J-Calc: A typed lambda calculus for Intuitionistic Justification Logic

Konstantinos Pouliasis
Department of Computer Science
The Graduate Center at City University of New York
NY, USA
Kpouliasis@gc.cuny.edu

Giuseppe Primiero
FWO - Research Foundation Flanders
Center for Logic and Philosophy of Science
Ghent University, Belgium
Giuseppe.Primiero@UGent.be

In this paper we offer a system J-Calc that can be regarded as a typed λ-calculus for the $\{\rightarrow, \bot\}$ fragment of Intuitionistic Justification Logic. We offer different interpretations of J-Calc, in particular, as a two phase proof system in which we proof check the validity of deductions of a theory T based on deductions from a stronger theory T'. We establish some first metatheoretic results.

1 Introduction

A plausible reading of Gödel's incompleteness results ([?]) is that the notion of "validity" diverges from that of "truth within a specific theory": given a theory that includes enough arithmetic, there are statements whose validity can only be established in a theory of larger proof-strength. This phenomenon can be shown even with non-Gödelian arguments in the relation e.g. between $\mathsf{I}\Delta_0$ and $\mathsf{I}\Sigma_1$ arithmetic [?], $\mathsf{I}\Sigma_1$ and PA, PA and ZF, etc. [?, ?]. The very same issues arise in automated theorem proving. A good example is given by type systems and interactive theorem provers (e.g. Coq, Agda) of the functional paradigm. In such systems, when termination of functions has to be secured, one might need to invoke stronger proof principles. The need for reasoning about two kinds of proof objects within a type system is apparent most of all when one wants to establish non-admissibility results for a theory T that can, in contrast, be proved in some stronger T'. The type system, then, has to reconcile the existence of a proof object of some type ϕ in some T' and a proof object of type $\neg\exists s.Prov_T(s,\phi)$ that witnesses the non-provability of ϕ (in T).

In this work, we argue that the explicit modality of Justification Logic [?] can be used to axiomatize relations between objects of two different calculi such as those mentioned above. It is well known that the provability predicate can be axiomatized using a modality [?], [?]. The Logic of Proofs LP [?] goes further and provides explicit proof terms (*proof polynomials*) to inhabit judgments on validity. By translating reasoning in Intuitionistic Propositional Calculus (IPC) to classical proofs, LP obtains classical semantics for IPC through a modality (inducing a BHK semantics). In this paper we axiomatize the relation of the two kinds of proof objects explicitly, by creating a modal type theory that reasons about bindings of objects from two calculi: a lower-level theory T, formulated as IPC with Church-style λ-terms representing intuitionistic proof objects; and a higher-level, possibly stronger theory T' in a classical setting, fixed as foundational and validity preserving, with *justifications* representing classical proof objects. These latter objects are axiomatized as justifications (i.e. terms of Justification Logic) and are used to interpret classically (meaning *truth-functionally*) the constructions of the intuitionistic natural deduction. A *binding witness* constructor *JBox* creates witnesses of bindings between objects of the two kinds. This is how necessity is introduced: by proof-checking deductions of T with deductions of T', we reason constructively (i.e. within T) about admissibility of valid (via T') statements in T. We thus assume that T and T' contain both enough arithmetic and hence they both *capture* provability predicates: their own and of one another. In turn, both can capture the existence of proof bindings, i.e. proofs that a sentence is provable in both T and in T'.

A possible application of the presented type theory comes from the computational reading of our extended Curry–Howard correspondance. We can read functions within $\Box$ types indexed by justifications as constructs that link processes for modular programming in functional languages like ML.[1] Such programs are used to map well–typed constructs importing and using module signatures into their residual programs. By residual programs we mean programs where all instances of module types and function calls are replaced by (i.e. *linked* to) their actual implementations, which remain hidden in the module. We believe that, with slight modifications, our type system can find a natural application in this setting. Here we focus on the type system itself and not on its operational semantics.

The backbone of this work is the idea of representing the proof theoretic semantics for IPC through modality that stems from [?],[?]. An operational approach to modality related to this work can be found in [?]. The modularity of LP, i.e. its ability to realize other kinds of modal reasoning with proper changes in the axiomatization of proof polynomials, was shown with the development of the family of Justification Logics [?]. This ability is easily seen to be preserved here. Our work tries to incorporate the rich type system and modularity of Justification Logic within the proofs-as-programs doctrine. For that reason, we obtain an extension of Curry-Howard correspondance ([?], [?]) and adopt the judgmental approach of Intuitionistic Type Theory ([?], [?], [?], [?], [?]). Our system borrows from other modal calculi developed within the judgmental approach (e.g. [?], [?],[?]), and especially [?] for the modal logic K. A main difference of our system with those systems, as well as with previous λ-calculi for LP ([?], [?]) is that our type system hosts a two-kinded typing relation for proof objects of the same formulae. It can be viewed as an attempt to add proof terms for validity judgments as presented in [?]. The resulting type system adopts dependent typing ([?], [?]) to relate the two kinds of proof objects with modality. The construction of the type universe as well as of justificational terms draws a lot from ideas in [?] and from [?]. Extending typed modal calculi with additional (contextual) terms of dependent typing can be also found in [?].

