Expressing Ecumenical Systems in the λΠ-Calculus Modulo Theory

Emilie Grienenberger
Université Paris-Saclay, ENS Paris-Saclay, Inria, CNRS, Laboratoire Méthodes Formelles, France

Abstract

Systems in which classical and intuitionistic logics coexist are called ecumenical. Such a system allows for interoperability and hybridization between classical and constructive propositions and proofs. We study Ecumenical STT, a theory expressed in the logical framework of the λΠ-calculus modulo theory. We prove soudness and conservativity of four subtheories of Ecumenical STT with respect to constructive and classical predicate logic and simple type theory. We also prove the weak normalization of well-typed terms and thus the consistency of Ecumenical STT.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Proof theory; Theory of computation → Higher order logic; Theory of computation → Constructive mathematics

Keywords and phrases dependent types, predicate logic, higher order logic, constructivism, interoperability, ecumenical logics

Digital Object Identifier 10.4230/LIPIcs.TYPES.2022.4

Acknowledgements The author thanks Gilles Dowek for his scientific advice and guidance, and the reviewers for their constructive and detailed returns.

1 Introduction

The λΠ-calculus modulo theory ($\lambda\Pi/\equiv$) [3] is a logical framework in which diverse systems – predicate logic, pure type systems [10], cumulative type systems [44], the ς-calculus [39], Matching Logic and more – can be expressed as theories. Using a common language to describe the logical foundations of various proof assistants allows more interoperability between the currently impermeable libraries of formal proofs. Indeed, it is a valuable tool in the design of translations [8, 43, 19, 24], the constitution of a common database of proofs [12], or even the hybridization of their proofs [9].

In the zoology of proof assistants, there are many examples of classical systems (the HOL family, PVS, etc) and constructive systems (Coq, Agda, Matita, etc). They rely on different sets of axioms: for instance, the axiom of the excluded-middle $\neg P \vee P$ is used in classical logic but not in intuitionistic logic [11]. These axioms define the *meaning* of logical symbols, thus constructive and classical disjunctions have different meanings. This observation does not only hold for disjunction and negation, but also for connectives that do not appear in the excluded-middle axiom. For example Peirce's formula $((P \Rightarrow Q) \Rightarrow Q) \Rightarrow P$, the equivalence $(\neg P \vee Q) \Leftrightarrow (P \Rightarrow Q)$ and the de Morgan laws hold classically but not intuitionistically. Using a unique symbol for two connectives with different significations is unsatisfactory, thus we can attempt to design logical systems where intuitionistic and classical symbols are written differently. Such logical systems are called *ecumenical* [38, 36].

An ecumenical expression of logic and of mathematics has many advantages. First, it allows to use the expressivity of classical logic while explicitely keeping constructive properties. For example, a program can be extracted from a proof of $\forall x.\ \exists y.\ S(x, y)$ – where $\forall$ and $\exists$ are intuitionistic – even if the specification $S(x, y)$ is classical, reflecting the fact that an algorithm can both be effective and have a classical correctness proof. Second, intuitionistic and classical proofs coexisting in the same logical system can be stored in a common database of formal proofs, while using two separate logical systems entails two separate databases – or the loss of readily available constructive information.

In [6] is introduced *theory* $\mathcal{U}$, which is a $\lambda\Pi/\equiv$ theory in which all proofs of minimal, constructive, classical, and ecumenical predicate logic, minimal, constructive, classical, and ecumenical simple type theory with or without prenex polymorphism or predicate subtyping, and the calculus of constructions be can expressed. More precisely, [6] includes a presentation of the axioms of theory $\mathcal{U}$ and a proof of well-typedness and modularity of the constructed theory. Many axioms and fragments of theory $\mathcal{U}$ have been studied in isolation [26, 43, 13, 3], however some of these studies lack proofs of normalization, consistency, soundness or conservativity with respect to appropriate reference systems. Some axioms of theory $\mathcal{U}$ have not been studied together; notably, there is currently no proof of normalization or consistency for the whole theory. In this paper, we study the ecumenical subtheory of theory $\mathcal{U}$, called Ecumenical STT, by reviewing existing results and establishing its soundness, conservativity, normalization, and consistency.

Related work

Examples of first order ecumenical systems are found in sequent calculi [34, 22, 14, 33, 37] and natural deduction [38, 36]. Some rely on double negation translations to define their classical connectives and rules, as in [14] and to a lesser extent [38]. Another point of difference is the way predicates and atoms are handled. In [38], there is a classical and an intuitionistic copy of each predicate symbol. The system described in [14] avoids this split by relying on total provability – the equivalence of provability in its classical fragment and in LK is valid for sequents with empty contexts only. Ecumenical STT does not rely on copies of predicates nor on total provability, as these features seem unsuitable for the specific purpose of interoperability between proof systems. To our knowledge, none of the aforementioned ecumenical systems are extended to the higher order.

The termination of rewriting systems being an important problem in logic and software verification, there is a multitude of dedicated theoretical results and automatized tools. Many are specific to first order rewriting [21, 32], or simply-typed theories [31, 5]. A few are designed for higher order and/or dependently typed systems [30, 28, 7], which is the framework of this paper. The existing tools using dependency pairs are currently unable to conclude to the normalization of Ecumenical STT, thus we base our normalization proof on models of $\lambda\Pi/\equiv$ theories in variants of pre-Heyting algebras developed in [15].

Normalization is also a valuable, if not crucial tool to establish the conservativity of $\lambda\Pi/\equiv$ theories as shown in [10, 2]. The conservativity proofs lead in this paper use the framework these aforementioned studies provide.

Outline

In Section 2 is presented the logical framework $\lambda\Pi$-calculus modulo theory ($\lambda\Pi/\equiv$) and the $\lambda\Pi/\equiv$ theory of Ecumenical STT [6]. Some meta-theoretical properties of Ecumenical STT are discussed in Section 3; notably we establish its weak normalization and the decidability of type-checking in Section 3.2. Finally, the soundness and conservativity with respect to appropriate reference systems – constructive and classical predicate logic and simple-type theory – of four subtheories of Ecumenical STT are proven in Section 4 and Section 5. We conclude as to the consistency of Ecumenical STT in Section 5.3.

2 Expressing ecumenism in $\lambda\Pi$-calculus modulo theory

2.1 The $\lambda\Pi$-calculus modulo theory

The logical framework $\lambda\Pi$-calculus modulo theory ($\lambda\Pi / \equiv$) [3], based on the Edinburgh Logical Framework (LF) [25], is an extension of simply-typed lambda calculus with dependent types and a primitive notion of computation via the definition of rewrite rules [42]. Formally, $\lambda\Pi / \equiv$ *terms* are defined inductively by

$$t, u, \ldots = \texttt{TYPE} \mid \texttt{KIND} \mid x \mid c \mid \lambda x : t.\, u \mid t\ u \mid (x : t) \rightarrow u$$

where x belongs to an infinite set of variables $\mathcal{V}$ and c belongs to a finite or infinite set of constants $\mathcal{C}$. The terms TYPE and KIND are respectively the type of $\lambda\Pi / \equiv$ types, and the type of *kinds*, that is of $\lambda\Pi / \equiv$ type families; both terms are called *sorts* and often denoted by s. Dependent products, *i.e.* terms of the form $(x : t) \rightarrow u$, allow the definition of indexed type families. For example, the family of vectors indexed by their length $n \in \mathbb{N}$ can be defined by declaring $\texttt{Vector} : (n : \mathbb{N}) \rightarrow \texttt{TYPE}$; in this context a vector of length 3 can be declared by $\texttt{v} : \texttt{Vector}\ 3$. In the pathological case where the variable x does not appear free in term u, we write $(x : t) \rightarrow u$ as a simple product $t \rightarrow u$. We denote by $fv(t)$ the set of free variables of a term t, and $(u/x)t$ the substitution of variable x by a term u in a term t.

A $\lambda\Pi / \equiv$ *theory* is defined conjointly by a finite set of declarations Σ and a finite set of rewrite rules $\mathcal{R}$. A *declaration* is the assignment of a type T to a constant $c \in \mathcal{C}$, denoted by $c : T$. We denote respectively by $const(\Sigma)$ and $\Lambda(\Sigma)$ the set of constants assigned in signature Σ and the set of terms written with the set $const(\Sigma)$ of constants. A *rewrite rule* is a pair of terms $\ell \hookrightarrow r$ such that $\ell = c\ t_1\ \ldots\ t_n$ where c is a constant and $t_1, \ldots, t_n$ are terms. For example, the rewrite rule $\texttt{Vector}\ (\texttt{n} + 1) \hookrightarrow \texttt{NonEmptyVector}$ assimilates all vector types of strictly positive length to a constant type NonEmptyVector.

We denote by $\hookrightarrow_\beta$ the β-reduction. Given a set of rewrite rules $\mathcal{R}$, the relation $\hookrightarrow_\mathcal{R}$ denotes the smallest relation closed by term constructors (λ-abstraction, application and dependent product) and substitution containing $\mathcal{R}$. Finally, we write $\hookrightarrow_{\beta\mathcal{R}}$ for the union $\hookrightarrow_\beta \cup \hookrightarrow_\mathcal{R}$ and $\equiv_{\beta\mathcal{R}}$ for the smallest equivalence relation containing $\hookrightarrow_{\beta\mathcal{R}}$.

Proofs in $\lambda\Pi / \equiv$ are similar to LF proofs. However, the *conversion* rule allows to assimilate types modulo $\equiv_{\beta\mathcal{R}}$ and not only modulo $\equiv_\beta$. Formally, *typing contexts* are finite sets of variable assignments of the form $x : T$ and are denoted in the following by Γ or Δ; the empty context is written $[]$. Typing *judgments* are written "$\vdash_{\Sigma,\mathcal{R}} \Gamma$ wf" and "$\Gamma \vdash_{\Sigma,\mathcal{R}} t : T$", where Γ is a typing context, t and T are terms of $\Lambda(\Sigma)$, and "wf" stands for *well-formed*. The typing rules of $\lambda\Pi / \equiv$ in a theory Σ, $\mathcal{R}$ are represented in Figure 1.

2.2 Ecumenical STT and its subtheories

The following section describes the expression of first and higher order ecumenism in theory $\mathcal{U}$ [6]. All associated declarations and rewrite rules are represented in Figure 2.

On Figure 2a is represented the base of the encoding. The type Set is the type of the object-types of the encoded theories (for example the sorts of predicate logic and simple types of STT). The symbol El embeds object-types into $\lambda\Pi / \equiv$ types: an object-type T and any one of its elements t can be manipulated as $\lambda\Pi / \equiv$ objects while still being linked by the typing relation $t : \text{El}\ T$. The type of propositions Prop and the embedding Prf of propositions into the type of their proofs are similarly defined.

