Lambda calculus and justification logic: A proposal

Konstantinos Pouliasis

October 1, 2016

Abstract

This work is structured in two parts: The first part is, essentially, my second examination paper with some revisions and additions. It introduces basic elements of my research topic which rests in the intersection of Justification Logic, Constructive Modality and Type Theory. I will present the relevant systems syntactically and I will pause on the basic metatheoretic proof techniques which will be useful in the rest of the text. In the second part I will delineate the current state of my research in the area. I will elaborate on a modal type system that enhances simple type theory with elements of justification logic. I will present its accompanying calculus obtained a la Curry-Howard and I will argue for its computational relevance. More specifically, I will show that the obtained calculus characterizes certain computational phenomena that abound in modern programming language semantics. I will omit full metatheoretic results here but I will hint to proof methods that can be adopted from the first part. Finally, I will propose certain directions for future work. Such developments together with the omitted metatheoretic results are expected to be the study of my dissertation thesis. This is a paper accompanying my dissertation proposal and a partial requirement for my Phd candidacy under the supervision of Distinguished Professor Sergei Artemov at the Department of Computer Science of the Graduate Center, CUNY.

Contents

Chapter 1

Intuitionistic Logic

1.1 Intuitionism

In this Chapter, I will be presenting foundational work in the intersection of *Intuitionistic Logic* and *Type Theory.* The presentation is scaffolding following Prof. Robert Harper's lecture videos in *Homotopy Type Theory* [18] and the accompanying notes by students of the class [21]. I will often diverge to standard textbooks in the field [9], [17], [32] to present further important results.

1.1.1 A bird's eye view

In a nutshell, *Intuitionistic mathematics* is a program in foundations of mathematics that extends *Brouwer's program* [12]. 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 adhere to computational interpretations. In the following I will be 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 [22] and Kolmogorov [24] [8], 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 [16]. For the reader interested also in the philosophical implications of constructive foundations and *antirealism*, Dummet’s treatment is a classic in the field [13].

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 typical notion of *proof in a formal system* [20, 19].

A formal proof is a proof given in a fixed formal system, such as the axiomatic theory of sets, and arises from the application of the inductively defined rules in that system. Formal proofs can, thus, be viewed as gödelizations of textual derivation in some fixed system.

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 cannot be given a formal proof in the formal system that is at stake. This *openness* of the nature of proofs is necessary for a foundational treatment of proofs that respects Gödelian phenomena. This is often coined as "Axiomatic Freedom" of intuitionistic foundations.

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.* Current developments in constructive foundations like *Homotopy Type Theory* and in general systems that rely on *Martin-Löf Type Theory* [26] do not necessarily rule out *LEM* but they might permit its usage locally, if needed, in a proof.

1.2 IPL

Intuitionistic Propositional Logic (IPL) can be viewed as "the logic of *proof relevance*" conforming with the intuitionistic view described in 1.1. 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 [25] 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 on 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

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

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

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

$$\frac{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1\ \mathsf{true}}\ \wedge\mathrm{E\scriptsize{L}} \qquad \frac{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_2\ \mathsf{true}}\ \wedge\mathrm{E\scriptsize{R}}$$

Disjunction Introduction and Elimination

$$\frac{\Gamma \vdash \phi_1\ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2\ \mathsf{true}}\ \vee\mathrm{I\scriptsize{L}} \qquad \frac{\Gamma \vdash \phi_2\ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2\ \mathsf{true}}\ \vee\mathrm{I\scriptsize{R}}$$

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

1.2.1 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

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

1.3 Order Theoretic Semantics: *Hayting Algebras*

IPL viewed order theoretically gives rise to a *Hayting 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 *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. $a \wedge x \leq b$

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

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.

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

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

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

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

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

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

$$\frac{}{\chi \leq 1}$$

which says that 1 is the greatest element.

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

$$\frac{}{0 \leq \chi}$$

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

$$\begin{array}{rcl} nil^+ & = & \top \\ (\Gamma, \phi)^+ & = & \Gamma^+ \wedge \phi * \end{array}$$

Chapter 2

Lambda Calculus With Types

2.1 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:

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

The rule says, “given the existence a proof of ϕ and a proof of ψ from assumptions Γ, there exists a proof of $\phi \wedge \psi$ 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 proof 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 are the driver idea in the "Curry–Howard Isomorphism" and its extensions.

Here we 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 one judgments J of the form $A\,\mathsf{true}$. Here are the the rules for constructing proof trees with labeled assumptions[1]. First, the deductions

[1]Essentially the constructs are directed acyclic graphs since assumptions with the same label are "bind" and substitutable together but we will be cavalier with such a details

using reflection on hypothesis are valid:

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

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

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

$$\frac{\begin{array}{c}\mathcal{D} \\ A\ \mathsf{true}\end{array}}{A \vee B\ \mathsf{true}} \qquad \frac{\begin{array}{c}\mathcal{D} \\ B\ \mathsf{true}\end{array}}{A \vee B\ \mathsf{true}}$$

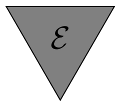
$$\dfrac{\begin{array}{c}\mathcal{D}\\ A \vee B\ \mathsf{true}\end{array} \qquad \begin{array}{c}A\ \mathsf{true}\\ \mathcal{E}\\ C\ \mathsf{true}\end{array} \qquad \begin{array}{c}B\ \mathsf{true}\\ \mathcal{F}\\ C\ \mathsf{true}\end{array}}{C\ \mathsf{true}}$$

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

2.1.1 Properties of Intuitionistic Entailment Redux

Proof trees by their nature satisfy the properties of entailment in 1.2.1. 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 with the proof 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 valid trees.

Theorem. If $\begin{array}{c}x : A\\ \mathcal{D}\\ B\ \mathsf{true}\end{array}$ and $\begin{array}{c}\bigtriangledown_{\mathcal{E}}\\ A\ \mathsf{true}\end{array}$ are valid proof trees their compo-



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

2.1.2 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 was stemmed from Gentzen's work on natural deduction and cut elimination and it is what gives to proofs computational content. Here 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 reframe them as equalities among proof terms in the next section. Nevertheless, they can be expressed in proof tree form. We show indicatively the equalities regarding the $\supset$ connective proofs reserving the rest for the more concise notation.

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

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

2.2 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

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

$$\frac{\phi_1 \in \mathsf{Type} \qquad \phi_2 \in \mathsf{Type}}{\phi_1 \to \phi_2 \in \mathsf{Type}}\ \textsc{Arr} \qquad \frac{\phi_1 \in \mathsf{Type} \qquad \phi_2 \in \mathsf{Type}}{\phi_1 \times \phi_2 \in \mathsf{Type}}\ \textsc{Prod}$$

$$\frac{\phi_1 \in \mathsf{Type} \qquad \phi_2 \in \mathsf{Type}}{\phi_1 + \phi_2 \in \mathsf{Type}}\ \textsc{Union}$$

Context Formation

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

Context Reflection

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

Top Introduction – Bottom Elimination

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

Function Construction and Application

$$\frac{\Gamma, x : \phi_1 \vdash M : \phi_2}{\Gamma \vdash \lambda x.M : \phi_1 \to \phi_2}\ \lambda\text{–}\textsc{Abs} \qquad \frac{\Gamma \vdash M : \phi_1 \to \phi_2 \qquad \Gamma \vdash M^{'} : \phi_1}{\Gamma \vdash (MM^{'}) : \phi_2}\ \textsc{App}$$

Tuple Construction and Projections

$$\frac{\Gamma \vdash M : \phi_1 \qquad \Gamma \vdash M^{'} : \phi_2}{\Gamma \vdash \langle M, M^{'}\rangle : \phi_1 \times \phi_2}\ \textsc{Tup}$$

$$\frac{\Gamma \vdash M : \phi_1 \times \phi_2}{\Gamma \vdash \mathrm{fst}(M) : \phi_1}\ \textsc{LPrj} \qquad\qquad \frac{\Gamma \vdash M : \phi_1 \times \phi_2}{\Gamma \vdash \mathrm{snd}(M) : \phi_2}\ \textsc{RPrj}$$

Union Construction and Elimination

$$\frac{\Gamma \vdash M : \phi_1}{\Gamma \vdash inj_l[\phi_2](M) : \phi_1 + \phi_2}\ \textsc{InjL} \qquad \frac{\Gamma \vdash M : \phi_2}{\Gamma \vdash inj_r[\phi_1](M) : \phi_1 + \phi_2}\ \textsc{InjR}$$

$$\frac{\Gamma \vdash M : \phi_1 + \phi_2 \qquad \Gamma, x : \phi_1 \vdash N : \phi \qquad \Gamma, y : \phi_2 \vdash O : \phi}{\Gamma \vdash \text{case } M \text{ of } inj_l(x) \longmapsto N \mid inj_r(y) \longmapsto O : \phi}\ \vee\text{E}$$

