $\theta(P_i) = 1$ if P_i is atomic
- $\theta(A \times B) = \theta(A \to B) = \theta(A) + \theta(B) + 1$

Definition The *degree of a redex* is defined as follows:

- Given that the type of $\lambda x.M$ is of type $A \to B$ then $d((\lambda x.M)N) = \theta(A \to B)$
- Similarly, $d(\text{fst}\langle M, N\rangle) = \theta(A \times B)$ where $A \times B$ is the type of $\langle M, N\rangle$

- Similarly for the other kinds of redexes.

Definition The *degree of a term* $d(t)$ is defined as the supremum of the degrees of its redexes.

Now we can prove the following facts:

Theorem. 1. The degree of redex r is strictly larger than the degree of its type A: $\theta(A) < d(r)$

2. The degree of a redex (r) seen as term (t) can be smaller than its redex degree since it might include other redexes: $d(r) \leq d(t)$.
3. The term resulting from a substitution $M[N/x]$ has degree : $d(M[N/x]) \leq max(d(M), d(N), \theta(A))$ where A is the declared type of x in the type context.

Theorem. If $M \mapsto M'$ then $d(M) < d(N)$ and hence, if $M \mapsto^{+} N$ then $d(M) < d(N)$.

As a result we get a weak normalization theorem by induction on pairs $(d(M), k)$ where k is the number of redexes with degree $d(M)$.

Theorem. Weak Normalization Theorem For every term $\Gamma \vdash M : A$ there exists a normalization strategy such that $M \mapsto_{\beta}^{*} N$ and N is a normal form.

Combining with previous results we get:

Theorem. Consistency There is no (closed) term M for which $\vdash M : \bot$

Suppose the opposite and obtain a contradiction using the previous theorem; there is no way to obtain a normal form of a bottom type from the rules.

A stronger result is the strong normalization theorem which states that *every* strategy is normalizing. This result is relevant to concurrent implementations of reduction since it implies that the order in which redexes are consumed does not matter during the evaluation of expression.

The important idea behind the technique – that generalizes to proof of strong normalization for more complex calculi– is the concept of reducibility predicates (see, e.g. [47]).

3.3.1 The essence of proofs–as–programs

The proofs of normalization above are essentially of the same "proof strength" as the logical proof of cut elimination. In a nutshell, eliminating cuts is the same as normalizing proof terms (or, correspondingly, proof trees).

In reality, the slogan of the Curry-Howard isomorphism and, in general, of a type theoretic treatment to logic should be "Normalization as Cut Elimination". This aspect of the isomorphism can be articulated nicely following Sieg's extraction method [48], which showcases how a construction of a Cut-free sequent calculus comes naturally from an analysis of normal proofs in natural deduction.

Chapter 4

Justification Logic

In this chapter I will give an overview of JL. I will emphasize LP, the very first logic of justification, and its deep relation with IPL. My scaffolding will be based upon [6], [5] that reflect this relation. Beforehand, I will allow for a more general discussion on JL following [3] and other relevant papers.

4.1 A bird's eye view

According to [3]"Justification logics are epistemic logics which allow knowledge and belief modalities to be "unfolded" into justification terms." More specifically, in JL the modality in question is witnessed by a reason and propositions of the kind $\Box\phi$ become $t : \phi$ that reads "ϕ is justified by reason t". Witnesses in JL have structure and operations. Different choices of operators result in logics that explicate different modalities (K, T, $S4$, $S5$). In general, there is an infinite family of justification logics. For our purposes, and in addition to type theoretic approaches to logic, JL reveals a computational content for *validity* in classical terms. As we will see following [1], JL and especially its $S4$ counterpart *The Logic of Proofs* (LP), can provide a unified classical *semantics* for type theoretic formulations of intuitionistic logic. In addition, following [7, 45], JL mechanics can be viewed type theoretically to provide for modal typed systems that enrich computational type theories with "semantical" notions such as explicit reflection and modular binding.

4.2 Minimal Justification Logic J_0

To permit for an account of reasons, the logic is enriched with an extra sort (j) for justifications. The sort of propositions is then enriched with propositions of the kind $j : \phi$ with ϕ being a proposition. Here is the abstract syntax:

$$
\begin{aligned}
j &:= s_i|\ C_i|j_1 * j_2|j_2 + j_2 \\
\phi &:= P_i|\ \bot|\ \phi_1 \wedge \phi_2|\ \phi_1 \vee \phi_2|\ \phi_2 \supset \phi_2|\ \neg\phi|\ j : \phi
\end{aligned}
$$

Constants C_i are symbols that can be assigned to logic axioms that are assumed to be necessary. Weaker justification logics exist without any assignment of constants (empty *constant specifications*) or with partial constant specifications. Nevertheless, in order for the *rule of necessitation* to be admissible each axiom instance of the underlying propositional logic has to be assigned a constant. We will be coming back to this topic in later sections. Symbols s_i stand for variables.

A Hilbert–style axiomatization of J_0 is given below. Its components are Hilbert's axioms for propositional logic together with two basic rules for justification: *applicativity* and *concatenation.* Concatenation internalizes weakening of proofs.

Propositional Axioms

P1. $\vdash \phi \supset (\psi \supset \phi)$

P2. $\vdash (\phi \supset (\psi \supset \chi)) \supset ((\phi \supset \psi) \supset (\phi \supset \chi))$

P3. $\vdash \phi \supset \psi \supset \phi \wedge \psi$

P4. $\vdash \phi \supset \psi \supset \psi \wedge \phi$

P5. $\vdash \phi \supset \phi \vee \psi$

P6. $\vdash \psi \supset \phi \vee \psi$

P7. $\vdash (\phi \supset \psi) \supset (\neg\psi \supset \neg\phi)$

Justification Axioms

Times. $\vdash \mathsf{j} : (\phi \supset \psi) \supset (\mathsf{j}' : \phi \supset \mathsf{j} * \mathsf{j}' : \psi)$

PlusL. $\vdash \mathsf{j} : \phi \supset (\mathsf{j} + \mathsf{j}' : \phi)$

PlusR. $\vdash \mathsf{j} : \phi \supset (\mathsf{j}' + \mathsf{j} : \phi)$

The rule of the system is *Modus Ponens*.

Modus Ponens

$$\frac{\phi \supset \psi \qquad \phi}{\psi}\,\mathsf{MP}$$

For the rule of necessitation to be admissible, we need necessitation of axioms to be admissible. For that reason a constant specification is required. We focus here on axiomatically appropriate constant specification CS because of its relation to combinatorial calculi. An axiomatization of axiomatically appropriate CS given below. Elements of CS are pairs (C, ϕ) of polymorphic (i.e. *parametrized* over propositions) constants and propositions. The ! operator relates to a concept of internalization of justified statements, i.e. witnessing the existence of a justified statement with a (higher order) justification. We demand that all justified axiomatic schemes can be internalized.

Axiomatic CS

$$\frac{}{\vdash (\mathsf{C}_1[\phi,\psi],\ \phi \to (\psi \to \phi)) \in \mathsf{CS}}\ \mathsf{C}_1$$

$$\frac{}{\vdash (\mathsf{C}_2[\phi,\psi,\chi],\ (\phi \supset (\psi \supset \chi)) \supset ((\phi \supset \psi) \supset (\phi \supset \chi)))) \in \mathsf{CS}}\ \mathsf{C}_2$$

$$\frac{}{\vdash (\mathsf{C}_3[\phi,\psi],\ \phi \supset \psi \supset \phi \wedge \psi) \in \mathsf{CS}}\ \mathsf{C}_3$$

$$\frac{}{\vdash (\mathsf{C}_4[\phi,\psi],\ \phi \supset \psi \supset \psi \wedge \phi) \in \mathsf{CS}}\ \mathsf{C}_4$$

$$\frac{}{\vdash (\mathsf{C}_5[\phi,\psi],\ \phi \supset \phi \vee \psi) \in \mathsf{CS}}\ \mathsf{C}_5 \qquad \frac{}{\vdash (\mathsf{C}_6[\phi,\psi],\ \psi \supset \phi \vee \psi) \in \mathsf{CS}}\ \mathsf{C}_6$$

$$\frac{}{\vdash (\mathsf{C}_7[\phi,\psi],\ (\phi \supset \psi) \supset (\neg\psi \supset \neg\phi) \in \mathsf{CS}}\ \mathsf{C}_7$$

$$\frac{}{\vdash (\mathsf{C}_8[\phi,\psi,j,j'],\ j : (\phi \supset \psi) \supset (j' : \phi \supset j * j' : \psi)) \in \mathsf{CS}}\ \mathsf{C}_8$$

$$\frac{\vdash (\mathsf{C},\phi) \in \mathsf{CS}}{\vdash (\mathsf{C}!,\ \mathsf{C} : \phi) \in \mathsf{CS}}\ C!$$

Finally we require reflection on CS:

Specification Reflection

$$\frac{\vdash (\mathsf{C},\phi) \in \mathsf{CS}}{\vdash \mathsf{C} : \phi}\ \mathrm{CSR}$$

The system can be given a Natural Deduction formulation *à la* IPL since the following theorem holds:

Deduction Theorem For any set of propositional assumptions Γ, $\Gamma, \phi \vdash \psi$ implies $\Gamma \vdash \phi \supset \psi$

4.3 Epistemic motivation

JL as an epistemic logic departs from previous traditions of logic of knowledge based on universality judgments. From [3]

> The modal approach to the logic of knowledge is, in a sense, built around the universal quantifier: X is known in a situation if X is true in all situations indistinguishable from that one. Justifications, on the other hand, bring an existential quantifier into the picture: X is known in a situation if there exists a justification for X in that situation

This fresh approach to the epistemic tradition has been utilized to solve many problems in formal epistemology (see [4]). We sketch here the solution to the famous *Red barn problem* that, also, provides a pedagogical example on how deduction in the system works.

The red barn problem can be stated as follows:

> Suppose I am driving through a neighborhood in which, unbeknownst to me, papier-mâché barns are scattered, and I see that

> the object in front of me is a barn. Because I have barn-before-me percepts, I believe that the object in front of me is a barn. Our intuitions suggest that I fail to know barn. But now suppose that the neighborhood has no fake red barns, and I also notice that the object in front of me is red, so I know a red barn is there. This juxtaposition, being a red barn, which I know, entails there being a barn, which I do not, "is an embarrassment"

The red barn example can be represented in a system of modal logic where $\Box\phi$ represents knowledge of ϕ that, in contrast to the the justified approach, is forgetful with respect to reasons. The formalization and the accompanying problem go as follows:

1. $\Box B$, 'I believe that the object in front of me is a red barn'.
2. $\Box(B \wedge R)$, 'I believe that the object in front of me is a red barn'.

   At the metalevel, 2 is actually knowledge, whereas by the problem description, 1 is not knowledge.
3. $\Box(B \wedge R \supset B)$, a knowledge assertion of a logical axiom.

> Within this formalization, it appears that epistemic closure in its modal form (2) is violated:line 2, $\Box(B\wedge R)$, and line 3, $(B\wedge R \supset B)$ are cases of knowledge whereas $\Box B$ (line 1) is not knowledge. The modal language here does not seem to help resolving this issue.

Of course, one can resolve this by introducing a second modality(e.g. for "I believe that"). But then similar problems can occur (e.g. by adding a third modality read as 'it should be'). Indexing of modalities with reasons solves this problem in its generality: by permitting the applicative closure only on reasons of the same sort one can overcome this defect.

1. $u : B$, 'u is a reason to believe that the object in front of me is a barn';
2. $v : (B \wedge R)$, 'v is a reason to believe that the object in front of me is a red barn';
3. $a : (B \wedge R \supset B)$, because of logical awareness.

   On the metalevel, the problem description states that 2 and 3 are cases of knowledge, and not merely belief, whereas 1 is belief which is not knowledge. Here is how the formal reasoning goes:
4. $a : (B \wedge R \supset B) \supset (v : (B \wedge R) \supset a * v : B)$, by Times
5. $v : (B \wedge R) \supset a * v : B$, from 3 and 4, by propositional logic; $a * v : B$, from 2 and 5, by propositional logic.

4.4 Proof theoretic view

In Chapter 2 we gave an analytic account of the BHK principles of constructive proofs. In the paper "Eine Interpretation des intuitionistischen Aussagenkalküls ", Göedel gave a classical provability interpretation of BHK using the modal system S4.

The standard axiomatization of S4 is given below:

The system S4

P1 – P7

K. $\vdash \Box(\phi \supset \psi) \supset (\Box\phi \supset \Box\psi)$

T. $\vdash \Box\phi \supset \phi$

4. $\vdash \Box\phi \supset \Box\Box\phi$

Modus Ponens

$$\frac{\phi \supset \psi \qquad \phi}{\psi}\,\mathsf{MP}$$

Göedel's result can be summarized in the following theorem:

Gödel-Tarski Translation of Intuitionistic Logic

$$\Gamma \vdash_{\mathsf{IPL}} \phi \rightharpoondown \Gamma \vdash_{\mathsf{S4}} \mathrm{tr}(\phi)$$

where tr(ϕ) is obtained by ϕ by $\Box$-ing its subformulas.

After this result the state of the project of a classical interpretation of BHK semantics was as follows: IPC $\hookrightarrow$ S4 $\hookrightarrow$? $\hookrightarrow$ CLASSICAL PROOFS. Filling the missing part was the motivation behind LP, the first Justification Logic.

4.5 The Logic of Proofs

An axiomatization of LP with axiomatically appropriate constant specification as defined in 4.2 can be given as follows:

The system LP

P1 – P7

Times. $\vdash j : (\phi \supset \psi) \supset (j' : \phi \supset j * j' : \psi)$

PlusL. $\vdash j : \phi \supset (j + j' : \phi)$

PlusR. $\vdash j : \phi \supset (j' + j : \phi)$

T. $\vdash j : \phi \supset \phi$

4. $\vdash j : \phi \supset (j! : j : \phi)$

4.6 Metatheoretic Results

The *Deduction Theorem* holds for LP

Deduction Theorem Any deduction of the kind $\Gamma, \phi \vdash \psi$ implies $\Gamma \vdash \phi \supset \psi$.

Also, the lifting property can be obtained:

Lifting Lemma

Any deduction of the kind $\vec{j} : \Gamma, \Delta \vdash \phi$ implies $\vec{j} : \Gamma, \vec{s} : \Delta \vdash j'(\vec{j}, \vec{s}) : \phi$ where $\vec{j}$ is a vector metavariables to be substituted for arbitrary polynomials and $\vec{s}$ is a vector of (object) variables.

In addition, LP is the forgetful projection of $S4$. More specifically, consider a formula of LP ϕ and the transformation $F_{\Box}(\phi)$ that replaces all subformulae of ϕ of the kind $j : \phi'$ with $\Box\phi'$. The following theorem holds:

Forgetful Projection Property

$\Gamma \vdash_{\mathsf{LP}} \phi$ implies $\Gamma \vdash_{\mathsf{S4}} F_{\Box}(\phi)$

The inverse also holds as the realization theorem says. Before introducing the realization procedure we give a motivating example.