On Figure 2c are defined the constructive connectives and quantifiers of theory $\mathcal{U}$; more precisely they are declared and then defined by rewriting. As an example, the definition of the implication is based on the Curry-De-Bruijn-Howard correspondance: the proofs of

$$\frac{}{\vdash_{\Sigma,\mathscr{R}} [\,]\ \mathrm{wf}}\ (\text{empty}) \qquad \frac{\Gamma \vdash_{\Sigma,\mathscr{R}} A : s}{\vdash_{\Sigma,\mathscr{R}} \Gamma, x : A\ \mathrm{wf}}\ (\text{decl}) \qquad \frac{\vdash_{\Sigma,\mathscr{R}} \Gamma\ \mathrm{wf}}{\Gamma \vdash_{\Sigma,\mathscr{R}} \mathtt{TYPE} : \mathtt{KIND}}\ (\text{sort})$$

$$\frac{\vdash_{\Sigma,\mathscr{R}} \Gamma\ \mathrm{wf} \qquad x : A \in \Gamma}{\Gamma \vdash_{\Sigma,\mathscr{R}} x : A}\ (\text{var}) \qquad \frac{\vdash_{\Sigma,\mathscr{R}} \Gamma\ \mathrm{wf} \qquad \vdash_{\Sigma,\mathscr{R}} A : s \qquad c : A \in \Sigma}{\Gamma \vdash_{\Sigma,\mathscr{R}} c : A}\ (\text{const})$$

$$\frac{\Gamma \vdash_{\Sigma,\mathscr{R}} A : \mathtt{TYPE} \qquad \Gamma, x : A \vdash_{\Sigma,\mathscr{R}} B : s}{\Gamma \vdash_{\Sigma,\mathscr{R}} (x : A) \to B : s}\ (\text{prod})$$

$$\frac{\Gamma \vdash_{\Sigma,\mathscr{R}} A : \mathtt{TYPE} \qquad \Gamma, x : A \vdash_{\Sigma,\mathscr{R}} B : s \qquad \Gamma, x : A \vdash_{\Sigma,\mathscr{R}} t : B}{\Gamma \vdash_{\Sigma,\mathscr{R}} \lambda x : A.\, t : (x : A) \to B}\ (\text{abs})$$

$$\frac{\Gamma \vdash_{\Sigma,\mathscr{R}} t : (x : A) \to B \qquad \Gamma \vdash_{\Sigma,\mathscr{R}} u : A}{\Gamma \vdash_{\Sigma,\mathscr{R}} t\ u : (u/x)B}\ (\text{app})$$

$$\frac{\Gamma \vdash_{\Sigma,\mathscr{R}} t : A \qquad \Gamma \vdash_{\Sigma,\mathscr{R}} B : s \qquad A \equiv_{\beta\mathscr{R}} B}{\Gamma \vdash_{\Sigma,\mathscr{R}} t : B}\ (\text{conv})$$

Figure 1 Typing rules of $\lambda\Pi/\equiv$ in theory $\Sigma, \mathscr{R}$.

$A \Rightarrow B$ are functions from proofs of A to proofs of B. We use a rewrite rule to identify the corresponding types $\mathrm{Prf}(A \Rightarrow B)$ and $\mathrm{Prf}(A) \to \mathrm{Prf}(B)$. All other connectives and quantifiers are defined *à la Russel*, mimicking the elimination rules of natural deduction; once again the relevant types are identified by rewriting. Note that the quantifiers $\forall$ and $\exists$ bind a variable from a sort in Set: we define them as taking as arguments an object-type $a : \mathrm{Set}$ and a function from elements of a to propositions. We call *Constructive Predicate Logic* the $\lambda\Pi/\equiv$ theory formed by

- $\Sigma^c_{FO} = \{\, (C\text{-decl}) \,:\, C \in \{\, \mathrm{Set}, \mathrm{El}, \iota, \mathrm{Prop}, \top, \bot, \neg, \wedge, \vee, \forall, \exists \,\}\,\}$
- and $\mathscr{R}^c_{FO} = \{\, (C\text{-red}) \,:\, C \in \{\, \top, \bot, \neg, \wedge, \vee, \forall, \exists \,\}\,\}$.

In Figure 2d are defined the classical connectives and quantifiers, using their constructive counterpart and double negations, which is a frequent strategy to build ecumenical logics. We begin by introducing a classical version of Prf, enabling us to add prenex double negations whenever necessary. This definition has many advantages: we are able to add double negations to isolated atomic formulas, which is a recurrent problem in the design of ecumenical systems, without adding too much heaviness to our system. The classical connectives and quantifiers are now defined using their constructive counterpart and internal double negations.

We call *Ecumenical Predicate Logic* the $\lambda\Pi/\equiv$ theory formed by $\Sigma^e_{FO} = \Sigma^c_{FO} \cup \{\, (C\text{-decl}) \,:\, C \in \{\, \mathrm{Prf}_c, \wedge_c, \vee_c, \forall_c, \exists_c \,\}\,\}$ and $\mathscr{R}^e_{FO} = \mathscr{R}^c_{FO} \cup \{\, (C\text{-red}) \,:\, C \in \{\, \mathrm{Prf}_c, \wedge_c, \vee_c, \forall_c, \exists_c \,\}\,\}$.

Simple type theory (STT) can be expressed either as a first order theory or as an extension of predicate logic. To avoid nestling multiple encodings, we choose the latter option, as shown in Figure 2b. Predicate Logic is extended with the object type of propositions o and the function type arrow $\rightsquigarrow$. These definitions allow to construct simple types and assimilate propositions to objects.

We respectively call *Constructive STT* and *Ecumenical STT* the $\lambda\Pi/\equiv$ theories $\Sigma^c_{HO} = \Sigma^c_{FO} \cup \{\, (C\text{-decl}) \,:\, C \in \{\, o, \rightsquigarrow \,\}\,\}, \mathscr{R}^c_{HO} = \mathscr{R}^c_{FO} \cup \{\, (C\text{-red}) \,:\, C \in \{\, o, \rightsquigarrow \,\}\,\}$ and $\Sigma^e_{HO} = \Sigma^e_{FO} \cup \Sigma^c_{HO}, \mathscr{R}^e_{HO} = \mathscr{R}^e_{FO} \cup \mathscr{R}^c_{HO}$. For readability purposes, we respectively denote by $\vdash^c_{HO}$ and $\vdash^e_{HO}$ the provability relations $\vdash_{\Sigma^c_{HO}, \mathscr{R}^c_{HO}}$ of Constructive STT and $\vdash_{\Sigma^e_{HO}, \mathscr{R}^e_{HO}}$ of Ecumenical STT.

(Set-decl)	$\mathsf{Set} : \mathtt{TYPE}$
(ι-decl)	$\iota : \mathsf{Set}$
(El-decl)	$\mathsf{El} : \mathsf{Set} \to \mathtt{TYPE}$
(Prop-decl)	$\mathsf{Prop} : \mathtt{TYPE}$
(Prf-decl)	$\mathsf{Prf} : \mathsf{Prop} \to \mathtt{TYPE}$

(a) Base of the encoding.

(o-decl)	$o : \mathsf{Set}$
(o-red)	$\mathsf{El}\ o \hookrightarrow \mathsf{Prop}$
($\leadsto$-decl)	$\leadsto\ : \mathsf{Set} \to \mathsf{Set} \to \mathsf{Set}$
($\leadsto$-red)	$\mathsf{El}\ (x \leadsto y) \hookrightarrow \mathsf{El}\ x \to \mathsf{El}\ y$

(b) Higher order.

($\top$-decl)	$\top : \mathsf{Prop}$
($\top$-red)	$\mathsf{Prf}\ \top \hookrightarrow (z : \mathsf{Prop}) \to \mathsf{Prf}\ z \to \mathsf{Prf}\ z$
($\bot$-decl)	$\bot : \mathsf{Prop}$
($\bot$-red)	$\mathsf{Prf}\ \bot \hookrightarrow (z : \mathsf{Prop}) \to \mathsf{Prf}\ z$
($\Rightarrow$-decl)	$\Rightarrow\ : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\Rightarrow$-red)	$\mathsf{Prf}\ x \Rightarrow y \hookrightarrow \mathsf{Prf}\ x \to \mathsf{Prf}\ y$
($\neg$-decl)	$\neg : \mathsf{Prop} \to \mathsf{Prop}$
($\neg$-red)	$\mathsf{Prf}\ (\neg\, x) \hookrightarrow \mathsf{Prf}\ x \to (z : \mathsf{Prop}) \to \mathsf{Prf}\ z$
($\wedge$-decl)	$\wedge : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\wedge$-red)	$\mathsf{Prf}\ (x \wedge y) \hookrightarrow (z : \mathsf{Prop}) \to (\mathsf{Prf}\ x \to \mathsf{Prf}\ y \to \mathsf{Prf}\ z) \to \mathsf{Prf}\ z$
($\vee$-decl)	$\vee : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\vee$-red)	$\mathsf{Prf}\ (x \vee y) \hookrightarrow (z : \mathsf{Prop}) \to (\mathsf{Prf}\ x \to \mathsf{Prf}\ z) \to (\mathsf{Prf}\ y \to \mathsf{Prf}\ z) \to \mathsf{Prf}\ z$
($\forall$-decl)	$\forall : (a : \mathsf{Set}) \to (\mathsf{El}\ a \to \mathsf{Prop}) \to \mathsf{Prop}$
($\forall$-red)	$\mathsf{Prf}\ (\forall\ a\ p) \hookrightarrow (z : \mathsf{El}\ a) \to \mathsf{Prf}\ (p\ z)$
($\exists$-decl)	$\exists : (a : \mathsf{Set}) \to (\mathsf{El}\ a \to \mathsf{Prop}) \to \mathsf{Prop}$
($\exists$-red)	$\mathsf{Prf}\ (\exists\ a\ p) \hookrightarrow (z : \mathsf{Prop}) \to ((x : \mathsf{El}\ a) \to \mathsf{Prf}\ (p\ x) \to \mathsf{Prf}\ z) \to \mathsf{Prf}\ z$

(c) Constructive connectives and quantifiers.