2.2.1 Definitional Equality: Proof tree equalities as term equalities

We want to think about when two proofs (resp. proof trees) $M : A$ and $M' : A$ are the same. In the following we elaborate Gentzen's principles introducing an equivalence relation called *definitional equality* that respects these principles, denoted $M \equiv M' : A$. We want definitional equality $\equiv$ to be the least congruence closed under the β rules. We will define what this means:

A *congruence* is an equivalence relation (i.e. reflexive transitive and antisymmetric) that respects our operators as formulated below.

$$\frac{}{\Gamma \vdash M \equiv M : \phi}\,\mathrm{R} \qquad \frac{\Gamma \vdash M \equiv M' : \phi}{\Gamma \vdash M' \equiv M : \phi}\,\mathrm{S}$$

$$\frac{\Gamma \vdash M \equiv M' : \phi \qquad \Gamma \vdash M' \equiv M'' : \phi}{\Gamma \vdash M \equiv M'' : \phi}\,\mathrm{T}$$

For the equivalence relation to respect operators we means that if $M \equiv M' : A$, then that their image under any operator should be equivalent. In other words, we should be able to replace M with M' everywhere.For example:

Congruence over fst

$$\frac{\Gamma \vdash M \equiv M' : A \wedge B}{\Gamma \vdash \mathrm{fst}(M) \equiv \mathrm{fst}(M') : A}$$

Inversion Principle

Gentzen's Inversion Principle captures the idea that "elim is post-inverse to intro," which is the informal notion that the elimination rules should cancel the introduction rules.

The β rules are as follows:

$$\frac{\Gamma \vdash M : \phi_1 \qquad \Gamma \vdash N : \phi_2}{\Gamma \vdash \text{fst}(\langle M, N \rangle) \equiv M : \phi_1}\ \beta\wedge_1$$

$$\frac{\Gamma \vdash M : \phi_1 \qquad \Gamma \vdash N : \phi_2}{\Gamma \vdash \text{snd}(\langle M, N \rangle) \equiv N : \phi_2}\ \beta\wedge_2 \qquad \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.M)(N) \equiv [N/x]M : B}\ \beta \supset$$

$$\frac{\Gamma, x : \phi_1 \vdash N : \psi \qquad \Gamma, y : \phi_2 \vdash O : \psi \qquad \Gamma \vdash P : \phi_1}{\Gamma \vdash (\text{case } inj_l(P) \text{ of } inj_l(x) \longmapsto N|\ inj_r(y) \longmapsto O) \equiv [P/x]N : \psi}\ \beta\vee_1$$

$$\frac{\Gamma, x : \phi_1 \vdash N : \psi \qquad \Gamma, y : \phi_2 \vdash O : \psi \qquad \Gamma \vdash Q : \phi_2}{\Gamma \vdash (\text{case } inr_r(Q) \text{ of } inj_l(x) \longmapsto N|\ inj_r(y) \longmapsto O) \equiv [Q/y]O : \psi}\ \beta\vee_2$$

Unicity Principle

Gentzen's Unicity Principles on the other hand capture the idea that "intro is post-inverse to elim". There should be only one way – modulo definitional equivalence – to prove something. The "β" rules give rise to computational interpretation. The "η" rules impose properties that the computational model should satisfy (e.g. Church-Rosser property).

The η rules are given below:

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

$$\frac{\Gamma \vdash M : \phi \supset \psi}{\Gamma \vdash M \equiv \lambda x.Mx : \phi \supset \psi}\ \eta \supset$$

$$\frac{\Gamma, z : \phi_1 + \phi_2 \vdash M : \psi \qquad \Gamma \vdash N : \phi_1 + \phi_2}{\begin{array}{ll}\Gamma \vdash M[N/z] \equiv \text{case } N \text{ of} & |inj_l(x) \mapsto M[inj_l(x)/z] \\ & inj_r(y) \mapsto M[inj_r(y)/z]) : \psi\end{array}}\ \eta\vee$$

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

Given the formulation of the λ-calculus we can think of *formulae-as-types* and *proof terms-as-programs*. This enriches logic with a computational aspect(*proof dynamics*) that is absent from other formulations. Dynamics stems from Gentzen's insight to give an equational system for proofs equipped with $\beta\eta$ rules for proof-tree conversion. These insights, give rise to λ-calculus dynamics if we devise an execution strategy (an *operational semantics*) for program reduction that respects these rules.

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*:

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

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

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

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

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 $\lambda x : []$, $(e[])$, $[]e$, 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 can be defined on terms that include redexes as subterms as follows :

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

$$(\lambda x : \phi.N)M \mapsto [M/x]N \ (\beta) \qquad \mathrm{fst}\langle M, N\rangle \mapsto M \ (\beta)$$

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

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

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 where $\Gamma \vdash M \equiv N : A$ based on the $\equiv$ 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 $\Gamma \vdash M \equiv N : A$ then there exists tern L s.t. $M, N \mapsto^*_{\beta\eta} L$

And as a result:

Theorem. Consistency of definitional equality of terms The definitional equality $\equiv$ 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. $M \mapsto_{\beta\eta} N_0 \mapsto_{\beta\eta} N_1 \mapsto_{\beta\eta} N_2 \mapsto_{\beta\eta} \ldots \mapsto_{\beta\eta} N_i$ where N_i is a $\beta\eta$ normal form.

It is common place in metatheotic proofs for such systems that induction on the structure of the term does not "go through". Intuitively, a reduction can be "enlarging" the term but, yet, it is doing progress based on a different kind of metric. We build this metric based on the following definitions and facts. 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 $\theta(A \to B)$ then $d(\lambda x.M)N = \theta(A \to B)$
- Similarly, $d(\pi\,\mathrm{fst}\langle M, N\rangle) = \theta(A \times B)$ 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.

Which give us the following fact that suggests the induction principle that succeeds toward the proof.

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 $M \mapsto^{*}_{\beta\eta} N$ 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 that says that *every* strategy in normalizing. This result is important in 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 strong normalization for more complex calculi- is the concept of a reducibility predicate. Reducibility predicates give a stronger induction principle that is able to give the desired result as a lemma. Similarly with the weak normalization reducibility predicates are sets of terms classified by their type. They are defined by induction on the structure of their type. We show strong normalization only for β reduction but and only for the $\rightarrow, \times$ fragment of the calculus but all techniques generalize (see, e.g. PRAWITZ).

Definition We define the predicate Red_A for a type A as follows and for closed terms M as follows:

- $M \in Red_{P_i}$ iff M is of atomic type P_i and M is normalizable.
- $M \in Red_{A \rightarrow B}$ iff M is of type $A \rightarrow B$ and $\forall N. N \in Red_A \implies (MN) \in Red_B$
- $M \in Red_{A \wedge B}$ iff M is of type $A \wedge B$, $\mathrm{fst}(M) \in Red_A$ and $\mathrm{snd}(M) \in Red_B$

Reducibility has the important following properties below where by *neutral* we mean terms of the form $\langle M, N\rangle$, $\lambda x.M$

Theorem. 1. $M \in Red_A$ implies that M is normalizable.

2. $M \in Red_A$ and $M \mapsto_\beta^* N$ then $N \in Red_A$

3. If M is neutral then, $\forall N \mathrm{s}.t M \mapsto^\beta N. N \in Red_A \implies M \in Red_A$

4. If M is neutral and normal, then $M \in Red_A$

The proof goes by induction on the structure of type A. Now we are able to prove the following theorems:

Theorem. • If M, N are reducible then so is $\langle M, N\rangle$

- For all reducible M of type A if $N[M/x]$ is reducible then so is $\lambda x.N$

These results suffice to show the following theorem: Given $x_1 : A_1, \ldots, x_i : A_i \vdash M : A$ then and reducible terms $v_1 \in Red_{A_1}, \ldots v_i \in Red_{A_i}$ then $M[v_1/x_1 \ldots v_i/x_i] \in Red_A$ Out of which we get:

Theorem. Strong Normalization Theorem

Every term $\vdash M : A$ is strongly normalizing

2.3.1 The essence of Proofs as program

The proofs of normalization above are essentially of the same ”proof strength” (induction principles) as the logical proof of cut elimination. In a nutshell, eliminating cuts is the same as normalizing proof terms (and the corresponding 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 shown explicitly follow Siegl's extraction method, that essentially showcases how a construction of the Cut-free sequent calculus comes naturally from an analysis of normal proofs in the natural deduction. This result has been extremely useful through my research and I am expecting to revisit it in more detail at my thesis work.

2.3.2 Propositions as Types

There is a correspondence between propositions and types:

Propositions	Types
$\top$	1
$A \wedge B$	$A \times B$
$A \supset B$	function $A \to B$ or B^A
$\bot$	0
$A \vee B$	$A + B$

2.4 Categories for proof relevant *IPL*

In a Heyting Algebra, we have a preorder (or, partial order in the "textbook" definition) $\phi \leq \psi$ when ϕ implies ψ. *HAs* are insufficient, however, for the treatment of proof objects (there can be at most one instance of $\phi \leq \psi$ for specific ϕ,ψ). We can keep track of proofs, so if M is a proof from Γ to ψ, we want to think of it as a map $M : \Gamma+ \to \psi+$. The category theoretic analog of a Heyting Algebra is a Bi–Cartesian Closed Category (*BiCCC*). That is a category with all finite products, co–products and exponentials. The axiomatization of a category (in general), finite (and nullary) products and co-products and exponentials is given in this section.

