Justification Logic And Type Theory: First Steps Towards Justified Typed Modality

Konstantinos Pouliaasis

June 30, 2014
Abstract

In this work, I will present the preliminary concepts of my research topic which rests in the intersection of Justification Logic, Constructive Modality and Type Theory. This is a survey paper towards the second exam requirement for my PhD candidacy under the supervision of Distinguished Professor Sergei Artemov at the Department of Computer Science of the Graduate Center at City University of New York.
Contents

1 Intuitionistic Logic 3
  1.1 Intuitionism ................................................. 3
    1.1.1 A bird’s eye view ...................................... 3
  1.2 IPL .......................................................... 5
    1.2.1 Basic Properties of Intuitionistic Entailment ........ 9
  1.3 Order Theoretic Semantics: Heyting Algebras .......... 10
  1.4 Adding proof terms ....................................... 13
  1.5 The computational view ................................... 16
    1.5.1 Definitional Equality .................................. 16
    1.5.2 Propositions as Types .................................. 19
  1.6 Categories for IPL ......................................... 20
    1.6.1 Definitions and Axioms of a Category ............... 20

2 Justification Logic 26
  2.1 A bird’s eye view .......................................... 26
  2.2 Minimal Justification Logic $J_0$ ......................... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3 Epistemic motivation</td>
<td>31</td>
</tr>
<tr>
<td>2.4 Proof theoretic view</td>
<td>33</td>
</tr>
<tr>
<td>2.5 The Logic of Proofs</td>
<td>35</td>
</tr>
<tr>
<td>2.6 Metatheoretic Results</td>
<td>35</td>
</tr>
<tr>
<td>2.6.1 Realization</td>
<td>37</td>
</tr>
<tr>
<td>2.6.2 Kripke - Fitting Semantics</td>
<td>39</td>
</tr>
<tr>
<td>3 Modal Type Theory</td>
<td>41</td>
</tr>
<tr>
<td>3.1 A bird’s eye view</td>
<td>41</td>
</tr>
<tr>
<td>3.2 Judgmental Reconstruction of Modal Logic</td>
<td>42</td>
</tr>
<tr>
<td>3.2.1 Properties of Entailment For the Judgmental Reconstruction</td>
<td>47</td>
</tr>
<tr>
<td>3.2.2 Adding proof terms</td>
<td>48</td>
</tr>
<tr>
<td>3.3 Computational Interpretation</td>
<td>50</td>
</tr>
<tr>
<td>3.3.1 Small Step Semantics</td>
<td>51</td>
</tr>
<tr>
<td>3.3.2 Operational Semantics for Source Expressions</td>
<td>53</td>
</tr>
</tbody>
</table>
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 math-
ematics 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 intuitionistic 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*, Dummett’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 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 Γ ctx.
Thus, in IPL we get the judgments \( \phi \in \text{Prop} \), \( \phi \text{ true} \) and \( \Gamma \text{ ctx} \):

\[
\begin{align*}
\phi \in \text{Prop} & \quad \phi \text{ is a (well-formed) proposition} \\
\phi \text{ true} & \quad \text{Proposition } \phi \text{ is true} \\
& \quad \text{i.e., has a proof.} \\
\Gamma \text{ ctx} & \quad \Gamma \text{ is a (well-formed) context of assumptions}
\end{align*}
\]
The natural deduction system of IPL is given below:

### Prop Formation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>$p_1 \in \text{Prop}$</td>
<td>$p_1 \in \text{Prop}$</td>
</tr>
<tr>
<td>TOP</td>
<td>$\top \in \text{Prop}$</td>
<td>$\top \in \text{Prop}$</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>$\bot \in \text{Prop}$</td>
<td>$\bot \in \text{Prop}$</td>
</tr>
<tr>
<td>ARR</td>
<td>$\phi_1 \in \text{Prop}$, $\phi_2 \in \text{Prop}$</td>
<td>$\phi_1 \supset \phi_2 \in \text{Prop}$</td>
</tr>
<tr>
<td>CONJ</td>
<td>$\phi_1 \in \text{Prop}$, $\phi_2 \in \text{Prop}$</td>
<td>$\phi_1 \land \phi_2 \in \text{Prop}$</td>
</tr>
<tr>
<td>DISJ</td>
<td>$\phi_1 \in \text{Prop}$, $\phi_2 \in \text{Prop}$</td>
<td>$\phi_1 \lor \phi_2 \in \text{Prop}$</td>
</tr>
</tbody>
</table>

### Context Formation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIL</td>
<td>nil ctx</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>ADD</td>
<td>$\Gamma \text{ctx}$, $\phi \in \text{Prop}$</td>
<td>$\Gamma, \phi \text{true ctx}$</td>
</tr>
</tbody>
</table>

### Context Reflection

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premises</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFL</td>
<td>$\Gamma \text{ctx}$, $\phi \text{true} \in \Gamma$</td>
<td>$\Gamma \vdash \phi \text{true}$</td>
</tr>
</tbody>
</table>
Top Introduction – Bottom Elimination

\[ \Gamma \vdash \top \quad \Gamma, \phi_1 \text{true} \vdash \phi_2 \text{true} \]

\[ \Gamma \vdash \bot \quad \Gamma \vdash \phi \text{true} \]

Implication Introduction and Elimination

\[ \Gamma, \phi_1 \text{true} \vdash \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_1 \supset \phi_2 \text{true} \quad \Gamma \vdash \phi_1 \text{true} \]

\[ \Gamma \vdash \phi_2 \text{true} \]

Conjunction Introduction and Elimination

\[ \Gamma \vdash \phi_1 \text{true} \quad \Gamma \vdash \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_1 \land \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_1 \text{true} \quad \Gamma \vdash \phi_1 \land \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_2 \text{true} \]