Example: Realization of $\vdash_{\mathsf{S4}} \Box\phi \vee \Box\psi \supset \Box(\phi \vee \psi)$

1. $\phi \supset \phi \vee \psi$, $\psi \supset \phi \vee \psi$ Prop. Axioms;
2. $C : (\phi \supset \phi \vee \psi)$, $C' : (\psi \supset \phi \vee \psi)$ From CS rules.
3. $s : \phi \supset C * s : \phi \vee \psi$, From 1,2 andTimes and MP
4. $t : \psi \supset C' * t : \phi \vee \psi$, Similarly
5. $C{*}s : \phi{\vee}\psi \supset (C{*}s{+}C'{*}t) : \phi{\vee}\psi$ and $C'{*}t : \phi{\vee}\psi \supset (C{*}s{+}C'{*}t) : \phi{\vee}\psi$, From Rplus, Lplus
6. $s : \phi \supset (C * s + C' * t) : \phi \vee \psi$, From 3,5 by Propositional Logic.
7. $t : \psi \supset (C * s + C' * t) : \phi \vee \psi$, From 4,5 by Propositional Logic.
8. $s : \phi \vee t : \psi \supset (C * s + C' * t) : \phi \vee \psi$, From 6,7 and Propositional Logic.

4.6.1 Realization

The realization theorem gives an algorithmic process for transforming cut-free deductions inS4 to LP. By an LP-realization of a modal formula ϕ we mean an assignment of proof polynomials to all occurrences of the modality inϕ. Let ϕ^r be the image of ϕ under a realization r.

The polarity of $\Box$s in a formula is relevant in realizations. We define positive and negative occurrences of modality in a formula and a sequent.

$\Box$ **Polarities**

1. The indicated occurrence of $\Box$ in $\Box\phi$ is of positive polarity;
2. any occurrence of $\Box$ in the subformula ϕ of $\psi \supset \phi$,$\psi \wedge \phi$, $\phi \wedge \psi$, $\psi \vee \phi$, $\phi \vee \psi$, $\Box\phi$,$\Gamma \Rightarrow \Delta, \phi$ – we will be defining $\Rightarrow$ momentarily – has the same polarity as the same occurrence of $\Box$ in ϕ.
3. any occurrence of $\Box$ in the subformula ϕ of $\neg\phi$, $\phi \supset \psi$, $\Gamma, \phi \Rightarrow \Delta$, has polarity opposite to the polarity of the very same occurrence of $\Box$ in ϕ.

Next we give a a cut-free sequent formulation of S4 (reference) with sequents $\Gamma \vdash \Delta$, where Γ and Δ are finite multisets of modal formulas. The left hand multisets are to be read conjunctively and the right hand ones disjunctively. The rules are the rules given below together with the typical structural ones.

$$\frac{}{\Gamma, \phi \vdash \phi, \Delta}\ \textsc{Refl} \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma, \neg\phi \vdash \Delta}\ \neg\mathrm{L} \qquad \frac{\phi, \Gamma \vdash \Delta}{\Gamma \vdash \neg\phi, \Delta}\ \neg\mathrm{R}$$

$$\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \wedge \psi \vdash \Delta}\ \wedge\mathrm{L} \qquad \frac{\Gamma \vdash \phi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \wedge \psi, \Delta}\ \wedge\mathrm{L}$$

$$\frac{\Gamma, \phi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \vee \psi \vdash \Delta}\ \vee\mathrm{L} \qquad \frac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \vee \psi, \Delta}\ \vee\mathrm{R}$$

$$\frac{\Gamma \vdash \phi, \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \supset \psi \vdash \Delta}\ \supset\mathrm{L} \qquad \frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \supset \psi, \Delta}\ \supset\mathrm{R}$$

$$\frac{\phi, \Gamma \vdash \Delta}{\Box\phi, \Gamma \vdash \Delta}\ \Box\mathrm{L} \qquad \frac{\Box\Gamma \vdash \phi}{\Box\Gamma \vdash \Box\phi}\ \Box\mathrm{R}$$

Relevant in the realization proof is the sequent formulation of LP, the system LPG which enjoys the cut-elimination property resulting in the system LPG^{-}. The rules relevant to justifications are given below.

$$\frac{\Gamma, \phi \vdash \phi, \Delta}{\Gamma, t : \phi \vdash \phi, \Delta} :\!\mathrm{L} \qquad \frac{\Gamma \vdash t : \phi, \Delta}{\Gamma \vdash !t : t : \phi, \Delta} !\mathrm{R} \qquad \frac{\Gamma \vdash t : \phi, \Delta}{\Gamma \vdash (t + s) : \phi, \Delta} +\mathrm{L}$$

$$\frac{\Gamma \vdash t : \phi, \Delta}{\Gamma \vdash (s + t) : \phi, \Delta} +\mathrm{R} \qquad \frac{\Gamma \vdash s : \phi \supset \psi, \Delta \qquad \Gamma \vdash t : \phi, \Delta}{\Gamma \vdash s * t : \psi, \Delta} *\mathrm{R}$$

$$\frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash c : \phi, \Delta} c\mathrm{R}$$

Utilizing the previous systems the realization theorem shows:

Realization Theorem If $\Gamma \vdash_{\mathsf{S4}} \phi$ then there is a *normal* realization s.t. $\Gamma \vdash_{\mathsf{LP}} \phi^r$. By normal we mean a realization for which all occurrences of $\Box$ are realized by proof variables and the corresponding constant specification is injective.

4.6.2 Kripke - Fitting Semantics

In this section I will be discussing Kripke – Fitting Semantics[19] for Justification Logic $\mathsf{J}_0 + \mathsf{CS}$ very briefly.

A possible world justification logic model for the system $\mathsf{J}_0 + \mathsf{CS}$ is a structure $M = \langle G, R, E, V \rangle$. $\langle G, R \rangle$ is a standard K frame, where G is a set of possible worlds and R is a binary relation on it. V is a mapping from propositional variables to subsets of G, specifying atomic truth at possible worlds. E is an evidence function that maps pairs of justification terms and formulas to sets of worlds.

Given such a model, we define the $\models$ relation as follows:

$\forall \Gamma \in G$

$M, \Gamma \models P$ iff $\Gamma \in V(P)$ for P a propositional letter

- It is not the case that $M, \Gamma \models \bot$
- $M, \Gamma \models \phi \supset \psi$ iff it is not the case that $M, \Gamma \models \phi$ or$M, \Gamma \models Y$
- $M, \Gamma \models (j : \phi)$ if and only if $\Gamma \in E(j, \phi)$ and, $\forall \Delta \in G$ with $\Gamma R \Delta$, we have that $M, \Delta \models \phi$.

The following conditions on evidence functions are assumed:

$E(j, \phi \supset \psi) \cap E(j', \phi) \subseteq E(j * j', \psi)$

$E(j, \phi) \cup E(j', \phi) \subseteq E(j + j', \phi)$

Finally, the Constant Specification CS should be taken into account. Recall that constants are intended to represent reasons for basic assumptions that are accepted outright. A model $M = \langle G, R, E, V \rangle$ meets Constant Specification CS provided: if $(C, \phi) \in CS$ then $E(c, \phi) = G$.

Typical, soundness and completeness results can be shown for such models. They can also be extended for all other justification logics.

Chapter 5

Curry – Howard view of justification logic

In this and the subsequent chapters we suggest reading a constructive necessity of a formula ($\Box A$) as internalizing a notion of constructive truth of A (a proof within a deductive system I) and a validation of A (a proof under an interpretation $[\![A]\!]_J$ within some system J). An example of such a relation is provided by the simply typed lambda calculus (as I) and its implementation in SK combinators (as J). We utilize justification logic to axiomatize the notion of validity-under-interpretation and, hence, treat a "semantical" notion in a purely proof-theoretic manner. We present the system in Gentzen-style natural deduction formulation and provide reduction and expansion rules for the $\Box$ connective. Finally, we add proof-terms and proof-term equalities to obtain a corresponding calculus (Jcalc) in the next chapter. The obtained system can be viewed as an extension of the Curry–Howard isomorphism with justifications. We provide standard metatheoretic results and suggest a programming language interpretation in languages with foreign function interfaces (*FFI*s).

5.1 Introduction: Necessity and Constructive Semantics

In his seminal "Explicit Provability and Constructive Semantics" [2] Artemov developed a constructive, proof-theoretic semantics for BHK proofs [51] in what turned out to be the first development of a family of logics that we now call justification logic. The general idea, upon which we build our calculus, is that any notion of semantics for a deductive system I involves mappings of proof constructs of I into another proof system J (which we call justifications) and can, thus, be viewed in a solely proof-theoretic manner. As an example one could think I being Heyting arithmetic and J some "stronger" system (e.g. a classical axiomatization of Peano arithmetic, a classical or intuitionistic set theory etc). In Artemov's work I is assumed to be based on intuitionistic logic and J on classical logic. We, initially, mute such assumptions to focus exclusively on the mechanics of necessity in this framework. We recover them later and study their relation to the Rule of Necessitation for our system. What's more, such a semantic relation can be treated logically giving rise to a modality of explicit necessity. Different sorts of necessity have been offered explicit counterparts under the umbrella of justification logic. In fact, there is an infinite family of logics of justification. Some of them have been studied within a Curry–Howard setting [8]. Our paper focuses on K modality and should be viewed as the counterpart of [12] with justifications as we explain in 7.4.

5.1.1 Deductive Systems, Validity and Necessity

Following a framework championed by Lambek [33, 34], let us assume two deductive systems I (with propositional universe U_I, a possibly non-empty signature of axioms Σ_I and an entailment relation $\Sigma_I; \Gamma \vdash_I A$) and J (resp. with U_J, Σ_J and $\Sigma_J; \Delta \vdash_J \phi$). We will be using Latin letters for the formulae of I and Greek letters for the formulae of J. We will be omitting the Σ signatures when they are not relevant.

For the entailment relations of the two systems we require the following elementary principles[1]:

1. *Reflexivity.* In both relations Γ and Δ are multisets of formulas (contexts) that enjoy reflexivity:

$$A \in \Gamma \Longrightarrow \Gamma \vdash_I A$$

$$\phi \in \Delta \Longrightarrow \Delta \vdash_J \phi$$

[1] We are not excluding other connectives but by imposing such minimal requirements we show that “necessity” ($\Box$) connective can be treated generically and orthogonally of the presence of other connectives

2. *Compositionality.* Both relations are closed under deduction composition:

$$\Gamma \vdash_I A \text{ and } \Gamma^{'}, A \vdash_I B \Longrightarrow \Gamma, \Gamma^{'} \vdash_I B$$

$$\Delta \vdash_J \phi \text{ and } \Delta^{'}, \phi \vdash_J \psi \Longrightarrow \Delta, \Delta^{'} \vdash_J \psi$$

3. *Top.* Both systems have a distinguished top formula $\top$ for which under any Γ, Δ:

$$\Gamma \vdash_I \top_I \text{ and } \ \Delta \vdash_J \top_J$$

Now we can define:

Definition. Given a deductive system I, an *interpretation for I*, noted by $[\![\bullet]\!]_J$, is a pair $(J, [\![\bullet]\!])$ of a deductive system J together with a (functional) mapping $[\![\bullet]\!] : U_I \to U_J$ on propositions of I into propositions of J extended to multisets of formulae of U_I with the following properties:

1. *Top preservation.* $[\![\top_I]\!] = \top_J$
2. *Structural interpretation of contexts.* For Γ contexts of the form $A_1, \ldots, A_n$:

$$[\![\Gamma]\!] = [\![A_1]\!], \ldots, [\![A_n]\!]$$

(trivially empty contexts map to empty contexts. As in [33] they can be treated as the $\top$ element).

Definition. Given a deductive system I and an interpretation $[\![\bullet]\!]_J$ for I we define a *corresponding validation of a deduction* $\Sigma_I; \Gamma \vdash_I A$ as a deduction

$\Sigma_J; \Delta \vdash_J \phi$ in J such that $[\![A]\!] = \phi$ and $\Delta = [\![\Gamma]\!]$. We will be writing $[\![\Sigma_I; \Gamma \vdash_I A]\!]_J$ to denote such a validation.

Definition. Given a deductive system I, we say that an interpretation $[\![\bullet]\!]_J$ is *logically complete* when for all purely logical deductions $\mathcal{D}$ (i.e. deductions that make no use of Σ_I) in I there exists a corresponding (purely logical) validation $[\![\mathcal{D}]\!]$ in J^2. i.e.

$$\forall \mathcal{D}.\ \mathcal{D} : \Gamma \vdash_I A \Longrightarrow \exists [\![\mathcal{D}]\!] : [\![\Gamma \vdash A]\!]_J$$

Examples of triplets $(I,\ J,\ [\![\bullet]\!]_J)$ of logical systems that fall under the definition above are: any intuitionistic system mapped to a classical one under the embedding $[\![A \supset B]\!] = \tilde{\neg} A \tilde{\vee} B$ where $\tilde{\neg}$ and $\tilde{\vee}$ are classical connectives, the opposite direction under double negation translation, an intuitionistic system mapped to another intuitionistic system (i.e. a mapping of atomic formulas of I to atomic formulas of J extended naturally to the intuitionistic connectives or, simply, the identity mapping) etc. A vacuous validation (when $[\![\bullet]\!]_J$ maps everything to $\top$) gives another example. Note that we do not require ”soundness” of the system J since all that is required for K modality is admissibility of necessitation which is obtained even in the extreme case

[2] Note, that we require existence but not uniqueness. Nevertheless, if we treat deductive systems in a proof irrelevant manner as preorders the above definition gives uniqueness vacuously. In a more refined approach where I and J are viewed as categories of proofs the above “logical completeness” translates to the requirement that if the set of (purely logical) arrows $Hom_I(\Gamma, A)$ is non empty then $Hom_J([\![\Gamma]\!], [\![A]\!]_J)$ cannot be empty (i.e. that $[\![\bullet]\!]_J$ can be extended to a functor). We leave a complete categorical semantics of our logic for future work but we expect a generalization of the endofunctorial interpretations of K modality appearing in [12, 31].

where J proves and, hence, can validate everything.

The main thesis outlined in this chapter is that this notion of “double proof” (reasoning about proofs that exists in two related systems) provides for an understanding of necessity in proof theoretic terms. In addition, we argue, that this is the driver of (at least) the simplest form of necessity (K) that appears in justification logic (*necessity as internalization*). We will focus on the case where I (the propositional part of our logic) is based on the implicative fragment of intuitionistic logic and show how justification logic provides for an axiomatization of such logically complete interpretations $[\![\bullet]\!]_J$ of implicative intuitionistic logic. In what follows we provide a natural deduction for an intuitionistic system I (truth), an axiomatization/specification of $[\![\bullet]\!]_J$ (treated abstractly as a function symbol on types) and a treatment of basic necessity that relates the two deductions by internalizing a notion of “double truth” (proof in I and existence of corresponding validation in J).

5.2 Judgments of Jcalc

We aim for a reading of necessity that internalizes a notion of “double proof” in two deductive systems. Motivated by the discussion and definitions in the previous section we will treat the notion of interpretation abstractly – as a function symbol on types – and axiomatize in accordance. Intuitively we want:

$$\Box A \ \textsf{true} := A \ \textsf{true} \ \& \ A \ \textsf{valid} = A \ \textsf{true in I} \ \& \ [\![A]\!] \ \textsf{true in J}$$

We will be dropping indexes I, J since they can be inferred by the different kinds of assumption contexts. In addition, we omit signatures Σ since they do not offer anything from a logical perspective.