2.4.1 Definitions and Axioms of a Category

A category has *objects* $\phi, \psi, \ldots$ and *arrows* $f, g, h \ldots$ Each arrow goes from an object to an object. To say that g goes from ϕ to ψ we write $g : \phi \to \psi$, or say that ϕ is the domain of g, and ψ the *co–domain.* We write $Dom(g) = \phi$

and $Cod(g) = \psi$. We say that two arrows f and g are *composable* with $Dom(f) = Cod(g)$. If f and g are composable, they have a *composite*, an arrow called $f \circ g$. There is an *identity* for every object ϕ.

$$\frac{}{\mathrm{id} : \phi \to \phi}\ \mathrm{ID}_{ex} \qquad \frac{f : \phi \to \psi \qquad g : \psi \to \chi}{g \circ f : \phi \to \chi}\ \textsc{Comp}$$

$$\frac{f : \phi \to \psi}{id_\psi \circ f = f : \phi \to \psi}\ \mathrm{ID}_l \qquad \frac{f : \phi \to \psi}{f \circ id_\phi = f : \phi \to \psi}\ \textsc{idr}$$

$$\frac{f : \phi \to \psi \qquad g : \psi \to \chi \qquad h : \chi \to \omega}{h \circ (g \circ f) = (h \circ g) \circ f : \phi \to \omega}\ \textsc{idr}$$

2.4.2 Terminal, Co-Terminal objects, Products and Co-Products

Now we can think about objects in the category that correspond to propositions given in the correspondence.

Terminal Object 1 is the terminal object, also called the final object, which corresponds to $\top$. For any object Γ there is a unique map $\Gamma \to 1$.

$$\frac{}{\langle\ \rangle : \phi \to 1}\ \textsc{Existence} \qquad \frac{M : \Gamma \to 1}{M = \langle\ \rangle : \Gamma \to 1}\ \textsc{Unicity}(\eta)$$

Product For any objects ϕ and ψ there is an object $\chi = \phi \times \psi$ equipped with arrows fst : $\phi \times \psi \to \phi$ and snd : $\phi \times \psi \to \psi$ that is the *product* of ϕ and ψ, which corresponds to the join $\phi \wedge \psi$. For any other object Γ with arrows $M : \Gamma \to \phi$ and $\Gamma \to \psi$ there exists *unique* arrow, $\langle M, N\rangle$ s.t. $\mathrm{fst} \circ \langle M, N\rangle = M(\beta\times_1)$ and $\mathrm{snd} \circ \langle M, N\rangle = N(\beta\times_2)$.

$$\frac{M : \Gamma \to \phi \qquad N : \Gamma \to \psi}{\langle M, N\rangle : \Gamma \to \phi \times \psi}\ \textsc{Exist}_1$$

$$\frac{M : \Gamma \to \phi \qquad N : \Gamma \to \psi}{\mathrm{fst} \circ \langle M, N\rangle : \Gamma \to \phi}\ \textsc{Exist}_2(\beta_1)$$

$$\frac{M : \Gamma \to \phi \qquad N : \Gamma \to \psi}{\mathrm{snd} \circ \langle M, N\rangle : \Gamma \to \phi}\ \textsc{Exist}_3(\beta_2)$$

$$\frac{P : \Gamma \to \phi \times \psi \qquad \mathrm{fst} \circ P = M : \Gamma \to \phi \qquad \mathrm{snd} \circ P = N : \Gamma \to \psi}{P = \langle M, N\rangle : \Gamma \to \phi \times \psi}\ \textsc{Un}(\eta)$$

Diagrammatically:

$$\begin{tikzcd}
 & \Gamma \arrow[dl, bend right, "M"'] \arrow[d, dashed, "\langle M{,}N \rangle"] \arrow[dr, bend left, "N"] & \\
 & \phi \times \psi \arrow[dl, "\mathrm{fst}"] \arrow[dr, "\mathrm{snd}"'] & \\
\phi & & \psi
\end{tikzcd}$$

Exponentials Given objects A and B, an exponential B^A (which corresponds to $A \supset B$) is an object with the following universal property:

$$\begin{tikzcd}
C \arrow[d, dashed, "\lambda(h)"'] & C \times A \arrow[d, dashed, "\lambda(h) \times \mathrm{id}_A"'] \arrow[dr, "h"] & \\
B^A & B^A \times A \arrow[r, "\mathrm{ap}"'] & B
\end{tikzcd}$$

such that the diagram commutes.

This means that there exists a map $\mathrm{ap} : B^A \times A \to B$ (application map) that corresponds to implication elimination.

The universal property is that for all objects C that have a map $h : C \times A \to B$, there exists a unique map $\lambda(h) : C \to B^A$ such that

$$\mathrm{ap} \circ (\lambda(h) \times \mathrm{id}_A) = h : C \times A \to B$$

This means that the diagram commutes. Another way to express the induced map is $\lambda(h) \times \mathrm{id}_A = \langle \lambda(h) \circ \mathrm{fst}, \mathrm{snd} \rangle$.

The map $\lambda(h) : C \to B^A$ is unique, meaning that

$$\frac{\mathrm{ap} \circ (g \times \mathrm{id_A}) = h : C \times A \to B}{g = \lambda(h) : C \to B^A}$$

Co–Products For any objects ϕ and ψ there is an object $\chi = \phi + \psi$ equipped with arrows $\mathrm{inl} : \phi \to \phi + \psi$ and $\mathrm{inr} : \psi \to \phi + \psi$ that is the *co-product* of ϕ and ψ, which corresponds to the meet $\phi \wedge \psi$. For any other object ω with arrows $M : \omega \to \phi \vee \psi$ and $N : \omega \to \phi \vee \psi$ there exists *unique* arrow, M, N s.t. $\{M, N\} \circ \mathrm{inl} = M$ and $\{M, N\} \circ \mathrm{inr} = N$.

$$\frac{O : \Gamma \to \phi}{\mathrm{inl} \circ O : \Gamma \to \phi + \psi}\ \textsc{Exist}_1 \qquad \frac{P : \Gamma \to \psi}{\mathrm{inr} \circ P : \Gamma \to \phi + \psi}\ \textsc{Exist}_2$$

$$\frac{O : \Gamma \to \phi \qquad M : \phi \to \omega \qquad N : \psi \to \omega}{\{M, N\} \circ \mathrm{inl} \circ O = M \circ O : \Gamma \to \omega}\ \textsc{Exist}_3(\beta_1)$$

$$\frac{P : \Gamma \to \psi \qquad M : \phi \to \omega \qquad N : \psi \to \omega}{\{M, N\} \circ \mathrm{inr} \circ P = N \circ P : \Gamma \to \omega}\ \textsc{Exist}_3(\beta_2)$$

$$\frac{M : \Gamma \to \phi \qquad N : \Gamma \to \psi}{\mathrm{snd} \circ \langle M, N \rangle : \Gamma \to \phi}\ \textsc{Exist}_3(\beta_2)$$

$$\frac{\begin{array}{c} O : \Gamma \to \phi \qquad P : \Gamma \to \psi \qquad U : \phi + \psi \to \omega \qquad M : \phi \to \omega \\ N : \psi \to \omega \qquad U \circ \mathrm{inl} \circ O = M \qquad\qquad U \circ \mathrm{inr} \circ N = M \end{array}}{U = \{M, N\}}\ \textsc{Un}(\eta)$$

Diagrammatically:

$$\begin{tikzcd}
\phi \arrow[dr, "\mathsf{inl}"] \arrow[ddr, bend right, "M"'] \arrow[ddr, phantom, "(\beta)"] & & \psi \arrow[dl, "\mathsf{inr}"'] \arrow[ddl, bend left, "N"] \arrow[ddl, phantom, "(\beta)"] \\
& \phi + \psi \arrow[d, bend right, "M"'] \arrow[d, bend left, "\{M,N\}"] \arrow[d, phantom, "(\eta)"] & \\
& C &
\end{tikzcd}$$

Chapter 3

Justification Logic

In the second part of this paper I will give an overview of *Justification Logic* (JL) highlighting the parts that are closely related to constructivity to remain coherent with 1. I will emphasize LP, the very first logic of justification, and its deep relation with IPL. My scaffolding will be based upon [6], [4] that reflect this relation. Beforehand, I will allow for a more general discussion on JL following [2] and other relevant papers.

3.1 A bird's eye view

According to [2]

> 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 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,$S4$,$S5$). 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] and [33], 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.