Disjunction Introduction and Elimination

\[ \Gamma \vdash \phi_1 \text{true} \quad \Gamma \vdash \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_1 \lor \phi_2 \text{true} \]

\[ \Gamma \vdash \phi_1 \lor \phi_2 \text{true} \]
<table>
<thead>
<tr>
<th>Reflexivity</th>
<th>Transitivity</th>
<th>Contraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Gamma \vdash \phi \lor \phi ) ( \Gamma, \phi \true \vdash \phi \true ) ( \Gamma, \phi_2 \true \vdash \phi \true ) [ \lor E ]</td>
<td>( \Gamma \vdash \psi \true ) ( \Gamma, \psi \true \vdash \phi \true ) ( \Gamma, \phi \true \vdash \phi \true )</td>
<td>( \Gamma, \phi \true, \phi \true \vdash \psi \true ) [ \lor E ]</td>
</tr>
</tbody>
</table>
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

\[ 0 \leq x \leq 1 \text{ for every } x \text{ in } L \]

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 \(a \rightarrow b\)) with the properties:

1. \(a \land x \leq b\)

---

\(^1\)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 \( \phi, \psi \) for any lower bound \( \chi \).

\[
\begin{align*}
\phi \wedge \psi & \leq \phi \\
\phi \wedge \psi & \leq \psi \\
\chi \leq \phi & \quad \chi \leq \psi \\
\hline
\chi \leq \phi \wedge \psi
\end{align*}
\]

We can axiomatize the join (\( \vee \)) (i.e. the least upper bound) of \( \phi, \psi \) for any upper bound \( \chi \) as follows.

\[
\begin{align*}
\phi & \leq \phi \vee \psi \\
\psi & \leq \phi \vee \psi \\
\phi \leq \chi & \quad \psi \leq \chi \\
\hline
\phi \vee \psi & \leq \chi
\end{align*}
\]

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

\[
\chi \leq 1
\]

which says that 1 is the greatest element.

We can axiomatize the existence of a least element as follows:
\[ 0 \leq \chi \]

which says that 0 is the least element.

Finally, to axiomatize $HAs$ we require the existence of exponentials for every $\phi$, $\psi$ as follows:

\[
\begin{array}{c}
\phi \land (\phi \supset \psi) \leq \psi \\
\chi \leq \phi \supset \psi
\end{array}
\]

Soundness and Completeness

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

\[
\begin{array}{c}
nil^+ = \top \\
(\Gamma, \phi)^+ = \Gamma^+ \land \phi^*
\end{array}
\]
1.4 Adding proof terms

*IPL* can be viewed as a minimal logic for proof constructs. Take the introduction rule for conjunction as an example:

\[
\Gamma \vdash \phi_1 \text{true} \quad \Gamma \vdash \phi_2 \text{true} \\
\Gamma \vdash \phi_1 \land \phi_2 \text{true}
\]

The rule says, “given a proof of \(\phi\) and a proof of \(\psi\) from assumptions \(\Gamma\), we should have a proof of \(\phi \land \psi\) from assumptions \(\Gamma\) at hand”. The simply typed lambda calculus makes this proof-tracking explicit by adding constructors and destructors on proof terms. Hypothetical assumptions are assigned unique variables as proof terms. The judgments of *IPL* are then transformed into judgments on proof terms. This is the well-known Curry–Howard Correspondence.

\[
\phi_1 \text{true}, \ldots, \phi_n \text{true} \vdash \phi \text{true} \quad \Rightarrow \quad x_1 : \phi_1, x_2 : \phi_2, \ldots, x_n : \phi_n \vdash M : \phi
\]

Simply typed lambda calculus
Type Formation

\[
\begin{align*}
\text{ATOM} & : P_i \in \text{Type} \\
\text{TOP} & : \top \in \text{Type} \\
\text{BOTTOM} & : \bot \in \text{Type} \\
\text{ARR} & : \phi_1 \in \text{Type}, \phi_2 \in \text{Type} \quad \phi_1 \rightarrow \phi_2 \in \text{Type} \\
\text{PROD} & : \phi_1 \in \text{Type}, \phi_2 \in \text{Type} \quad \phi_1 \times \phi_2 \in \text{Type} \\
\text{UNION} & : \phi_1 \in \text{Type}, \phi_2 \in \text{Type} \quad \phi_1 + \phi_2 \in \text{Type}
\end{align*}
\]

Context Formation

\[
\begin{align*}
\text{NIL} & : \text{nil ctx} \\
\text{ADD} & : \Gamma \text{ ctx} \quad \phi \in \text{Type} \quad x \text{ fresh in } \Gamma \quad \Gamma, x : \phi \text{ ctx}
\end{align*}
\]

Context Reflection

\[
\begin{align*}
\text{REFL} & : \Gamma \text{ ctx} \quad x : \phi \in \Gamma \quad \Gamma \vdash x : \phi
\end{align*}
\]

Top Introduction – Bottom Elimination

\[
\begin{align*}
\text{TII} & : \Gamma \vdash \langle \rangle : \top \\
\text{LE} & : \Gamma \vdash M : \bot \quad \Gamma \vdash \text{abort}[^φ](M) : \phi
\end{align*}
\]
Function Construction and Application

\[ \frac{\Gamma, x : \phi_1 \vdash M : \phi_2}{\lambda_{\text{ABS}} \quad \Gamma \vdash \lambda x.M : \phi_1 \rightarrow \phi_2} \quad \frac{\Gamma \vdash M : \phi_1 \rightarrow \phi_2}{\text{APP} \quad \Gamma \vdash (MM') : \phi_2} \]