(Prf_c-decl)	$\mathsf{Prf}_c : \mathsf{Prop} \to \mathtt{TYPE}$
(Prf_c-red)	$\mathsf{Prf}_c \hookrightarrow \lambda x : \mathsf{Prop}.\, \mathsf{Prf}\ (\neg\neg\, x)$
($\Rightarrow_c$-decl)	$\Rightarrow_c : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\Rightarrow_c$-red)	$\Rightarrow_c \hookrightarrow \lambda x : \mathsf{Prop}.\, [\lambda y : \mathsf{Prop}.\, (\neg\neg\, x) \Rightarrow_c (\neg\neg\, y)]$
($\wedge_c$-decl)	$\wedge_c : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\wedge_c$-red)	$\wedge_c \hookrightarrow \lambda x : \mathsf{Prop}.\, [\lambda y : \mathsf{Prop}.\, (\neg\neg\, x) \wedge (\neg\neg\, y)]$
($\vee_c$-decl)	$\vee_c : \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop}$
($\vee_c$-red)	$\vee_c \hookrightarrow \lambda x : \mathsf{Prop}.\, [\lambda y : \mathsf{Prop}.\, (\neg\neg\, x) \vee (\neg\neg\, y)]$
($\forall_c$-decl)	$\forall_c : (a : \mathsf{Set}) \to (\mathsf{El}\ a \to \mathsf{Prop}) \to \mathsf{Prop}$
($\forall_c$-red)	$\forall_c \hookrightarrow \lambda a : \mathsf{Set}.\, [\lambda p : \mathsf{El}\ a \to \mathsf{Prop}.\, \forall\ a\ (\lambda x : \mathsf{El}\ a.\, \neg\neg(p\ x))]$
($\exists_c$-decl)	$\exists_c : (a : \mathsf{Set}) \to (\mathsf{El}\ a \to \mathsf{Prop}) \to \mathsf{Prop}$
($\exists_c$-red)	$\exists_c \hookrightarrow \lambda a : \mathsf{Set}.\, [\lambda p : \mathsf{El}\ a \to \mathsf{Prop}.\, \exists\ a\ (\lambda x : \mathsf{El}\ a.\, \neg\neg(p\ x))]$

(d) Classical connectives and quantifiers.

Figure 2 Definition of Ecumenical STT in $\lambda\Pi / \equiv$.

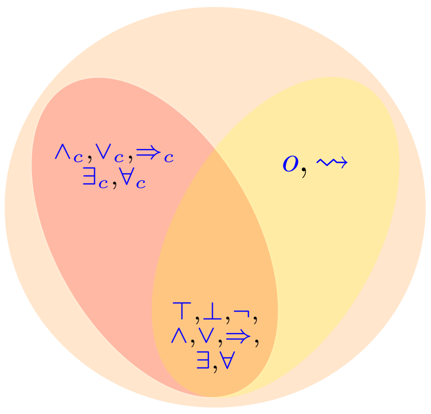

Figure 3 Logical fragments of theory $\mathcal{U}$.

We call the set of these four theories, represented in Figure 3, the "logical fragments of theory $\mathcal{U}$".

In Ecumenical Predicate Logic and Ecumenical STT, hybrid propositions and proofs can be expressed, for example in context $P : \mathrm{Prop}, Q : \mathrm{Prop}$, one can prove that $\mathrm{Prf}((P \wedge Q) \Rightarrow_c P)$ is inhabited and that $\mathrm{Prf}((P \wedge_c Q) \Rightarrow P)$ is not.

3 Properties of the logical fragments of theory $\mathcal{U}$

The system $\lambda\Pi / \equiv$ is very permissive due to the minimal restrictions on the user-defined rewriting system $\mathcal{R}$. In general, there is no guarantee of properties such as subject reduction, type uniqueness, or the decidability of type-checking [40, 2]. To ensure that the theories defined in the previous section are well-behaved, further properties need to be established such as the confluence, well-typedness, and normalization of the associated rewriting system.

3.1 Well-typedness

A $\lambda\Pi / \equiv$ theory $\Sigma, \mathcal{R}$ is said to be *well-typed* if

1. the rewriting system $\hookrightarrow_{\beta\mathcal{R}}$ is *confluent* [42, 40, Definition 1.1.5.],
2. for every declaration $c : T$ in signature Σ, term T is typed by a sort s in theory $\Sigma, \mathcal{R}$,
3. for every rule $\ell \hookrightarrow r$ in $\mathcal{R}$, typing context Γ, type T, and substitution σ such that $\Gamma \vdash_{\Sigma,\mathcal{R}} \sigma\ell : T$, then $\Gamma \vdash_{\Sigma,\mathcal{R}} \sigma r : T$.

Item 3 ensures that $\hookrightarrow_{\mathcal{R}}$ enjoys *subject reduction* [40, Definition 2.4.4], *i.e.* $\Gamma \vdash_{\Sigma,\mathcal{R}} t : T$ and $t \hookrightarrow_{\mathcal{R}} u$ implies $\Gamma \vdash_{\Sigma,\mathcal{R}} u : T$ for any Γ, t, u, T. Item 1 ensures that the product is *injective* in theory $\Sigma, \mathcal{R}$ [4], *i.e.* $(x : t_1) \to u_1 \equiv_{\beta\mathcal{R}} (x : t_2) \to u_2$ implies $t_1 \equiv_{\beta\mathcal{R}} t_2$ and $u_1 \equiv_{\beta\mathcal{R}} u_2$ for any x, t_1, t_2, u_1, u_2. These properties guarantee in turn that $\hookrightarrow_\beta$ preserves typing. Note that the injectivity of the product is also called *product compatibility* in the literature [40, 44].

The well-typedness of all fragments of theory $\mathcal{U}$, including the four logical fragments studied in this paper, is established in [6]. Alternatively, the framework of *strongly well-formed* $\lambda\Pi / \equiv$ theories [40] could also be used in the specific case of the logical fragments to ensure well-typedness. This framework is not sufficient for the entirety of theory $\mathcal{U}$, as fragments of theory $\mathcal{U}$ that include the definition of *predicate subtyping* [26] are not strongly well-formed.

▶ **Lemma 1** (Well-typedness [6, Theorem 9]). *All logical fragments of theory $\mathcal{U}$ are well-typed.*

▶ **Corollary 2** (Subject reduction). *Let $\Sigma, \mathcal{R}$ be one of the four logical fragments of theory $\mathcal{U}$. Let t, u, T be terms of $\Lambda(\Sigma)$ and Γ a typing context such that $\Gamma \vdash_{\Sigma,\mathcal{R}} t : T$ and $t \hookrightarrow_{\beta\mathcal{R}} u$, then $\Gamma \vdash_{\Sigma,\mathcal{R}} u : T$.*

▶ **Corollary 3** (Fragment theorem [6, Theorem 7]). *Let $\Sigma_1, \mathcal{R}_1$ and $\Sigma_2, \mathcal{R}_2$ be two of the four logical fragments of theory $\mathcal{U}$ such that $\Sigma_1 \subseteq \Sigma_2$. Let t, T be terms of $\Lambda(\Sigma_1)$ and Γ a typing context such that codom$(\Gamma) \subset \Lambda(\Sigma_1)$. If $\Gamma \vdash_{\Sigma_2,\mathcal{R}_2} t : T$, then $\Gamma \vdash_{\Sigma_1,\mathcal{R}_1} t : T$.*

3.2 Normalization

A sufficient condition for the decidability of type-checking in a well-typed theory $\Sigma, \mathcal{R}$ is the *normalization* [42] of the rewriting system $\hookrightarrow_{\beta\mathcal{R}}$. Indeed, using a normalization strategy and the confluence of $\hookrightarrow_{\beta\mathcal{R}}$, the convertibility modulo $\equiv_{\beta\mathcal{R}}$ of two terms $A, B \in \Lambda(\Sigma)$ is decidable by computing the normal forms of A and B and testing if they are equal (up to α-renaming). In this case, the applicability of the conversion rule (conv) is decidable, ensuring in turn the decidability of type-checking modulo $\Sigma, \mathcal{R}$. Normalization is also a first step towards a proof of consistency for a given theory.

The strong normalization of β-reduction is established for the $\lambda\Pi$-calculus [25], and the additional rewriting systems defined by the four logical fragments of theory $\mathcal{U}$ are obviously normalizing. However, weak and strong normalization are not modular in higher-order rewriting settings [1]; as a consequence, the normalization of $\hookrightarrow_{\beta\mathcal{R}}$ cannot be deduced from the fact that $\hookrightarrow_{\beta}$ and $\hookrightarrow_{\mathcal{R}}$ are both normalizing.

Some theoretical results and tools have been developed in order to prove normalization of term rewriting systems in dependent type theories [20, 35, 7]. However, these results cannot be directly used to prove the normalization of the logical fragments of $\mathcal{U}$, and more generally of theory $\mathcal{U}$. The rule ($\rightsquigarrow$-red) is one of the many problematic rules: the right-hand side is a product, thus the rule is not *arity-preserving* [18], and is pinpointed by SizeChangeTool [7] as being self-looping. As a consequence, the consistency and type-checking decidability of theory $\mathcal{U}$ and many of its fragments is still an open question.

In the following sections, we begin to answer this question by proving the weak normalization of Ecumenical STT, from which the decidability of type-checking and consistency will ensue. The following normalization proof relies on a notion of *models* of $\lambda\Pi/\equiv$ theories valued in structures named Π-*algebras*, which are similar to pre-Heyting algebras [27, 17]. Any $\lambda\Pi/\equiv$ theory which admits a model in every full ordered and complete Π-algebra is called *super-consistent*. In [15], the author establishes the strong normalization of $\hookrightarrow_{\beta}$ over well-typed terms for any super-consistent theory using *reducibility candidates* [41, 23]. Moreover, the super-consistency of an expression of minimal STT with parametric quantifiers is proven. Sections 3.2.1 and 3.2.2 extend the models of minimal STT with parametric quantifiers described in [15] to Constructive STT, thus proving that Constructive STT is super-consistent. Finally, the strong normalization of $\hookrightarrow_{\beta}$ over $\lambda\Pi/\equiv$ terms well-typed in Constructive STT is used to prove weak normalization for all logical fragments of theory $\mathcal{U}$.

3.2.1 Super-consistency

In the following section, we define all the notions necessary to state and prove the super-consistency of Constructive STT.

▶ **Definition 4.** *A* full, complete, and ordered Π-algebra *is formed with*
- *a preordered set $(\mathcal{B}, \leq)$ with a maximal element $\tilde{\top}$ of $\mathcal{B}$,*
- *a function $\tilde{\wedge} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$ such that $a \mathbin{\tilde{\wedge}} b$ is a greatest lower bound of $\{ a, b \}$ for $\leq$,*

- *a function $\tilde{\Pi} : \mathcal{B} \times (\mathcal{P}(\mathcal{B}) \setminus \emptyset) \to \mathcal{B}$ such that $a \leq \tilde{\Pi}(b, S)$ if for all $c \in S$, $a \tilde{\wedge} b \leq c$,*
- *and an order relation $\sqsubseteq$ over $\mathcal{B}$ with respect to which $\tilde{\Pi}$ is left anti-monotonic and right monotonic, and for which every subset of $\mathcal{B}$ has a least upper bound.*

Note that relations $\leq$ and $\sqsubseteq$ need not be in any way related.