3.2 Minimal Justification Logic J_0

To permit for an account of reasons, the logic is enriched with an extra sort for 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:

$$j := s_i |\ C_i | j_1 * j_2 | j_2 + j_2$$

$$\phi := P_i |\ \bot |\ \phi_1 \wedge \phi_2 |\ \phi_1 \vee \phi_2 |\ \phi_2 \supset \phi_2 |\ \neg\phi |\ j : \phi$$

Constants C_i are symbols that can be assigned to logic axioms that are assumed to be necessary. Weaker justifications 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

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

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

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

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 constants and propositions:

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 a la IPL since the following theorem holds:

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

3.3 Epistemic motivation

JL as an epistemic logic departs from previous traditions of logic of knowledge based on universality judgments. From [2]

> 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 on epistemic tradition has been utilized to solve many problems in formal epistemology (see [3]). We give here a solution to the famous 'Red barn problem' that is also 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.

3.4 Proof theoretic view

In 1 we gave an analytic account of the *Brower-Heyting-Kolmogorov* (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

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

where $\mathrm{tr}(\phi)$ 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.

3.5 The Logic of Proofs

An axiomatization of LP with axiomatically appropriate constant specification as defined in 3.2 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. $\vdash j : \phi \supset (j! : j : \phi)$

3.6 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 and 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.

3.6.1 Realization

The realization gives an algorithmic process of transforming 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.

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

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

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

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

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

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.

3.6.2 Kripke - Fitting Semantics

In this section I will be discussing Kripke – Fitting Semantics[15] 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 4

Modal Type Theory

In this chapter I will give a short, example-driven introduction to typed modality and its applications. The discussion will remain informal when it comes to metatheory and will be focused on usage of modal typed calculi in applications. Apropos, I will discuss Operational Semantic as the essence of the computational approach to logic.

4.1 A bird's eye view

Significant in the development of modal logic within a type-theoretic framework is the work of Moggi [27]. In his seminal work he mentioned the need of shifting from simply typed calculi to systems that can capture notions of computation such exceptions, partial functions, binding constructs etc. Moggi's initiative stems from a categorical point of view.

In the realm of deductive systems with explicit witnesses, Artemov's work on Operational Modal Logic[5], and the "Gang of four" [11],[?] revived the interest in constructive modality and its computational view. Of course earlier work in the intersection of modal and constructive logic exists (cf. [34],[14]) but its relation with programming languages is not yet explicit. It's worth mentioning that the work by DePaiva initiates from a categorical view too. That is Categorical Semantics for Linear Logic.

Since it is of great importance in my work, I will be presenting here the *judgmental* approach followed by Pfenning's "Judgmental Reconstruction of Modal Logic" [31] which is a foundational approach that captures previous work on $\Box$ Calculi for $S4$. Although the system presented is a judgmental reconstruction of the system $S4$ the approach can be used to host other modalities.

4.2 Judgmental Reconstruction of Modal Logic

The $\Box$ fragment of Pfenning's Judgmental reconstruction, consists of the judgments of IPL as developed in the first Chapter together with judgments of validity. The definition of validity is given in a proof theoretic manner. In a nutshell, judgments of validity internalize judgments of proof without assumptions. "Evidence for validity of ϕ, is simply unconditional evidence of truth of ϕ".

1. If $\mathsf{nil} \vdash \phi\,\mathsf{true}$ then $\phi\,\mathsf{Valid}$
2. If $\phi\,\mathsf{Valid}$ then $\Gamma \vdash \phi\,\mathsf{true}$

The logical judgments are now extended in the form:

$$\phi'_1\,\mathsf{Valid}, \phi'_2\,\mathsf{Valid}, \ldots, \phi'_n\,\mathsf{Valid}; \phi_1\,\mathsf{true}\phi_2\,\mathsf{true}, \ldots \phi_m\,\mathsf{true} \vdash \phi\,\mathsf{true}$$

In the rules, we restrict ourselves to proving judgments of the form $\phi\,\mathsf{true}$ (rather than $\phi\,\mathsf{Valid}$), which is possible since the latter is directly defined in terms of the former. The meaning of hypothetical judgments yields the general substitution principle.

$\Delta \vdash \phi\,\mathsf{Valid}$ and $\Delta, \phi\,\mathsf{Valid} \vdash J$ then $\Delta \vdash J$

This principle can be rewritten utilizing the definition of validity:

Substitution Principle For Validity $\Delta; \mathsf{nil} \vdash \phi\mathsf{true}$ and $\Delta, \phi\mathsf{Valid}; \Gamma \vdash J$ then $\Delta; \Gamma \vdash J$

Additionally, we add the following hypothesis reflection rule for valid contexts:

Validity Context Reflection

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

In the typability rules we add the following:

$$\frac{\phi\ \mathsf{Prop}}{\Box\phi\ \mathsf{Prop}}\ \Box\mathrm{F}$$

The $\Box$ introduction rule just allows the internalization of the validity of ϕ as truth of $\Box\phi$, according to the definition of validity.

Necessity Introduction

$$\frac{\Delta;\ \mathsf{nil} \vdash \phi\ \mathsf{true}}{\Delta; \Gamma \vdash \Box\phi\ \mathsf{true}}\ \Box\mathrm{I}$$

The elimination rule is harder. A simplified version like the one below is unsound since the hypotheses in Γ are unjustified.:

$$\frac{\Delta; \Gamma \vdash \Box\phi\ \mathsf{true}}{\Delta; \vdash \Box\phi\ \mathsf{true}}\ \Box\mathrm{E}\text{-}\textsc{Unsound}$$

Another approach would be the rule below. Which is locally sound but not complete Gentzen's inversion principle is not satisfied. After eliminating the $\Box$ we cannot re-introduce it.

$$\frac{\Delta; \vdash \Box\phi\ \mathsf{true}}{\Delta; \Gamma \vdash \phi\ \mathsf{true}}\ \Box\textsc{E-Incomplete}$$

The proposed rule that satisfies local reduction and local expansion is :

$$\boxed{\frac{\Delta; \Gamma \vdash \Box\phi\ \mathsf{true} \qquad \Delta, \phi\ \mathsf{Valid}; \Gamma \vdash \psi\ \mathsf{true}}{\Delta; \Gamma \vdash \psi\ \mathsf{true}}\ \Box\mathrm{E}}$$

The negative fragment of the system is, thus, as follows:

Prop Formation

$$\frac{}{P_i \in \mathsf{Prop}}\ \textsc{Atom} \qquad \frac{}{\top \in \mathsf{Prop}}\ \textsc{Top} \qquad \frac{\phi_1 \in \mathsf{Prop} \qquad \phi_2 \in \mathsf{Prop}}{\phi_1 \supset \phi_2 \in \mathsf{Prop}}\ \textsc{Arr}$$

Context Γ Formation

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

Context Δ Formation

$$\frac{}{\mathsf{nil}\ \ \mathsf{ctx}}\ \textsc{Nil} \qquad\qquad \frac{\Delta\ \mathsf{ctx} \qquad \phi \in \mathsf{Prop}}{\Delta, \phi\, \mathsf{Valid}\ \ \mathsf{ctx}}\ \Delta\text{-}\textsc{Add}$$

Compound $\Gamma;\Delta$ Context

$$\frac{\Delta\ \mathsf{ctx} \qquad \Gamma\ \mathsf{ctx}}{\Delta;\Gamma \vdash \mathsf{ctx}}\ \Gamma;\Delta\text{-F}$$

Context Γ Reflection

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

Context Δ Reflection

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

Top Introduction

$$\frac{}{\Delta;\Gamma \vdash \top\ \mathsf{true}}\ \top\mathrm{I}$$

Implication Introduction and Elimination

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

Necessity Introduction and Elimination

$$\frac{\Delta;\ \mathsf{nil}\vdash\phi\ \mathsf{true}}{\Delta;\Gamma\vdash\Box\phi\ \mathsf{true}}\Box\mathrm{I}\qquad\frac{\Delta;\Gamma\vdash\Box\phi\ \mathsf{true}\qquad\Delta,\phi\ \mathsf{Valid};\Gamma\vdash\psi\ \mathsf{true}}{\Delta;\Gamma\vdash\psi\ \mathsf{true}}\Box\mathrm{E}$$

4.2.1 Properties of Entailment For the Judgmental Reconstruction

The guiding transitivity principles, weakening, contraction, and exchange can be proved for the system:

Transitivity The guiding substitution principle can be expressed as a property of this formal system and also be proven by induction over the structure of derivations

- If $\Delta;\Gamma,\phi\ \mathsf{true},\Gamma'\vdash\psi\ \mathsf{true}$ and $\Delta;\Gamma\vdash\phi\ \mathsf{true}$ then $\Delta;\Gamma,\Gamma'\vdash\psi\ \mathsf{true}$
- If $\Delta,\phi\ \mathsf{Valid},\Delta';\Gamma\vdash\psi\ \mathsf{true}$ and $\Delta;\ \mathsf{nil}\vdash\phi\ \mathsf{true}$ then $\Delta\Delta';\Gamma\vdash\psi\ \mathsf{true}$

Weakening, Contraction and Exchange properties can be also shown to hold.