Tuple Construction and Projections

\[ \frac{\Gamma \vdash M : \phi_1}{\text{TUP} \quad \Gamma \vdash \langle M, M' \rangle : \phi_1 \times \phi_2} \quad \frac{\Gamma \vdash M : \phi_1 \times \phi_2}{\text{LPRJ} \quad \Gamma \vdash \text{fst}(M) : \phi_1} \quad \frac{\Gamma \vdash M : \phi_1 \times \phi_2}{\text{RPRJ} \quad \Gamma \vdash \text{snd}(M) : \phi_2} \]

Union Construction and Elimination

\[ \frac{\Gamma \vdash M : \phi_1 \quad \Gamma \vdash \text{inj} \_ [\phi_2](M) : \phi_1 + \phi_2}{\text{INJL}} \quad \frac{\Gamma \vdash M : \phi_2 \quad \Gamma \vdash \text{inj} \_ [\phi_1](M) : \phi_1 + \phi_2}{\text{INJR}} \]

\[ \frac{\Gamma \vdash M : \phi_1 + \phi_2 \quad \Gamma, x : \phi_1 \vdash N : \phi \quad \Gamma, y : \phi_2 \vdash O : \phi}{\text{V\!E} \quad \Gamma \vdash \text{case } M \text{ of } \text{inj}_l(x) \leftrightarrow N | \text{inj}_r(y) \leftrightarrow O : \phi} \]
1.5 The computational view

Given the formulation of the $\lambda$-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 $\lambda$-calculus dynamics if we devise an execution strategy (an operational semantics) for program reduction that respects these rules. The correspondence between computations and Gentzen equational principles for proof terms is enlightened by specific metatheoretic results.

1.5.1 Definitional Equality

We want to think about when two proofs $M : A$ 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 $\beta$ 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.
The β rules are as follows:

\[
\frac{\Gamma \vdash M \equiv M' : \phi}{\Gamma \vdash M \equiv M'' : \phi}
\]

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 \( \text{fst} \)**

\[
\frac{\Gamma \vdash M \equiv M' : A \land B}{\Gamma \vdash \text{fst}(M) \equiv \text{fst}(M') : A}
\]

**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 \( \beta \) rules are as follows:
\[
\frac{
\Gamma \vdash M : \phi_1 \quad \Gamma \vdash N : \phi_2
}{
\Gamma \vdash \text{fst}(\langle M, N \rangle) \equiv M : \phi_1}
\]

\[
\frac{
\begin{array}{c}
\Gamma \vdash M : \phi_1 \\
\Gamma \vdash N : \phi_2
\end{array}
\quad
\frac{
\Gamma, x : A \vdash M : B \\
\Gamma \vdash N : A
}{
\Gamma \vdash \text{snd}(\langle M, N \rangle) \equiv N : \phi_2}
\quad
\frac{
\Gamma \vdash (\lambda x. M)(N) \equiv [N/x]M : B
}{
\beta \supset
}\]

\[
\frac{
\begin{array}{c}
\Gamma, x : \phi_1 \vdash N : \psi \\
\Gamma, y : \phi_2 \vdash O : \psi
\end{array}
\quad
\frac{
\Gamma \vdash P : \phi_1
}{
\Gamma \vdash (\text{case inj}_l(P) \text{ of } \text{inj}_l(x) \mapsto N \mid \text{inj}_r(y) \mapsto O) \equiv [P/x]N : \psi}
\]

\[
\frac{
\begin{array}{c}
\Gamma, x : \phi_1 \vdash N : \psi \\
\Gamma, y : \phi_2 \vdash O : \psi
\end{array}
\quad
\frac{
\Gamma \vdash Q : \phi_2
}{
\Gamma \vdash (\text{case inj}_r(Q) \text{ of } \text{inj}_l(x) \mapsto N \mid \text{inj}_r(y) \mapsto O) \equiv [Q/y]O : \psi}
\]

**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 “\(\beta\)” rules give rise to computational interpretation. The “\(\eta\)” rules impose properties that the computational model should satisfy (e.g. Church-Rosser property).

The \(\eta\) rules are given below:
\[
\begin{array}{c}
\Gamma \vdash M : T \\
\Gamma \vdash M \equiv \langle \rangle : T \\
\eta^\top \\

gamma \vdash \langle \text{fst}(M), \text{snd}(M) \rangle : A \land B \\
\eta^\land \\
\Gamma \vdash M : A \land B \\
\Gamma \vdash M : \phi \supset \psi \\
\eta^{\supset} \\
\Gamma \vdash \lambda x.Mx : \phi \supset \psi \\
\Gamma, z : \phi_1 + \phi_2 \vdash M : \psi \\
\Gamma \vdash N : \phi_1 + \phi_2 \\
\eta^{\lor} \\
\Gamma \vdash M[N/z] \equiv \text{case } N \text{ of } \\
| \text{inj}_l(x) \mapsto M[\text{inj}_l(x)/z] \\
| \text{inj}_r(y) \mapsto M[\text{inj}_r(y)/z] : \psi
\end{array}
\]

### 1.5.2 Propositions as Types

There is a correspondence between propositions and types:

<table>
<thead>
<tr>
<th>Propositions</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\top)</td>
<td>1</td>
</tr>
<tr>
<td>(A \land B)</td>
<td>(A \times B)</td>
</tr>
<tr>
<td>(A \supset B)</td>
<td>(\text{function } A \rightarrow B \text{ or } B^A)</td>
</tr>
<tr>
<td>(\bot)</td>
<td>0</td>
</tr>
<tr>
<td>(A \lor B)</td>
<td>(A + B)</td>
</tr>
</tbody>
</table>
1.6 Categories for IPL

In a Heyting Algebra, we have a preorder (or, partial order in the “textbook”
definition) \( \phi \leq \psi \) when \( \phi \) implies \( \psi \). HAs are insufficient, however, for the
treatment of proof objects (there can be at most one instance of \( \phi \leq \psi \) for
specific \( \phi, \psi \)). We can keep track of proofs, so if \( M \) is a proof from \( \Gamma \) to \( \psi \),
we want to think of it as a map \( M : \Gamma^+ \rightarrow \psi^+ \). 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.