Logical entailment for the proposed $\Box$ connective can be summarized easily given our previous discussion. Given a deduction $\mathcal{D} : A \vdash B$ and the existence of validation $[\![\mathcal{D}]\!] : [\![A]\!] \vdash [\![B]\!]$ then given $\Box A$ (i.e. a proof of a $\vdash A$ and a validation $\vdash [\![A]\!]$) we obtain a double proof of B (and hence, $\Box B$) by *compositionality* of the underlying systems. Using standard, proof tree notation with labeled assumptions we formulate our rule of the connective in natural deduction:

$$\dfrac{\begin{array}{c}\vdots\\ \Box A\end{array} \qquad \begin{array}{c}\overline{}\ x\\ A\\ \vdots\\ B\end{array} \qquad \begin{array}{c}\overline{}\ s\\ [\![A]\!]\\ \vdots\\ [\![B]\!]\end{array}}{\Box B}\ I_{\Box B}E^{x,s}_{\Box A}$$

We can, easily, generalize to $\Box$ed contexts (of the form $\Box A_1, \ldots, \Box A_i$) of arbitrary length:

$$\dfrac{\begin{array}{c}\vdots\\ \Box A_1 \ldots \Box A_i\end{array} \qquad \begin{array}{c}\overline{}\ \vec{x}\\ \Gamma' : A_1, \ldots A_i\\ \vdots\\ B\end{array} \qquad \begin{array}{c}\overline{}\ \vec{s}\\ [\![\Gamma']\!] : [\![A_1]\!], \ldots [\![A_i]\!]\\ \vdots\\ [\![B]\!]\end{array}}{\Box B}\ I_{\Box B}E^{\vec{x},\vec{s}}_{\Box A_1 \ldots \Box A_i}$$

We read as "Introducing $\Box B$ after eliminating $\Box A_1 \ldots \Box A_i$ crossing out (vectors of) labels $\vec{x}, \vec{s}$ ". Interestingly, the same rule eliminates boxes and introduces new ones. This is not surprising for K modality (it is a left-right rule as we will see (5.2.4). See also discussion in [12, 13]). We will be referring to this rule as "$\Box$ Intro–After–Elim" or, simply $\Box_{IE}$, from now on.

Note that we define the $\Box$ connective negatively, yet (pure) introduction rules for the $\Box$ connective are derivable. Such are instances of the previous Intro–After–Elim rule when Γ' is empty which conforms exactly with the idea of necessity internalizing double theoremhood.

$$\frac{\vdash B \qquad \vdash [\![B]\!]}{\Box B}\, I_{\Box B}$$

In the next section, we provide the whole calculus in natural deduction format. As expected we will extend the implicational fragment of intuitionistic logic with

- Judgments about validity (justification logic).
- Judgments that relate truth and validity (modal judgments).

5.2.1 Natural Deduction for Jcalc

The treatment of necessity in the previous section is completely orthogonal to the underlying systems (it just assumes the basic requirements stated for the behavior $[\![\bullet]\!]$). In this section we will provide a full natural deduction and in congruence with justification logic we will assume that the underlying system (I) is a fragment of intuitionistic logic (the 'negative' to be precise). The host theory J can still remain unspecified, but the choice of I informs for some specifications (in order to preserve completeness of logical deductions).

Following type theory conventions, we first provide rules underlying type construction, then rules for well-formedness of (labeled) assumption contexts and rules introducing and eliminating connectives. The rules below should be obvious except for small caveat. On the one hand, the type universe of U_I and the proof trees of I are inductively defined as usual; on the other hand, the host theory J (its corresponding universe, connectives and proof trees) is "black boxed". What we actually axiomatize are the properties that all (logic preserving) interpretations of I should conform to, independently of the specifics of the host theory. Validity judgments should thus be read as specifications of provability (existence of proofs) of any candidate J.

When we write $[\![\Gamma]\!] \vdash [\![\phi]\!]$ it reads as there exists derivation $\mathcal{D} : \Delta \vdash_J \psi$ s.t. $\Delta = [\![\Gamma]\!]$ and $\psi = [\![\phi]\!]$). We use $\mathsf{Prop_0}$ to denote the type universe of I and $[\![\mathsf{Prop_0}]\!]$ to denote its image under an interpretation, $\mathsf{Prop_1}$ denotes modal ("boxed") types and Prop the union of $\mathsf{Prop_0}, \mathsf{Prop_1}$. We write P_k with k ranging in some subset of natural numbers to denote atomic propositions in I.

Judgments on Type Universe(s)

$$\frac{}{P_k \in \mathsf{Prop_0}}\textsc{Atom} \qquad \frac{}{\top \in \mathsf{Prop_0}}\textsc{Top} \qquad \frac{A \in \mathsf{Prop_i} \qquad B \in \mathsf{Prop_i}}{A \wedge B \in \mathsf{Prop_i}}\textsc{Conj}$$

$$\frac{A \in \mathsf{Prop_0}}{\Box A \in \mathsf{Prop_1}}\textsc{Box} \qquad \frac{A \in \mathsf{Prop_i} \qquad B \in \mathsf{Prop_j}}{A \supset B \in \mathsf{Prop_{max(i,j)}}}\textsc{Arr} \qquad \frac{A \in \mathsf{Prop_0}}{[\![A]\!] \in [\![\mathsf{Prop_0}]\!]}\textsc{Brc}$$

For labeled contexts of assumptions we require standard wellformedness conditions (i.e. uniqueness of labels). We use letters x_i, or simply x, for labels of contexts with assumptions in $\mathsf{Prop_0}$, x'_i or simply x' for contexts with assumptions in $\mathsf{Prop_1}$ and s_i, or simply s, for $[\![\mathsf{Prop_0}]\!]$ contexts. We use $\circ$ and $\Box\circ$ for the empty context of $\mathsf{Prop_0}$ and $\mathsf{Prop_1}$ respectively and $\dagger$ for the empty context of $[\![\mathsf{Prop_0}]\!]$. We abuse notation and write $x : A \in \Gamma$ (or, similarly, $s : [\![A]\!] \in \Delta$) to denote that the label x is assigned type A in Γ; or $\Gamma \in \mathsf{Prop_0}$ instead of $\Gamma \vdash \mathsf{wf_0}$ (resp. $\Gamma \in \mathsf{Prop_1}$, $\Delta \in [\![\mathsf{Prop_0}]\!]$) to denote that Γ is a wellformed context with co–domain of elements in $\mathsf{Prop_0}$ (resp. in $\mathsf{Prop_1}$, $[\![\mathsf{Prop_0}]\!]$). For $\Gamma \in \mathsf{Prop_0}$ we define $[\![\Gamma]\!]$ as the lifting of the context Γ through the $[\![\bullet]\!]$ symbol (with appropriate renaming of variables – e.g. $x_i \rightsquigarrow s_i$), similarly we define the $\Box\Gamma$ operation . For the vacuous cases when Γ is the empty context we require $[\![\circ]\!] = \dagger$ and $\Box\Gamma = \Box\circ$to be well formed.

Judgments on Context Wellformedness

$$\frac{}{\circ \vdash \mathsf{wf}_0}\ \textsc{Nil} \qquad\qquad \frac{\Gamma \vdash \mathsf{wf}_0 \qquad A \in \mathsf{Prop}_0 \qquad x \notin \Gamma}{\Gamma, x : A \vdash \mathsf{wf}_0}\ \Gamma\text{-}\textsc{Ext}$$

$$\frac{}{\dagger = [\![\circ]\!] \vdash [\![\mathsf{wf}_0]\!]}\ [\![\ \textsc{Nil}\ ]\!] \qquad \frac{\Gamma \vdash \mathsf{wf}_0}{[\![\Gamma]\!] \vdash [\![\mathsf{wf}_0]\!]}\ [\![\Gamma]\!] \qquad \frac{}{\Box\circ \vdash \mathsf{wf}_1}\ \Box\ \textsc{Nil} \qquad \frac{\Gamma \vdash \mathsf{wf}_0}{\Box\Gamma \vdash \mathsf{wf}_1}\ \Box\Gamma$$

In the following entry we define proof trees (in turnstile representation) of the intuitionistic source theory I. For all following rules we assume $\Gamma, A, B \in \mathsf{Prop}_0$:

Judgments on Truth $\Gamma, A, B \in \mathsf{Prop}_0$

$$\frac{x : A \in \Gamma}{\Gamma \vdash A}\ \Gamma_0\text{-}\textsc{Refl} \qquad\qquad \frac{}{\Gamma \vdash \top}\ \top_0\mathrm{I} \qquad\qquad \frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge_0\mathrm{I}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge_0\mathrm{E1} \qquad\qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge_0\mathrm{E2} \qquad\qquad \frac{\Gamma, x : A \vdash B}{\Gamma \vdash A \supset B}\ \supset_0\mathrm{I}$$

$$\frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \supset_0\mathrm{E}$$

For the calculus of interpretation (validity) we demand context reflexivity, compositionality and logical completeness with respect to intuitionistic implication. Logical completeness is specified axiomatically, since the host theory is “black boxed”. Following justification logic, we use an axiomatic characterization of combinatory logic (for $\supset$) together with the requirement that the interpretation preserves modus ponens:

Judgments on Validity with $\Delta \in [\![\mathsf{Prop}_0]\!]$

$$\frac{s : [\![A]\!] \in \Delta}{\Delta \vdash [\![A]\!]}\ \Delta\text{-}\textsc{Refl} \qquad \frac{}{\Delta \vdash [\![\top]\!]}\ \mathrm{Ax}_1 \qquad \frac{A, B \in \mathsf{Prop}_0}{\Delta \vdash [\![A \supset (B \supset A)]\!]}\ \mathrm{Ax}_2$$

$$\frac{A, B, C \in \mathsf{Prop}_0}{\Delta \vdash [\![A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C))]\!]}\ \mathrm{Ax}_3$$

$$\frac{A, B \in \mathsf{Prop}_0}{\Delta \vdash [\![A \supset (B \supset A \wedge B)]\!]}\ \mathrm{Ax}_4 \qquad \frac{A, B \in \mathsf{Prop}_0}{\Delta \vdash [\![A \wedge B \supset A]\!]}\ \mathrm{Ax}_5$$

$$\frac{A, B \in \mathsf{Prop}_0}{\Delta \vdash [\![A \wedge B \supset B]\!]}\ \mathrm{Ax}_6 \qquad \frac{\Delta \vdash [\![A \supset B]\!] \qquad \Delta \vdash [\![A]\!]}{\Delta \vdash [\![B]\!]}\ \mathrm{MP}$$

Finally, we have judgments in the □ed universe (Prop_1). These are context reflection, the □ Intro-After-Elim rule, and the rules for intuitionistic implication between □ed types [3].

[3] The implication and elimination rules in Prop_1 actually coincide with the ones in Prop_0 since we are focusing on the case where I is intuitionistic. This need not necessarily be the case as we have explained. Intuitionistic implication among □ types should be read as "double proof of A implies double proof of B" and would still be defined even if we did not observe any kind of implication in I. Similarly, one could provide intuitionistic conjunction or disjunction between □ types independently of I and, vice versa, one could add connectives in I that are not observed between □ed types.

Judgments on Necessity with

$\Gamma \in \mathsf{Prop}_1$**,** $\mathsf{length}(\Gamma) = i$**,** $1 \leq k \leq i$ **and,** $\Gamma', A, A_k, B \in \mathsf{Prop}_0$

$$\frac{x' : \Box A \in \Gamma}{\Gamma \vdash \Box A}\ \Gamma_1\text{-}\textsc{Refl}$$

$$\frac{(\forall A_i \in \Gamma'.\ \Gamma \vdash \Box A_i) \qquad \Gamma' \vdash B \qquad [\![\Gamma']\!] \vdash [\![B]\!]}{\Gamma \vdash \Box B}\ I_{\Box B} E^{\vec{x},\vec{s}}_{\Box A_1 \dots \Box A_i}$$

$$\frac{\Gamma, x' : \Box A \vdash \Box B}{\Gamma \vdash \Box A \supset \Box B} \supset_1 \mathrm{I} \qquad\qquad \frac{\Gamma \vdash \Box A \supset \Box B \qquad \Gamma \vdash \Box A}{\Gamma \vdash \Box B} \supset_1 \mathrm{E}$$

(Pure) $\Box I$ as derivable rule

We stress here that $\Box$ can be introduced positively with the previous rule with $\Gamma' = \circ$. The first premise reduces to a simple requirement that $\Gamma \in \mathsf{Prop}_1$.

$$\frac{\circ \vdash A \qquad \dagger \vdash [\![A]\!]}{\Gamma \vdash \Box A}\ I_{\Box A}$$

A simple derivation

We show here that the K axiom of modal logic is a theorem (omitting some obvious steps). In the following

$\Gamma := x_1' : \Box(A \supset B), x_2' : \Box A,\ \Gamma' = x_1 : A \supset B, x_2 : A,\ [\![\Gamma']\!] = s_1 : [\![A \supset B]\!], s_2 : [\![A]\!]$

$$\dfrac{\dfrac{\dfrac{\Gamma\vdash\Box(A\supset B)\qquad \Gamma\vdash\Box A\qquad \Gamma'\vdash B\qquad [\![\Gamma']\!]\vdash[\![B]\!]}{\Box(A\supset B),\Box A\vdash\Box B}\,I_{\Box A}E^{x_1,x_2,s_1,s_2}_{\Box A\supset B,\Box A}}{\Box(A\supset B)\vdash\Box A\supset\Box B}\supset_1\mathrm{I}}{\circ\vdash\Box(A\supset B)\supset\Box A\supset\Box B}\supset_1\mathrm{I}$$

5.2.2 Logical Completeness, Admissibility of Necessitation and Completeness with respect to Hilbert Axiomatization

In this section, we give a Hilbert axiomatization of the $\supset$ fragment of intuitionistic K logic in order to compare it with our system. Here $\vdash^{\mathcal{H}}$ captures the textbook (metatheoretic) notion of "deduction from assumptions" in a Hilbert style axiomatization. We assume the restriction of the system to formulas up to modal degree 1 as we have done throuhgout. This restriction is, firstly, for pedagogical purposes and, secondly, for pragmatic purposes related to the programming language applications that we are targeting. Nevertheless, in chapter 8 we sketch how such restrictions can be dropped.

Hilbert Style Formulation

AX1. $A\supset(B\supset A)$ AX2. $(A\supset(B\supset C))\supset((A\supset B)\supset(A\supset C))$

AX3. $(A\supset(B\supset A\wedge B))$ AX4. $(A\wedge B)\supset A)$ AX4. $(A\wedge B)\supset B)$

K. $\Box(A\supset B)\supset\Box A\supset\Box B$ MP $\dfrac{A\supset B\qquad A}{B}$ NEC $\dfrac{\vdash^{\mathcal{H}} A}{\Box A}$

It is easy to verify that axioms 1, 2 are derived theorems of Jcalc in $\mathsf{Prop_0}$. The rule Modus Ponens is trivially admissible, whereas axiom K was shown to be a theorem in the previous section (5.2.1). The rule of Necessitation is not obviously admissible though. In our reading of necessity the admissibility of this rule is directly related to the requirement of "logical completeness of the interpretation" i.e. preservation of logical theoremhood. In general, adding more connectives in I would require additional specifications for the host theory to obtain necessitation.