4.2.2 Adding proof terms

The system can be assigned proof terms as follows:

Context Γ Formation

$$\frac{}{\mathsf{nil}\ \ \mathsf{ctx}}\ \textsc{Nil} \qquad \frac{\Gamma\ \mathsf{ctx} \qquad \phi \in \mathsf{Prop} \qquad x \notin \Gamma}{\Gamma, x : \phi\ \ \mathsf{ctx}}\ \Gamma\text{-}\textsc{Add}$$

Context Δ Formation

$$\frac{}{\mathsf{nil}\ \ \mathsf{ctx}}\ \textsc{Nil} \qquad \frac{\Delta\ \mathsf{ctx} \qquad \phi \in \mathsf{Prop} \qquad s \notin \Delta}{\Delta, s :: \phi\ \ \mathsf{ctx}}\ \Delta\text{-}\textsc{Add}$$

Compound $\Gamma; \Delta$ Context

$$\frac{\Delta\ \mathsf{ctx} \qquad \Gamma\ \mathsf{ctx}}{\Delta; \Gamma \vdash \mathsf{ctx}}\ \Gamma; \Delta\text{-F}$$

Context Γ Reflection

$$\frac{}{\Delta; \Gamma, x : \phi, \Gamma' \vdash x : \phi}\ \Gamma\text{-}\textsc{Refl}$$

Context Δ Reflection

$$\frac{}{\Delta, s :: \phi, \Delta'; \Gamma \vdash s : \phi}\ \Delta\text{-}\textsc{Refl}$$

Top Introduction

$$\frac{}{\Delta; \Gamma \vdash \langle\rangle : \top}\ \top\mathrm{I}$$

Implication Introduction and Elimination

$$\frac{\Delta; \Gamma, x : \phi_1 \vdash M : \phi_2}{\Delta; \Gamma \vdash \lambda x : \phi_1 . M : \phi_1 \supset \phi_2}\ \supset\mathrm{I}$$

$$\frac{\Delta; \Gamma \vdash M : \phi_1 \supset \phi_2 \qquad \Delta; \Gamma \vdash N : \phi_1}{\Delta; \Gamma \vdash (MN) : \phi_2}\ \supset\mathrm{E}$$

Necessity Introduction and Elimination

$$\frac{\Delta;\ \mathsf{nil} \vdash M : \phi}{\Delta; \Gamma \vdash \mathrm{box}(M) : \Box\phi\ \mathsf{true}}\ \Box\mathrm{I}$$

$$\frac{\Delta; \Gamma \vdash M : \Box\phi\ \mathsf{true} \qquad \Delta, s :: \phi; \Gamma \vdash N : \psi}{\Delta; \Gamma \vdash \mathrm{let}\,\mathrm{box}(s) = M\,\mathrm{in}\,N : \psi}\ \Box\mathrm{E}$$

4.3 Computational Interpretation

One of the possible ways to read $\Box\phi$ is as representing *source* code of type ϕ. This makes the *Box* calculus given a framework for typing programs with explicit *staged computation*. Explicit staging exists in many languages. One of its most characteristic implementations is the *quote* constructs in Lisp [10]. We introduce the concept and the application of the calculus with a motivating example following [28].

Consider the exponential function exp : $nat \to nat \to nat$ and the two definitions

```
exp(0) = lm x -> 1
exp(s(n)) = lm x -> x* exp n x

exp’(0) = lm x -> 1
exp(s(n)) = let f = exp’n in lm x -> x* f x
```

The two functions although behaviorally equivalent have a completely different operational behavior. For the first function applied to a $s(s(0))$ will unfold to

```
lm x-> x* exp(s(0)) x
```

the second though recurs completely on its argument unfolding to:

```
lm x-> x* (lm x-> x *( lm x->1) x) x
```

which after reduction under λ can be reduced to:

```
lm x-> x* x*1
```

We can see that the second version does a lot more computation than the first. However, if the resulting function is applied many times, to many different bases, then the second can be more efficient. We want to extend our language with types that discriminate between the two cases.

We will try to explore this and show the how $\Box$ types can be useful to discriminate between the two. Prior to this we have to speak about about operational semantics.

4.3.1 Small Step Semantics

Small steps semantics, is a transition system that describes how an abstract state machine would execute well typed programs expressed in a λ calculus. Small steps semantics gives local reductions and follows a deterministic evaluation principle. Other kind of operational semantics exist (e.g big step or non-deterministic. The Church-Rosser theorem for a calculus can give a proof that all evaluation strategies are equivalent for a calculus). We also need a notion of value. That is a term that accepts no more reductions under our strategy. We work here with call-by-value semantics. That is functions in a λ form are not further reduced and when the term is a function call the arguments are reduced to values before application.

Here is an example of transition system for small step semantics of the negative fragment of IPL:

$$\frac{}{\langle\rangle\ value} \qquad \frac{}{(\lambda x:\phi.M)\ value} \qquad \frac{M\ value}{(\lambda x:\phi.N)M \mapsto [M/x]N}$$

$$\frac{M\ value \qquad N\ value}{\text{fst}\langle M,N\rangle \mapsto M} \qquad \frac{N\ value \qquad N\ value}{\text{snd}\langle M,N\rangle \mapsto M}$$

$$\frac{M \mapsto M'}{\langle M,N\rangle \mapsto \langle M',M\rangle} \qquad \frac{M\ value \qquad N \mapsto N'}{\langle M,N\rangle \mapsto \langle M,N'\rangle} \qquad \frac{M \mapsto M'}{\text{fst}(M) \mapsto fst(M')}$$

$$\frac{M \mapsto M'}{\text{snd}(M) \mapsto snd(M')} \qquad \frac{M \mapsto M'}{(MN) \mapsto (M'N)} \qquad \frac{M value \qquad N \mapsto N'}{(MN) \mapsto (MN')}$$

Now we have *preservation* and *progress* property. Those are standard properties for any small step semantics transition system. They are formulated only on closed terms because, unlike the process of proof reduction, we only evaluate expressions that are closed.

Preservation If $\mathsf{nil} \vdash M : \phi$ and $M \mapsto M'$ then $\mathsf{nil} \vdash M' : \phi$

Progress If $\mathsf{nil} \vdash M : \phi$ then either $\exists M'.M \mapsto M'$ or $M\ value$

Finally we have the *weak normalization theorem* or *termination theorem*. That is pertinent to the specific choice of semantics. In the simply typed lambda calculus the reduction strategy does not change the normalization property. That is *strong normalization* can be shown. Moreover, from the

Church–Rosser theorem and the normalization property one can deduce the existence and unicity of canonical forms. For other systems this might not be the case.

Termination If $\mathsf{nil} \vdash M : \phi$ then $\exists V.V$ *value and* $M \mapsto^* V$. Where $\mapsto^*$ is the reflexive, transitive closure of $\mapsto$

4.3.2 Operational Semantics for Source Expressions

We extend the computational interpretation sketched above to encompass the necessity modality The interpretation goes as follows:

$x : \phi$	x stands for value of type ϕ
$s :: \phi$	s stands for a source expression ϕ
$[\![M/s]\!]N$	substitute the source expression M for s in N
$\mathrm{box}\, M$	quote the closed term M as a source expression
$\mathrm{let}\,\mathrm{box}(s) = M\,\mathrm{in}\,N$	evaluate M up to the (quoted expression of the) form $\mathrm{box}(M')$ and then evaluate $[\![M/s]\!]N$

We add the following values, a congruence rule and a reduction rule:

$$\frac{}{\text{box}\,(M)\ \textit{value}} \qquad \frac{M \mapsto M'}{\text{let box}(s) = M \text{ in } N \mapsto \text{let box}(u) = M' \text{ in } N}$$

$$\frac{}{\text{let box}(s) = \text{box}(M) \text{ in } N \mapsto [\![M/s]\!]N}$$

The importance of the typing is explained by Pfenning:

> The crucial restriction of the typing rules ensures that in an expression box(M), the term M does not refer to any free variables x that stand for values. It can, however, mention variables s that stand for source expressions. So when we substitute $[\![N/s]\!]\,\text{box}(M)$ then we are building a larger source expression from two smaller ones, N and M. Conversely, when we substitute a value $[V/x]\,\text{box}\,M = \text{box}(M)$ the source expression is not affected.