1.6.1 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 \( \phi \) to \( \psi \) we write \( g : \phi \rightarrow \psi \), or
say that \( \phi \) is the domain of \( g \), and \( \psi \) the co–domain. We write \( \text{Dom}(g) = \phi \)
and \( \text{Cod}(g) = \psi \). We say that two arrows \( f \) and \( g \) are composable with
\( \text{Dom}(f) = \text{Cod}(g) \). If \( f \) and \( g \) are composable, they have a composite, an
arrow called \( f \circ g \). There is an identity for every object \( \phi \).
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 \( \Gamma \) there is a unique map \( \Gamma \to 1 \).

**Product** For any objects \( \phi \) and \( \psi \) there is an object \( \chi = \phi \times \psi \) equipped with arrows \( \text{fst} : \phi \times \psi \to \phi \) and \( \text{snd} : \phi \times \psi \to \psi \) that is the product of \( \phi \) and \( \psi \), which corresponds to the join \( \phi \land \psi \). For any other object \( \Gamma \) with arrows \( M : \Gamma \to \phi \) and \( \Gamma \to \psi \) there exists unique arrow, \( \langle M, N \rangle \) s.t.
\[ \text{fst} \circ \langle M, N \rangle = M(\beta \times 1) \text{ and } \text{snd} \circ \langle M, N \rangle = N(\beta \times 2). \]

\[
\begin{array}{c}
M : \Gamma \to \phi \quad N : \Gamma \to \psi \\
\hline
\langle M, N \rangle : \Gamma \to \phi \times \psi
\end{array}
\] \hspace{1cm} \text{EXIST}_1

\[
\begin{array}{c}
M : \Gamma \to \phi \quad N : \Gamma \to \psi \\
\hline
\text{fst} \circ \langle M, N \rangle : \Gamma \to \phi
\end{array}
\] \hspace{1cm} \text{EXIST}_2(\beta_1)

\[
\begin{array}{c}
M : \Gamma \to \phi \quad N : \Gamma \to \psi \\
\hline
\text{snd} \circ \langle M, N \rangle : \Gamma \to \phi
\end{array}
\] \hspace{1cm} \text{EXIST}_3(\beta_2)

\[
\begin{array}{c}
P : \Gamma \to \phi \times \psi \quad \text{fst} \circ P = M : \Gamma \to \phi \quad \text{snd} \circ P = N : \Gamma \to \psi \\
\hline
P = \langle M, N \rangle : \Gamma \to \phi \times \psi
\end{array}
\] \hspace{1cm} \text{UN}(\eta)

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

\[
\begin{array}{ccc}
C \quad & C \times A \\
\downarrow \lambda(h) \quad & \downarrow \quad \lambda(h) \times \text{id}_A \\
B^A \quad & B^A \times A \\
\end{array}
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \q
Co–Products  For any objects $\phi$ and $\psi$ there is an object $\chi = \phi + \psi$ equipped with arrows $\text{inl} : \phi \to \phi + \psi$ and $\text{inr} : \psi \to \phi + \psi$ that is the co-product of $\phi$ and $\psi$, which corresponds to the meet $\phi \land \psi$. For any other object $\omega$ with arrows $M : \omega \to \phi \lor \psi$ and $N : \omega \to \phi \lor \psi$ there exists unique arrow, $M, N$ s.t. $\{M, N\} \circ \text{inl} = M$ and $\{M, N\} \circ \text{inr} = N$. 

$$
\begin{array}{c}
\text{Exist}_1 \\
\text{inl} \circ O : \Gamma \to \phi + \psi \\
\hline
O : \Gamma \to \phi \quad P : \Gamma \to \psi \\
\text{Exist}_2 \\
\text{inr} \circ P : \Gamma \to \phi + \psi
\end{array}
$$

$$
\begin{array}{c}
\text{Exist}_3(\beta_1) \\
\{M, N\} \circ \text{inl} \circ O = M \circ O : \Gamma \to \omega \\
O : \Gamma \to \phi \\
M : \phi \to \omega \\
N : \psi \to \omega
\end{array}
$$

$$
\begin{array}{c}
\text{Exist}_3(\beta_2) \\
\{M, N\} \circ \text{inr} \circ P = N \circ P : \Gamma \to \omega \\
P : \Gamma \to \psi \\
M : \phi \to \omega \\
N : \psi \to \omega
\end{array}
$$

$$
\begin{array}{c}
\text{Exist}_3(\beta_2) \\
\text{snd} \circ \langle M, N \rangle : \Gamma \to \phi \\
M : \Gamma \to \phi \\
N : \Gamma \to \psi
\end{array}
$$

$$
\begin{array}{c}
\text{UN}(\eta) \\
O : \Gamma \to \phi \\
P : \Gamma \to \psi \\
U : \phi + \psi \to \omega \\
M : \phi \to \omega \\
N : \psi \to \omega \\
U \circ \text{inl} \circ O = M \\
U \circ \text{inr} \circ N = M \\
U = \{M, N\}
\end{array}
$$
Diagrammatically:
Chapter 2

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.

2.1 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 \( \mathcal{JL} \) the modality in question is witnessed by a reason and propositions of the kind become \( t : \phi \) that reads “\( \phi \) is justified by reason \( t \)”.

Witnesses in \( \mathcal{JL} \) 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, \( \mathcal{JL} \) 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], \( \mathcal{JL} \) 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.

### 2.2 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 \( \phi \) being a proposition. Here is the abstract syntax:

\[
egin{align*}
  j & := s_i | C_i | j_1 * j_2 | j_2 + j_2 \\
  \phi & := P_i | \bot | \phi_1 \land \phi_2 | \phi_1 \lor \phi_2 | \phi_2 \supset \phi_2 | \neg \phi | j : \phi
\end{align*}
\]
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.