▶ **Definition 5.** *A* model valued in a full, ordered, and complete Π-algebra *is a triplet of interpretation functions $(\llbracket\cdot\rrbracket^i)_{1\leq i\leq 3}$ and a set $\mathcal{V}$. The i^{th} interpretation $\llbracket\cdot\rrbracket^i$ takes as arguments a term t and $i-1$ variable assignments $(\phi_j)_{1\leq j<i}$ such that $fv(t) \subseteq \bigcap_{1\leq j<i} dom(\phi_j)$ and returns a value $\llbracket t\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} \in \mathcal{V}$ such that*

- *$\llbracket KIND\rrbracket^2_{\phi_1} = \llbracket TYPE\rrbracket^2_{\phi_1} = \mathcal{B}$ and $\llbracket KIND\rrbracket^3_{\phi_1,\phi_2} = \llbracket TYPE\rrbracket^3_{\phi_1,\phi_2} = \tilde{\top}$,*
- *$\llbracket(x:t) \to u\rrbracket^3_{\phi_1,\phi_2} = \tilde{\Pi}(\llbracket t\rrbracket^3_{\phi_1,\phi_2}, \{\, \llbracket u\rrbracket^3_{(\phi_1,x=c_1),(\phi_2,x=c_2)} : c_1 \in \llbracket t\rrbracket^1, c_2 \in \llbracket t\rrbracket^2_{\phi_1} \,\})$*

This definition is a simplification of its counterpart in [15], which allows models with an arbitrary number of interpretation functions. However, only three levels of interpretation are required to show the super-consistency of Constructive STT.

▶ **Definition 6** (Compatibility). *Let $(\llbracket\cdot\rrbracket_i)_{1\leq i\leq 3}$ be a model valued in a full, ordered, and complete Π-algebra $\mathcal{B}$ and $i \in \{1,2,3\}$. The variable assignments $\phi_1,\ldots,\phi_{i-1}$ are* compatible *with a typing context Δ if for all $1 \leq j < i$ and $(x : A) \in \Delta$ we have $\phi_j(x) \in \llbracket A\rrbracket^j_{\phi_1,\ldots,\phi_{j-1}}$.*

▶ **Definition 7** (Model of a theory). *A model valued in a full, ordered, and complete Π-algebra $\mathcal{B}$ is a model of a theory $\Sigma, \mathcal{R}$ if and only if the following conditions are met for every $i \in \{1,2,3\}$, typing context Δ, terms $t, u, A, B, C \in \Lambda(\Sigma)$, variables $x, y \in \mathcal{V}$, and compatible variable assignments $\phi_1,\ldots,\phi_{i-1}$:*

Variable assignment: *if $i \geq 2$, then for every $(x : A) \in \Delta$, we have $\llbracket x\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} = \phi_{i-1}(x)$.*

Well-typedness: *if $i \geq 2$ and $\Delta \vdash_{\Sigma,\mathcal{R}} t : A$, then $\llbracket t\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} \in \llbracket A\rrbracket^{i-1}_{\phi_1,\ldots,\phi_{i-2}}$.*

Weakening: *if $\Delta \vdash_{\Sigma,\mathcal{R}} t : A$ and $y \notin dom(\Delta)$, then $\llbracket t\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} = \llbracket t\rrbracket^i_{(\phi_1,y=a_1),\ldots,(\phi_{i-1},y=a_{n-1})}$.*

Substitution: *if $\Delta(y : C) \vdash_{\Sigma,\mathcal{R}} t : B$, $\Delta \vdash_{\Sigma,\mathcal{R}} u : C$, then*
$\llbracket(u/y)t\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} = \llbracket t\rrbracket^i_{(\phi_1,y=\llbracket u\rrbracket^2_{\phi_1}),\ldots,(\phi_{i-1},y=\llbracket u\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}})}$.

Validity of the congruence: *if $\Delta \vdash_{\Sigma,\mathcal{R}} A : C$, $\Delta \vdash_{\Sigma,\mathcal{R}} B : C$, and $A \equiv_{\beta\mathcal{R}} B$, then $\llbracket A\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}} = \llbracket B\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}}$*

Validity of the axioms: *if $(c : A) \in \Gamma$, then $\llbracket A\rrbracket^3 \geq \tilde{\top}$.*

▶ **Definition 8** (Super-consistency). *A $\lambda\Pi / \equiv$ theory $\Sigma, \mathcal{R}$ admitting a model valued in every full, ordered, and complete Π-algebra is called* super-consistent.

▶ **Theorem 9** ([15, Theorem 5.1]). *$\to_\beta$ is strongly normalizing over well-typed terms in super-consistent $\lambda\Pi / \equiv$ theories.*

Note that this result does not yield normalization for the whole rewriting system $\hookrightarrow_{\beta\mathcal{R}}$. This theorem is more generally applicable to non-terminating user-defined rewriting systems $\hookrightarrow_{\mathcal{R}}$, which is useful when $\equiv_{\beta\mathcal{R}}$ is a decidable congruence but $\hookrightarrow_{\mathcal{R}}$ is non-terminating. A trivial example would be $\mathcal{R} = \{\, x \hookrightarrow x \,\}$.

▶ **Theorem 10** ([15, Theorem 4.3]). *Minimal STT is super-consistent.*

3.2.2 Models of Constructive STT

In this section, we extend the model of Minimal STT valued in a given arbitrary full, ordered, and complete Π-algebra $\mathcal{B}$ defined in [15] to a model of Ecumenical STT.

3.2.2.1 Model of Minimal STT

Let $\{ e \}$ be an arbitrary one-element set and $\mathcal{A}$ the smallest set containing $\mathcal{B}$ and $\{ e \}$, and closed by cartesian product (denoted $\times$) and exponentiation (denoted $\mathcal{F}$). Let $\mathcal{V}$ be the smallest set containing $\mathcal{A}$, $\mathcal{B}$, and $\{ e \}$ and closed by cartesian product and *dependent function space*, *i.e.* if S is an element of $\mathcal{V}$ and T a family of elements of $\mathcal{V}$ indexed by S, then the set $\mathcal{F}_d(S, T)$ of functions mapping each element s of S to an element of T_s is in $\mathcal{V}$. We define the interpretation functions $(\llbracket\cdot\rrbracket^i)_{1\leq i\leq 3}$ as described in Figure 4.

▶ **Lemma 11** ([15, Theorem 4.2]). *The model* $(\llbracket\cdot\rrbracket^i)_{1\leq i\leq 3}$ *is a model of Minimal STT in* $\mathcal{B}$.

3.2.2.2 Extension of the model to Constructive STT

The three interpretation functions of the model defined in Figure 4 are extended to define a model of Constructive STT valued in the given Π-algebra $\mathcal{B}$ with the following method:

1. we extend the model to interpret the constructive connectives $\top$, $\bot$, $\neg$, $\exists$, $\wedge$, and $\vee$;
2. we prove that the typing of these connectives is preserved by these interpretations;
3. we prove that the axioms declaring these connectives, *i.e.* ($\top$-decl), ($\bot$-decl), ($\neg$-decl), ($\exists$-decl), ($\wedge$-decl), and ($\vee$-decl), are valid in this extension.
4. we prove that reduction by the rewrite rules defining these connectives, *i.e.* ($\top$-red), ($\bot$-red), ($\neg$-red), ($\exists$-red), ($\wedge$-red), and ($\vee$-red), leave the interpretations invariant.

These verifications ensure that our extension is a model of Constructive STT. The most problematic step of this method is Item 4. As an example, constructive conjunction is not directly defined by a rewrite rule of the form $\wedge \hookrightarrow t$ or $a \wedge b \hookrightarrow t$, but by a rule of the form Prf $(a \wedge b) \hookrightarrow t$ which needs to preserve interpretation. As a consequence, the interpretation of Prf and of the application need to be taken into account while accomplishing Item 1.

For example, as Prf $\top \hookrightarrow (z : \text{Prop}) \to \text{Prf}\ z \to \text{Prf}\ z$, we need to define $\llbracket\top\rrbracket^3_{\phi,\psi}$ such that $\llbracket\text{Prf}\ \top\rrbracket^3_{\phi,\psi} = \llbracket(z : \text{Prop}) \to \text{Prf}\ z \to \text{Prf}\ z\rrbracket^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{ \tilde{\Pi}(c, \{ c \}) \ : \ c \in \mathcal{B} \})$; here we conclude using $\llbracket\text{Prf}\ \top\rrbracket^3_{\phi,\psi} = \llbracket\top\rrbracket^3_{\phi,\psi}$. The interpretations of all constructive connectives and quantifiers are described in Figure 5.

▶ **Lemma 12** (Well-typedness). *For any* $i \in \{ 1, 2 \}$ *and declaration* $c : T$ *in* ($\top$-decl), ($\bot$-decl), ($\neg$-decl), ($\exists$-decl), ($\wedge$-decl), *and* ($\vee$-decl), *we have* $\llbracket c\rrbracket^{i+1}_{\phi_1,\ldots,\phi_i} \in \llbracket T\rrbracket^i_{\phi_1,\ldots,\phi_{i-1}}$.

Proof. We check every case.

If $i = 1$: $\llbracket c\rrbracket^2_\phi = e$ and $\llbracket T\rrbracket^1 = \{ e \}$

If $i = 2$: $\llbracket\top\rrbracket^3_{\phi,\psi}$ and $\llbracket\bot\rrbracket^3_{\phi,\psi}$ are elements of $\mathcal{B}$, which is equal to $\llbracket\text{Prop}\rrbracket^2_\phi$,

- $\llbracket\neg\rrbracket^3_{\phi,\psi}$ is an element of $\mathcal{F}(\{ e \} \times \mathcal{B}, \mathcal{B})$, which is equal to $\llbracket\text{Prop} \to \text{Prop}\rrbracket^2_\phi$,
- $\llbracket\wedge\rrbracket^3_{\phi,\psi}$ and $\llbracket\vee\rrbracket^3_{\phi,\psi}$ are elements of $\mathcal{F}(\{ e \} \times \mathcal{B}, \mathcal{F}(\{ e \} \times \mathcal{B}, \mathcal{B}))$, which is equal to $\llbracket\text{Prop} \to \text{Prop} \to \text{Prop}\rrbracket^2_\phi$,
- and $\llbracket\exists\rrbracket^3_{\phi,\psi}$ is an element of $\mathcal{F}(\mathcal{A} \times \mathcal{B}, \mathcal{F}(\{ e \} \times \mathcal{F}(\{ e \} \times S, \mathcal{B}), \mathcal{B}))$, which is equal to $\llbracket(x : \text{Set}) \to (\text{El}\ x \to \text{Prop}) \to \text{Prop}\rrbracket^2_\phi$. ◀

▶ **Lemma 13** (Validity of the axioms). *For all declarations* $c : T$ *in* ($\top$-decl), ($\bot$-decl), ($\neg$-decl), ($\exists$-decl), ($\wedge$-decl), *and* ($\vee$-decl), *then* $\llbracket T\rrbracket^3_{\phi,\psi} \geq \tilde{\top}$.