The steps of the proof are given in the Appendix, but this is essentially the "lifting lemma" in justification logic [2]. The proof makes essential use of the provability requirements imposed in the $[\![\mathsf{Prop_0}]\!]$ fragment.

$\Box$Lifting Lemma In Jcalc, for every $\Gamma, A \in \mathsf{Prop_0}$ if $\Gamma \vdash A$ then $[\![\Gamma]\!] \vdash [\![A]\!]$ and, hence, $\Box\Gamma \vdash \Box A$.

We get admissibility of necessitation as a lemma for Γ empty:

Admissibility of Necessitation For $A \in \mathsf{Prop_0}$, if $\circ \vdash A$ then $\circ \vdash \Box A$.

As a result:

Completeness Jcalc is complete with respect to the Hilbert style formulation of degree-1 intuitionistic K modal logic.

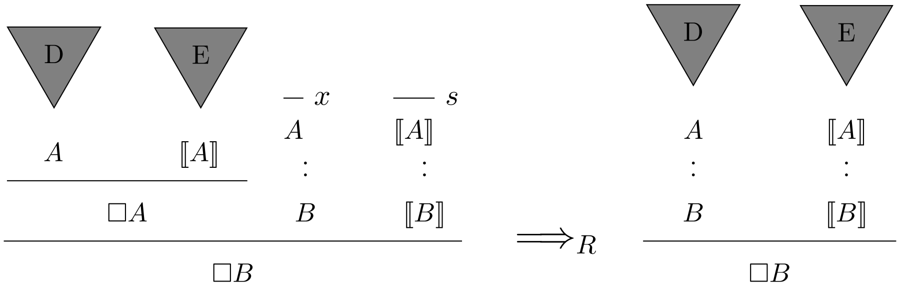
5.2.3 Harmony: Local Soundness and Local Completeness

Before we move on to show (Global) Soundness we provide evidence for the so-called "local soundness" and "local completeness" of the □ connective following Gentzen's dictum. The local soundness and completeness for the ⊃ connective is given elsewhere (e.g. [46]) and in Gentzen's original [20]. Gentzen's program can be described with the following two slogans:

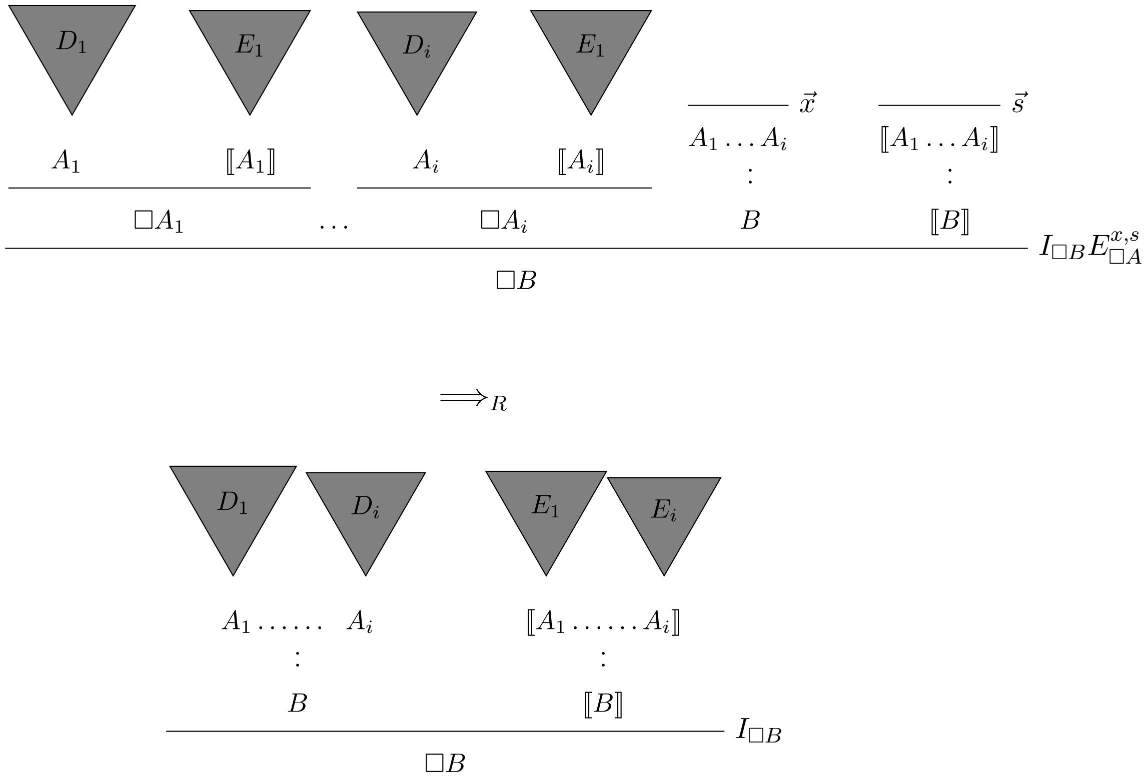
a. Elim is left-inverse to Intro

b. Intro is right-inverse to Elim

Applied to the □ connective, the first principle says that introducing a $\Box A$ (resp. many $\Box A_1, \ldots, \Box A_i$) only to eliminate it (resp. them) directly is redundant. In other words, the elimination rule cannot give you more data than what were inserted in the introduction rule(s) ("elimination rules are not *too* strong"). We show first the "Elim-After-Singleton-Intro" sub-case.



The exact same principle applies in the "Elim-after-Intro" of multiple □s:



These equalities are of importance since they dictate (together with the corresponding principles for the $\supset$, $\wedge$ connectives) the proof dynamics of the calculus. The proof term assignment and the corresponding computational (β-)rules are directly instructed by these reduction principles. We see that eliminating (using) an introduced $\Box$ corresponds to double substitution in the corresponding judgments.

Dually, the second principle says eliminating a $\Box A$, should give enough information to directly reintroduce it ("elimination rules are not *too weak*"). This is an expansion principle.

$$\begin{array}{c} \mathcal{D} \\ \Box A \end{array} \quad \Longrightarrow_E \quad \dfrac{\begin{array}{c}\mathcal{D}\\ \Box A\end{array} \qquad \dfrac{}{A}\,x \qquad \dfrac{}{[\![A]\!]}\,s}{\Box A}\, I_{\Box A} E^{x,s}_{\Box A}$$

5.2.4 (Global) Soundness

Soundness is shown by proof theoretic techniques. Standardly, we add the bottom type ($\bot$) to Jcalc together with its elimination rule and show that the system is consistent ($\nvdash \bot$) by devising a sequent calculus and showing admissibility of cut. We only present the calculus here and collect the theorems towards consistency in the Appendix.

In the following, we use $\Gamma \Rightarrow A$ (where $\Gamma, A \in \mathsf{Prop_0} \cup \mathsf{Prop_1}$) to denote sequents modulo Γ permutations where Γ is a multiset of Prop (no labels) and $\Delta \Rightarrow [\![\mathsf{A}]\!]$ for sequents corresponding to $[\![\mathsf{judgments}]\!]$ of the calculus modulo Δ permutations (with Δ (unlabeled) multiset of $[\![\mathsf{Prop_0}]\!]$). The multiset/ modulo permutation approach is instructed by standard structural properties. All properties are stated formally and proved in the Appendix.

The $[\![\Gamma]\!] \Rightarrow [\![A]\!]$ relation is defined directly from $\vdash$:

Sequent Calculus ($[\![\mathsf{Prop_0}]\!]$)

$$[\![\Gamma]\!] \Rightarrow [\![A]\!] := \quad \exists \Gamma' \in \pi([\![\Gamma]\!]) \text{ s.t } \Gamma' \vdash [\![A]\!]$$

where $\pi([\![\Gamma]\!])$ is the collection of permutations of $[\![\Gamma]\!]$.

Sequent Calculus (Prop)

$$\dfrac{}{\Gamma, A \Rightarrow A}\ Id \qquad \dfrac{\Gamma, A \supset B, B \Rightarrow C \qquad \Gamma, A \supset B \Rightarrow A}{\Gamma, A \supset B \Rightarrow C}\ \supset_L$$

$$\dfrac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\ \supset_R \qquad \dfrac{}{\Gamma, \bot \Rightarrow A}\ \bot_L \qquad \dfrac{\Gamma \Rightarrow A \qquad [\![\Gamma]\!] \Rightarrow [\![A]\!]}{\Box\Gamma \Rightarrow \Box A}\ \Box_{LR}$$

Standardly, we extend the system with the Cut rule and we obtain the extended system $\Gamma \Rightarrow^{+} A := \Gamma \Rightarrow A + \mathsf{Cut}$. We show Completeness of $\Rightarrow^{+}$ with respect to Natural Deduction and Admissibility of Cut that leads to the consistency result.

Consistency of Jcalc $\nvdash \bot$

Chapter 6

Order theoretic semantics

The previous chapter started by introducing mappings between deductive systems and motivating the reading of necessity as "double-proof under a map". As a result, it is unsurprising that the calculus is amenable to order theoretic semantics. We present them in this chapter.

6.1 semi-Heyting algebras

In order to progress we first define the notion of a *semi-Heyting Algebra (semi-HA)*. To define semi-HA we need the notion of a *(meet) semi-lattice*.

Definition: A *(meet) semi-lattice* is a non-empty *partial order* (i.e. reflexive, antisymmetric and transitive) with finite meets.

In addition, we define *meet semi-lattice* as follows:

Definition: A *bounded (meet) semi-lattice* $(L, \leq)$ is a (meet) semi-lattice that additionally has a *greatest element* (we name it 1), which satisfies

$x \leq 1$ for every x in L

Finally, we can define *semi-HA*:

Definition: A *semi-HA* is a bounded (meet) semi-lattice $(L, \leq, 1)$ s.t. for every $a, b \in L$ there exists an *exponential* (we name it $a \to b$) with the properties:

1. $a \to b \times a \leq b$
2. $a \to b$ is the greatest such element

6.2 Jcalc-triplets

Given two *semi-HAs*, we are interested in order preserving functions (functors) F that also preserve products and exponentials:

Definition A function F between two (semi)-HAs (HA_1, HA_2) is order preserving and commutes with top, products and exponentials *iff* for every $\phi, \psi \in HA_1$

1. $\phi \leq_{HA_1} \psi \Rightarrow F\phi \leq_{HA_2} F\psi$
2. $F\top_{HA_1} = \top_{HA_2}$
3. $F(\phi \times \psi) = F(\phi) \times (F(\psi)$
4. $F(\phi \to \psi) = F(\psi) \to F(\phi)$

For the order theoretic models of Jcalc the following structures (triplets) are of interest. We define a *Jcalc*-triplet as follows:

Definition A *Jcalc-triplet* is

1. A semi-Heyting algebra HA
2. A partial order J
3. An order preserving function F from HA to J s.t.
   (a) The image $F(HA)$ forms a semi-Heyting Algebra
   (b) F preserves top, products and exponentials

We are going to utilize the following definition:

Definition Given two partial orders $(K, \leq_K)$, $(L, \leq_L)$ and a function $(F : K \to L)$ we can define the algebra of F-points $(F : K \to L, \leq_{F:K\to L})$ where:

1. Elements of $F : K \to L$ are pairs of the form $\langle k, Fk \rangle$
2. $\langle k_1, Fk_1 \rangle \leq_F \langle k_2, Fk_2 \rangle$ *iff* $k_1 \leq_K k_2$ and $Fk_1 \leq_L Fk_2$

Theorem. For any triplet (K, L, F) of HAs with an order preserving function $F : K \to L$ the algebra of F-points is a partial order.

Proof. It is trivial to show that the algebra of F-points "inherits" reflexivity, transitivity and antisymmetry from the underlying algebras. □

Given a *Jcalc*-triplet there is an induced F-point algebra:

Definition Given a $Jcalc$-triplet we define the algebra $\Box^F HA$ as the induced F-point algebra.

By definition, the $\Box^F HA$ point algebra has the following properties:

1. Elements are pairs $\langle A, FA \rangle$ (name them $\Box^F A$) where $A \in HA$ and FA its image
2. For every two elements $\Box^F A$, $\Box^F B$:
   $\Box^F A \leq \Box^F B$ *iff* $A \leq_{HA} B$ *and* $FA \leq_J FB$
3. It is a Heyting algebra with:
   - $\Box^F \top := \langle \top_{HA}, F\top_{HA} = \top_J \rangle$
   - Elements of the form $\Box^F(A \times B)$ forming products (we name them $\Box^F A \times \Box^F B$)
   - Elements of the form $\Box^F(A \to B)$ forming exponentials (name them $\Box^F A \to \Box^F B$)

The last property is not obvious so we will sketch the proof. We will be omitting indexes in the $\leq$ relations since they can be trivially inferred:

$\Box^F HA$ is Heyting

Proof. $\Box^F \top$ is a top element since for any $A \in HA$, $A \leq \top$ and thusly $FA \leq F\top = \top_J$ and thus by definition $\Box^F A \leq \Box^F \top$ for any $\Box^F A$.

For any two elements $\Box^F A$, $\Box^F B$, the element $\Box^F(A \times B)$ forms their product since, $A \times B \leq A$ in HA and $F(A \times B) = FA \times FB \leq FA$ in J, and thusly, $\Box^F(A \times B) \leq \Box^F A$ (in $\Box^F HA$). Analogously, $\Box^F(A \times B) \leq \Box^F A$.

In addition, $\Box^F(A \times B)$ is the product we need to show that is the greatest element with the previous property. I.e. for any $\Box^F C$ s.t. $\Box^F C \leq \Box^F A$ and $\Box^F C \leq \Box^F B$ we get $\Box^F C \leq \Box^F(A \times B)$. By the definition for any such $\Box^F C$ we have $C \leq A \times B$ and $FC \leq F(A \times B)$ that imply $\Box^F C \leq \Box^F(A \times B)$.

To show that $\Box^F(A \to B)$ is the exponential of $\Box^F A$, $\Box^F B$ we have to show first that $\Box^F(A \to B) \times \Box^F A \leq \Box^F B$. By the $\Box^F HA$ product definition $\Box^F(A \to B) \times \Box^F A := \Box^F((A \to B) \times A)$. Also, by the underlying exponentials we have $(A \to B) \times A \leq B$ and $F((A \to B) \times A) = (FA \to FB) \times FA \leq FB$ that by definition of $\Box^F HA$ gives $\Box^F((A \to B) \times A) \leq \Box^F B$ and hence, by definition, $\Box^F(A \to B) \times \Box^F A \leq \Box^F B$.