<table>
<thead>
<tr>
<th>Propositional Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P1.</strong> $\vdash \phi \supset (\psi \supset \phi)$</td>
</tr>
<tr>
<td><strong>P2.</strong> $\vdash (\phi \supset (\psi \supset \chi)) \supset ((\phi \supset \psi) \supset (\phi \supset \chi))$</td>
</tr>
<tr>
<td><strong>P3.</strong> $\vdash \phi \supset \psi \supset \phi \land \psi$</td>
</tr>
<tr>
<td><strong>P4.</strong> $\vdash \phi \supset \psi \supset \psi \land \phi$</td>
</tr>
<tr>
<td><strong>P5.</strong> $\vdash \phi \supset \phi \lor \psi$</td>
</tr>
<tr>
<td><strong>P6.</strong> $\vdash \psi \supset \phi \lor \psi$</td>
</tr>
<tr>
<td><strong>P7.</strong> $\vdash (\phi \supset \psi) \supset (\neg \psi \supset \neg \phi)$</td>
</tr>
</tbody>
</table>
Justification Axioms

\[\text{Times. } \vdash j : (\phi \supset \psi) \supset (j' : \phi \supset j \ast j' : \psi)\]

\[\text{PlusL. } \vdash j : \phi \supset (j + j' : \phi)\]

\[\text{PlusR. } \vdash j : \phi \supset (j' + j : \phi)\]

The rule of the system is Modus Ponens.

\[
\begin{array}{c}
\phi \supset \psi \\
\hline
\phi \\
\hline
\psi \\
\end{array}
\quad \text{MP}
\]

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 \((C, \phi)\) of constants and propositions:
Axiomatic CS

\[ \vdash (C_1[\phi, \psi], \phi \rightarrow (\psi \rightarrow \phi)) \in \text{CS} \]

\[ \vdash (C_2[\phi, \psi, \chi], (\phi \supset (\psi \supset \chi)) \supset ((\phi \supset \psi) \supset (\phi \supset \chi))) \in \text{CS} \]

\[ \vdash (C_3[\phi, \psi], \phi \supset \psi \supset \phi \wedge \psi) \in \text{CS} \]

\[ \vdash (C_4[\phi, \psi], \phi \supset \psi \supset \psi \wedge \phi) \in \text{CS} \]

\[ \vdash (C_5[\phi, \psi], \phi \supset \phi \lor \psi) \in \text{CS} \]

\[ \vdash (C_6[\phi, \psi], \psi \supset \phi \lor \psi) \in \text{CS} \]

\[ \vdash (C_7[\phi, \psi], (\phi \supset \psi) \supset (\neg \psi \supset \neg \phi)) \in \text{CS} \]

\[ \vdash (C_8[\phi, \psi, j, j'], j : (\phi \supset \psi) \supset (j' : \phi \supset j \ast j' : \psi)) \in \text{CS} \]

\[ \vdash (C, \phi) \in \text{CS} \]

Finally we require reflection on CS:

Specification Reflection

\[ \vdash (C, \phi) \in \text{CS} \]

\[ \vdash C : \phi \]
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, Γ, φ ⊨ ψ implies Γ ⊨ φ ⊃ ψ |

2.3 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, unknownst 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 □\( \phi \) represents knowledge of \( \phi \) that, in contrast to the justified approach, is forgetful with respect to reasons. The formalization and the accompanying problem go as follows:

1. □\( B \), ‘I believe that the object in front of me is a red barn’.
2. □\( (B \land R) \), ‘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. □\( (B \land R \supset B) \), a knowledge assertion of a logical axiom.

   Within this formalization, it appears that epistemic closure in its modal form (2) is violated: line 2, □\( (B \land R) \), and line 3, \( (B \land R \supset B) \) are cases of knowledge whereas □\( B \) (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. \( v : (B \land R) \), ‘\( v \) is a reason to believe that the object in front of me is a red barn’;
3. \( a : (B \land R \supset B) \), 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. \( a : (B \land R \supset B) \supset (v : (B \land R) \supset a \ast v : B) \), by Times
5. \( v : (B \land R) \supset a \ast v : B \), from 3 and 4, by propositional logic; \( a \ast v : B \), from 2 and 5, by propositional logic.

### 2.4 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 ”, Gödel gave a classical provability interpretation of BHK using the modal system S4.

The standard axiomatization of S4 is given below:
The system $\text{S4}$

$\text{P1} - \text{P7}$

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

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

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

**Modus Ponens**

$$
\begin{array}{c}
\phi \supset \psi \\
\phi \\
\hline
\psi
\end{array}
$$

MP

Gödel’s result can be summarized in the following theorem:

**Gödel-Tarski Translation of Intuitionistic Logic**

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

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

After this result the state of the project of a classical interpretation of $\mathbf{BHK}$ semantics was as follows: IPC $\leftrightarrow$ S4 $\leftrightarrow$ ? $\leftrightarrow$ CLASSICAL PROOFS. Filling the missing part was the motivation behind $\mathbf{LP}$ the first Justification Logic.
2.5 The Logic of Proofs

An axiomatization of $\text{LP}$ with axiomatically appropriate constant specification as defined in 2.2 can be given as follows:

<table>
<thead>
<tr>
<th>The system $\text{LP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{P1} - \text{P7}$</td>
</tr>
<tr>
<td>$\text{Times.} \vdash j : (\phi \supset \psi) \supset (j' : \phi \supset j \ast j' : \psi)$</td>
</tr>
<tr>
<td>$\text{PlusL.} \vdash j : \phi \supset (j + j' : \phi)$</td>
</tr>
<tr>
<td>$\text{PlusR.} \vdash j : \phi \supset (j' + j : \phi)$</td>
</tr>
<tr>
<td>$\text{T.} \vdash j : \phi \supset \phi$</td>
</tr>
<tr>
<td>$\text{4.} \vdash j : \phi \supset (j! : j : \phi)$</td>
</tr>
</tbody>
</table>