Proof. There are four possibilities.

- For ($\top$-decl) and ($\bot$-decl), we have $T = \text{Prop}$ and by definition $\llbracket T\rrbracket^3_{\phi,\psi} = \tilde{\top} \geq \tilde{\top}$.
- For ($\neg$-decl), we have $T = \text{Prop} \to \text{Prop}$, and as a consequence $\llbracket T\rrbracket^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{ \tilde{\top} \}) \geq \tilde{\top}$.
- For ($\wedge$-decl) and ($\vee$-decl), we have $T = \text{Prop} \to \text{Prop} \to \text{Prop}$ and by definition $\llbracket T\rrbracket^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{ \tilde{\Pi}(\tilde{\top}, \{ \tilde{\top} \}) \}) \geq \tilde{\top}$.
- For ($\exists$-decl), we have $T = (x : \text{Set}) \to (\text{El}\ x \to \text{Prop}) \to \text{Prop}$, and as a consequence $\llbracket T\rrbracket^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{ \tilde{\Pi}(\tilde{\Pi}(c, \{ \tilde{\top} \}), \{ \tilde{\top} \}) \ : \ c \in \mathcal{B} \}) \geq \tilde{\top}$. ◀

$$
\begin{aligned}
\llbracket \mathtt{TYPE} \rrbracket^1 = \llbracket \mathtt{KIND} \rrbracket^1 &= \mathcal{V} \\
\llbracket \mathrm{Set} \rrbracket^1 &= \mathcal{A} \\
\llbracket (x : C) \to D \rrbracket^1 &= \begin{cases} \{\, e \,\} & \text{if } \llbracket D \rrbracket^1 = \{\, e \,\} \\ \mathcal{F}(\llbracket C \rrbracket^1, \llbracket D \rrbracket^1) & \text{else} \end{cases} \\
\llbracket \lambda x : C.\, t \rrbracket^1 = \llbracket t\ u \rrbracket^1 &= \llbracket t \rrbracket^1 \\
\llbracket t \rrbracket^1 &= \{\, e \,\} \text{ in every other case}
\end{aligned}
$$

(a) First level of interpretation.

$$
\begin{aligned}
\llbracket \mathtt{TYPE} \rrbracket^2_\phi = \llbracket \mathtt{KIND} \rrbracket^2_\phi = \llbracket \mathrm{Set} \rrbracket^2_\phi = \llbracket o \rrbracket^2_\phi &= \mathcal{B} \\
\llbracket \mathrm{El} \rrbracket^2_\phi &= S \mapsto S \in \mathcal{F}(\mathcal{A}, \mathcal{V}) \\
\llbracket \mathrm{Prf} \rrbracket^2_\phi &= e \mapsto \{\, e \,\} \in \mathcal{F}(\{\, e \,\}, \mathcal{V}) \\
\llbracket \iota \rrbracket^2_\phi &= \{\, e \,\} \\
\llbracket (x : C) \to D \rrbracket^2_\phi &= \begin{cases} \{\, e \,\} \text{ if for all } c' \in \llbracket C \rrbracket^1,\ \llbracket D \rrbracket^2_{\phi, x = c'} = \{\, e \,\} \\ \mathcal{F}_d(\llbracket C \rrbracket^1 \times \llbracket C \rrbracket^2_\phi, (\llbracket D \rrbracket^2_{\phi, x = c'})_{\langle c, c' \rangle}) \text{ else} \end{cases} \\
\llbracket \rightsquigarrow \rrbracket^2_\phi &= \begin{cases} \{\, e \,\} \text{ if } T = \{\, e \,\} \\ \langle S, T \rangle \in \mathcal{A} \times \mathcal{A} \mapsto \mathcal{F}(\{\, e \,\} \times S, T) \text{ else} \end{cases} \\
\llbracket x \rrbracket^2_\phi &= \phi(x) \\
\llbracket \lambda x : C.\, t \rrbracket^2_\phi &= \begin{cases} e \text{ if for all } c \in \llbracket C \rrbracket^1,\ \llbracket t \rrbracket^2_{\phi, x = c} = e \\ c \in \llbracket C \rrbracket^1 \mapsto \llbracket t \rrbracket^2_{\phi, x = c} \text{ else} \end{cases} \\
\llbracket t\ u \rrbracket^2_\phi &= \begin{cases} e \text{ if } \llbracket t \rrbracket^2_\phi = e \\ \llbracket t \rrbracket^2_\phi\ \llbracket u \rrbracket^2_\phi \text{ else} \end{cases} \\
\llbracket t \rrbracket^2_\phi &= e \text{ in every other case}
\end{aligned}
$$

(b) Second level of interpretation.

$$
\begin{aligned}
\llbracket \mathtt{TYPE} \rrbracket^3_{\phi,\psi} = \llbracket \mathtt{KIND} \rrbracket^3_{\phi,\psi} &= \tilde{\top} \\
\llbracket \mathrm{Set} \rrbracket^3_{\phi,\psi} = \llbracket \iota \rrbracket^3_{\phi,\psi} = \llbracket o \rrbracket^3_{\phi,\psi} &= \tilde{\top} \\
\llbracket \rightsquigarrow \rrbracket^3_{\phi,\psi} &= \langle \langle S, a \rangle, \langle T, b \rangle \rangle \in (\mathcal{A} \times \mathcal{B})^2 \mapsto \tilde{\Pi}(a, \{\, b \,\}) \in \mathcal{B} \\
\llbracket \mathrm{El} \rrbracket^3_{\phi,\psi} &= \langle S, a \rangle \in \mathcal{A} \times \mathcal{B} \mapsto a \in \mathcal{B} \\
\llbracket \mathrm{Prf} \rrbracket^3_{\phi,\psi} &= \langle e, a \rangle \in \{\, e \,\} \times \mathcal{B} \mapsto a \in \mathcal{B} \\
\llbracket \Rightarrow \rrbracket^3_{\phi,\psi} &= \langle e, a \rangle \in \{\, e \,\} \times \mathcal{B} \mapsto \langle e, b \rangle \in \{\, e \,\} \times \mathcal{B} \mapsto \tilde{\Pi}(a, \{\, b \,\}) \\
\llbracket \forall \rrbracket^3_{\phi,\psi} &= \langle S, a \rangle \in \mathcal{A} \times \mathcal{B} \mapsto \langle e, g \rangle \in \{\, e \,\} \times \mathcal{F}(\{\, e \,\} \times S, \mathcal{B}) \mapsto \\
&\ \tilde{\Pi}(a, \{\, g\ \langle e, s \rangle \ :\ s \in S \,\}) \\
\llbracket x \rrbracket^3_{\phi,\psi} &= \psi(x) \\
\llbracket (x : C) \to D \rrbracket^3_{\phi,\psi} &= \tilde{\Pi}\left(\llbracket C \rrbracket^3_{\phi,\psi}, \{\, \llbracket D \rrbracket^3_{\phi(x=c'),\psi(x=c)} \ :\ c' \in \llbracket C \rrbracket^1, c \in \llbracket C \rrbracket^2_\phi \,\} \right) \\
\llbracket \lambda x : C.\, t \rrbracket^3_{\phi,\psi} &= \begin{cases} e \text{ if for all } \langle c', c \rangle \in \llbracket C \rrbracket^1 \times \llbracket C \rrbracket^2_\phi,\ \llbracket t \rrbracket^3_{\phi(x=c'),\psi(x=c)} = e \\ \langle c', c \rangle \in \llbracket C \rrbracket^1 \times \llbracket C \rrbracket^2_\phi \mapsto \llbracket t \rrbracket^3_{\phi(x=c'),\psi(x=c)} \text{ else} \end{cases} \\
\llbracket t\ u \rrbracket^3_{\phi,\psi} &= \begin{cases} e \text{ if } \llbracket t \rrbracket^3_{\phi,\psi} = e \\ \llbracket t \rrbracket^3_{\phi,\psi} \langle \llbracket u \rrbracket^2_\phi, \llbracket u \rrbracket^3_{\phi,\psi} \rangle \text{ else} \end{cases}
\end{aligned}
$$

(c) Third level of interpretation.

Figure 4 Model of minimal STT.

$$[\![t]\!]^1 = \{\, e \,\} \text{ if } t \in \{\, \top, \bot, \neg, \exists, \wedge, \vee, \mathrm{Prop} \,\}$$

(a) First level.

$$\begin{aligned} [\![\mathrm{Prop}]\!]^2_\phi &= \mathcal{B} \\ [\![t]\!]^2_\phi &= e \text{ if } t \in \{\, \top, \bot, \neg, \exists, \wedge, \vee \,\} \end{aligned}$$

(b) Second level.

$$\begin{aligned}
[\![\mathrm{Prop}]\!]^3_{\phi,\psi} &= \tilde{\top} \\
[\![\top]\!]^3_{\phi,\psi} &= \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(c, \{\, c \,\}) \ : \ c \in \mathcal{B} \,\}) \\
[\![\bot]\!]^3_{\phi,\psi} &= \tilde{\Pi}(\tilde{\top}, \mathcal{B}) \\
[\![\wedge]\!]^3_{\phi,\psi} &= \langle e, a\rangle \mapsto \langle e, b\rangle \mapsto \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(\tilde{\Pi}(a, \{\, \tilde{\Pi}(b, \{\, c \,\}) \,\}), \{\, c \,\}) \ : \ c \in \mathcal{B} \,\}) \\
[\![\vee]\!]^3_{\phi,\psi} &= \langle e, a\rangle \mapsto \langle e, b\rangle \mapsto \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(\tilde{\Pi}(a, \{\, c \,\}), \{\, \tilde{\Pi}(\tilde{\Pi}(b, \{\, c \,\}), \{\, c \,\}) \,\}) \ : \ c \in \mathcal{B} \,\}) \\
[\![\neg]\!]^3_{\phi,\psi} &= \langle e, a\rangle \mapsto \tilde{\Pi}(a, \{\, \tilde{\Pi}(\tilde{\top}, \mathcal{B}) \,\}) \\
[\![\exists]\!]^3_{\phi,\psi} &= \langle S, a\rangle \in \mathcal{A} \times \mathcal{B} \mapsto \langle e, g\rangle \in \{\, e \,\} \times \mathcal{F}(\{\, e \,\} \times S, \mathcal{B}) \mapsto \\
&\qquad \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(a, \{\, \tilde{\Pi}(g\ \langle e, s\rangle, \{\, c \,\}) \ : \ s \in S \,\}), \{\, c \,\}) \ : \ c \in \mathcal{B} \,\})
\end{aligned}$$

(c) Third level.