Returning to our example, the system is able to discriminate between the two examples. The first version of exp still has type $nat \to (nat \to nat)$ whereas the second can be rewritten in the new syntax and has type $nat \to \Box(nat \to nat)$. It is crucial that the first version fails to be written as a source code generator since if we try to re-implement the definition as:

```
exp(s(n))= box(lm x-> x* exp n x)
```

the expression is ill-typed due to the reference to the value variable n. The second version can be written in our extension of the language as a code generator and it is well typed:

```
exp'(0)= box(lm x-> 1): Box(nat -> nat)
exp(s(n))= let box(f)= exp'(n) in box(lm x-> x* f  x): Box(nat -> nat)
```

The computational reading sheds new light to modal theorems as programming combinators. The canonical inhabitant of Axiom 4 of modal logic (seen in the Curry–Howard fashion) is the type of the polymorphic metaprogram that quotes a quoted source code expression:

$$\frac{D}{\mathit{quote} = \lambda x : \Box\phi.\, \mathrm{let\,box}(s) = x \,\mathrm{in\,box\,box}(x) : \Box\phi \supset \Box\Box\phi}$$

The K axiom corresponds to the combinator that applies applicative source expressions and results to a larger source expression:

$$\lambda x : \Box(\phi \supset \psi).\lambda y : \Box\phi.\, \mathrm{let\,box}(s) = x \,\mathrm{in\,let\,box}(t) = y \,\mathrm{in\,box}(st) : \Box(\phi \supset \psi) \supset \Box\phi \supset \Box\psi$$

Finally the factivity axiom corresponds to unquoting a quoted source code expression:

$$\frac{D}{\mathit{unquote} = \lambda x : \Box\phi.\, \mathrm{let\,box}(s) = x \,\mathrm{in}\, s : \Box\phi \supset \phi}$$

These operations resemble monadic combinators in languages like Haskell or Scala (cf. [35],[36]). The connection of modality and monadic computation is thoroughly explored in [23]. This discussion, pushes towards a judgmental reconstruction of the possibility modality which we will not be discussing here.

Chapter 5

Proposal for Thesis contribution

In this paper 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 validity 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^{-}) that 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" Artemov2001 Artemov developed a constructive, proof-theoretic semantics for BHK proofs troelstra1988constructivism 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 semantics of a deductive system I can be viewed in a solely proof-theoretic manner as mappings of proof constructs of I into another proof system J (which we call justifications). 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). What's more, such a semantic relation can be treated logically giving rise to a modality of explicit necessity. Different sorts of necessity (K, D, $S4$, $S5$) have been offered an explicit counterpart under the umbrella of justification logic. Some of them have been studied within a Curry–Howard setting ArtBon07LFCS. Our paper focuses on K modality and should be viewed as the counterpart of Bellin2001 with justifications as we explain in ??.

5.1.1 Deductive Systems, Validity and Necessity

Following a framework championed by Lambek lambek1968deductive,lambek1969deductive, 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. *Reflexivity.* In both relations Γ and Δ are multisets of formulas (contexts) that enjoy reflexivity:

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

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

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 lambek1968deductive 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. i.e.

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

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 $Hom_J(\llbracket\Gamma\rrbracket, \llbracket A\rrbracket_J)$ cannot be empty (i.e. that $\llbracket\bullet\rrbracket_J$ 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 Bellin2001,kavvos2016system.

Examples of triplets $(I,\ J,\ \llbracket\bullet\rrbracket_J)$ of logical systems that fall under the definition above are: any intuitionistic system mapped to a classical one under the embedding $\llbracket A \supset B\rrbracket = \tilde{\neg} A \tilde{\vee} B$ 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 $\llbracket\bullet\rrbracket_J$ maps everything to $\top$) gives another example.

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 $\llbracket\bullet\rrbracket_J$ of implicative intuitionistic logic. In what follows we provide a natural deduction for an intuitionistic system I (truth), an axiomatization/specification of $\llbracket\bullet\rrbracket_J$ (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. Schematically we want:

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

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:

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

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

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

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 (**??**). See also discussion in Bellin2001,bierman2000intuitionistic). 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^{-}

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.

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)

$$\frac{}{P_k \in \mathsf{Prop_0}}\ \textsc{Atom} \qquad \frac{}{\top \in \mathsf{Prop_0}}\ \textsc{Top} \qquad \frac{A \in \mathsf{Prop_0}}{\Box A \in \mathsf{Prop_1}}\ \textsc{Box}$$

$$\frac{A \in \mathsf{Prop_0} \qquad B \in \mathsf{Prop_0}}{A \supset B \in \mathsf{Prop_0}}\ \textsc{Arr}_0 \qquad \frac{A \in \mathsf{Prop_1} \qquad B \in \mathsf{Prop_1}}{A \supset B \in \mathsf{Prop_1}}\ \textsc{Arr}_1$$

$$\frac{A \in \mathsf{Prop_0}}{[\![A]\!] \in [\![\mathsf{Prop_0}]\!]}\ \textsc{Brc}$$

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 $\circ$ for the empty context of $\mathsf{Prop_0}$ and $\mathsf{Prop_1}$ 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 $\Gamma \in \mathsf{Prop_0}$ (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. $x_i \rightsquigarrow s_i$). For the vacuous case when Γ is empty we require $[\![\circ]\!] = \dagger$ to be well formed.

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

$$\frac{x : A \in \Gamma}{\Gamma \vdash_{\mathsf{IPC}} A}\ \Gamma_0\text{-}\textsc{Refl} \qquad \frac{}{\Gamma \vdash_{\mathsf{IPC}} \top}\ \top_0\mathrm{I} \qquad \frac{\Gamma, x : A \vdash_{\mathsf{IPC}} B}{\Gamma \vdash_{\mathsf{IPC}} A \supset B}\ \supset_0\mathrm{I}$$

$$\frac{\Gamma \vdash_{\mathsf{IPC}} A \supset B \qquad \Gamma \vdash_{\mathsf{IPC}} A}{\Gamma \vdash_{\mathsf{IPC}} B}\ \supset_0\mathrm{E}$$

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 $\Delta \in [\![\mathsf{Prop}_0]\!]$

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

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

$$\frac{\Delta \vdash_{\mathsf{IPC}} [\![A \supset B]\!] \qquad \Delta \vdash_{\mathsf{IPC}} [\![A]\!]}{\Delta \vdash_{\mathsf{IPC}} [\![B]\!]}\ \mathrm{MP}$$

Finally, we have judgments in the $\Box$ed universe ($\mathsf{Prop_1}$). These are context reflection, the $\Box$ Intro-After-Elim rule, and the rules for intuitionistic implication between $\Box$ed types [1].

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

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

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

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

(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 $\Gamma \in \mathsf{Prop_1}$.

$$\frac{\circ \vdash_{\mathsf{IPC}} A \qquad \dagger \vdash_{\mathsf{IPC}} [\![A]\!]}{\Gamma \vdash_{\mathsf{IPC}} \Box A}\ I_{\Box A}$$

[1] The implication and elimination rules in $\mathsf{Prop_1}$ actually coincide with the ones in $\mathsf{Prop_0}$ 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 $\Box$ 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 $\Box$ types independently of I and, vice versa, one could add connectives in I that are not observed between $\Box$ed types.

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

Here 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.

Hilbert Style Formulation

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

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

It is easy to verify that axioms 1, 2 are derived theorems of Jcalc^{-} in Prop_0. The rule Modus Ponens is also admissible trivially, 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 Artemov2001. The proof fully depends on the provability requirements imposed in the $[\![\mathsf{Prop}_0]\!]$ fragment. [$\Box$Lifting Lemma]lemmaboxlift 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]lemmafirstadm

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.

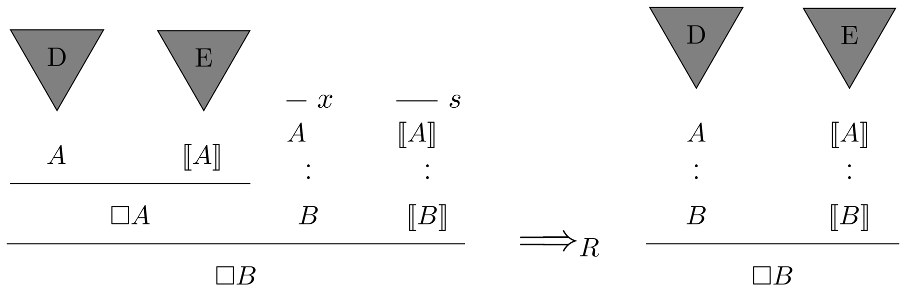
5.2.3 Harmony: Local Soundness and Local Completeness

Before we move on to show (Global) Soundness we provide evidence for the so called "local soundness" and "local completeness" of the $\Box$ connective following Gentzen's dictum. The local soundness and completeness for the $\supset$ connective is given elsewhere (e.g. bo:prawitz65a) and in Gentzen's original gentzen1935untersuchungen. 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 $\Box$ 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 here the "Elim-After-Singleton-Intro" sub-case. The exact same principle applies in the "Elim-after-Intro" of multiple $\Box$s shown in the Appendix (**??**).



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)

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

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

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^-]theoremfirstcon $\vdash \bot$

5.3 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 (i.e. extensionality η-rules) equalities for the 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.

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.

5.3.1 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 here the corresponding β, η equality rules since they are standard.