2.6 Metatheoretic Results

The Deduction Theorem holds for $\text{LP}$

**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 \([LP] \phi \) and the transformation \( F_{[\Box]}(\phi) \) that replaces all subformulae of \( \phi \) of the kind \( j : \phi' \) with \( \Box \phi' \). The following theorem holds:

**Forgetful Projection Property**

\[
\Gamma \vdash_{[LP]} \phi \implies \Gamma \vdash_{S4} F_{[\Box]}(\phi)
\]

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

**Example:** Realization of \( \vdash_{S4} [\Box] \phi \lor [\Box] \psi \supset [\Box] ([\Box] \phi \lor [\Box] \psi) \)

1. \( \phi \supset [\Box] \phi \lor [\Box] \psi, \psi \supset [\Box] \phi \lor [\Box] \psi \) Prop. Axioms;
2. \( C : (\phi \supset [\Box] \phi \lor [\Box] \psi), C' : (\psi \supset [\Box] \phi \lor [\Box] \psi) \) From CS rules.
3. \( s : [\Box] \phi \supset C \ast s : [\Box] \phi \lor [\Box] \psi, \) From 1,2 and Times and MP
4. \( t : [\Box] \psi \supset C' \ast t : [\Box] \phi \lor [\Box] \psi, \) Similarly
5. \( C \ast s : [\Box] \phi \lor [\Box] \psi \supset (C \ast s + C' \ast t) : [\Box] \phi \lor [\Box] \psi \) and \( C' \ast t : [\Box] \phi \lor [\Box] \psi \supset (C \ast s + C' \ast t) : [\Box] \phi \lor [\Box] \psi, \) From Rplus, Lplus
6. \( s : [\Box] (C \ast s + C' \ast t) : [\Box] \phi \lor [\Box] \psi, \) From 3,5 by Propositional Logic.
7. \( t : [\Box] (C \ast s + C' \ast t) : [\Box] \phi \lor [\Box] \psi, \) From 4,5 by Propositional Logic.
8. \( s : [\Box] \phi \lor t : [\Box] (C \ast s + C' \ast t) : [\Box] \phi \lor [\Box] \psi, \) From 6,7 and Propositional Logic.
2.6.1 Realization

The realization gives an algorithmic process of transforming deductions in $\mathbf{S4}$ to $\mathbf{LP}$. By an $\mathbf{LP}$-realization of a modal formula $\phi$ we mean an assignment of proof polynomials to all occurrences of the modality $\square \phi$. Let $\phi^r$ be the image of $\phi$ under a realization $r$.

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

### $\square$ Polarities

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

Next we give a cut-free sequent formulation of $\mathbf{S4}$ (reference) with sequents $\Gamma \vdash \Delta$, where $\Gamma$ and $\Delta$ 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.
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.
Utilizing the previous systems the realization theorem shows:

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

### 2.6.2 Kripke - Fitting Semantics

In this section I will be discussing Kripke – Fitting Semantics\[15\] for Justification Logic $J_0 + CS$ very briefly.

A possible world justification logic model for the system $J_0 + CS$ 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:

\[
\forall \Gamma \in G \\
M, \Gamma \models P \iff \Gamma \in V(P) \text{ for } P \text{ a propositional letter} \\
\bullet \text{ It is not the case that } M, \Gamma \models \bot \\
\bullet \text{ } M, \Gamma \models \phi \supset \psi \text{ iff it is not the case that } M, \Gamma \models \phi \text{ or } M, \Gamma \models Y \\
\bullet \text{ } M, \Gamma \models (j : \phi) \text{ if and only if } \Gamma \in E(j, \phi) \text{ and, } \forall \Delta \in G \text{ with } \Gamma R \Delta, \text{ we have that } M, \Delta \models \phi.
\]

The following conditions on evidence functions are assumed:

\[
E(j, \phi \supset \psi) \cap E(j', \phi) \subseteq E(j \ast j', \psi) \\
E(j, \phi) \cup E(j', \phi) \subseteq E(j + j', \phi)
\]

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 3

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.

3.1 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\cite{Artemov}, and the “Gang of four” \cite{Gang} revived the interest in constructive modality and its computational view. Of course earlier work in the intersection of modal and constructive logic exists (cf. \cite{34},\cite{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” \cite{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.

### 3.2 Judgmental Reconstruction of Modal Logic

The $\Box$ fragment of Pfenning’s Judgmental reconstruction, consists of the judgments of $\text{IPL}$ 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 $\phi$, is simply unconditional evidence of truth of $\phi$".

1. If nil ⊢ φ true then φ Valid
2. If φ Valid then Γ ⊢ φ true

The logical judgments are now extended in the form:

\[ \phi_1 \text{ Valid}, \phi_2 \text{ Valid}, \ldots, \phi_n \text{ Valid}; \phi_1 \text{ true} \phi_2 \text{ true}, \ldots \phi_m \text{ true} \vdash \phi \text{ true} \]

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

\[ \Delta \vdash \phi \text{ Valid} \text{ and } \Delta, \phi \text{ Valid} \vdash J \text{ then } \Delta \vdash J \]

This principle can be rewritten utilizing the definition of validity:

**Substitution Principle For Validity** \( \Delta; \text{nil} \vdash \phi \text{ true} \text{ and } \Delta, \phi \text{ Valid}; \Gamma \vdash J \text{ then } \Delta; \Gamma \vdash J \)

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

**Validity Context Reflection**
In the typability rules we add the following:

\[
\frac{\phi \text{ Prop}}{\Box F} \quad \Box I
\]