2 A road map for the type system

The present system can be viewed as a calculus of reasoning in three interleaving phases. Firstly, reasoning about proof objects in the implicational fragment of an intuitionistic theory T in absence of any metatheoretic assumptions of validity (*Here*), introduced in Section 3. This calculus

[1] See [?].

is formalized by the turnstile $\Gamma \vdash_{\mathsf{IPC}}$[2] where Γ contains assumptions on proofs of sentences in T. The underlying logic is intuitionistic, the system corresponding to the implicational fragment of simply typed lambda calculus. Secondly, reasoning with justifications, corresponding to reasoning about proof objects in some fixed foundational system T' (*There*), introduced in Section 4. We suppose that T' provides the intended semantics for the intuitionistic system T. The corresponding turnstile is $\Delta \vdash_{\mathsf{J}}$. Abstracting from any specific metatheory, all that matters from a purely logical point of view is that the theory of the interpretation should – at least – include as much logic as the implicational fragment of T and it should satisfy some minimal provability conditions applied to the provability predicate of T. Finally, reasoning (in *both* systems) about existence of bindings [3] between proof objects in the implicational fragment of the two axiomatic systems, introduced in Section 5. This mode of reasoning is axiomatized within the full turnstile $\Delta; \Gamma \vdash \mathsf{JC}$. The core of that system is the $\Box$-Introduction rule, which allows to express constructive reasoning on binding existence. The idea is – ignoring contextual reasoning for simplicity – that binding a construction in T with a justification of the same type from T' we obtain a proof of a constructive (or, admissible in T) validity. The full turnstile $\Gamma; \Delta \vdash$ is a modal logic that "zips" mutual reasoning between the two calculi. Within this framework we obtain the Curry-Howard correspondence for justification logic under K modal reasoning. Before presenting this mutual reasoning at any arbitrary level of nesting, we first introduce JCalc_1 which is a restriction of the calculus up to degree 1.

We fix a countable universe of propositions (P_i) that corresponds to sentences of T together with the bottom type ($\bot$). The elements of this universe can be inhabited either by constructions or justifications.[4] We will need, accordingly, two kinds of inhabitation relations for each proposition. We will be writing $M : \phi$ for a construction M of type ϕ in T. Such construction in T does not entail its necessity: a corresponding justification from T' has to be obtained. We will be writing $j : \mathsf{Just}\ \phi$ to express the fact that j is a justification (proof in T') of the proposition ϕ. When there is no confusion we will be abbreviating this by $j :: \phi$. The justification (j) of ϕ in T' alone entails its validity but not its admissibility in T (*constructive necessity*). This is expressed by the proposition – type $\Box^{\mathsf{j}}\phi$. A construction of $\Box^{\mathsf{j}}\phi$ can be obtained only when the (weaker) theory T actually "responds" with a construction M of the type ϕ to the valid fact ϕ known from T' by deducing j. Hence, once (and only if) we have $j :: \phi$ then $\Box^{\mathsf{j}}\phi$ can be regarded as a well formed proposition. The stronger theory might be able to judge about $\Box^{\mathsf{j}}\phi$ (given $j :: \phi$) and prove e.g, $u :: \Box^{\mathsf{j}}\phi$. In that case T' "knows" that ϕ is admissible in T. Or, more interestingly, we might have $u :: \neg\Box^{\mathsf{j}}\phi$, meaning that T' proves that ϕ cannot be proved in T. In other words, when reasoning with justifications, the universe of types is *contextual.* To speak about an admissible (or, constructive) necessity of a proposition we need a corresponding proof object j in T' that establishes its validity.

[2]One could alternatively use an additional constant symbol null and write $\mathsf{null}; \Gamma \vdash_{\mathsf{IPC}}$ to denote reasoning purely in T and, thus, in absence of any metatheoretic environment.

[3]The reason we insist on *binding witnesses* and not bindings themselves should become clear when we introduce them.

[4]Or, equivalently, there is a bijection between sentences of a theory T and their intended interpretations in T'.

3 Reasoning without foundational assumptions: IPC

Reasoning about the implicational fragment of the constructive theory (T) (i.e. without formulating provability statements) is done within the implicational fragment of the simply typed lambda calculus. We start by giving the grammar for the metavariable ϕ used in the rules.

$$\phi := P_i | \bot | \phi \to \phi$$

The calculus is presented by introducing: the universe of types Prop_0; rules for constructing well-formed contexts of simple propositional assumptions Γ_0; the rules governing $\vdash_{\mathsf{IPC}}$.

$$\frac{}{P_i \in \mathsf{Prop}_0}\ \textsc{Atom}_0 \qquad \frac{}{\bot \in \mathsf{Prop}_0}\ \textsc{Bot}_0 \qquad \frac{\phi_1 \in \mathsf{Prop}_0 \qquad \phi_2 \in \mathsf{Prop}_0}{\phi_1 \to \phi_2 \in \mathsf{Prop}_0}\ \textsc{Impl}_0$$

$$\frac{}{\mathsf{nil} \vdash_{\mathsf{IPC}} \mathsf{wf}}\ \textsc{Nil}_0 \qquad \frac{\Gamma_0 \vdash_{\mathsf{IPC}} \mathsf{wf} \qquad \phi \in \mathsf{Prop}_0}{\Gamma_0, x : \phi \vdash_{\mathsf{IPC}} \mathsf{wf}}\ \Gamma_0\text{-}\textsc{Exp}$$

$$\frac{\Gamma_0 \vdash_{\mathsf{IPC}} \mathsf{wf} \qquad x : P_i \in \Gamma_0}{\Gamma_0 \vdash_{\mathsf{IPC}} x : P_i}\ \Gamma\text{-}\textsc{Refl} \qquad \frac{\phi \in \mathsf{Prop}_0 \qquad \Gamma_0 \vdash_{\mathsf{IPC}} M : \bot}{\Gamma_0 \vdash_{\mathsf{IPC}} \mathit{abort}^{\phi} M : \phi}\ \textsc{False}$$

$$\frac{\Gamma_0, x : \phi_1 \vdash_{\mathsf{IPC}} M : \phi_2}{\Gamma_0 \vdash_{\mathsf{IPC}} \lambda x : \phi_1 . M : \phi_1 \to \phi_2}\ {\to}\mathrm{I} \qquad \frac{\Gamma_0 \vdash_{\mathsf{IPC}} M : \phi_1 \to \phi_2 \qquad \Gamma_0 \vdash_{\mathsf{IPC}} M' : \phi_1}{\Gamma_0 \vdash_{\mathsf{IPC}} (MM') : \phi_2}\ {\to}\mathrm{E}$$