In addition, we have to show that $\Box^F(A \to B)$ is the greatest element with the previous property. Consider any other $\Box^F C$ s.t. $\Box^F C \times \Box^F A \leq \Box^F B$, by definitions of $\Box^F$ and its products we obtain: $C \times A \leq B$ and $FC \times FA \leq FB$. By the definitions of the underlying exponentials we get $C \leq A \to B$ and $FC \leq FA \to FB = F(A \to B)$. And again by definition of $\Box^F HA$, $\Box^F C \leq \Box^F(A \to B)$. □

6.3 Jcalc algebras: Soundness and completeness

Given a *Jcalc*-triplet we can define a *Jcalc* algebra:

Definition Given a *Jcalc*-triplet we define the corresponding *Jcalc*-algebra as the union of the underlying relations of HA, $F(HA)$, $\Box^F HA$

Theorem. Soundness and completeness

$\Gamma \vdash_{Jcal} \phi$ iff for any *Jcalc algebra JC* (HA, F, J) and any $*$ map that extends a map of atomic Props (p_i) to elements of HA with properties shown below and $(+)$ is defined inductively on the length of Γ as shown below then $\Gamma^+ \leq \phi^*$.

$$\begin{aligned}
(\top)* &= \top \\
(A \wedge B \in Prop_0)* &= A * \times_{HA} B* \\
(A \supset B \in Prop_0)* &= A* \rightarrow_{HA} B* \\
(\llbracket A \rrbracket)* &= F(A*) \\
(\Box A)* &= \Box^F A* \\
(\Box A \supset \Box B)* &= \Box^F A* \rightarrow \Box^F B* \\
(\Box A \wedge \Box B)* &= \Box^F A * \times \Box^F B*
\end{aligned}$$

$$\begin{array}{rcl}
\circ^{+} & = & \top \\
\dagger^{+} & = & [\![\top]\!] \\
(\Box\circ)^{+} & = & \Box^{F}\top \\
(\Gamma, \phi \in \mathsf{Prop_0})^{+} & = & \Gamma^{+} \times_{HA} \phi* \\
([\![\Gamma]\!], [\![\phi]\!] \in [\![\mathsf{Prop_0}]\!])^{+} & = & \Gamma^{+} \times_{J} F\phi* \\
(\Box\Gamma, \Box\phi \in \mathsf{Prop_1})^{+} & = & \Gamma^{+} \times_{\Box^{F}HA} \Box^{F}(\phi)*
\end{array}$$

Proof. To prove soundness we go by induction on the derivations. For the $\mathsf{Prop_0}$ fragment the proof is well-known from intuitionistic logic semantics $(\Gamma \in \mathsf{Prop_0} \vdash \phi \in \mathsf{Prop_0} \Rightarrow \Gamma^{+} \leq_{HA} \phi*)$. For the $[\![\mathsf{Prop_0}]\!]$ part of the calculus again by induction. Reflection, of contexts is trivial. For the axiomatic cases, it is a well known result that in any Heyting algebra (and, subsequently, in $F(HA)$ of any Jcalc algebra) elements of the shape of the axiomatic combinators are equivalent (equiprovable) to $\top$. For example in any Heyting algebra we have $\top \leq A \to (B \to A)$ (using the definition of exponentials twice from the fact $\top \times A \times B \leq A$), the modus ponens case is handled by induction and the properties of F (preserving exponentials). Hence, $([\![\Gamma]\!])^{+}+ \leq ([\![\phi]\!])^{*}$ for any deduction in $[\![\mathsf{Prop_0}]\!]$.

The interesting part of the proof is the $\Box$ rule which we present again here for readability:

Judgments on Necessity

with $\Gamma \in \mathsf{Prop_1}$**,** $\mathsf{length}(\Gamma) = i$**,** $1 \leq k \leq i$ **and,** $\Gamma', A, A_k, B \in \mathsf{Prop_0}$

$$\frac{(\forall A_i \in \Gamma'.\ \Gamma \vdash \Box A_i) \qquad \Gamma' \vdash B \qquad [\![\Gamma']\!] \vdash [\![B]\!]}{\Gamma \vdash \Box B}\ I_{\Box B} E^{\vec{x},\vec{s}}_{\Box A_1 \ldots \Box A_i}$$

By the induction hypothesis we have $(\Gamma^{'})^{+} \leq B^{*}$ and $([\![\Gamma^{'}]\!])^{+} \leq FB^{*}$ or equivalently by the properties of F $F((\Gamma^{'})^{+}) \leq F(B^{*})$ which gives $\Box^{F}(\Gamma^{'})^{+} \leq \Box^{F}B$ Additionally from induction hypothesis, for every A_i in $\Gamma^{+}\Box^{F}A_i$ and by the product definition $\Gamma^{+} \leq (\Gamma^{'})^{+}$ and thus $\Gamma^{+} \leq \Box^{F}B*$.

For the inverse we create a Lindenbaum construction. We sketch the construction:

- Create a preorder *pre-HA* with underlying set (isomorphic to) $\mathsf{Prop_0}$
- Define $\phi \leq \psi$ *iff* $\phi \vdash \psi$
- Define the equivalence relation $\phi \equiv \psi$ *iff* $\phi \leq \psi$ and $\psi \leq \phi$
- Define the quotient *pre-HA*$/_{\equiv}$
- Show that it is a Heyting Algebra with products the elements of of shape $\phi \wedge \psi$, top $\top$ and exponentials $\phi \supset \psi$
- Repeat the construction for the syntactic elements of $[\![\mathsf{Prop_0}]\!]$, with $[\![\phi]\!] \leq [\![\psi]\!]$ *iff* $[\![\phi]\!] \vdash [\![\psi]\!]$. Show that it is a Heyting algebra J
- Repeat the construction for the syntactic elements of $\mathsf{Prop_1}$ and $\Box\phi \leq \Box\psi$ *iff* $\Box\phi \vdash \Box\psi$
- Show that the union of the three relations above forms a Jcalc algebra with

$F := A \mapsto [\![A]\!]$. I.e. show that:

- $A \vdash B \Rightarrow [\![A]\!] \vdash [\![B]\!]$ (Holds by the lifting lemma)
- $[\![A \wedge B]\!]$ is product (trivial) and $[\![A \supset B]\!]$ (trivial given the deduction theorem which we have shown) in J
- $\Box A \vdash \Box B$ *iff* $A \vdash B$ and $[\![A]\!] \vdash [\![B]\!]$ Easy by induction on the derivations and usage of the lifting lemma

Now assume that $\Gamma^+ \leq \phi*$ for any Jcalc algebra and mapping $*$; consider $*$ to extend the identity mapping into the (free) Jcalc algebra defined above. It is trivial to see that in Jcalc $\Gamma \vdash \phi$. □

Chapter 7

The computational side of Jcalc

In this section we add proof terms to represent natural deduction constructions. The meaning of these terms emerges naturally from Gentzen's principles that give reduction (computational β-rules) and expansion (extensionality η-rules) equalities for each construct. We focus on the new constructs of the calculus that emerge from the judgmental interpretation of the $\Box$ connective as explained in section 5.2. In addition, we focus on the *implicational* part. The proof term assignment for $\wedge$ rules is standard and can be added.

There will be no computational (reduction) rules on provability terms. This conforms with our reading of these terms as *references* to proof constructs of an *abstracted* theory J that can be realized differently for a concrete J.

7.1 Proof term assignment

The following rules and their correspondence with natural deduction constructs (5.2.1) should be obvious to the reader familiar with the simply typed λ-calculus and basic justification logic. We do not repeat the corresponding β, η equality rules since they are standard.

Judgments on Truth $\Gamma, A, B \in \mathsf{Prop}_0$ **and** $M := x_i \mid \ <> \ \mid \lambda x : A.\ M \mid (MM)$

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \Gamma_0\text{-}\textsc{Refl} \qquad \frac{}{\Gamma \vdash <>: \top}\ \top_0\mathrm{I} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.\ M : A \supset B}\ \supset_0\mathrm{I}$$

$$\frac{\Gamma \vdash M : A \supset B \qquad \Gamma \vdash M' : A}{\Gamma \vdash (MM^{'}) : B}\ \supset_0\mathrm{E} \qquad + \ \beta\eta \text{ equalities for } \ \top, \supset$$

For judgments of $[\![\mathsf{Prop}_0]\!]$, we assume a countable set of constant names and demand that every combinatorial axiom of intuitionistic logic has a witness under the interpretation $[\![\bullet]\!]$. This is what justification logicians call "axiomatically appropriate constant specification". As usual we demand reflection of contexts in J and preservation of modus ponens – closedness under some notion of application (which we denote as $*$).

Judgments on Validity $\Delta \in [\![\mathsf{Prop}_0]\!]$ **and** $\mathsf{J} := s_i \mid C_i \mid \mathsf{J} * \mathsf{J}$

$$\frac{s : [\![A]\!] \in \Delta}{\Delta \vdash s : [\![A]\!]}\ \Delta\text{-}\textsc{Refl} \qquad\qquad \frac{}{\Delta \vdash C_{\top} : [\![\top]\!]}\ \textsc{Ax}_1$$

$$\frac{A, B \in \mathsf{Prop}_0}{\Delta \vdash C_{K^{A,B}} : [\![A \supset (B \supset A)]\!]}\ \textsc{Ax}_2$$

$$\frac{A, B, C \in \mathsf{Prop}_0}{\Delta \vdash C_{S^{A,B,C}} : [\![A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C))]\!]}\ \textsc{Ax}_3$$

$$\frac{\Delta \vdash \mathsf{J} : [\![A \supset B]\!] \qquad \Delta \vdash \mathsf{J}' : [\![A]\!]}{\Delta \vdash \mathsf{J} * \mathsf{J}' : [\![B]\!]}\ \textsc{App}$$

If J is a proof calculus and $[\![\bullet]\!]_J$ is an interpretation such that the specifications above are realized, then J can witness intuitionistic provability. This can be shown by the proof relevant version of the lifting lemma that states:

$[\![\bullet]\!]$**Lifting Lemma** Given $\Gamma, A \in \mathsf{Prop}_0$ s.t. and a term M s.t. $\Gamma \vdash M : A$ then there exists J s.t $[\![\Gamma]\!] \vdash \mathsf{J} : [\![A]\!]$.

7.1.1 Proof term assignment and Gentzen Equalities for $\Box$ Judgments

Before we proceed, we will give a small primer of *let*-bindings as used in modern programming languages to provide for some intuition on how such terms work. Let us assume a rudimentary programming language that supports some basic types, say integers (int), as well as pairs of such types. Moreover, let us define a datatype Point as a pair of int i.e. as (int, int) In a language with *let*-bindings one can define a simple function that takes a Point and "shifts" it by adding 1 to each of its x and y coordinates as follows:

```
def shift (p:Point) =
let  (x,y) be p
in
(x+1,y+1)
```

If we call this function on the point (2,3), the computation

```
let (x,y) be (2,3) in (x+1,y+1)
```

is invoked. This expression reduces following the *let* reduction rule (i.e. pattern matching and substitution) to (2+1,3+1); and as a result we obtain the value (3,4). As we will see, *let* bindings – with appropriate typing restrictions for our system – are used in the assignment of proof terms for the $\Box_{IE}$ rule. Moreover, the reduction principle for such terms (β-rule) – obtained following Gentzen's equalities for the $\Box$ connective – is exactly the one that we just informally described.

We can now move forward with the proof term assignment for the $\Box_{IE}$ rule. We show first the sub-cases for Γ' empty (pure $\Box_I$) and Γ' singleton and explain the computational significance utilizing Gentzen's principles appropriated for the □ connective. We are directly translating proof tree equalities from 5.2.3 to proof term equalities. We generalize for arbitrary Γ' in the following subsection. We have, respectively, the following instances:

$$\frac{\Gamma \in \mathsf{Prop}_1 \qquad \circ \vdash M : B \qquad \dagger \vdash \mathsf{J} : [\![B]\!]}{\Gamma \vdash M \& \mathsf{J} : \Box B}$$

$$\frac{\Gamma \vdash N : \Box A \qquad x : A \vdash M : B \qquad s : [\![A]\!] \vdash \mathsf{J} : [\![B]\!]}{\Gamma \vdash \mathsf{let}\ (x \& s\ \ \mathsf{be}\ N)\ \mathsf{in}\ (M \& \mathsf{J}) : \Box B}$$

7.1.2 Gentzen's Equalities for (□ terms)

Gentzen's reduction and expansion principles give computational meaning (dynamics) and an extensionality principle for linking terms. We omit naming the empty contexts for economy.

$$\dfrac{\Box_I\,\dfrac{\Gamma \in \mathsf{Prop}_1 \qquad \vdash M : A \qquad \vdash \mathsf{j} : [\![A]\!]}{\Gamma \vdash M \& \mathsf{j} : \Box A} \qquad x : A \vdash M' : B \qquad s : [\![A]\!] \vdash \mathsf{j}' : [\![B]\!]}{\Gamma \vdash \mathsf{let}\ \ (x \& s)\ \ \mathsf{be}\ \ (M \& \mathsf{J})\ \ \mathsf{in}\ \ (M' \& \mathsf{J}') : \Box B}\ I_{\Box B} E^{x,s}_{\Box A}$$

$$\Longrightarrow_R$$

$$\frac{\Gamma \in \mathsf{Prop}_1 \qquad \vdash M'[M/x] : B \qquad \vdash \mathsf{J}'[\mathsf{J}/s] : [\![B]\!]}{\Gamma \vdash M'[M/x] \& \mathsf{J}'[\mathsf{j}/s] : \Box B}\ I_{\Box B}$$

Where the expressions $M'[M/x]$ and $\mathsf{J}'[\mathsf{J/s}]$ denote capture avoiding substitution, reflecting proof compositionality of the two calculi.

Following the expansion principle we obtain:

$$\Gamma \vdash M : \Box A \quad \Longrightarrow_E$$

$$\dfrac{\Gamma \vdash M : \Box A \qquad x : A \vdash x : A \qquad s : [\![A]\!] \vdash s : [\![A]\!]}{\Gamma \vdash \mathsf{let}\ (x\&s\ \mathsf{be}\ M)\ \mathsf{in}\ (x\&s) : \Box A}\ I_{\Box A}E^{x,s}_{\Box A}$$

That gives an η-equality principle as follows:

$$M : \Box A =_\eta \ \ \mathsf{let}\ \ (x\&s\ \mathsf{be}\ M)\ \ \mathsf{in}\ \ (x\&s) : \Box A$$

The η equality demands that every $M : \Box A$ should be reducible to a form $M'\&\mathsf{J}'$.