Figure 5 Interpretations of Constructive STT connectives.

▶ **Lemma 14** (Validity of the congruence). *For all rewrite rules $\ell \hookrightarrow r$ in ($\top$-red), ($\bot$-red), ($\neg$-red), ($\exists$-red), ($\wedge$-red), and ($\vee$-red), then for all $i \in \{\, 1, 2, 3 \,\}$, $[\![\ell]\!]^i_{\phi_1,\dots,\phi_{i-1}} = [\![r]\!]^i_{\phi_1,\dots,\phi_{i-1}}$.*

Proof. We check every case.
If $i = 1$: in all cases $[\![\ell]\!]^1 = [\![r]\!]^1 = \{\, e \,\}$.
If $i = 2$: in all cases $[\![\ell]\!]^2_\phi = [\![r]\!]^2_\phi = e$.
If $i = 3$: we check the equality for every rule.

($\top$-red): $[\![\mathrm{Prf}\ \top]\!]^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(c, \{\, c \,\}) \ : \ c \in \mathcal{B} \,\}) = [\![(z : \mathrm{Prop}) \to \mathrm{Prf}\ z \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

($\bot$-red): $[\![\mathrm{Prf}\ \bot]\!]^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \mathcal{B}) = [\![(z : \mathrm{Prop}) \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

($\neg$-red): $[\![\mathrm{Prf}\ \neg A]\!]^3_{\phi,\psi} = \tilde{\Pi}([\![A]\!]^3_{\phi,\psi}, \{\, \tilde{\Pi}(\tilde{\top}, \mathcal{B}) \,\}) = [\![\mathrm{Prf}\ A \to (z : \mathrm{Prop}) \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

($\wedge$-red): $[\![\mathrm{Prf}\ A \wedge B]\!]^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(\tilde{\Pi}([\![A]\!]^3_{\phi,\psi}, \{\, \tilde{\Pi}([\![B]\!]^3_{\phi,\psi}, \{\, c \,\}) \,\}), \{\, c \,\}) \ : \ c \in \mathcal{B} \,\}) = [\![(z : \mathrm{Prop}) \to (\mathrm{Prf}\ A \to \mathrm{Prf}\ B \to \mathrm{Prf}\ z) \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

($\vee$-red): $[\![\mathrm{Prf}\ A \vee B]\!]^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}(\tilde{\Pi}([\![A]\!]^3_{\phi,\psi}, \{\, c \,\}), \{\, \tilde{\Pi}(\tilde{\Pi}([\![B]\!]^3_{\phi,\psi}, \{\, c \,\}), \{\, c \,\}) \,\}) \ : \ c \in \mathcal{B} \,\}) = [\![(z : \mathrm{Prop}) \to (\mathrm{Prf}\ A \to \mathrm{Prf}\ z) \to (\mathrm{Prf}\ B \to \mathrm{Prf}\ z) \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

($\exists$-red): $[\![\mathrm{Prf}\ (\exists\ T\ P)]\!]^3_{\phi,\psi} = \tilde{\Pi}(\tilde{\top}, \{\, \tilde{\Pi}([\![T]\!]^3_{\phi,\psi}, \{\, \tilde{\Pi}([\![P]\!]^3_{\phi,\psi}\ \langle e, s\rangle, \{c\}) \ : \ s \in [\![T]\!]^2_\phi \,\}), \{c\}) \ : \ c \in \mathcal{B} \,\}) = [\![(z : \mathrm{Prop}) \to ((y : \mathrm{El}\ T) \to \mathrm{Prf}\ P\ y \to \mathrm{Prf}\ z) \to \mathrm{Prf}\ z]\!]^3_{\phi,\psi}$

In all cases, reduction preserves interpretation. ◀

▶ **Proposition 15.** *$\hookrightarrow_\beta$ strongly terminates on well-typed terms of Constructive STT.*

Proof. Constructive STT has models valued in all full, ordered, and complete Π-algebras. Thus, the theory is super-consistent and we conclude by Theorem 9. ◀

▶ **Corollary 16.** *Ecumenical STT is weakly normalizing.*

Proof. We exhibit a normalizing strategy for a given well-typed term t of Ecumenical STT.

1. First, normalize t with respect to every rule defining classical connectives and quantifiers, *i.e.* ($\Rightarrow_c$-red), ($\wedge_c$-red), ($\vee_c$-red), ($\forall_c$-red), and ($\exists_c$-red). This procedure terminates as the number of classical connectives and quantifiers stricly decreases with every reduction. Note that the result t' of this procedure is well-typed in Constructive STT.
2. Second, β-normalize t'. This procedure terminates by Proposition 15, yielding a term t''.

3. Finally, normalize t'' with respect to ($\top$-red), ($\bot$-red), ($\Rightarrow$-red), ($\wedge$-red), ($\vee$-red), ($\forall$-red), and ($\exists$-red). The number of connectives and quantifiers strictly decreases with each reduction. Each of these reduction step using rules ($\forall$-red) and ($\exists$-red) might create one β-redex; however it is of the form $(\lambda x : t.\, u)\ \ y$, which can be immediately reduced without increasing the number of connectives and quantifiers.

The term resulting from this procedure is $\hookrightarrow_{\beta\mathscr{R}^e_{HO}}$-normal. ◀

We can conclude that type-checking in the logical fragments of theory $\mathcal{U}$ is decidable.

▶ **Corollary 17.** *Type-checking in Ecumenical STT is decidable.*

In practice, this normalization strategy is not implemented to type-check modulo Ecumenical STT. However, a considerable number of proofs, notably the standard HOL Light library, have been type-checked in this theory [12]. The diversity and size of these developments provide no counter-example to the strong normalization of Ecumenical STT.

▶ **Conjecture 18.** *Ecumenical STT is strongly normalizing.*

4 First order ecumenism

Some fragments of theory $\mathcal{U}$ have been previously studied separately in [3, 26, 43]. Soundness and conservativity of these expressions with respect to a reference system, and consistency have not all been established for all fragments; the consistency of theory $\mathcal{U}$ is still an open question.

In the following sections, we study the soundness and conservativity of all logical fragments of $\mathcal{U}$ with respect to appropriate reference systems (first order constructive and classical logics, and higher-order constructive and classical logics). We also establish the consistency of the logical fragments of theory $\mathcal{U}$, which is a first step towards the consistency of the whole theory. In the current section, we focus on the first order fragments, ie Constructive Predicate Logic and Ecumenical Predicate Logic.

4.1 Reference systems: constructive and classical predicate logic

As reference systems for Constructive Predicate Logic and Ecumenical Predicate Logic, we choose the systems NJ and NK [11].

In these systems, terms are defined over a first order language $\mathcal{L}$ containing *function* and *predicate* symbols with their arity. Terms are of the form $t, u, \dots = x \mid f(t_1, \dots, t_n)$ where x is a variable and $f \in \mathcal{L}$ is a function symbol of arity n. Formulas are defined by $A, B, \dots = P(t_1, \dots, t_n) \mid \top \mid \bot \mid A \wedge B \mid A \vee B \mid A \Rightarrow B \mid \neg A \mid \forall x.\ A \mid \exists x.\ A$, where x is a variable and $P \in \mathcal{L}$ is an n-ary predicate symbol. Typing contexts are defined by $\Gamma, \Delta, \dots = [] \mid \Gamma, A$. The rules of the NJ proof system are described in Figure 6. System NK is the extension of NJ with the excluded-middle rule (EM) of conclusion $A \vee \neg A$ and without premises. We respectively write $\Gamma \vdash_{\mathsf{NJ}} A$ and $\Gamma \vdash_{\mathsf{NK}} A$ if the judgment $\Gamma \vdash A$ is derivable in NJ and NK.

4.2 Soundness and conservativity of Constructive Predicate Logic

Soundness and conservativity of Constructive Predicate Logic were first proved in [13], and restated and reproven in [40]. The proof if soundness is tedious and error prone: in both aforementioned proofs, the free variables occuring in the constructive natural deduction proof are not accurately taken into account. In the following, we reprove soundness and conservativity of Constructive and highlight the errors made in previous proofs.

$$\frac{A \in \Gamma}{\Gamma \vdash A}\text{ axiom} \qquad \frac{}{\Gamma \vdash \top}\ \top\text{-intro} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-elim} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-elim} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro} \qquad \frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg\text{-intro} \qquad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash \bot}\ \neg\text{-elim} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-intro}$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\text{-elim} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-intro} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-intro}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-elim} \qquad \frac{\Gamma \vdash \forall x.\ A}{\Gamma \vdash A}\ \forall\text{-elim} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \exists x.\ A}\ \exists\text{-intro}$$

$$\frac{\Gamma \vdash A \quad x \notin fv(\Gamma)}{\Gamma \vdash \forall x.\ A}\ \forall\text{-intro} \qquad \frac{\Gamma \vdash \exists x.\ A \quad \Gamma, A \vdash B \quad x \notin fv(\Gamma, B)}{\Gamma \vdash B}\ \exists\text{-elim}$$

Figure 6 Rules of constructive predicate logic NJ.

$$\begin{array}{rcl} |x|_c & \triangleq & x \\ |f(t_1,\dots,t_n)|_c & \triangleq & \dot{f}\ |t_1|_c \dots |t_n|_c \\ |P(t_1,\dots,t_n)|_c & \triangleq & \dot{P}\ |t_1|_c \dots |t_n|_c \\ |\top|_c & \triangleq & \top \\ |\bot|_c & \triangleq & \bot \\ |\neg A|_c & \triangleq & \neg\, |A|_c \end{array} \qquad \begin{array}{rcl} |A \wedge B|_c & \triangleq & |A|_c \wedge |B|_c \\ |A \vee B|_c & \triangleq & |A|_c \vee |B|_c \\ |A \Rightarrow B|_c & \triangleq & |A|_c \Rightarrow |B|_c \\ |\forall x.\ A|_c & \triangleq & \forall\ \iota\ (\lambda x : \mathrm{El}\ \iota.\ |A|_c) \\ |\exists x.\ A|_c & \triangleq & \exists\ \iota\ (\lambda x : \mathrm{El}\ \iota.\ |A|_c) \end{array}$$

Figure 7 Translation $|\cdot|_c$ of NJ to its expression in $\lambda\Pi / \equiv$.