4 Reasoning in the Presence of Foundations: A calculus of Justifications J

Reasoning in the presence of minimal foundations corresponds to reasoning on the existence of proof objects in the foundational theory T'. The minimal foundational assumptions from the logical point of view is that T' knows at least as much logic as T does. The more non-logical axioms in T, the more the specifications T' should satisfy (one needs stronger foundations to justify stronger theories). Abstracting from any particular T and T' though, and assuming only that T is a constructive theory, the minimal specifications about existence of proofs in T' are:

- to have "enough" types to provide – at least – an intended interpretation of every type ϕ of T to a unique type $\mathsf{Just}\ \phi$. In other words a subset of the types of T' should serve as interpretations of types in T;
- to have – at least – proof objects for all the instances of the axiomatic characterization of the IPC fragment described above;[5]

[5]If we extend our fragment we should extend our specifications accordingly but this can be easily done directly as in full justification logic. We choose to remain within this fragment for economy of presentation.

- to include some modus ponens rule which translates as: the existence of proof objects of types Just $(\phi \to \psi)$ and of type Just ϕ in T' should imply the existence of a proof object of the type Just ϕ.

4.1 Minimal Justification Logic J-Calc$_1$

Under these minimal requirements, we develop a minimal justification logic that is able to realize modal reasoning as reasoning on the existence of bindings between proofs of T and T'. We first realize modal reasoning restricted to formulae of degree (i.e. level of $\Box$-nesting) 1. Such a calculus will be used as a base to build a full modal calculus with justifications for formulae of arbitrary degree.

4.1.1 Reasoning on minimal foundations J$_0$

Reasoning about such a minimal metatheory is axiomatized in its own turnstile ($\vdash_{J_0}$). Henceforth, judgments on the justificational type universe of J$_0$ (an adequate subset of types in T') together with wf predicate for Δ_0 contexts go as follows:

$$\frac{}{\mathsf{nil} \vdash_{\mathsf{J}_0} \mathsf{wf}}\ \textsc{Nil} \qquad \frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{wf} \qquad \Delta_0 \vdash_{\mathsf{J}_0} \phi \in \mathsf{Prop}_0}{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Just}\ \phi \in \mathsf{jtype}_0}\ \textsc{Simple}$$

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Just}\ \phi \in \mathsf{jtype}_0 \qquad s \notin \Delta_0}{\Delta_0, s :: \phi \vdash_{\mathsf{J}_0} \mathsf{wf}}\ \Delta_0\text{-}\textsc{App} \qquad \frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{wf} \qquad s :: \phi \in \Delta}{\Delta_0 \vdash_{\mathsf{J}_0} s :: \phi}\ \Delta_0\text{-}\textsc{Refl}$$

We add logical constants to satisfy the requirement that J$_0$ includes an axiomatic characterization of – at least – a fragment of IPC. Following justification logic, we define a signature of polymorphic constructors including K, S from combinatory logic, and Falsum for the explosion principle. The values of those constructors are axiomatic constants that witness existence of proofs in T' of all instances of the corresponding logical validities. This axiomatic characterization of intuitionistic logic in J$_0$ together with rule scheme TIMES (*applicativity of justifications*) satisfy the minimal requirement for T' to reason logically.

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Just}\ \ \phi_1 \to \phi_2 \to \phi_1 \in \mathsf{jtype}_0}{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{K}[\phi_1, \phi_2] :: \phi_1 \to \phi_2 \to \phi_1}\ \mathrm{K} \qquad \frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Just}\ \ \bot \to \phi \in \mathsf{jtype}_0}{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Falsum}[\phi] :: \bot \to \phi}\ \textsc{Falsum}$$

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{Just}\ \ (\phi_1 \to \phi_2 \to \phi_3) \to (\phi_1 \to \phi_2) \to (\phi_1 \to \phi_3) \in \mathsf{jtype}_0}{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{S}[\phi_1, \phi_2, \phi_3] :: (\phi_1 \to \phi_2 \to \phi_3) \to (\phi_1 \to \phi_2) \to (\phi_1 \to \phi_3)}\ \mathrm{S}$$

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} j_2 :: \phi_1 \to \phi_2 \qquad \Delta_0 \vdash_{\mathsf{J}_0} j_1 :: \phi_1}{\Delta_0 \vdash_{\mathsf{J}_0} j_2 * j_1 :: \phi_2}\ \textsc{Times}$$

The substitution principle for justifications is expressed explicitly by the following rule:[6]

$$\frac{\Delta_0, s :: \phi \vdash_{\mathsf{J}_0} j :: \phi \qquad \Delta_0 \vdash_{\mathsf{J}_0} j' :: \phi}{\Delta_0 \vdash_{\mathsf{J}_0} j[s := j'] :: \phi}\ \Delta_0\text{-}\textsc{Subst}$$

$$\frac{\Delta, s :: \phi \vdash_{\mathsf{J}_0} \mathsf{J} \in \mathsf{jtype}_0}{\Delta_0 \vdash_{\mathsf{J}_0} \Pi s :: \phi.\ \mathsf{J} \in \mathsf{type}_0}\ \Pi\text{-}\textsc{Abs} \qquad \frac{\Delta_0 \vdash_{\mathsf{J}_0} \Pi s :: \phi.\ \mathsf{J} \in \mathsf{evftype} \qquad \Delta, s :: \phi \vdash_{sfJ_0} evf : \mathsf{J}}{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{J}s.\ evf : \Pi s :: \phi.\ \mathsf{J}}\ \mathsf{J}\text{-}\textsc{Abs}$$

4.1.2 Zipping: $\mathsf{J\text{-}Calc}_1 = \mathsf{IPC} + \mathsf{J}_0 + \Box\text{−}\mathsf{Intro}$

In this section we introduce $\mathsf{J\text{-}Calc}_1$ for reasoning on the existence of bindings i.e. constructions that witness the existence of proofs both in IPC (*here*: T) and J_0 (*there*: T'). By constructing a binding we have a proof of a constructive necessity of a formula, showing that it is true and valid. Bindings have types of the form $\Box^{\mathsf{j}}\phi$ where j is a justification of the appropriate type. $\mathsf{J\text{-}Calc}_1$ realizes modal logic theoremhood in K up to degree 1.