7.1.3 Proof term assignment for the $\Box$ rule (Generically)

After understanding the computational meaning of let expressions in the $\Box_{IE}$ rule we can now give proof term assignment for the rule in the general case(i.e. for Γ' of arbitrary length). We define a helper syntactic construct –$\mathsf{let}^* \ldots\ \mathsf{in}$ – as syntactic sugar for iterative let bindings based on the structure of contexts. The let^* macro takes four arguments: a context $\Gamma \in \mathsf{Prop_0}$, a context $\Delta \in [\![\mathsf{Prop_1}]\!]$, a possibly empty ([ ]) list of terms $Ns := N_1, \ldots, N_i$ - all three of the same length - and a term M. It is defined as follows for the empty and non-empty cases:

$\mathsf{let}^*\ (\circ;\ \dagger;\ [\ ])\ \mathsf{in}\ M := M$

$\mathsf{let}^*\ (x_1 : A_1, \ldots, x_i : A_i\ ;\ s_1 : \phi_1, \ldots, s_i : \phi_i;\ N_1, \ldots, N_i)\ \mathsf{in}\ M :=$

$\mathsf{let}\ \{(x_1 \& s_1)\ \mathsf{be}\ N_1, \ldots, (x_i \& s_i)\ \mathsf{be}\ N_i\}\ \mathsf{in}\ M$

Using this syntactic definition the rule $\Box_{IE}$ rule can be written compactly:

$\Box_{IE}$ **With** $\Gamma \in \mathsf{Prop}_1$**,** $\Gamma' \in \mathsf{Prop}_0$**,** $\mathsf{length}(\Gamma) = i$**,** $Ns := N_1 \ldots\ N_i$**,** $1 \leq k \leq i$

$$\frac{\forall A_k \in \Gamma'.\ \Gamma \vdash N_k : \Box A_k \qquad \Gamma' \vdash M : B \qquad [\![\Gamma']\!] \vdash \mathsf{J} : [\![B]\!]}{\Gamma \vdash \mathsf{let}^*\ (\Gamma', [\![\Gamma']\!], Ns)\ \mathsf{in}\ (M \& \mathsf{J}) : \Box B}\ I_{\Box B}E^{\vec{x},\vec{s}}_{\Box A_1 \ldots \Box A_i}$$

It is obvious that all previously mentioned cases are captured with this formulation. The rule of β-equality can be given for multi-let bindings directly from Gentzen's reduction principle (5.2.3) generalized for the multiple intro case:

$$\mathsf{let}\{(x_1 \& s_1)\ \mathsf{be}\ (M_1 \& \mathsf{J}_1), \ldots, (x_i \& s_i)\ \mathsf{be}\ (M_i \& \mathsf{J_i})\}\ \mathsf{in}\ (M \& \mathsf{J}) \quad =_\beta$$
$$M[M_1/x_1, \ldots, M_i/x_i] \& \mathsf{J}[\mathsf{J}_1/s_1, \ldots, \mathsf{J_i}/s_i]$$

7.2 Strong Normalization and small-step semantics

In the appendix (B) we provide a proof of normalization for natural deduction (via cut elimination). This is, mutatis mutandis, a strong normalization result for the proof term system too. A weaker result is normalization under a deterministic,“call-by-value” reduction strategy for β-rules. This gives an idea of how the system computes and we can use it in the applications in the next section. As usual we characterize a subset of the closed terms as values and we provide rules for the reduction of the non-value closed terms. Note that for the constants of validity and their applicative closure we do not observe reduction properties but treat them as values – again conforming with the idea of J (and its reduction principles) being “black boxed”.

Small step, call-by-value reduction $\to$

$$\frac{}{\lambda x.M\ \mathsf{value}} \qquad \frac{}{C_i\ \ \mathsf{value}} \qquad \frac{\mathsf{J}_1\ \mathsf{value} \qquad \mathsf{J}_2\ \mathsf{value}}{\mathsf{J}_1 * \mathsf{J}_2\ \mathsf{value}}$$

$$\frac{M\ \mathsf{value} \qquad \mathsf{J}\ \mathsf{value}}{M\&\mathsf{J}\ \mathsf{value}} \qquad \frac{M \to M'}{M\&\mathsf{J} \to M'\&\mathsf{J}}$$

$$\frac{N_1\ \mathsf{value}\ \ldots\ N_{k-1}\ \mathsf{value} \qquad N_k \to N_k'}{\begin{array}{c}\mathsf{let}\{(x_1\&s_1)\ \mathsf{be}\ N_1, \ldots,\ (x_k\&s_k)\ \mathsf{be}\ N_k, \ldots\}\ \mathsf{in}\ M \to \\ \mathsf{let}\{(x_1\&s_1)\ \mathsf{be}\ N_1, \ldots,\ (x_k\&s_k)\ \mathsf{be}\ N_k', \ldots\}\ \mathsf{in}\ M\end{array}}$$

$$\frac{M_1\&\mathsf{J}_1\ \mathsf{value}\ \ldots\ M_i\&\mathsf{J}_\mathsf{i}\ \mathsf{value}}{\begin{array}{c}\mathsf{let}\{(x_1\&s_1)\ \mathsf{be}\ (M_1\&\mathsf{J}_1), \ldots, (x_i\&s_i)\ \mathsf{be}\ (M_i\&\mathsf{J}_\mathsf{i})\}\ \mathsf{in}\ (M\&\mathsf{J}) \to \\ M[M_1/x_1, \ldots, M_i/x_i]\&\mathsf{J}[\mathsf{J}_1/s_1, \ldots, \mathsf{J}_\mathsf{i}/s_i]\end{array}}$$

$$\frac{M \to M'}{(MN) \to (M'N)} \qquad \frac{N \to N'}{((\lambda x.M)N) \to ((\lambda x.M)N')}$$

$$\frac{N\ \mathsf{value}}{((\lambda x.M)N) \to [N/x]M}$$

Using the reducibility candidates proof method [22]) we show:

Termination Under Small Step Reduction With $\to^*$ being the reflexive transitive closure of $\rightarrow$: for every closed term M and $A \in \mathsf{Prop}$ if $\vdash M : A$ then there exists N value s.t. $\vdash N : A$ and $M \rightarrow^* N$.

7.3 A programming language view: Dynamic Linking and separate compilation

Our type system can be related to programming language design when considering *Foreign Function Interfaces.* This is a typical scenario in which a language I interfaces another language J which is essentially "black boxed". For example, OCaml code might call C code to perform certain computations. In such cases I is a client and J is a host that provides implementations for an interface utilized by the client. In the course of software development, the implementations of such an interface might often change (i.e. a new version of the host language, or more dramatically, a complete switch of host language). We want a language design that satisfies two interconnected properties. Firstly, *separate compilation* i.e. when implementations change we do not have to recompile client code and, secondly, *dynamic linking* we want the client code to be linked dynamically to its new "meaning".

We will assume that both languages are functional and based on the lambda calculus. I.e. our interpretation function should have the property $[\![A \supset B]\!]_J = [\![A]\!]_J [\![\supset]\!]_J [\![B]\!]_J$ where $[\![\supset]\!]_J$ is the implication type constructor in J. The specifics of the host J and the concrete implementations are unknown to I but during the linker construction we assume that both languages share some basic types for otherwise typed "communication" of the two languages would be impossible. Simplifying, we consider that the only shared type is (int), i.e. the linker construction assumes $\bar{n} : [\![\mathsf{int}]\!]$ for every integer $n : \mathsf{int}$. Let us now assume source code in I that is interfacing a simple data structure, say an integer stack[1], with the following signature $\mathbf{\Sigma}$:

```
using type intstack
empty: intstack , push: int -> intstack -> intstack ,
pop: intstack -> int
```

And let us consider a simple program in I that is using the signature say,

```
pop(push (1+1) empty):int
```

This program involves two kinds of computations: a redex $(1 + 1)$ that can be reduced using the internal semantics of the language $1 + 1 \rightsquigarrow_I 2$ and the signature calls `pop (push 2 empty)` that are to be performed externally in whichever host language implements them. We treat dynamic linkers as "term re-writers" that map a computation to its meaning(s) based on different implementations. In the following we consider $\mathbf{\Sigma}$ to be the signature of the interface. Here are the steps towards the linker construction.

1. Reduce the source code based on the operational semantics of I until it

[1] The details of the stack implementation do not really matter here. It is only "gluing" types together correctly that is observed by our type system. Nevertheless, to avoid usage of pair types we assume that the `pop` operations are "impure". I.e. that they modify the very same object and return its top element.

doesn't have a redex:

$$\Sigma; \bullet \vdash \texttt{pop(push } (1+1) \texttt{ Empty)} \leadsto \texttt{pop(push 2 Empty)} : \texttt{int}$$

2. Contextualize the use of the signature at the final term in step 1:

$$\begin{aligned} \Sigma; \mathtt{x}_1 : \texttt{intstack}, \mathtt{x}_2 : \texttt{int} \to \texttt{intstack} \to \texttt{intstack}, \mathtt{x}_3 : \texttt{intstack} \to \texttt{int} \\ \vdash \mathtt{x}_3(\mathtt{x}_2\ 2\ \mathtt{x}_1) : \texttt{int} \end{aligned}$$

3. Rewrite the previous judgment assuming (abstract) implementations for the corresponding missing elements using the "known" specification for the shared elements.

$$\begin{aligned} \mathtt{s}_1 : [\![\texttt{instack}]\!], \mathtt{s}_2 : [\![\texttt{int} \to \texttt{intstack} \to \texttt{intstack}]\!], \mathtt{s}_3 : [\![\texttt{intstack} \to \texttt{int}]\!] \\ \vdash \mathtt{s}_3 * (\mathtt{s}_2 * \bar{2} * \mathtt{s}_1) : [\![\texttt{int}]\!] \end{aligned}$$

4. Combine the two previous judgments using the $\Box_{IE}$ rule.

$$\begin{aligned} \Sigma; \mathtt{x}_1' : \Box\texttt{intstack}, \mathtt{x}_2' : \Box(\texttt{int} \to \texttt{intstack} \to \texttt{intstack}), \\ \mathtt{x}_3' : \Box(\texttt{intstack} \to \texttt{int}) \\ \vdash \texttt{let}\{\mathtt{x}_1 \& \mathtt{s}_1 \texttt{ be } \mathtt{x}_1',\ \mathtt{x}_2 \& \mathtt{s}_2 \texttt{ be } \mathtt{x}_2',\ \mathtt{x}_3 \& \mathtt{s}_3 \texttt{ be } \mathtt{x}_3'\} \texttt{ in} \\ (\mathtt{x}_3(\mathtt{x}_2\ 2\ \mathtt{x}_1)\ \&\ \mathtt{s}_3 * (\mathtt{s}_2 * \bar{2} * \mathtt{s}_1)) : \Box\texttt{int} \end{aligned}$$

5. Using λ-abstraction three times we obtain the dynamic linker:

$$\begin{aligned}\Sigma;\circ \vdash \texttt{linker} = \lambda \texttt{x}'_1.\ \lambda \texttt{x}'_2.\lambda x'_3.\texttt{let}\{\texttt{x}_1\&\texttt{s}_1 \texttt{ be } \texttt{x}'_1,\ \texttt{x}_2\&\texttt{s}_2 \texttt{ be } \texttt{x}'_2,\ \texttt{x}_3\&\texttt{s}_3 \texttt{ be } \texttt{x}'_3\}\ \texttt{in}\\ (\texttt{x}_3(\texttt{x}_2\ 2\ \texttt{x}_1)\ \&\ \texttt{s}_3 * (\texttt{s}_2 * \bar{2} * \texttt{s}_1))\\ : \Box(\texttt{instack}) \to \Box(\texttt{int} \to \texttt{intstack} \to \texttt{intstack})\\ \to \Box(\texttt{intstack} \to \texttt{int}) \to \Box\texttt{int}\end{aligned}$$

Let us see how it can be used in the presence of different implementations:

1. Suppose the developer responsible for the implementation of the interface is providing an array based implementation for the stack in some language J i.e. we get references to type-checked code fragments of J as follows[2]:

$$\texttt{create}() : \texttt{intarray},\ \texttt{add\_array} : \texttt{int}_\texttt{J} \to_\texttt{J} \texttt{intarray} \to_\texttt{J} \texttt{intarray}$$

$$\texttt{pop\_array} : \texttt{intarray} \to_\texttt{J} \texttt{int}$$

2. A unification algorithm check is performed to verify the conformance of the implementations to the signature taking into account fixed type sharing equalities ($[\![\mathsf{int}]\!] = \mathsf{int}_\mathsf{J}$). In our case it produces:

$$[\![\to]\!] = \to_\mathsf{J}, [\![\mathsf{intstack}]\!] = \mathsf{intarray}$$

3. We thus obtain type-checked links using the $\Box_I$ rule. For example:

[2] We have changed the return type of pop to avoid products. This is just for economy and products can easily be handled.

$$\frac{\begin{array}{c}\Sigma;\circ \vdash \mathtt{push} : \mathtt{int} \to \mathtt{intstack} \to \mathtt{intstack} \\ \bullet \vdash \mathtt{add\_array} : [\![\mathtt{int} \to \mathtt{intstack} \to \mathtt{intstack}]\!]\end{array}}{\Sigma;\circ \vdash \mathsf{push} \;\&\; \mathsf{add\_array} : \Box(\mathtt{int} \to \mathtt{intstack} \to \mathtt{intstack})}$$

And analogously:

$$\Sigma;\circ \vdash \mathsf{pop} \;\&\; \mathsf{pop\_array} : \Box(\mathtt{intstack} \to \mathtt{int})$$

$$\Sigma;\circ \vdash \mathsf{empty} \;\&\mathsf{create}() : \Box\mathtt{intstack}$$

4. Finally we can compute the next step in the computation for the expression applying the linker to the obtained pairings:

$$\Sigma;\bullet\vdash (\mathtt{linker}\ (\mathtt{empty}\ \&\ \mathtt{create}())\ (\mathtt{push}\ \&\ \mathsf{add\_array})\ (\mathtt{pop}\ \&\ \mathtt{pop\_array})) \\ : \Box\mathtt{int}$$

which reduces to:

$$\begin{aligned}\Sigma;\bullet \vdash &\mathtt{let}\{(\mathtt{x}_1\&\mathtt{s}_1)\ \mathsf{be}\ (\mathsf{empty}\ \&\mathsf{create}()),\ (\mathtt{x}_2\&\mathtt{s}_2)\ \mathsf{be}\ (\mathsf{push}\ \&\ \mathtt{add\_array}), \\ &(\mathtt{x}_3\&\mathtt{s}_3)\ \mathsf{be}\ (\mathsf{pop}\ \&\ \mathtt{pop\_array})\} \\ &\mathtt{in}\ (\mathtt{x}_3(\mathtt{x}_2\ 2\ \mathtt{x}_1)\ \&\ \mathtt{s}_3 * (\mathtt{s}_2 * \bar{2} * \mathtt{s}_1)) : \Box\mathtt{int}\end{aligned}$$

The last expression reduces to (β-reduction for let):

$$\Sigma;\bullet \vdash\ \mathsf{pop}(\mathsf{push}\ 2\ \mathsf{empty})\ \&\ \mathtt{pop\_array} * (\mathsf{add\_array} * \bar{2} * \mathsf{empty}) : \Box\mathtt{int}$$

giving exactly the next step of the computation for the source expression. The good news is that the linker computes correctly the next step given any conforming set of implementations. It is easy to see that given a list implementation the very same process would produce a different computation step:

$$\Sigma; \bullet \vdash \ \mathtt{pop}(\mathtt{push}\ 2\ \mathtt{empty})\ \&\ \mathtt{pop\_list} * (\mathtt{Cons} * \bar{2} * []) : \Box\mathtt{int}$$

We conclude with some remarks that:

- The construction gives a mechanism of abstractions that works not only over different implementations in the same language but even for implementations in different (applicative) languages.
- We assumed in the example that the two languages are based on the lambda calculus and implement a curried, higher-order function space. It is easy to see that such host satisfies the requirements for the $[\![\bullet]\!]$ (with C_S, C_K being the S, K combinators in λ form and $*$ translating to λ application).
- Often, the host language of a foreign call is not a language that satisfies such specifications. This situation occurs when we have bindings from a functional language to a lower level language [3]. Such cases can be captured by adding conjunction (and pairs), tuning the specifications of J accordingly and loosening the assumption that $[\![\bullet]\!]$ is total on types.
- Introduction of modal types is clearly relative to the $[\![\bullet]\!]$ function on types. It would be interesting to consider examples where $[\![\bullet]\!]$ is realized by non-trivial mappings such as $[\![A \supset B]\!] = !A \multimap B$ from the embedding on intuitionistic logic to intuitionistic linear logic. That will showcase an

[3] In this setting the type signature of push would be: $\mathsf{int} \times \mathsf{intstack} \rightarrow \mathsf{instack}$

example of modality that works when lifting to a completely different logic or, correspondingly, to an essentially different computational model.