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

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

\[
\frac{\Delta; \Gamma \vdash \Box \phi \text{ true}}{\Delta; \vdash \Box \phi \text{ true}} \quad \Box \text{E-UNSound}
\]

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

\[
\Delta; \Gamma \vdash \Box \text{true} \quad \Box \text{E-INCOMPLETE} \\
\Delta; \Gamma \vdash \text{true}
\]

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

\[
\Delta; \Gamma \vdash \Box \text{true} \quad \Delta, \Box \text{Valid}; \Gamma \vdash \psi \text{true} \\
\Delta; \Gamma \vdash \psi \text{true}
\]

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

**Prop Formation**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>( P \in \text{Prop} )</td>
</tr>
<tr>
<td>TOP</td>
<td>( \top \in \text{Prop} )</td>
</tr>
<tr>
<td>ARR</td>
<td>( \phi_1 \in \text{Prop}, \phi_2 \in \text{Prop} )</td>
</tr>
<tr>
<td>( \phi_1 \supset \phi_2 \in \text{Prop} )</td>
<td></td>
</tr>
</tbody>
</table>

**Context \( \Gamma \) Formation**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIL</td>
<td>( \text{nil ctx} )</td>
</tr>
<tr>
<td>( \Gamma \text{ctx} \phi \in \text{Prop} )</td>
<td></td>
</tr>
<tr>
<td>( \Gamma, \phi \text{true ctx} )</td>
<td></td>
</tr>
</tbody>
</table>

\( \Gamma \text{-ADD} \)

\( \Gamma \text{-ADD} \)
Context $\Delta$ Formation

\[
\begin{array}{c}
\text{nil ctx} \\
\hline
\text{NIL}
\end{array}
\quad \Delta \text{ ctx} \quad \phi \in \text{Prop} \quad \Delta \text{-ADD}
\quad \Delta, \phi \text{ Valid ctx}
\]

Compound $\Gamma; \Delta$ Context

\[
\Delta \text{ ctx} \quad \Gamma \text{ ctx} \\
\hline
\Gamma; \Delta \text{-F}
\Delta; \Gamma \vdash \text{ctx}
\]

Context $\Gamma$ Reflection

\[
\Delta; \Gamma \text{ ctx} \quad \phi \text{ true} \in \Gamma \\
\hline
\Gamma \text{-REFL}
\Delta; \Gamma \vdash \phi \text{ true}
\]

Context $\Delta$ Reflection

\[
\Delta; \Gamma \text{ ctx} \quad \phi \text{ Valid} \in \Delta \\
\hline
\Delta \text{-REFL}
\Delta; \Gamma \vdash \phi \text{ true}
\]

Top Introduction

\[
\text{TI} \\
\Delta; \Gamma \vdash \top \text{ true}
\]
Implication Introduction and Elimination

\[
\begin{array}{c}
\Delta; \Gamma, \phi_1 \text{ true} \vdash \phi_2 \text{ true} \\
\hline
\Delta; \Gamma \vdash \phi_1 \supset \phi_2 \text{ true} \\
\end{array}
\]

\[
\begin{array}{c}
\Delta; \Gamma \vdash \phi_1 \supset \phi_2 \text{ true} \\
\hline
\Delta; \Gamma \vdash \phi_2 \text{ true} \\
\end{array}
\]

Necessity Introduction and Elimination

\[
\begin{array}{c}
\Delta; \text{nil} \vdash \phi \text{ true} \\
\hline
\Delta; \Gamma \vdash \Box \phi \text{ true} \\
\end{array}
\]

\[
\begin{array}{c}
\Delta; \Gamma \vdash \Box \phi \text{ true} \\
\hline
\Delta; \Gamma \vdash \psi \text{ true} \\
\end{array}
\]

3.2.1 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 \( \Delta; \Gamma, \phi \text{ true}, \Gamma' \vdash \psi \text{ true} \) and \( \Delta; \Gamma \vdash \phi \text{ true} \) then \( \Delta; \Gamma, \Gamma' \vdash \psi \text{ true} \)
- If \( \Delta, \phi \text{ Valid}, \Delta'; \Gamma \vdash \psi \text{ true} \) and \( \Delta; \text{nil} \vdash \phi \text{ true} \) then \( \Delta \Delta', \Gamma \vdash \psi \text{ true} \)

Weakening, Contraction and Exchange properties can be also shown to hold.
### 3.2.2 Adding proof terms

The system can be assigned proof terms as follows:

<table>
<thead>
<tr>
<th>Context $\Gamma$ Formations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Context } \Gamma$</td>
<td>$\text{Formation}$</td>
</tr>
<tr>
<td>$\text{nil ctx}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\Gamma, x : \phi \text{ ctx}$</td>
<td>$\Gamma, x : \phi \text{ ctx}$</td>
</tr>
<tr>
<td>$\text{ctx}$</td>
<td>$\phi \in \text{Prop}$</td>
</tr>
<tr>
<td>$\phi \notin \Gamma$</td>
<td>$\Gamma$-ADD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Context $\Delta$ Formations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Context } \Delta$</td>
<td>$\text{Formation}$</td>
</tr>
<tr>
<td>$\text{nil ctx}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\Delta, s :: \phi \text{ ctx}$</td>
<td>$\Delta, s :: \phi \text{ ctx}$</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>$\phi \in \text{Prop}$</td>
</tr>
<tr>
<td>$s \notin \Delta$</td>
<td>$\Delta$-ADD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compound $\Gamma; \Delta$ Context</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$ ctx</td>
<td>$\Gamma$ ctx</td>
</tr>
<tr>
<td>$\Delta, \Gamma \vdash \text{ctx}$</td>
<td>$\Delta,-\text{F}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Context $\Gamma$ Reflection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta; \Gamma, x : \phi, \Gamma' \vdash x : \phi$</td>
<td>$\Gamma$-REFL</td>
</tr>
</tbody>
</table>

| $\Delta, x : \phi, \Gamma \vdash x : \phi$ | $\Gamma$-REFL |
Context \[\Delta\text{ Reflection}\]

\[
\begin{array}{c}
\Delta, s :: \phi, \Delta', \Gamma \vdash s : \phi \\
\hline
\Delta - \text{Refl}
\end{array}
\]