As in [40], a first order language $\mathcal{L}$ is encoded in $\lambda\Pi / \equiv$ by a context $\Delta_{\mathcal{L}}$ declaring an n-ary function $\dot{f} : \mathrm{El}\ \iota \to \dots \to \mathrm{El}\ \iota \to \mathrm{El}\ \iota$ for every n-ary function symbol $f \in \mathcal{L}$, and an n-ary function $\dot{P} : \mathrm{El}\ \iota \to \dots \to \mathrm{El}\ \iota \to \mathrm{Prop}$ for every n-ary predicate symbol $P \in \mathcal{L}$. For example, the first order language $\mathcal{L}$ containing a nullary predicate P and a unary predicate Q is expressed in $\lambda\Pi / \equiv$ by the context $\Delta_{\mathcal{L}}$ represented in Figure 8. Terms and formulas of the intuitionistic first order system NJ are naturally embedded into Constructive Predicate Logic using transformation $|.|_c$ defined in Figure 7. Finally, for any context $\Gamma = A_1, \dots, A_n$ and proposition A, denoting by $y_1, \dots, y_k$ the free variables of Γ, A, we define $|\Gamma|_c^A = y_1 : \mathrm{El}\ \iota, \dots, y_k : \mathrm{El}\ \iota, x_1 : \mathrm{Prf}\ |A_1|_c, \dots, x_n : \mathrm{Prf}\ |A_n|_c$. Again, Figure 8 provides an example: the NJ context $\forall x.\ [Q(x) \wedge P]$ is translated into the $\lambda\Pi / \equiv$ context Γ.

In [40], the NJ proof represented in Figure 8 is expressed by the term π in Constructive Predicate Logic. Note that the variable y is not free in the final judgment $\forall x.\ [Q(x) \wedge P] \vdash P$, thus is not declared in the typing context $\Delta_{\mathcal{L}}, \Gamma$. However, y is free in the NJ proof, thus appears free in term π. As a consequence, π is not well-typed in $\Delta_{\mathcal{L}}, \Gamma$ and the soundness result of [40] does not hold. To handle such free variables, we suggest the following slight alteration: we add a witness $w : \mathrm{El}\ \iota$ to context $\Delta_{\mathcal{L}}$ and substitute whenever necessary. As an example, we express the NJ proof of Figure 8 with the term $\pi(y/w)$, well-typed in $\Delta_{\mathcal{L}}, \Gamma$. Note that this presentation can be easily extended to many sorted natural deduction by adding a constant $s : \mathrm{El}$ and a witness $w_s : \mathrm{El}\ s$ for every additional sort.

▶ **Lemma 19** (Soundness). *If A is an NJ formula over a first-order language $\mathcal{L}$ such that $\Gamma \vdash_{NJ} A$, then there is a term $t \in \Lambda(\Sigma^c_{FO})$ such that $\Delta_{\mathcal{L}}; |\Gamma|_c^A \vdash^c_{FO} t : \mathrm{Prf}\ |A|_c$.*

$$\dfrac{\dfrac{\forall x.\ [Q(x) \wedge P] \vdash \forall x.\ [Q(x) \wedge P]}{\forall x.\ [Q(x) \wedge P] \vdash Q(y) \wedge P}\ \forall\text{-elim}}{\forall x.\ [Q(x) \wedge P] \vdash P}\ \wedge\text{-elim}$$

$$\begin{aligned} \Delta_{\mathcal{L}} &= (P : \mathrm{Prop}), (Q : \mathrm{El}\ \iota \to \mathrm{Prop}) \\ \Gamma &= (H : \mathrm{Prf}\ (\forall\ \iota\ (\lambda x : \mathrm{El}\ \iota.\, [(Q\ x) \wedge P]))) \\ \pi &= (H\ y)\ P\ (\lambda x_1 : \mathrm{Prf}\ (Q\ y).\, \lambda x_2 : \mathrm{Prf}\ P.\, x_2) \end{aligned}$$

Figure 8 Counter-example to soundness proofs of constructive predicate logic from [13, 40].

Proof. By induction on the derivation of $\Gamma \vdash A$. We develop the case of the left elimination of the conjunction $\wedge$-elim to illustrate the use of the witness $w : \mathrm{El}\ \iota$. By induction hypothesis on the proof π_B of $\Gamma \vdash A \wedge B$, there is a term t_B such that $\Delta_{\mathcal{L}}; |\Gamma|_c^{A\wedge B} \vdash^c_{FO} t_B : \mathrm{Prf}\ |A \wedge B|_c$.

Term $t = t_B\ |A|_c\ (\lambda z_1 : \mathrm{Prf}\ |A|_c.\, \lambda z_2 : \mathrm{Prf}\ |B|_c.\, z_1)$ is of type $\mathrm{Prf}\ |A|_c$ in context $\Delta_{\mathcal{L}}, |\Gamma|_c^{A\wedge B}$. Let $y_1, \dots, y_k = fv(B) \backslash fv(\Gamma, A)$. As $|\Gamma|_c^{A\wedge B} = |\Gamma|_c^A, y_1 : \mathrm{El}\ \iota, \dots, y_k : \mathrm{El}\ \iota$, and using the substitution lemma [40, Lemma 2.6.9.], $\Delta_{\mathcal{L}}; |\Gamma|_c^A \vdash (w/y_1, \dots, w/y_k)t : \mathrm{Prf}\ |A|_c$. ◀

The conservativity of Constructive Predicate Logic with respect to NJ, which is the converse statement to Lemma 19, has been established in [13, 40] and can be seen as a specific case of the proof of conservativity of Constructive STT, further developed in Section 5.2.2.

▶ **Lemma 20** (Conservativity). *Let $\mathcal{L}$ be a first-order language and A a NK formula over $\mathcal{L}$. If there is a term $t \in \Lambda(\Sigma^c_{FO})$ such that $\Delta_{\mathcal{L}}; |\Gamma|_c^A \vdash^c_{FO} t : \mathrm{Prf}\ |A|_c$, then $\Gamma \vdash_{NJ} A$.*

4.3 Soundness and conservativity of Ecumenical Predicate Logic

In the following section, the soudness and conservativity of Ecumenical Predicate Logic with respect to NK are established using the analogous results already established for Constructive Predicate Logic, *i.e.* Lemmas 19 and 20, and the properties of double-negation translations.

Figure 9a defines the embedding $|.|_e$ of NK formulas into Ecumenical terms, which maps every NK connective to the corresponding classical connective. Considering the construction of the classical connectives, this transformation mimicks the Kolmogorov double negation translation $A \mapsto \neg\neg(A^\perp)$ [29] represented in Figure 9b. In the following, the Kolmogorov translation $\neg\neg(\cdot^\perp)$ is naturally extended to contexts.

$$\begin{array}{rcl} |x|_e &=& x \\ |f(t_1, \dots, t_n)|_e &=& \dot{f}\ |t_1|_e \dots |t_n|_e \\ |P(t_1, \dots, t_n)|_e &=& \dot{P}\ |t_1|_e \dots |t_n|_e \\ |\square|_e &=& \square \\ |\neg A|_e &=& \neg\, |A|_e \\ |A \bowtie B|_e &=& |A|_e \bowtie_c |B|_e \\ |Qx.\ A|_e &=& Q_c\ \iota\ (\lambda x : \mathrm{El}\ \iota.\, |A|_e) \end{array}$$

(a) Expressing NK into Ecumenical Predicate Logic.

$$\begin{array}{rcl} P(t_1, \dots, t_n)^\perp &=& P(t_1, \dots, t_n) \\ \square^\perp &=& \square \\ (\neg A)^\perp &=& \neg A \\ (A \bowtie B)^\perp &=& (\neg\neg A) \bowtie (\neg\neg B) \\ (Qx.\ A)^\perp &=& Qx.\ (\neg\neg A) \end{array}$$

(b) The $\cdot^\perp$ translation from NK to NJ.

Figure 9 Translations of NK propositions, where $\square \in \{\top, \bot\}$, $\bowtie \in \{\wedge, \vee, \Rightarrow\}$, and $Q \in \{\forall, \exists\}$.

▶ **Lemma 21.** *For every NK proposition A, $\mathrm{Prf}_c\ \ |A|_e \equiv_{\beta\mathcal{R}^e_{FO}} |\neg\neg A^\perp|_c$.*

Proof. By a straightforward induction on the structure of formula A. ◀

▶ **Lemma 22** ([29]). *For every NK proposition A, if $\Gamma \vdash_{NK} A$ then $\neg\neg\Gamma^{\perp} \vdash_{NJ} \neg\neg A^{\perp}$.*

The soundness of Ecumenical Predicate Logic with respect to NK is immediate using Lemmas 21 and 22. Formally, for any context $\Gamma = A_1, \ldots, A_n$ and proposition A, denoting by $y_1, \ldots, y_k$ the free variables of Γ, A, we define $|\Gamma|_e^A = y_1 : \mathrm{El}\ \iota, \ldots, y_k : \mathrm{El}\ \iota, x_1 : \mathrm{Prf}_c\ |A_1|_e, \ldots, x_n : \mathrm{Prf}_c\ |A_n|_e$.

▶ **Lemma 23** (Soundness). *If A is NK formula over language $\mathscr{L}$ such that $\Gamma \vdash_{NK} A$ is provable in NK, then there exists a term $t \in \Lambda(\Sigma_{FO}^e)$ such that $\Delta_{\mathscr{L}}; |\Gamma|_{A,c} \vdash_{FO}^e t : Prf_c\ |A|_e$.*

Proof. By Lemma 22, $\neg\neg\Gamma^{\perp} \vdash_{NJ} \neg\neg A^{\perp}$ is provable in NJ. By the soundness of Constructive Predicate Logic, $\Delta_{\mathscr{L}}; |\neg\neg\Gamma^{\perp}|_c^{\neg\neg A} \vdash_{FO}^c t : \mathrm{Prf}\ |\neg\neg A^{\perp}|_c$. By Lemma 21 and the fact that $fv(\neg\neg A) = fv(A)$, we conclude that $\Delta_{\mathscr{L}}; |\Gamma|_e^A \vdash_{FO}^e t : \mathrm{Prf}_c\ |A|_e$. ◀

▶ **Lemma 24** (Conservativity). *Let $\mathscr{L}$ be a first-order language and A a NK formula over $\mathscr{L}$. If there is a term $t \in \Lambda(\Sigma_{FO}^e)$ such that $\Delta_{\mathscr{L}}, |\Gamma|_e^A \vdash_{FO}^e t : Prf_c\ |A|_e$, then $\Gamma \vdash_{NK} A$.*

Proof. By the conversion rule and Lemma 21, $\Delta_{\mathscr{L}}, |\neg\neg\Gamma^{\perp}|_c^A \vdash_{FO}^e t : \mathrm{Prf}\ (|\neg\neg A^{\perp}|_c)$. By Corollary 3, there is a reduct of $\mathrm{Prf}\ (|\neg\neg A^{\perp}|_c)$, which we will denote by T, such that $\Delta_{\mathscr{L}}, |\neg\neg\Gamma^{\perp}|_c^A \vdash_{FO}^c t : T$. By conversion, $\Delta_{\mathscr{L}}, |\neg\neg\Gamma^{\perp}|_c^A \vdash_{FO}^c t : \mathrm{Prf}\ (|\neg\neg A^{\perp}|_c)$. By Lemma 20, $\neg\neg\Gamma^{\perp} \vdash_{NJ} \neg\neg A^{\perp}$ is derivable in NJ, and we conclude by Lemma 22 that $\Gamma \vdash_{NK} A$. ◀

5 Higher order ecumenism

In the following section, we study the soundness and conservativity of the higher order logical fragments of $\mathcal{U}$ with respect to higher-order constructive and classical logics. We will then conclude that Ecumenical STT is consistent.