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} \mathsf{wf}}{\Delta_0; \mathsf{nil} \vdash_{\mathsf{JC}_1} \mathsf{wf}}\ \textsc{ImpWf} \qquad \frac{\Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} \mathsf{wf} \qquad \Delta_0 \vdash_{\mathsf{J}_0} j :: \phi}{\Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} j :: \phi}\ \textsc{ImpJust}$$

$$\frac{\phi \in \mathsf{Prop}_0 \qquad \Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} j :: \phi}{\Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} \Box^j \phi \in \mathsf{Prop}_1}\ \textsc{Prop}_1\text{-}\textsc{Intro}$$

$$\frac{\Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} \phi \in \{\mathsf{Prop}_0, \mathsf{Prop}_1\} \qquad x \notin \Gamma_1}{\Delta_0; \Gamma_1, x : \phi \vdash_{\mathsf{JC}_1} \mathsf{wf}}\ \Gamma_1\text{-}\textsc{App}$$

$$\frac{\Delta_0 \vdash_{\mathsf{J}_0} evf : \mathsf{J} \qquad \Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} \mathsf{wf}}{\Delta_0; \Gamma_1 \vdash_{\mathsf{JC}_1} evf : \mathsf{J}}\ \textsc{ImpEvf}$$

From justifications of formulas in Prop_0, we can reason about their admissibility in T. Hence, Γ_1 might include assumptions from the sorts Prop_0 and Prop_1. For the inhabitation of $\mathsf{Prop}_0, \mathsf{Prop}_1$, we first we accumulate intuitionistic reasoning extended to the new type universe (Prop_1), adapt-

[6]We could have introduced a function space among justificational types in the form of *evidence functions* (objects appearing in the semantics of Justification Logic) as first class citizens of the theory, with terms corresponding to abstraction from Δ, together with an elimination rule corresponding to substitution. For economy of presentation, and in accordance with justification logic syntax, we prefer here a single substitution rule.

ing the rules from Section 3:

$$\frac{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} \mathsf{wf} \qquad x:\phi \in \Gamma_1}{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} x:\phi}\ \Gamma\text{-}\textsc{Refl}$$

$$\frac{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} \phi \in \{\mathsf{Prop}_0,\mathsf{Prop}_1\} \qquad \Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} M:\bot}{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} abort^{\phi} M:\phi}\ \textsc{False}$$

$$\frac{\Delta_0;\Gamma_1,x:\phi_1 \vdash_{\mathsf{JC}_1} M:\phi_2}{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} \lambda x:\phi_1.\,M:\phi_1\rightarrow\phi_2}\ {\rightarrow}\mathrm{I} \qquad \frac{\Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} M:\phi_1\rightarrow\phi_2 \qquad \Delta_0;\Gamma_1 \vdash_{\mathsf{JC}_1} M':\phi_1}{\Delta_0;\Gamma \vdash_{\mathsf{JC}_1} (MM'):\phi_2}\ {\rightarrow}\mathrm{E}$$

For relating the two calculi, a lifting rule is formulated for turning strictly Prop_0 judgments to judgments on proof bindings (Prop_1). Note that since the K modality does not require factivity, we do not keep the construction M in the conclusion of the rule. That is, when the existence of a proof match is obtained the T construct becomes irrelevant. This is why we speak of binding witnesses instead of bindings themselves.

In the rule, the $\downharpoonleft$-operator, to be fully defined in Section 5.2, ensures that context list $\downharpoonleft\Gamma$ includes assumptions strictly in Prop_0. Finally, since justifications now appear in constructions (via "boxing") we have to extend the substitution rule accordingly as shown for the full calculus in Section 5.1. [7]

$$\frac{\downharpoonleft\Gamma_1 \vdash_{\mathsf{IPC}} M:\phi \qquad \Delta_0 \vdash_{\mathsf{J}_0} j::\phi \qquad \Delta_0;\Gamma_1,\Gamma_1' \vdash_{\mathsf{JC}_1} \mathsf{wf}}{\Delta_0;\Gamma_1,\Gamma_1' \vdash_{\mathsf{JC}_1} Jbox\ j:\Box^{j}\phi}\ \Box\text{-}\textsc{Intro}$$

5 The Full Calculus: J-Calc

$\mathsf{J\text{-}Calc}_1$ motivates the generalization to modal reasoning of arbitrary nesting: J-Calc. To allow such generalization, we need justifications of types of the form Just $\Box^{j}\phi$. Let us revise: If ϕ is a proposition (or, a sentence in the language of T), then *Just* ϕ corresponds to the intended interpetation of ϕ in some metatheory T'. In $\mathsf{J\text{-}Calc}_1$ we could reason logically about the constructive admissibility of (valid according to T') facts of T. The existence of a binding (or proofcheck or, mapping) of a proof in T with an existing proof of the same type in T' would lead to constructions of a type of the form $\Box^{j}\phi$ with ϕ a simple type. To get modal theoremhood of degree 2 or more we have to assume that T' can express the existence of such bindings in itself. That is to say that T' can express the provability predicates both of T *and* of itself. Hence, supposing that $j::\phi$, we can read a justification term of type Just $\Box^{j}\phi$ as a witness of a proof in T' of the fact $\exists x.Proof_T(x,\underline{\phi}) \wedge \exists x.Proof_{T'}(x,\ \underline{Just\ \phi})$ expressed in T'. We will specify which of those types T' is expected to *capture* by introducing additional appropriate constants. Having this kind of