- Finally, it should be clear from the operational semantics and this example that we did not demand any equalities (or, reduction rules) for the proofs in J, but mere existence of specific terms. This is in accordance to justification logic. Analogously, we did not observe computation in the host language but only the construction of the linkers as program transformers. We were careful, to say that our calculus corresponds to the dynamic linking part of separate compilation. This, of course, does not tell the whole story of program execution in such cases. Foreign function calls, return the control to the client after the result gets calculated in the external language. For example, the execution of the program `pop (push 2 empty) + 2` should "escape" the client to compute the stack calls and then return for the last addition. Our modality captures exclusively the passing of control from the client to the host dynamically and, as such, is a K (non-factive) modality. Capturing the continuation of the computation and the return of the control back to the source would require a factive modality and a notion of "reverse" of the mapping $[\![\bullet]\!]$. We touch on this subject in the next chapter and we would like to explore such an extension in future work.

7.4 Related and Future Work

Directly related work with our calculus, in the same fashion that justification logic and LP [2] are related to modal logic, is [12]. The work in [12] provides a calculus for explicit assignments (substitutions) which is actually a sub-case of Jcalc with $[\![\bullet]\!]$ being identity. This sub-case captures dynamic linking where the host language is the very same one; such need appears in languages with module mechanisms (i.e. implementation hiding and separate compilation within the very same language). In general, the judgmental approach to modality is heavily influenced by [42]. In a sense, our treatment of validity-as-explicit-provability also generalizes the approach there without having to commit to a "factive" modality. Finally, important results on programming paradigms related to justification logic have been obtained in [8, 14, 11]. Immediate future developments would be to interpret modal formulas of higher degree under the same principles. This corresponds to dynamic linking in two or more steps (i.e., when the host becomes itself a client of another interface that is implemented dynamically in a third level and, so on). Some preliminary results towards this direction have been developed in [45] and we sketch them in the next section.

Chapter 8

Notes on extending the calculus

In this chapter we will make an informal case about the scalability of the presented system. We will sketch how the calculus can easily be extended in different ways and support that such extensions are of interest from a trinitarian (logic/ type theory/ category theory) point of view.

8.1 Extending on higher order modal types

We saw in Chapter 6 how the calculus corresponds to Jcalc algebras which are essentially pairs of Heyting algebras under an order preserving function. The points of such functions correspond to □ed types.

$$\begin{tikzcd}
A \arrow[r, equal, "\Box A"] \arrow[d, dash, "\vdash"] & FA \arrow[d, dash, "\vdash"] \\
B \arrow[r, equal, "\Box B"] & FB
\end{tikzcd}$$

This structure is easily extensible to account for □ed types of higher degree. Instead of a pair of Heyting algebras we could have a tower of Heyting algebras related with order preserving functions as shown in the schema.

$$\begin{tikzcd}
A \arrow[r, equal, "\Box A"] \arrow[d, dash, "\vdash"] \arrow[rr, bend left, dash, "\Box\Box A"] & F_0 A \arrow[r, equal, "\Box F A"] \arrow[d, dash, "\vdash"] & F_1 F_0 A \arrow[d, dash, "\vdash"] \\
B \arrow[r, equal, "\Box B"] \arrow[rr, bend right, dash, "\Box\Box B"'] & F_0 B \arrow[r, equal, "\Box F B"] & F_1 F_0 B
\end{tikzcd}$$

In a nutshell, instead of one function symbol $[\![\bullet]\!]$ the system can be axiomatized to reason about chains of composable (provability) preserving functions. The modifications required are minor to obtain such a system. Instead of one function symbol $[\![\bullet]\!]$ we have a collection $F_0, F_1 \ldots F_j$ and we define for any formula $\Box A \in \mathsf{Prop_i}$, $F_i \Box A := \Box F_i A$ (and, similarly, lifting over the connectives: $F_i(\Box A \supset \Box B) := \Box F_i A \supset \Box F_i B$). The rule can then be written:

Judgments on Necessity

with $\Gamma \in \mathsf{Prop_i}$**,** $\mathsf{length}(\Gamma) = j$**,** $1 \leq k \leq j$ **and,** $\Gamma', A, A_k, B \in \mathsf{Prop_{i-1}}$

$$\frac{(\forall A_i \in \Gamma'.\ \Gamma \vdash \Box A_i) \qquad \Gamma' \vdash B \qquad F_i \Gamma' \vdash F_i B}{\Gamma \vdash \Box B}\ I_{\Box B} E^{\vec{x},\vec{s}}_{\Box A_1 \ldots \Box A_j}$$

8.2 From order theory to category theory

There is a classic passage from orders to categories, which corresponds to the passage of provability to proof relevance. The main idea is that instances $(\Gamma \leq \phi)$ of the inequality relation arising from an order theoretic treatment of a deductive system are now refined to (equivalent classes) of arrows $M : \Gamma \Longrightarrow \phi$. Each term M is corresponding to a different deduction of ϕ from Γ. Order preserving functions become functors in the categorical scenario. But functors behave functionally on terms; they preserve proof equalities or, essentially, normalization principles of the cut elimination process. To account, hence, for a categorical semantics of the system one has to account for equality in the higher level of the system (i.e. on justifications).

This idea is actually not foreign in literature that explores the relation between lambda calculus and (typed) combinatory logic and, moreover, it is tempting to introduce equality between justifications so that one could more accurately describe computational phenomena arising when a language interacts with another language (or, its own modules).

Generalizing from the order theoretic semantics, we would expect a system in which $[\![\bullet]\!]$ would correspond to functors (preserving the connectives and hence, $\beta\eta$ equalities). We are expecting an extension of Jcalc with rules for $\beta\eta$ equalities on the level of justifications to fit exactly the bill.

8.3 Factivity and adjunctions

Having an understanding of the system in order theoretic/ categorical terms facilitates thinking about extensions. In any category theoretic textbook, the next “tighter” relation between categories is that of an adjunction. Interestingly, adjunctions arise from the relation of classical and intuitionistic proofs. Also, adjunctions play an important role in functional programming theory as the backbone of monadic computation.

We would expect that our view of necessity as a means of controlling two proof systems could be extended to cover such a notion. In order-theoretic terms, an adjunction between two partial orders $\mathcal{C}, \mathcal{D}$ is a pair of order preserving functions $L : \mathcal{D} \to C$, $R : \mathcal{C} \to D$ such that there is an isomorphism: $\forall d \in D, c \in C.\ Ld \leq c \longleftrightarrow d \leq Rc$. A logically interesting example of an adjunction is that between intuitionistic proofs and classical proofs where the left adjunct is inclusion and the right one is double negation translation and we have:

$$\frac{I(\Gamma) \vdash \phi}{\Gamma \vdash \neg\neg\phi} \downarrow\uparrow$$

To axiomatize such notions in $Jcalc$ one should add another function symbol to correspond to R and add the rule:

$$\frac{[\![\Gamma]\!] \vdash_J \mathsf{j} : \phi}{\Gamma \vdash \mathsf{return}\ (\mathsf{j}) : \mathsf{R}\phi}\ R$$

This is enough to obtain a generalized notion of Factivity:

$$\frac{[\![\Gamma]\!] \vdash M : \Box\phi}{\Gamma \vdash \mathsf{let\_s} = M\ \mathsf{in\ return}\ (\mathsf{s}) : R[\![\phi]\!]}\ R$$

Such an extension gives the standard factivity rule if the composition $R[\![\bullet]\!] = id$. With such a rule, a system can capture phenomena in which a language is giving control to another language partially to calculate a result but it retains control back for the continuation its program.

To conclude, our system captures a basic fragment of justification logic within a Curry–Howard isomorphism. Justification logic provides a rich, higher-order system with reflection. We expect that it can be deployed to provide foundations for richer typed programming language interaction.

Appendices

Appendix A

Theorems

Deduction Theorem for Validity Judgments With $\Delta \vdash [\![\mathsf{wf}]\!]$, if $\Delta, s : [\![A]\!] \vdash [\![B]\!]$ then $\Delta \vdash [\![A \supset B]\!]$.

Proof. The proof is essentially the deduction theorem for a Hilbert style formulation of the corresponding fragment of propositional logic and we do not show it here for economy. Note that this theorem cannot be proven without the logical specification Ax_1, Ax_2. I.e. it is exactly the requirements of the logical specification that ensure that all interpretations should adequately embed intuitionistic implication. □

$[\![\cdot]\!]$**Lifting Lemma** Given any wellformed context of assumptions $\Gamma \vdash \mathsf{wf}$ and $\Gamma, A \in \mathsf{Prop_0}$ then $\Gamma \vdash A \Longrightarrow [\![\Gamma]\!] \vdash [\![A]\!]$.

Proof. The proof goes by induction on the derivations trivially for all the cases($\supset_E$ is treated using the App rule that internalizes Modus ponens). For the $\supset I$ the previous theorem has to be used. □

□Lifting Lemma For $\Gamma, A \in \mathsf{Prop_0}$, then $\Gamma \vdash A$ implies $\Box\Gamma \vdash \Box A$.

Proof. Assuming a derivation $\mathcal{D} :: \Gamma \vdash A$, from the previous item, there exists corresponding validity derivation $\mathcal{E} :: [\![\Gamma]\!] \vdash [\![A]\!]$. Using the two as premises in the $\Box_{IE}$ with $\Gamma := \Box\Gamma$ we obtain $\Box\Gamma \vdash \Box A$. □

Let us show an inverse principle to the □ Lifting Lemma. We define for A in Prop_1:

$$\downharpoonleft (A_1 \supset A_2) = \downharpoonleft A_1 \supset \downharpoonleft A_2$$

$$\downharpoonleft \Box A = \downharpoonleft A$$

And the lifting of the $\downharpoonleft$ over $\Gamma \in \mathsf{Prop}$. We get:

Collapse □ Lemma If $\Gamma \vdash A$ for $\Gamma, A \in \mathsf{Prop_1}$ then $\downharpoonleft \Gamma \vdash \downharpoonleft$ A.

Weakening For the N.D. system of Jcalc, with $\Gamma, \Gamma' \vdash \mathsf{wf}$ and $\Delta, \Delta^{'} \vdash [\![\mathsf{wf}]\!]$.

1. If $\Gamma \vdash A$ then $\Gamma, \Gamma' \vdash A$.
2. If $\Delta \vdash [\![A]\!]$ then $\Delta, \Delta^{'} \vdash [\![A]\!]$.

Proof. By induction on derivations. □

Contraction For the N.D. system of Jcalc, with $\Gamma, x : A, x' : A, \Gamma' \vdash \mathsf{wf}$ and $\Delta, s : [\![A]\!], s' : [\![A]\!], \Delta^{'} \vdash [\![\mathsf{wf}]\!]$.

1. If $\Gamma, x : A, x' : A, \Gamma' \vdash B$ then $\Gamma, x : A, \Gamma' \vdash B$.
2. If $\Delta, s : [\![A]\!], s' : [\![A]\!], \Delta^{'} \vdash [\![B]\!]$ then $\Delta, s : [\![A]\!], \Delta^{'} \vdash [\![B]\!]$.

Proof. By induction on derivations. □

Permutation For the N.D. system of Jcalc, with $\Gamma \vdash \mathsf{wf}$ and $\Delta \vdash [\![\mathsf{wf}]\!]$ and $\pi\Gamma$ and $\pi\Delta$ the collection of well-formed contexts of assumptions with the same co-domain of Γ, Δ we get

1. If $\Gamma \vdash A$ and $\Gamma' \in \pi\Gamma$ then $\Gamma' \vdash A$.
2. If $\Delta \vdash [\![A]\!]$ and $\Delta^{'} \in \pi\Delta$ then $\Delta' \vdash [\![A]\!]$.

Proof. By induction on derivations. □

Substitution Principle The following hold for both kinds of judgments:

1. If $\Gamma, x : A \vdash B$ and $\Gamma \vdash A$ then $\Gamma \vdash B$
2. If $\Delta, s : [\![A]\!] \vdash [\![B]\!]$ and$\Delta \vdash [\![A]\!]$ then $\Delta \vdash [\![B]\!]$

All previous theorems can be stated for proof terms too. Specifically:

Deduction Theorem / Emulation of λ abstraction With $\Delta \vdash [\![\mathsf{wf}]\!]$, if $\Delta, s : [\![A]\!] \vdash \mathsf{j} : [\![B]\!]$ then there exists j' s.t. $\Delta \vdash \mathsf{j}' : [\![A \supset B]\!]$.

$[\![\cdot]\!]$**Lifting Lemma for terms** If $\Gamma A \in \mathsf{Prop_0}$ and $\Gamma \vdash M : A$ then there exists j s.t. $[\![\Gamma]\!] \vdash \mathsf{j} : [\![A]\!]$.

In both theorems the existence of this j, j' is algorithmic following the induction principle of the proof.

Appendix B

Notes on the cut elimination proof and normalization of natural deduction

Standardly, we add the bottom type and elimination rule in the natural deduction and show that in Jcalc + ⊥: $\nvdash \bot$. The addition goes as follows:

$$\frac{}{\bot \in \mathsf{Prop}_0}\ \textsc{Bot} \qquad\qquad \frac{\Gamma \vdash \bot \qquad A \in \mathsf{Prop}}{\Gamma \vdash A}\ E_{\bot}$$

Our proof strategy follows directly [39]. We construct an intercalation calculus [48] corresponding to the Prop fragment with the following two judgments:

$A \Uparrow$ for “Proposition A has normal deduction”.

$A^{\downarrow}$ for “Proposition A is extracted from hypothesis”.

This calculus is, essentially, restricting the natural deduction to canonical derivations. The $[\![\mathsf{judgments}]\!]$ are not annotated and are directly ported from the natural deduction since we observe consistency in Prop. The construction is identical to [39] (Chapter 3) for the $\mathsf{Hypotheses}, \mathsf{Coercion}, \supset, \perp$ cases, we add the $\Box$ case.

$$\frac{x : A \downarrow \in \Gamma^{\downarrow}}{\Gamma^{\downarrow} \vdash^{-} A \downarrow}\ \Gamma\text{-}\textsc{hyp} \qquad\qquad \frac{\Gamma^{\downarrow} \vdash^{-} A \downarrow}{\Gamma^{\downarrow} \vdash^{-} A \Uparrow}\ \downarrow\Uparrow$$

$$\frac{\Gamma^{\downarrow}, x : A \downarrow \vdash^{-} B \Uparrow}{\Gamma^{\downarrow} \vdash^{-} A \supset B \Uparrow} \supset\mathrm{I}^{x} \frac{\Gamma^{\downarrow} \vdash^{-} A \supset B \downarrow \qquad \Gamma^{\downarrow} \vdash^{-} A \Uparrow}{\Gamma^{\downarrow} \vdash^{-} B \downarrow} \supset\mathrm{E}$$

$$\frac{\Gamma^{\downarrow} \vdash^{-} \perp \downarrow \qquad A \in \mathsf{Prop}}{\Gamma^{\downarrow} \vdash^{-} A \Uparrow}\ E_{\perp} \qquad\qquad \frac{\Gamma \downarrow\vdash A \Uparrow \qquad [\![\Gamma]\!] \vdash [\![A]\!]}{\Box\Gamma \downarrow\vdash \Box A \Uparrow}\ \Box_{IE}$$

We prove simultaneously by induction:

Soundness of Normal Deductions The following hold:

1. If $\Gamma^{\downarrow} \vdash^{-} A \Uparrow$ then $\Gamma \vdash A$, and
2. If $\Gamma^{\downarrow} \vdash^{-} A \downarrow$ then $\Gamma \vdash A$.