Judgments on Truth $\Gamma, A, B \in \mathsf{Prop_0}$ **and** $M := x_i \mid <> \mid \lambda x : A.\ M \mid (MM)$

$$\frac{x : A \in \Gamma}{\Gamma \vdash_{\mathsf{IPC}} x : A}\ \Gamma_0\text{-}\textsc{Refl} \qquad \frac{}{\Gamma \vdash_{\mathsf{IPC}} <>: \top}\ \top_0\mathrm{I}$$

$$\frac{\Gamma, x : A \vdash_{\mathsf{IPC}} M : B}{\Gamma \vdash_{\mathsf{IPC}} \lambda x : A.\ M : A \supset B}\ \supset_0\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathsf{IPC}} M : A \supset B \qquad \Gamma \vdash_{\mathsf{IPC}} M' : A}{\Gamma \vdash_{\mathsf{IPC}} (MM') : B}\ \supset_0\mathrm{E}$$

$+\ \beta\eta$ 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** $\mathsf{J} := s_i \mid C_i \mid \mathsf{J} * \mathsf{J}$

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

$$\frac{A, B \in \mathsf{Prop_0}}{\Delta \vdash C_{K^{A,B}} : [\![A \supset (B \supset A)]\!]}\ \textsc{Ax}_2$$

$$\frac{A, B, C \in \mathsf{Prop_0}}{\Delta \vdash C_{S^{A,B,C}} : [\![A \supset (B \supset C) \supset ((A \supset B) \supset (A \supset C))]\!]}\ \textsc{Ax}_3$$

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

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 $[\![\Gamma]\!] \vdash \mathsf{J} : [\![A]\!]$.

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 $(\mathsf{int}, \mathsf{int})$ 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)`, then 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 $\Box$ connective. We are directly translating proof tree equalities from **??** to proof term equalities. We generalize for arbitrary Γ' in the following subsection. We have, respectively, the following instances:

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

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

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

Following the expansion principle we obtain:

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

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

That gives an η-equality 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 $M' \& \mathsf{J}'$.

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

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

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 (**??**) generalized for the multiple intro case shown in the appendix (**??**).

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

5.3.2 Strong Normalization and small-step semantics

In the appendix (**??**) we provide a proof of normalization for natural deduction (via cut elimination). This is "essentially" a strong normalization result for the proof term system also. In general we have shown the congruence obtained from $=_{\beta\eta}$ rules gives a consistent equational system. Nevertheless, we leave this for an extended version of this paper. Instead, we sketch briefly a weaker result: 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 $\rightarrow$

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

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

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

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

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

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

Using the reducibility candidates proof method citeulike:993095) we show:

Termination Under Small Step Reduction With $\rightarrow^*$ 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 $M \rightarrow^* N$.

5.4 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. Through software development, often the implementations of such an interface might 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. First, *separate compilation* i.e. when implementations change we do not have to recompile client code and, yet, 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, with the following signature Σ: 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 Σ 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 doesn't have a redex: $\Sigma; \bullet \vdash \mathtt{pop(push}\ (1+1)\ \mathtt{Empty)} \rightsquigarrow \mathtt{pop(push}\ 2\ \mathtt{Empty)} : \mathtt{int}$
2. Contextualize the use of the signature at the final term in step 1:

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

3. Rewrite the previous judgment assuming (abstract) implementations for the

corresponding missing elements using the "known" specification for the shared elements.

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

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

$$\Sigma; \mathtt{x}'_1 : \Box\mathtt{intstack}, \mathtt{x}'_2 : \Box(\mathtt{int} \to \mathtt{intstack} \to \mathtt{intstack}), \mathtt{x}'_3 : \Box(\mathtt{intstack} \to \mathtt{int}) \vdash$$

$$\mathtt{let}\{\mathtt{x}_1\&\mathtt{s}_1\ \mathtt{be}\ \mathtt{x}'_1,\ \mathtt{x}_2\&\mathtt{s}_2\ \mathtt{be}\ \mathtt{x}'_2,\ \mathtt{x}_3\&\mathtt{s}_3\ \mathtt{be}\ \mathtt{x}'_3\}\ \mathtt{in}\ (\mathtt{x}_3(\mathtt{x}_2\ 2\ \mathtt{x}_1)\ \&\ \mathtt{s}_3 * (\mathtt{s}_2 * \bar{2} * \mathtt{s}_1)) : \Box\mathtt{int}$$

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

$$\Sigma; \circ \vdash$$

$$\mathtt{linker} = \lambda\mathtt{x}'_1.\ \lambda\mathtt{x}'_2.\lambda x'_3.$$

$$\mathtt{let}\{\mathtt{x}_1\&\mathtt{s}_1\ \mathtt{be}\ \mathtt{x}'_1,\ \mathtt{x}_2\&\mathtt{s}_2\ \mathtt{be}\ \mathtt{x}'_2,\ \mathtt{x}_3\&\mathtt{s}_3\ \mathtt{be}\ \mathtt{x}'_3\}\ \mathtt{in}(\mathtt{x}_3(\mathtt{x}_2\ 2\ \mathtt{x}_1)\ \&\ \mathtt{s}_3 * (\mathtt{s}_2 * \bar{2} * \mathtt{s}_1))$$

$$: \Box(\mathtt{instack}) \to \Box(\mathtt{int} \to \mathtt{intstack} \to \mathtt{intstack}) \to \Box(\mathtt{intstack} \to \mathtt{int}) \to \Box\mathtt{int}$$

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 typechecked code fragments of J as follows[2]:

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

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

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:

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

3. We thus obtain typechecked links using the $\Box_I$ rule. For example:

$$\frac{\Sigma; \circ \vdash_{\mathsf{IPC}} \texttt{push} : \texttt{int} \to \texttt{intstack} \to \texttt{intstack} \qquad \bullet \vdash_{\mathsf{IPC}} \mathsf{add\_array} : [\![\texttt{int} \to \texttt{intstack} \to \texttt{intstack}]\!]}{\Sigma; \circ \vdash_{\mathsf{IPC}} \texttt{push}\ \&\ \mathsf{add\_array} : \Box(\texttt{int} \to \texttt{intstack} \to \texttt{intstack})}$$

And analogously:

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

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

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

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

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

which reduces to:

$$\Sigma;\bullet \vdash \texttt{let}\{(\texttt{x}_1\&\texttt{s}_1)\ \texttt{be}\ (\texttt{empty}\ \&\texttt{create}()),\ (\texttt{x}_2\&\texttt{s}_2)\ \texttt{be}\ (\texttt{push}\ \&\ \texttt{add\_array}),\ \ (\texttt{x}_3\&\texttt{s}_3)\ \texttt{be}\ (\texttt{pop}\ \&\ \texttt{pop\_}$$

$$\texttt{in}\ (\texttt{x}_3(\texttt{x}_2\ 2\ \texttt{x}_1)\ \&\ \texttt{s}_3 * (\texttt{s}_2 * \bar{2} * \texttt{s}_1)) : \Box\texttt{int}$$

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

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

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:

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

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 girard1987linear. That will showcase an 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

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

"escape" the client to compute the stack calls and then return for the last addition. Our modality is concerned only with passing the 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 would like to explore such an extension in future work.

5.5 Related and Future Work

Directly related work with our calculus, in the same fashion that justification logic and LP Artemov2001 are related to modal logic, is Bellin2001. The work in Bellin2001 provides a calculus for explicit assignments (substitutions) which is actually a sub-case of Jcalc^- with $[\![\bullet]\!]$ 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 citeulike:5447115. 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 ArtBon07LFCS,bavera2015justification. 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 Pouliasis201471.

plainnat jcalcminustex

.1 Appendix

.1.1 Theorems

Deduction Theorem for Validity Judgments Given any $\Gamma, A, B \in \mathsf{Prop_0}$ then $\Gamma, x : A \vdash B \Longrightarrow [\![\Gamma]\!] \vdash [\![A \supset B]\!]$.

Proof. The proof proceeds by induction on the derivations $\Gamma, A, B \in \mathsf{Prop_0}$. Note that the axiomatization of $[\![\mathsf{Prop_0}]\!]$ derives the sequents:$\Delta \vdash [\![A \supset A]\!]$ for any $\Delta \in [\![\mathsf{Prop_0}]\!]$ (as in combinatory logic the I combinator is derived from SK). This handles the reflection case. The rest of the cases are treated exactly as in the proof of completeness of combinatorial axiomatization with respect to the natural deduction in intuitionistic logic. 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 be complete with respect to intuitionistic implication. □

$[\![\bullet]\!]$**Lifting Lemma** Given $\Gamma, A \in \mathsf{Prop_0}$ then $\Gamma \vdash A \Longrightarrow [\![\Gamma]\!] \vdash [\![A]\!]$.