[7] The (currently in progress) operational semantics of the calculus reads *Jbox j* as the return value of an interpretation procedure that "consumes" constructs in T and maps them, forgetfully, to constructs in T'. The return value witnesses a success in this mapping process. The connection with IO and Lift monads has to be explored and utilized for a more explicit version of this rule.

justifications we can obtain $\mathsf{Prop_i}$ for any finite i as slices of a type universe in a mutual inductive construction. Schematically: $\mathsf{Prop_0} \Rightarrow \mathsf{Just\ Prop_0} \Rightarrow \mathsf{Prop_1} \Rightarrow \mathsf{Just\ Prop_1}$ and so on. This way we obtain full modal logic with justifications and Curry-Howard Isomorphism for intuitionistic justification logic. As different kinds of judgments are kept separated by the different typing relations, we do not need to provide distinct calculi as we did for $\mathsf{J\text{-}Calc_1}$ but we provide one "zipped" calculus directly. [8]

5.1 Justificational (Validity) Judgments

The justificational type system has to include: judgments on the wellformedness of contexts (wf);[9] judgments on what T' can reason about (jtype) under the requirement that it is a metatheory of T; judgments on the construction of the justificational type universe (jtype) and minimal requirements about its inhabitation. (i.e, *a minimal signature of logical constants*). Here is the grammar for the metavariables appearing below:

$$\begin{aligned}
\phi :=& P_i|\bot|\Box^j\phi|\phi_1 \to \phi_2\\
j :=& s_i|C|j_1 * j_2\\
C :=& \mathsf{K}[\phi_1,\phi_2]|\mathsf{S}[\phi_1,\phi_2,\phi_3]|\mathsf{Falsum}[\phi]|\mathsf{Kappa}[j_1,j_2,\phi_1,\phi_2]|!C\\
\mathsf{J} :=& \Pi s :: \phi_1.\mathsf{Just}\ \phi_2|\Pi s :: \phi_1.\ \mathsf{J}\\
evf :=& \mathsf{J}s :: \phi.\ j|\mathsf{J}s :: \phi.\ evf\\
s :=& s_i
\end{aligned}$$

We introduce progressively: formation rules for Prop (where we treat negation definitionally: $\neg\phi =^{def} \phi \to \bot$); the formation rule for jtype; rules to build well-formed contexts of propositions and justifications (where we will be abbreviating using the following equational rule: $\mathsf{nil}, s_1 :: \phi_1, s_2 :: \phi_2, \ldots =^{def} s_1 :: \phi_1, s_2 :: \phi_2, \ldots$).

$$\frac{}{\mathsf{nil};\mathsf{nil} \vdash_{\mathsf{JC}} \mathsf{wf}}\ \text{NIL} \qquad \frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \mathsf{wf}}{\Delta;\Gamma \vdash_{\mathsf{JC}} P_i \in \mathsf{Prop}}\ \textsc{Atom} \qquad \frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \mathsf{wf}}{\Delta;\Gamma \vdash_{\mathsf{JC}} \bot \in \mathsf{Prop}}\ \textsc{Bot}$$

$$\frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \phi_1 \in \mathsf{Prop} \qquad \Delta;\Gamma \vdash_{\mathsf{JC}} \phi_2 \in \mathsf{Prop}}{\Delta;\Gamma \vdash_{\mathsf{JC}} \phi_1 \to \phi_2 \in \mathsf{Prop}}\ \textsc{Impl} \qquad \frac{\Delta;\Gamma \vdash_{\mathsf{JC}} j :: \phi}{\Delta;\Gamma \vdash_{\mathsf{JC}} \Box^j\phi \in \mathsf{Prop}}\ \textsc{Box}$$

$$\frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \phi \in \mathsf{Prop}}{\Delta;\Gamma \vdash_{\mathsf{JC}} \mathsf{Just}\ \phi \in \mathsf{jtype}}\ \textsc{Jtype} \qquad \frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \mathsf{Just}\ \phi \in \mathsf{jtype} \qquad s \notin \Delta}{\Delta, s :: \phi;\Gamma \vdash_{\mathsf{JC}} \mathsf{wf}}\ \Delta\text{-}\textsc{App}$$

$$\frac{\Delta;\Gamma \vdash_{\mathsf{JC}} \phi \in \mathsf{Prop} \qquad x \notin \Gamma}{\Delta_0;\Gamma, x : \phi \vdash \mathsf{wf}}\ \Gamma\text{-}\textsc{App}$$

[8]In fact, adjoining Γ contexts when reasoning within justifications is pure weakening so we could have kept those judgments separated in a single–context $\vdash$ relation. We gain something though: we can squeeze two premises ($\Delta \vdash j :: \phi$, $\Delta;\Gamma \vdash \mathsf{wf}$) to a single one ($\Delta;\Gamma \vdash j :: \phi$).

[9] Analogous treatments of judgments on the validity of contexts can be found e.g. in [?].