Top Introduction

\[
\begin{array}{c}
\Delta; \Gamma \vdash \langle \rangle : \top \\
\hline
\top I
\end{array}
\]

Implication Introduction and Elimination

\[
\begin{array}{c}
\Delta; \Gamma, x : \phi_1 \vdash M : \phi_2 \\
\Delta; \Gamma \vdash \lambda x : \phi_1. M : \phi_1 \supset \phi_2 \quad \supset I \\
\Delta; \Gamma \vdash M : \phi_1 \supset \phi_2 \\
\Delta; \Gamma \vdash N : \phi_1 \\
\hline
\Delta; \Gamma \vdash (MN) : \phi_2 \\
\supset E
\end{array}
\]

Necessity Introduction and Elimination

\[
\begin{array}{c}
\Delta; \text{nil} \vdash M : \phi \\
\hline
\square I
\end{array}
\]

\[
\begin{array}{c}
\Delta; \Gamma \vdash \text{box}(M) : \square \phi \text{ true} \\
\hline
\Delta; \Gamma \vdash M : \square \phi \text{ true} \\
\Delta, s :: \phi; \Gamma \vdash N : \psi \\
\hline
\Delta; \Gamma \vdash \text{let box}(s) = M \text{ in } N : \psi \\
\square E
\end{array}
\]
3.3 Computational Interpretation

One of the possible ways to read $\Box \phi$ is as representing source code of type $\phi$. 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 [11]. We introduce the concept and the application of the calculus with a motivating example following [28].

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

\[
\begin{align*}
\text{exp}(0) &= \lambda m \ x \rightarrow 1 \\
\text{exp}(s(n)) &= \lambda m \ x \rightarrow x \ast \text{exp} \ n \ x \\
\text{exp}'(0) &= \lambda m \ x \rightarrow 1 \\
\text{exp}(s(n)) &= \text{let } f = \text{exp}' n \text{ in } \lambda m \ x \rightarrow x \ast f \ x
\end{align*}
\]

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

\[
\lambda m \ x \rightarrow x \ast \text{exp}(s(0)) \ x
\]

the second though recurs completely on its argument unfolding to:

\[
\lambda m \ x \rightarrow x \ast (\lambda m \ x \rightarrow x \ast (\lambda m \ x \rightarrow 1)) \ x \ x
\]

which after reduction under $\lambda$ can be reduced to:

\[
\lambda m \ x \rightarrow x \ast x \ast 1
\]
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 how □ types can be useful to discriminate between the two. Prior to this we have to speak about about operational semantics.

### 3.3.1 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:
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 \( \text{nil} \vdash M : \phi \) and \( M \rightarrow M' \) then \( \text{nil} \vdash M' : \phi \)

**Progress** If \( \text{nil} \vdash M : \phi \) then either \( \exists M'. M \rightarrow M' \) or \( M \text{ value} \)

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

### 3.3.2 Operational Semantics for Source Expressions

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x : \phi$</td>
<td>$x$ stands for value of type $\phi$</td>
</tr>
<tr>
<td>$s :: \phi$</td>
<td>$s$ stands for a source expression $\phi$</td>
</tr>
<tr>
<td>$[M/s]N$</td>
<td>substitute the source expression $M$ for $s$ in $N$</td>
</tr>
<tr>
<td>box $M$</td>
<td>quote the closed term $M$ as a source expression</td>
</tr>
<tr>
<td>let box$(s) = M$ in $N$</td>
<td>evaluate $M$ up to the (quoted expression of the)</td>
</tr>
<tr>
<td></td>
<td>form box$(M')$ and then evaluate $[M/s]N$</td>
</tr>
</tbody>
</table>

We add the following values, a congruence rule and a reduction rule:
The importance of the typing is explained by Pfenning:

The crucial restriction of the typing rules ensures that in an expression \( \text{box}(M) \), 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 \( \text{nat} \to (\text{nat} \to \text{nat}) \) whereas the second can be rewritten in the new syntax and has type \( \text{nat} \to \square(\text{nat} \to \text{nat}) \). 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:

\[
\text{exp}(s(n)) = \text{box}(lm \ x \to x \times \text{exp} \ n \ x)
\]
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:

\[
\begin{align*}
\text{exp}'(0) &= \text{box}(\lambda m \ x \rightarrow 1) : \text{Box}(\text{nat} \rightarrow \text{nat}) \\
\text{exp}(s(n)) &= \text{let box}(f)= \text{exp}'(n) \text{ in box}(\lambda m \ x \rightarrow x \cdot f \ x) : \text{Box}(\text{nat} \rightarrow \text{nat})
\end{align*}
\]

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:

\[
D
\]

\[
\begin{align*}
\text{quote} &= \lambda x : \Box \phi. \text{let box}(s) = x \text{ in box}(x) : \Box \phi \supset \Box \Box \phi
\end{align*}
\]

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

\[
\begin{align*}
\lambda x : \Box(\phi \supset \psi). \lambda y : \Box \phi. \text{let box}(s) = x \text{ in let box}(t) = y \text{ in box}(st) : \Box(\phi \supset \psi) \supset \Box \phi \supset \Box \psi
\end{align*}
\]

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

\[
D
\]

\[
\begin{align*}
\text{unquote} &= \lambda x : \Box \phi. \text{let box}(s) = x \text{ in } s : \Box \phi \supset \phi
\end{align*}
\]
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.
Bibliography