Proof. The proof goes by induction on the derivations trivially for all the cases($\supset_{E_0}$ is treated using the App rule that internalizes Modus ponens). For the $\supset_{I_0}$ the previous theorem has to be used. $\square$

*

Proof. Assuming a derivation $\mathcal{D} :: \Gamma \vdash A$ from **??** 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 $\Box\Gamma \vdash \Box A$. $\square$

From the previous we get: *

Collapse $\Box$ Lemma If $\Box\Gamma \vdash \Box A$ for $\Gamma, A \in \mathsf{Prop_0}$ then $\Gamma \vdash A$.

Weakening For the N.D. system of Jcalc^{-}, with $\Gamma, \Gamma' \in \mathsf{Prop_0}$.

1. If $\Gamma \vdash A$ then $\Gamma, \Gamma' \vdash A$.
2. If $\Box\Gamma \vdash \Box A$ then $\Box\Gamma, \Box\Gamma' \vdash \Box A$.

Proof. By induction on derivations for the first item. For the second item, given $\Box\Gamma \vdash \Box A$ by the collapse lemma we get $\Gamma \vdash A$ which by the previous item gives $\Gamma, \Gamma' \vdash A$. Using the lifting lemma we get $[\![\Gamma, \Gamma']\!] \vdash [\![A]\!]$. Using the last two items we and the $\Box$ rule gives the result. $\square$

Contraction For the N.D. system of Jcalc, with $\Gamma, x : A, B \in \mathsf{Prop_0}$

If $\Gamma, x : A, x' : A, \Gamma' \vdash B$ then $\Gamma, x : A, \Gamma' \vdash B$.

2. If $\Box\Gamma, x : \Box A, x' : \Box A, \Box\Gamma' \vdash \Box B$ then $\Box\Gamma, x : \Box A, \Box\Gamma' \vdash \Box B$

Proof. Similarly with previous theorem. $\square$

Permutation For the N.D. system of Jcalc, with $\Gamma \in \mathsf{Prop_0}$ and $\pi\Gamma$ the collection of permutations of Γ.

1. If $\Gamma \vdash A$ and $\Gamma' \in \pi\Gamma$ then $\Gamma' \vdash A$.
2. If $\Box\Gamma \vdash \Box A$ then $\pi\Box\Gamma \vdash \Box A$.

Proof. As in the previous item. $\Box$

Substitution Principle The following hold for both kinds of judgments:

1. If $\Gamma, x : A \vdash M : B$ and $\Gamma \vdash N : A$ then $\Gamma \vdash M[N/x] : B$
2. If $[\![\Gamma]\!], s : [\![A]\!] \vdash \mathsf{J} : [\![B]\!]$ and $[\![\Gamma]\!] \vdash \mathsf{J}^{'} : [\![B]\!]$ then $[\![\Gamma]\!] \vdash \mathsf{J}[\mathsf{J}^{'}/\mathsf{s}][\![B]\!]$

All previous theorems can actually be stated for proof terms too. We should discuss the following:

Deduction Theorem / Emulation of λ abstraction If $\Gamma, A \in \mathsf{Prop_0}$ and $\Gamma, x : A \vdash M : B$ then there exists J s.t. $[\![\Gamma]\!] \vdash \mathsf{J} : [\![A \supset B]\!]$.

$[\![\bullet]\!]$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 proof.

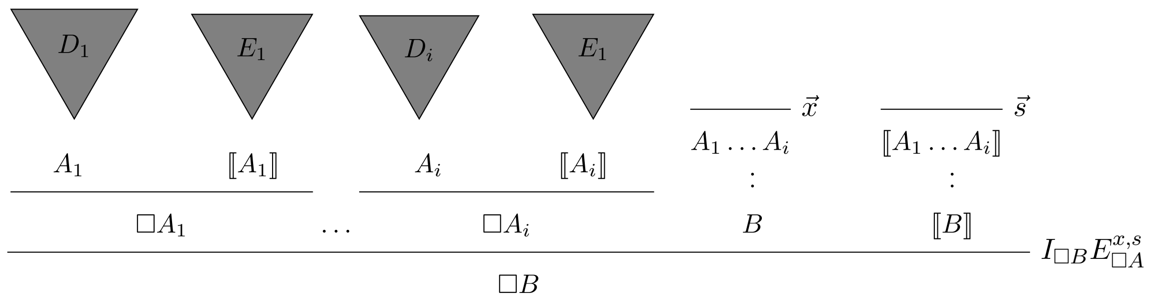
.1.2 Linking on the function space

The above mentioned algorithms permit for translating λ abstractions to polynomials of S, K combinators which is a standard result in the literature.

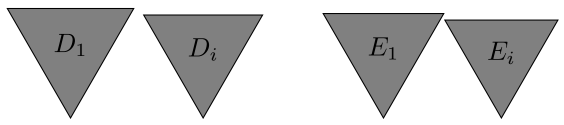
We do not give the details here but the translation is syntax driven as it can be seen by the inductive nature of the proofs.

Henceforth, we can generalize the construction in **??** so that it permits for dynamic linking of functions of the client (with missing implementations) such as λ`n` : `int`.`push n empty` dynamically given that the host actually implements a higher-order function space (that is it implements the combinators S, K in, say, own lambda calculus λ^J). Given implementations of `push_impl`, `empty_impl` the linker produces an application expression consisting of `push_impl`, `empty_impl`, S and K. The execution of the target expression will happen in the host after dereferencing push_impl, empty_impl (dynamic part) and the combinators S, K (constant part) as, say, lambdas (e.g. $K = \lambda^J x.\lambda^J y.x$).

.1.3 Gentzen's reduction Principle for □(General)



$$\Longrightarrow_R$$



$$\dfrac{\begin{matrix} A_1 \ldots\ldots\ A_i \\ \vdots \\ B \end{matrix} \qquad \begin{matrix} [\![A_1 \ldots\ldots A_i]\!] \\ \vdots \\ [\![B]\!] \end{matrix}}{\Box B}\ I_{\Box B}$$

.1.4 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 + $\bot$: $\vdash \bot$. The addition goes as follows:

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

Our proof strategy follows directly [?]. We construct an intercalation calculus [?] 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 [?] (Chapter 3) for the Hypotheses, Coercion, $\supset$, $\bot$ cases, we add the $\Box$ case.

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

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

$$\frac{\Gamma^{\downarrow} \vdash^{-} \bot \downarrow \qquad A \in \mathsf{Prop}}{\Gamma^{\downarrow} \vdash^{-} A \Uparrow}\ E_{\bot}$$

$$\frac{\Gamma^{\downarrow} \vdash \Box\Gamma' \downarrow \qquad \Gamma'^{\downarrow} \vdash A \Uparrow \qquad [\![\Gamma']\!] \vdash [\![A]\!]}{\Gamma^{\downarrow} \vdash \Box A \Uparrow}\ \Box_{IE}$$

Where $\Gamma^{\downarrow} \vdash \Box\Gamma'$ abbreviates $\forall A_i \in \Gamma'.\ \Gamma^{\downarrow} \vdash \Box A_i \downarrow$. 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. $\Box$

It is easy to see that this restricted proof system $\vdash^{-} \bot \Uparrow$. 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.[4]

To obtain completeness we add the rule:

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

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

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.

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

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

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

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

Sequent Calculus (Prop)

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

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

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 $A \Uparrow$. $\Box$

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

Collapse principle for normal deductions The following hold:

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

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:

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

Proof. As before. The cut rule case is handled by the $\Uparrow\downarrow$ and substitution for extractions principle showcasing that 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. $\square$

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 triple induction on the structure of the formula, and the given derivations and we leave it for a technical report. 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 **??** gives $\Gamma \vdash^{+} A \Uparrow$. By **??** and Cut Elimination we obtain $\Gamma \Rightarrow A$ which by **??** completes the proof. $\square$

As a result we obtain: *

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

Bibliography

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

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

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

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

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

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

[10] Alan Bawden et al. Quasiquotation in lisp. In *PEPM*, pages 4–12. Citeseer, 1999.

[11] Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. *Term Assignment for Intuitionistic Linear Logic: Preliminary Report*. 1992.

[12] 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.

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

[14] Melvin Fitting. *Proof methods for modal and intuitionistic logics*, volume 169. Springer, 1983.

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

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

[17] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*, volume 7. Cambridge University Press Cambridge, 1989.

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

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

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

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

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

[23] Satoshi Kobayashi. Monad as modality. *Theoretical Computer Science*, 175(1):29–74, 1997.

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

[25] 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.

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

[27] Eugenio Moggi. Notions of computation and monads. *Information and computation*, 93(1):55–92, 1991.

[28] Frank Pfenning. Computational interpretations of modalities. `http://www.cs.cmu.edu/~fp/courses/15816-s10/lectures/04-compmodal.pdf`, August 2009.

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

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

[31] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical structures in computer science*, 11(04):511–540, 2001.

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

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

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

[35] Philip Wadler. Comprehending monads. *Mathematical Structures in Computer Science*, 2(04):461–493, 1992.

[36] Philip Wadler. The essence of functional programming. In *Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '92, pages 1–14, New York, NY, USA, 1992. ACM.