While inhabitation of Prop and jtype have not yet been presented, substitution on justificational terms is treated as a context operation and introduced first. In the following, the Γ' context in the second premise is rudimentary and could be assumed to be empty. In addition, having second degree assumptions implies dependencies of propositions on the succeeding slice of the context Δ_2 by s. As a result, we apply substitution uniformly from the "point" $s :: \phi$ on.

$$\frac{\Delta_1, s :: \phi_1, \Delta_2; \Gamma \vdash_{\mathsf{JC}} j :: \phi_2 \qquad \Delta_1; \Gamma' \vdash_{\mathsf{JC}} j' :: \phi_1}{\Delta_1, \Delta_2[s := j']; \Gamma[s := j'] \vdash_{\mathsf{JC}} j[s := j'] :: \phi_2[s := j']}\ \Delta\text{-}\textsc{Subst}_1$$

Analogously we have the same substitution principle for propositional judgments. Substitution should be applied in the constructions too, since in the general case they include justificational terms (cf. *JBox*):

$$\frac{\Delta_1, s :: \phi_1, \Delta_2; \Gamma \vdash_{\mathsf{JC}} M : \phi_2 \qquad \Delta_1; \Gamma' \vdash_{\mathsf{JC}} j' :: \phi_1}{\Delta_1, \Delta_2[s := j']; \Gamma[s := j'] \vdash_{\mathsf{JC}} M[s := j'] :: \phi_2[s := j]}\ \Delta\text{-}\textsc{Subst}_2$$

In this sort we classify the types of evidence functions. Dependent typing is at work here since the return type of an evidence function might depend on its arguments.

$$\frac{\Delta, s :: \phi_1; \Gamma \vdash_{\mathsf{JC}} \mathsf{jtype}}{\Delta \vdash_{\mathsf{J}} \Pi s :: \phi_1.\ \mathsf{Just}\ \phi_2 \in \mathsf{evftype}}\ \Delta\text{-}\textsc{Red} \qquad \frac{\Delta, s :: \phi \vdash_{\mathsf{J}} \mathsf{J} \in \mathsf{evftype}}{\Delta \vdash_{\mathsf{J}} \Pi s :: \phi.\ \mathsf{J} \in \mathsf{evftype}}\ \Pi\text{-}\textsc{Abs2}$$

In this sort we classify the types of evidence functions.

5.1.1 Prop Inhabitation

Here is the first part of logical propositional reasoning of the system.

$$\frac{\Delta; \Gamma \vdash_{\mathsf{JC}} \mathsf{wf} \qquad x : \phi \in \Gamma}{\Delta; \Gamma \vdash_{\mathsf{JC}} x : \phi}\ \Gamma\text{-}\textsc{Refl} \qquad \frac{\Delta; \Gamma \vdash_{\mathsf{JC}} \phi \in \mathsf{Prop} \qquad \Delta; \Gamma \vdash_{\mathsf{JC}} M : \bot}{\Delta; \Gamma \vdash_{\mathsf{JC}} \mathit{abort}^{\phi} M : \phi}\ \textsc{False}$$

$$\frac{\Delta; \Gamma, x : \phi_1 \vdash_{\mathsf{JC}} M : \phi_2}{\Delta; \Gamma \vdash_{\mathsf{JC}} \lambda x : \phi_1.\ M : \phi_1 \to \phi_2}\ \to\text{I} \qquad \frac{\Delta; \Gamma \vdash_{\mathsf{JC}} M : \phi_1 \to \phi_2 \qquad \Delta; \Gamma \vdash_{\mathsf{JC}} M' : \phi_1}{\Delta; \Gamma \vdash_{\mathsf{JC}} (MM') : \phi_2}\ \to\text{E}$$

5.1.2 jtype Inhabitation

Now we move to the core of the system. In the judgments below we provide the constructions of canonical elements of justificational types (jtype). The judgments reflect the minimal requirements for T' to be a metatheory of some T as presented in Section 4.1.1 together with specifications on internalizing proof binding reasoning in itself. More specifically, we demand that T' can *capture* reasoning on bindings (between proof objects of T and itself) *within* itself and also, internalize modus ponens of T. To capture these provability conditions we add the constant constructors ! (*bang*) and Kappa. Although introduction of bindings is axiomatized in the next section, these judgments are judgments on the existence on proofterms of T' and, thus, are presented together.

$$\frac{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Just}\ \ \phi_1\rightarrow\phi_2\rightarrow\phi_1\in\mathsf{jtype}}{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{K}[\phi_1,\phi_2]::\phi_1\rightarrow\phi_2\rightarrow\phi_1}\ \textrm{K}\qquad\qquad \frac{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Just}\ \ \bot\rightarrow\phi\in\mathsf{jtype}}{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Falsum}[\phi]::\bot\rightarrow\phi}\ \textrm{FALSUM}$$

$$\frac{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Just}\ \ (\phi_1\rightarrow\phi_2\rightarrow\phi_3)\rightarrow(\phi_1\rightarrow\phi_2)\rightarrow(\phi_1\rightarrow\phi_3)\in\mathsf{jtype}}{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{S}[\phi_1,\phi_2,\phi_3]::(\phi_1\rightarrow\phi_2\rightarrow\phi_3)\rightarrow(\phi_1\rightarrow\phi_2)\rightarrow(\phi_1\rightarrow\phi_3)}\ \textrm{S}$$

$$\frac{\Delta;\Gamma\vdash_{\mathsf{JC}} j_2::\phi_1\rightarrow\phi_2\qquad \Delta;\Gamma\vdash_{\mathsf{JC}} j_1::\phi_1}{\Delta\vdash_{\mathsf{J}} j_2*j_1::\phi_2}\ \textrm{TIMES}\qquad\qquad \frac{\Delta;\Gamma\vdash_{\mathsf{JC}} M:\Box^{C}\phi}{\Delta;\Gamma\vdash_{\mathsf{JC}} C!::\Box^{\mathsf{C}}\phi}\ \textrm{BANG}$$

$$\frac{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Just}\ \Box^{j'}\phi_1\in\mathsf{jtype}\qquad \Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Just}\ \ \Box^{j}(\phi_1\rightarrow\phi_2)\in\mathsf{jtype}}{\Delta;\Gamma\vdash_{\mathsf{JC}} \mathsf{Kappa}[j,j',\phi_1,\phi_2]::\Box^{j}(\phi_1\rightarrow\phi_2)\rightarrow\Box^{j'}\phi_1\rightarrow\Box^{j*j'}\phi_2}\ \textrm{KAPPA}$$