Proof. Simultaneously by induction on derivations. □

It is easy to see that this restricted proof system $\nvdash\ ^{-}\bot \Uparrow$. It is hard to show its completeness to the non-restricted natural deduction ($\vdash +\bot_E$ of Jcalc) directly. For that reason we add a rule to make it complete ($\vdash^{+}$) preserving soundness and get a system of Annotated Deductions. We show the correspondence of the restricted system ($\vdash^{-}$) to a cut-free sequent calculus (JSeq), the correspondence of the extended system ($\vdash^{+}$) to $\mathsf{Jseq} + \mathsf{Cut}$ and show cut elimination.[1]

To obtain completeness we add the rule:

$$\frac{\Gamma^{\downarrow} \vdash A \Uparrow}{\Gamma^{\downarrow} \vdash A \downarrow} \Uparrow\downarrow$$

We define $\vdash^{+} := \quad \vdash^{-}$ with $\Uparrow\downarrow\mathsf{Rule}$. We show:

Soundness of Annotated Deductions The following hold:

1. If $\Gamma^{\downarrow} \vdash^{+} A \Uparrow$ then $\Gamma \vdash A$, and
2. If $\Gamma^{\downarrow} \vdash^{+} A \downarrow$ then $\Gamma \vdash A$.

Proof. As previous item. □

Completeness of Annotated Deductions The following hold:

1. If $\Gamma \vdash A$ then, $\Gamma \downarrow \vdash^{+} A \Uparrow$, and
2. If $\Gamma \vdash A$, then $\Gamma \downarrow \vdash^{+} A \downarrow$.

Proof. By induction over the structure of the $\Gamma \vdash A$ derivation. □

[1] In reality, the sequent calculus formulation is built exactly upon intuitions on the intercalation calculus. We refer the reader to the references.

Next we move with devising a sequent calculus formulation corresponding to normal proofs $\Gamma^{\downarrow} \vdash^{-} A \Uparrow$. The calculus that is given in the main body of this theorem. We repeat it here for completeness.

Sequent Calculus ($\llbracket \mathsf{Prop_0} \rrbracket$)

$$\Delta \Rightarrow \llbracket A \rrbracket := \quad \exists \Delta' \in \pi(\Delta) \text{ s.t } \Delta' \vdash \llbracket A \rrbracket$$

where $\pi(\Delta)$ is the collection of wellformed $\llbracket \mathsf{Prop_0} \rrbracket$ contexts $\Delta' \vdash \llbracket \mathsf{wf} \rrbracket$ with some permutation of the multiset Δ as co–domain.

Sequent Calculus (Prop)

$$\frac{}{\Gamma, A \Rightarrow A}\,Id \qquad \frac{\Gamma, A \supset B, B \Rightarrow C \qquad \Gamma, A \supset B \Rightarrow A}{\Gamma, A \supset B \Rightarrow C}\,\supset_L$$

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\,\supset_R \qquad \frac{}{\Gamma, \bot \Rightarrow A}\,\bot_L \qquad \frac{\Gamma \Rightarrow A \qquad \llbracket \Gamma \rrbracket \Rightarrow \llbracket A \rrbracket}{\Box\Gamma \Rightarrow \Box A}\,\Box_{LR}$$

We want to show correspondence of the sequent calculus w.r.t normal proofs ($\vdash^{-}$). Two lemmas are required to show soundness.

Substitution principle for extractions The following hold:

1. If $\Gamma_1^{\downarrow}, x : A^{\downarrow}, \Gamma_2^{\downarrow} \vdash^{-} B \Uparrow$ and $\Gamma_1^{\downarrow} \vdash^{-} A \Uparrow$ then $\Gamma_1^{\downarrow}, \Gamma_2^{\downarrow} \vdash^{-} B \Uparrow$
2. If $\Gamma_1^{\downarrow}, x : A^{\downarrow}, \Gamma_2^{\downarrow} \vdash^{-} B \downarrow$ and $\Gamma_1^{\downarrow} \vdash^{-} A \downarrow$ then $\Gamma_1^{\downarrow}, \Gamma_2^{\downarrow} \vdash^{-} B \Uparrow$

Proof. Simultaneously by induction on the derivations $A \downarrow$ and $A \Uparrow$. □

And making use of the previous we can show, with ($\downharpoonright A$ defined previously):

Collapse principle for normal deductions The following hold:

1. If $\Gamma^{\downarrow}, \vdash^{-} A \Uparrow$ then $\downharpoonright \Gamma^{\downarrow} \vdash^{-} \downharpoonright A \Uparrow$ and,
2. If $\Gamma^{\downarrow} \vdash^{-} A \downarrow$ then $\downharpoonright \Gamma^{\downarrow} \vdash^{-} \downharpoonright A \downarrow$

Using the previous lemmas and by induction we can show :

Soundness of the Sequent Calculus If $\Gamma \Rightarrow B$ then $\Gamma^{\downarrow} \vdash^{-} B \Uparrow$.

Soundness of the Sequent Calculus with Cut If $\Gamma \Rightarrow^{+} B$ then $\Gamma^{\downarrow} \vdash^{+} B \Uparrow$.

Next we define the $\Gamma \Rightarrow^{+} A$ as $\Gamma \Rightarrow A$ plus the rule:

$$\dfrac{\Gamma \Rightarrow^{+} A \qquad \Gamma, A \Rightarrow^{+} B}{\Gamma \Rightarrow^{+} B}\ \textsc{Cut}$$

Proof. As before. The cut rule case is handled by the $\Uparrow\downarrow$ and substitution for extractions principle showcasing the correspondence of the cut rule to the coercion from normal to extraction derivations. □

Standard structural properties (*Weakening, Contraction*) to show completeness. We do not show these here but they hold.

Completeness of the Sequent Calculus The following hold:

1. If $\Gamma^{\downarrow} \vdash^{-} B \Uparrow$ then $\Gamma \Rightarrow B$ and,
2. If $\Gamma^{\downarrow} \vdash^{-} A \downarrow$ and $\Gamma, A \Rightarrow B$ then $\Gamma \Rightarrow B$

Proof. Simultaneously by induction on the given derivations making use of the structural properties. □

Similarly we show for the extended systems.

Completeness of the Sequent Calculus with Cut The following hold:

1. If $\Gamma^{\downarrow} \vdash^{+} B \Uparrow$ then $\Gamma \Rightarrow^{+} B$ and,
2. If $\Gamma^{\downarrow} \vdash^{+} A \downarrow$ and $\Gamma, A \Rightarrow^{+} B$ then $\Gamma \Rightarrow^{+} B$.

Proof. As before. The extra case is handled by the Cut rule. □

After establishing the correspondence of $\vdash^{-}$ with $\Rightarrow$ and of $\vdash^{+}$ with $\Rightarrow^{+}$ we move on with:

Admissibility of Cut If $\Gamma \Rightarrow A$ and $\Gamma, A \Rightarrow B$ then $\Gamma \Rightarrow B$.

The proof is by double induction on the structure of the formula, its (sub-)derivations. This gives easily:

Cut Elimination If $\Gamma \Rightarrow^{+} A$ then $\Gamma \Rightarrow A$.

Which in turn gives us:

Normalization for Natural Deduction If $\Gamma \vdash A$ then $\Gamma^{\downarrow} \vdash^{-} A \Uparrow$

Proof. From assumption $\Gamma \vdash A$ which by completeness of annotated deductions gives $\Gamma \vdash^{+} A \Uparrow$. Then by completeness of sequent calculus and Cut Elimination we obtain $\Gamma \Rightarrow A$ which by soundness of sequent calculus completes the proof. □

As a result we obtain:

Proof. By contradiction, assume $\vdash \bot$ then $\Rightarrow \bot$ which is not possible. $\square$

Appendix C

Makam Implementation

A *Github* repo is preserved for *Jcalc* which includes an implementation in the metaprogramming framework Makam [49]. The implementation is currently developed by Antonis Stampoulis and the author. A type checker for *Jcalc* terms has been implemented and the call-by-value evaluation strategy is under current development.

The interested reader should install Makam (in a unix environment) and then she can run the `run.sh` file in the repository. A successful run will verify a number of Jcalc theorems encoded in the file `just.md` and will additionally output an html file (already included in the repo as `just.html`). The obtained file is a notebook of (pretty-printed) Latex and Makam code that can be opened in any modern browser. It showcases how the implementation faithfully follows the rules of the formal system.

Bibliography

[1] S Artemov. Unified semantics for modality and lambda terms via proof polynomials. *Logic, Language and Computation*, 97.

[2] S. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001.

[3] Sergei Artemov and Melvin Fitting. Justification logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2012 edition, 2012.

[4] Sergei Artemov and Roman Kuznets. Logical omniscience as infeasibility. *Annals of Pure and Applied Logic*, 165(1):6–25, 2014.

[5] Sergei N. Artemov. Operational modal logic. Technical Report MSI 95–29, Cornell University, December 1995.

[6] Sergei N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, March 2001.

[7] Sergei N. Artëmov and Eduardo Bonelli. The intensional lambda calculus. In *LFCS*, pages 12–25, 2007.

[8] Sergei [N.] Artemov and Eduardo Bonelli. The intensional lambda calculus. In Sergei N. Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science, International Symposium, LFCS 2007, New York, NY, USA, June 4–7, 2007, Proceedings*, volume 4514 of *Lecture Notes in Computer Science*, pages 12–25. Springer, 2007.

[9] Sergei Nikolaevich Artemov. Kolmogorov and gödel's approach to intuitionistic logic: current developments. *Russian Mathematical Surveys*, 59(2):203, 2004.

[10] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Sole Distributors for the U.S.A. And Canada, Elsevier Science Pub. Co., 1984.

[11] Francisco Bavera and Eduardo Bonelli. Justification logic and history based computation. In *International Colloquium on Theoretical Aspects of Computing*, pages 337–351. Springer, 2010.

[12] G. Bellin, V. de Paiva, and E. Ritter. Extended Curry-Howard Correspondence for a Basic Constructive Modal Logic. preprint; presented at M4M-2, ILLC, UvAmsterdam, 2001, 2001.

[13] G.M. Bierman and V. de Paiva. On an intuitionistic modal logic. *Studia Logica*, (65):383–416, 2000.

[14] Eduardo Bonelli and Federico Feller. Justification logic as a foundation for certifying mobile computation. *Annals of Pure and Applied Logic*, 163(7):935 – 950, 2012.

[15] Luitzen Egbertus Jan Brouwer and A Heyting. *Collected Works: Vol.: 1.: Philosophy and Foundations of Mathematics*. North-Holland Publishing Company, American Elsevier Publishing Company, Incorporated, 1975.

[16] Iliano Cervesato and Andre Scedrov. Relating state-based and process-based concurrency through linear logic (full-version). *Information and Computation*, 207(10):1044 – 1077, 2009.

[17] Haskell B Curry. Functionality in combinatory logic. *Proceedings of the National Academy of Sciences*, 20(11):584–590, 1934.

[18] Michael AE Dummett. *Elements of intuitionism*, volume 39. Oxford University Press, 2000.

[19] Melvin Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, 132(1):1–25, 2005.

[20] G. Gentzen. Untersuchungen über das logische schließen i. *Mathematische Zeitschrift*, 39:176–210, 1935.

[21] Gerhard Gentzen. The collected papers of gerhard gentzen. 1970.

[22] Jean Y. Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.

[23] Timothy Griffin. A formulae-as-types notion of control. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990*, pages 47–58, 1990.

[24] Professor Robert Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, New York, NY, USA, 2012.

[25] Robert Harper. Homotopy type theory seminar. `http://www.cs.cmu.edu/~rwh/courses/hott/notes/notes_week1.pdf`.

[26] Robert Harper. Extensionality, intensionality, and Brouwer's dictum. `http://existentialtype.wordpress.com/2012/08/11/extensionality-intensionality-and-brouwers-dictum/`, August 2012.

[27] Robert Harper. Constructive mathematics is not metamathematics. `http://existentialtype.wordpress.com/2013/07/10/constructive-mathematics-is-not-meta-mathematics/`, July 2013.

[28] S.Balzer H.DeYoung. Homotopy type theory seminar. `http://http://www.cs.cmu.edu/~rwh/courses/hott/`.

[29] Arend Heyting. *Intuitionism: an introduction*, volume 41. Elsevier, 1966.

[30] William A Howard. The formulae-as-types notion of construction. 1995.

[31] GA Kavvos. System k: a simple modal {\ lambda}-calculus. *arXiv preprint arXiv:1602.04860*, 2016.

[32] Andrey Kolmogorov. O principe tertium non datur. mathematicheskij sbornik 32: 646–667. *English trans. in van Heijenoort [1967, 414-437]*, 1925.

[33] J. Lambek. Deductive systems and categories I. Syntactic calculus and residuated categories. 2(4):287–318, 1968.

[34] Joachim Lambek. *Deductive systems and categories II. Standard constructions and closed categories*, pages 76–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 1969.

[35] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60.

[36] Per Martin-Lof and Giovanni Sambin. *Intuitionistic type theory*, volume 17. Bibliopolis Naples, 1984.

[37] C.-H. L. Ong and C. A. Stewart. A curry-howard foundation for functional computation with control. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '97, pages 215–227, New York, NY, USA, 1997. ACM.

[38] Frank Pfenning. Structural cut elimination: I. intuitionistic and classical logic. *Information and Computation*, 157(1):84–141, 2000.

[39] Frank Pfenning. Automated theorem proving. 2004. http://www.cs.cmu.edu/afs/.cs.cmu.edu/Web/People/fp/courses/atp/handouts/atp.pdf.

[40] Frank Pfenning. Lecture notes on harmony. http://www.cs.cmu.edu/~fp/courses/15317-f09/lectures/03-harmony.pdf, September 2009.

[41] Frank Pfenning. Lecture notes on natural deduction. http://www.cs.cmu.edu/~fp/courses/15317-f09/lectures/02-natded.pdf, August 2009.

[42] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical. Structures in Comp. Sci.*, 11(04):511–540, August 2001.

[43] Benjamin C. Pierce. *Types and Programming Languages.* MIT Press, Cambridge, MA, USA, 2002.

[44] Konstantinos Pouliasis. *A Curry–Howard View of Basic Justification Logic*, pages 316–337. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[45] Konstantinos Pouliasis and Giuseppe Primiero. J-calc: A typed lambda calculus for intuitionistic justification logic. *Electr. Notes Theor. Comput. Sci.*, 300:71–87, 2014.

[46] Dag Prawitz. Natural deduction: A proof-theoretical study. *AMC*, 10:12.

[47] Dag Prawitz. Ideas and results in proof theory. *Studies in Logic and the Foundations of Mathematics*, 63:235–307, 1971.

[48] Wilfried Sieg and John Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica*, 60(1):67–106, 1998.

[49] Antonis Stampoulis. Makam.

[50] Morten Heine B. Sørensen and Pawel Urzyczyn. Lectures on the curry-howard isomorphism, 1998.

[51] A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics: An Introduction*, volume I,II. North-Holland, Amsterdam, 1988.