5.1.3 Inhabiting evftype

Abstraction from the context stack is only possible from the top element. That is to avoid leaving dangling references in the next elements. The return type of an “arrow” created this way might depend on the value of its argument and that’s why Π typing is at work here. Those abstractions form evidence functions. The dual rule is applying an evidence function to a justification.

$$\frac{\Delta\vdash_{\mathsf{J}} \Pi s::\phi_1.\ \mathsf{Just}\ \phi_2\in\mathsf{evftype}\qquad \Delta,s::\phi\vdash_{\mathsf{J}} j::\phi_2}{\Delta\vdash_{\mathsf{J}} \mathsf{J}s.\ j:\Pi s::\phi.\ \mathsf{Just}\ \phi_2}\ \textrm{J-ABS}_1$$

$$\frac{\Delta\vdash_{\mathsf{J}} \Pi s::\phi.\ \mathsf{J}\in\mathsf{evftype}\qquad \Delta,s::\phi\vdash_{\mathsf{J}} evf:\mathsf{J}}{\Delta\vdash_{\mathsf{J}} \mathsf{J}s.\ evf:\Pi s::\phi.\ \mathsf{J}}\ \textrm{J-ABS}_2$$

For the dual rule :

$$\frac{\Delta\vdash_{\mathsf{J}} evf:\Pi s::\phi.\ \mathsf{J}\qquad \Delta\vdash_{\mathsf{J}} j::\phi}{\Delta\vdash_{\mathsf{J}} \mathsf{let}\ u=(ev\ j)\ \mathsf{in}\ u:\mathsf{J[s:=j]}}\ \textrm{EVFUN-APP}$$

5.2 Proof Bindings

Our next task is to formulate the main rule for the *K* modality as a lifting rule for going from reasoning about constructions to reasoning about admissibility of validities via proof matching. To reflect the modal axiom *K* in Natural Deduction we have to obtain a rule that reflects the following provability principle:

$$\frac{\phi_1\ \mathsf{true},\ldots,\phi_n\ \mathsf{true}\vdash\phi\ \mathsf{true}\qquad \phi_1\ \mathsf{valid},\ldots,\phi_n\ \mathsf{valid}\vdash\phi\ \mathsf{valid}}{\Box\phi_1\ \mathsf{true},\ldots,\Box\phi_n\ \mathsf{true},\ldots\vdash\Box\phi\ \mathsf{true}}\ \Box\textrm{-INTRO}$$

We proceed with giving inhabitants analogously to what was explained in Section 4.1.2:

$$\frac{\Delta; \downharpoonright\Gamma \vdash_{\mathsf{JC}} M : \phi \qquad \Delta'; \Gamma, \Gamma_w \vdash_{\mathsf{JC}} j :: \phi}{\Delta'; \Gamma, \Gamma_w \vdash \mathit{JBox}\ j : \square^j \phi}\ \square\text{-}\textsc{Intro}$$

[10] The operator $\downharpoonright$ can be viewed as the opposite of *lift* operation applied on context lists erasing one level of boxed assumptions at the top level as described below. The addition of Γ_w is to obtain weakening.

$$\begin{array}{rl} \downharpoonright\Gamma\ := & \textbf{match}\ \Gamma\ \textbf{with} \\ & \mathsf{nil} \Rightarrow \mathsf{nil} \\ & \mid \Gamma', x_i : \square^j \phi_i \Rightarrow \downharpoonright\Gamma',\ x_i : \phi_i \\ & \mid \Gamma', \_ \Rightarrow \downharpoonright\Gamma' \end{array}$$

5.2.1 An Example

Let us give the main steps of realizing the K modal theorem $\square\square(P_1 \to P_2) \to \square\square P_1 \to \square\square P_2$ in JCalc.

[Z]=

$$\frac{\dfrac{\dfrac{\mathit{skip}}{\mathsf{nil}; x : P_1, y : P_1 \to P_2 \vdash_{\mathsf{JC}} \mathsf{wf}}\ \Gamma\text{-}\textsc{App}}{\vdots}}{\mathsf{nil}; x : P_1, y : P_1 \to P_2 \vdash_{\mathsf{JC}} (yx) : P_2}\ \textsc{App}$$

Letting:

$\Delta = \{s :: P_1, t :: P_1 \to P_2\}$ and $\Gamma = \{x : P_1 \to P_2, y : P_1\}$, $\Gamma' = \{x : \square^{\mathsf{t}}(P_1 \to P_2), y : \square^{\mathsf{s}} P_1,\}$

[V]=

$$\frac{[Z] \qquad \dfrac{\dfrac{\dfrac{\mathit{skip}}{\Delta; \Gamma' \vdash_{\mathsf{JC}} \mathsf{wf}}}{\vdots}}{\Delta; \Gamma' \vdash_{\mathsf{JC}} t * s :: P_2}\ \textsc{Times}}{\Delta; \Gamma' \vdash_{\mathsf{JC}} \mathit{JBox}\ \ t * s : \square^{\mathsf{t*s}} P_2}\ \square\text{-}\textsc{Intro}$$

Letting:

[10]We prefer this to the mouthful but equivalent:

$$\frac{\begin{array}{c}\Delta; x_1 : \phi_1, \ldots, x_i : \phi_i\ \ as\ \Gamma \vdash M : \phi \\ \forall \phi_i \in \Gamma.\ \Delta'; \mathsf{nil} \vdash j_i :: \phi_i \qquad \Delta'; \mathsf{nil} \vdash j :: \phi \qquad \Delta'; x_1 : \square^{j_1} \phi_i, \ldots, \square^{j_i} \phi_i, \Gamma_w \vdash \mathsf{wf}\end{array}}{\Delta'; x_1 : \square^{j_1} \phi_i, \ldots, \square^{j_i} \phi_i, \Gamma_w \vdash \mathit{JBox}\ j : \square^{\mathsf{j}} \phi}\ \square\text{-}\textsc{Intro}$$

$\Delta' = \{s :: P_1, t :: P_1 \to P_2, u :: \Box^{\mathsf{t}}(P_1 \to P_2)\}, v :: \Box^{\mathsf{s}}P_1, \Gamma'' = \{x : \Box^{\mathsf{v}}\Box^{\mathsf{t}}(P_1 \to P_2), y : \Box^{\mathsf{u}}\Box^{\mathsf{s}}P_1, \}$, $\mathsf{Kp}_1 = \mathsf{Kappa}[t, s, P_1, P_2]$ and with $\Gamma_w = \mathsf{nil}$ in $\Box$-Intro.

$$
\cfrac{
\begin{array}{c}
\cfrac{
[V] \qquad
\cfrac{
\begin{array}{c}
\cfrac{
\begin{array}{c}
\cfrac{\mathit{skip}}{\Delta';\Gamma'' \vdash_{\mathsf{JC}} \mathsf{wf}} \\
\vdots
\end{array}
}{\Delta';\Gamma'' \vdash_{\mathsf{JC}} \mathsf{Kp}_1 :: \Box^{\mathsf{t}}(P_1 \to P_2) \to \Box^{\mathsf{s}}P_1 \to \Box^{\mathsf{t*s}}P_2}\ \textsc{Kappa} \\
\vdots
\end{array}
}{\Delta';\Gamma'' \vdash_{\mathsf{JC}} \mathsf{Kp}_1 * u * v :: \Box^{\mathsf{t*s}}P_2}\ \textsc{Times}
}{\Delta';\Gamma'' \vdash_{\mathsf{JC}} \mathit{JBox}\ \ \mathsf{Kp}_1 * u * v : \Box^{\mathsf{Kp_1*u*v}}\Box^{\mathsf{t*s}}P_2}\ \Box\text{-}\textsc{Intro} \\
\vdots
\end{array}
}{\Delta';\mathsf{nil} \vdash_{\mathsf{JC}} \lambda x.\lambda y.\ \mathit{JBox}\ \ \mathsf{Kp}_1 * u * v : \Box^{\mathsf{u}}\Box^{\mathsf{t}}(P_1 \to P_2) \to \Box^{\mathsf{v}}\Box^{\mathsf{s}}P_1 \to \Box^{\mathsf{Kp_1*u*v}}\Box^{\mathsf{t*s}}P_2}\ {\to}\mathrm{I}
$$

6 Further Results and Conclusions

Standard meta-theoretical results can be proven for J-Calc. We just mention here that the *Jbox* operator satisfies standard commutativity with the substitution rule for justifications and that structural rules can be proven. We will be skipping the index in $\vdash_{\mathsf{JC}}$.

Theorem 1 (Weakening)**.** J-Calc *satisfies Weakening in both modes of reasoning:*

1. *If* $\Delta;\mathsf{nil} \vdash j :: \phi$*, and* $\Delta;\Gamma \vdash \mathsf{wf}$ *then,* $\Delta;\Gamma \vdash j :: \phi$*.*
2. *If* $\Delta;\Gamma \vdash j :: \phi$*, then* $\Delta, s :: \phi';\Gamma \vdash j :: \phi$*, with s fresh.*
3. *If* $\Delta;\Gamma \vdash M : \phi$*, then* $\Delta;\Gamma, x : \phi' \vdash M : \phi$*, with x fresh.*

Proof. For all items by structural induction on the derivation trees of the two kinds of constructions. The proof of the first is vacuous since Γ contexts are irrelevant in justification formation. As a result, its inverse can also be shown. $\Box$

Theorem 2 (Contraction)**.** J-Calc *satisfies Contraction:*

1. *If* $\Delta, s :: \phi, t :: \phi;\mathsf{nil} \vdash j :: \phi'$*, then* $\Delta, u :: \phi;\mathsf{nil} \vdash j[s \equiv t/u] :: \phi'$*.*
2. *If* $\Delta, s :: \phi, t :: \phi;\Gamma \vdash \mathsf{wf}$*, then,* $\Delta, u :: \phi;\Gamma[s \equiv t/u] \vdash \mathsf{wf}$*.*
3. *If* $\Delta, s :: \phi, t :: \phi;\Gamma \vdash M : \phi'$*, then,* $\Delta, u :: \phi;\Gamma[s \equiv t/u] \vdash M[s \equiv t/u] : \phi'[s \equiv t/u]$*.*
4. *If* $\Delta;\Gamma, x : \phi, y : \phi \vdash M : \phi'$*, then* $\Delta;\Gamma, z : \phi \vdash M[x \equiv y/z] : \phi'$*.*

Proof. First item by structural induction on the derivation trees of justifications (validity judgments). Note, as mentioned in the previous theorem, that it can be shown for arbitary Γ. For the second, nested induction on the structure of context Γ (treated as list) and the complexity of formulas. Vacuously in the nil case. For the non-empty case: case analysis on the complexity of the head formula using the inductive hypothesis on the tail. Cases of interest are with $\Box^{\mathsf{s}}\phi$ or $\Box^{\mathsf{t}}\phi$ as subformulae. Use the previous item and judgments for wf contexts. For the third, structural induction on the derivation. In all cases except for $\Box$-Intro, $M[s \equiv t/u]$ is vacuous. For the fourth, again by structural induction on the derivation. $\Box$

We additionally mention that the calculus satisfies *exchange* up to wellformedness preservation. Moreover, a forgetful projection of the calculus (i.e omitting justificational judgments, Δ contexts, all j annotations in $\Box$ types, as well as, proof terms) gives exactly a natural deduction for modal logic K. This is a weak argument towards soundness. Further results shall include a translation to a procedural semantics with appropriate type safety preservation and termination. Preferably, as mentioned before, we would like to strengthen the connection of our lifting rule with programming procedures of monadic kind such as linkers, that replace parts of the code of a program by programming constructs defined elsewhere (e.g. linking process in a language with modules).

References