5.1 Reference systems: constructive and classical HOL-λ

As reference systems for Constructive and Ecumenical STT, we consider the intentional version of HOL-λ [16], that is the system obtained by removing the η-expansion rule.

The types of the system HOL-λ are the *simple types* defined by $T, U, \dots = \iota \mid o \mid T \to U$. Terms of HOL-$\lambda$ are λ-terms with additional constants, among which figure the logical connectives. Formally, terms and their associated types are inductively defined by:

- a set of typed variables $\mathcal{X}$, such that every variable $x \in \mathcal{X}$ of type T is a term of type T;
- a set of typed constants $\mathscr{L}$, such that every constant $c \in \mathscr{L}$ of type T is a term of type T;
- for every term t of type U and variable $x \in \mathcal{X}$ of type T, $\lambda x.\ t$ is a term of type $T \to U$;
- for every pair of terms t and u of respective types $T \to U$ and T, $t\ u$ is a term of type U;
- $\dot{\Rightarrow}$, $\dot{\wedge}$, and $\dot{\vee}$ are terms of type $o \to o \to o$, $\dot{\perp}$ and $\dot{\top}$ of type o, and $\dot{\neg}$ of type $o \to o$;
- $\dot{\forall}_T$ and $\dot{\exists}_T$ are terms of type $(T \to o) \to o$ for every simple type T.

We assume that there is an infinite number of variables associated to each simple type. Terms of type o are called *propositions.* Note that we will use the infix notation for the binary connectives $\dot{\Rightarrow}$, $\dot{\wedge}$, and $\dot{\vee}$ for readability purposes. We write the substitution of variable x by a similarly typed term u in a term t by $(u/x)t$.

The β-reduction is defined by the rewrite rule $(\lambda x.\ t)\ u \hookrightarrow (u/x)t$. The rewriting system $\hookrightarrow_{\beta}$, *i.e.* the smallest relation containing β-reduction and closed by term constructors and substitution, is confluent and strongly normalizing [23]. In the following, the $\beta-$normal form of a term t will be denoted by $t{\downarrow}$. The proof system of HOL-λ is shown on Figure 10; all propositions appearing in a proof are normal. The constructive subsystem HOL-λI of HOL-λ is the system obtained by removing the excluded-middle rule (EM) from its proof system.

$$\frac{}{\Gamma \vdash A}\ \text{axiom} \qquad \frac{}{\Gamma \vdash A \mathbin{\dot{\vee}} \dot{\neg} A}\ \text{EM} \qquad \frac{}{\Gamma \vdash \dot{\top}}\ \dot{\top}\text{-intro} \qquad \frac{\Gamma \vdash \dot{\bot}}{\Gamma \vdash A}\ \dot{\bot}\text{-elim} \qquad \frac{\Gamma \vdash A \mathbin{\dot{\wedge}} B}{\Gamma \vdash A}\ \dot{\wedge}\text{-elim}$$

$$\frac{\Gamma \vdash A \mathbin{\dot{\wedge}} B}{\Gamma \vdash B}\ \dot{\wedge}\text{-elim} \qquad \frac{\Gamma \vdash B \quad \Gamma \vdash A}{\Gamma \vdash A \mathbin{\dot{\wedge}} B}\ \dot{\wedge}\text{-intro} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \mathbin{\dot{\vee}} B}\ \dot{\vee}\text{-intro} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \mathbin{\dot{\vee}} B}\ \dot{\vee}\text{-intro}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \mathbin{\dot{\Rightarrow}} B}\ \dot{\Rightarrow}\text{-intro} \qquad \frac{\Gamma \vdash A \mathbin{\dot{\Rightarrow}} B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \dot{\Rightarrow}\text{-elim} \qquad \frac{\Gamma \vdash (A\ t)\!\downarrow}{\Gamma \vdash \dot{\exists}_T A}\ \dot{\exists}\text{-intro} \qquad \frac{\Gamma \vdash \dot{\forall}_T A}{\Gamma \vdash (A\ t)\!\downarrow}\ \dot{\forall}\text{-elim}$$

$$\frac{\Gamma \vdash A \mathbin{\dot{\vee}} B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \dot{\vee}\text{-elim} \qquad \frac{\Gamma, A \vdash \dot{\bot}}{\Gamma \vdash \dot{\neg} A}\ \dot{\neg}\text{-intro} \qquad \frac{\Gamma \vdash \dot{\neg} A \quad \Gamma \vdash A}{\Gamma \vdash \dot{\bot}}\ \dot{\neg}\text{-elim}$$

$$\frac{\Gamma \vdash (A\ x)\!\downarrow \quad x \notin fv(\Gamma)}{\Gamma \vdash \dot{\forall}_T A}\ \dot{\forall}\text{-intro} \qquad \frac{\Gamma \vdash \dot{\exists}_T A \quad \Gamma, (A\ x)\!\downarrow \vdash B \quad x \notin fv(\Gamma, B)}{\Gamma \vdash B}\ \dot{\exists}\text{-elim}$$

Figure 10 Rules of the HOL-λ proof system.

$$\begin{array}{ccc} |\iota| = \iota & |x|_c = x & |\dot{C}|_c = C \\ |o| = o & |t\ u|_c = |t|_c\ |u|_c & |c|_c = \dot{c} \\ |T \to U| = |T| \rightsquigarrow |U| & |\lambda x.\ t|_c = \lambda x : \widetilde{T}.\,|t|_c & |\dot{Q}_T|_c = Q \end{array}$$

Figure 11 Shallow embedding of HOL-λI types and terms in the expression of constructive STT, where $C \in \{\Rightarrow, \wedge, \vee, \top, \bot, \neg\}$, $c \in \mathcal{L}$, $Q \in \{\forall, \exists\}$, and x is a HOL-λ variable of type T.

5.2 Soundness and conservativity of Constructive STT

Let us establish the direct correspondance between Constructive STT and HOL-λI.

5.2.1 Soundness of Constructive STT

We choose a shallow expression of HOL-λI in Constructive STT via a translation $|.|_c$ represented on Figure 11. This translation preserves λ-abstractions, applications, and β-conversion. We express simple types in $\lambda\Pi / \equiv$ using the translation $|\cdot|$ shown in Figure 11. We denote by $\widetilde{T}$ the normal form of El $|T|$, where T is a simple type. This notation allows to use normalized type anotations in order to map normal HOL-λI terms to normal terms of Constructive STT. Finally, we define a context $\Delta_{\mathcal{L}}$ declaring a symbol $\dot{c} : \widetilde{T}$ for every constant $c \in \mathcal{L}$ of type T and a witness w : El ι.

▶ **Lemma 25** (Preservation of $\hookrightarrow_\beta$). *If $t \hookrightarrow_\beta t'$ in HOL-λI, then $|t|_c \hookrightarrow_\beta |t'|_c$ in $\lambda\Pi / \equiv$.*

Proof. By the shallowness of translation $|\cdot|_c$. ◀

▶ **Corollary 26** (Preservation of β-conversion). *If t and t' are two convertible HOL-λI terms, then $|t|_c \equiv_{\beta\mathcal{R}^c_{HO}} |t'|_c$.*

▶ **Lemma 27** (Preservation of normality). *If t is a HOL-λI term, then t is β-normal if and only if $|t|_c$ is normal in Constructive STT.*

Proof. The direct implication is immediate by using Lemma 25. The converse statement is established by a straightforward induction on t. ◀

▶ **Lemma 28** (Preservation of types). *Let t be a HOL-λI term of type T, and $x_1, \ldots, x_n$ its free variables of respective types $T_1, \ldots, T_n$. We can type $|t|_c$ with: $\Delta_{\mathcal{L}}, x_1 : \mathsf{El}\ |T_1|, \ldots, x_n : \mathsf{El}\ |T_n| \vdash^c_{HO} |t|_c : \mathsf{El}\ |T|$.*

Proof. By induction over the proof of typability of term t. ◀

Lemma 28 notably entails that for any HOL-λI proposition A, the $\lambda\Pi / \equiv$ term $\mathsf{Prf}\ |A|_c$ is well-typed in any context declaring the free variables of A. Note that using the witness $w : \mathsf{El}\ \iota$, every type $|T|$, where T is a HOL-λI simple type, has an element $w(T)$ in Constructive STT:

- if $T = \iota$, then w has type $\mathsf{El}\ \iota$.
- if $T = o$, then $\top$ has type $\mathsf{El}\ o \equiv_{\beta\mathcal{R}^c_{HO}} \mathsf{Prop}$.
- if $T = T_1 \to T_2$, then $\lambda x : \mathsf{El}\ |T_1|.\ w(T_2)$ has type $\mathsf{El}\ |T| \equiv_{\beta\mathcal{R}^c_{HO}} \mathsf{El}\ |T_1| \to \mathsf{El}\ |T_2|$.

We define a transformation from HOL-λI contexts into $\lambda\Pi / \equiv$ contexts: if $\Gamma = A_1, \ldots, A_n$ is a HOL-λI context and A is a HOL-λI proposition, denoting by $y_1, \ldots, y_k$ the free variables of Γ and A, of respective types $T_1, \ldots, T_k$, then $|\Gamma|^A_c = y_1 : \mathsf{El}\ |T_1|, \ldots, y_k : \mathsf{El}\ |T_k|, x_1 : \mathsf{Prf}\ |A_n|_c, \ldots, x_n : \mathsf{Prf}\ |A_n|_c$. Observe that contrary to the case of predicate logic, some of the free variables may be of type $\mathsf{El}\ o \equiv_{\beta\mathcal{R}^c_{HO}} \mathsf{Prop}$.

▶ **Lemma 29** (Soundness). *If $\Gamma \vdash A$ is provable in HOL-λI, then there is a term $t \in \Lambda(\Sigma^c_{HO})$ such that $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} t : \mathsf{Prf}\ |A|_c$.*

Proof. By induction on the derivation of $\Gamma \vdash A$. We develop the cases of the introduction and elimination of the universal quantifier.

Rule $\dot{\forall}$-elim: By induction hypothesis, there is t_A such that $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} t_A : \mathsf{Prf}\ |\dot{\forall}_T A|_c$. By weakening and the application rule, $\Delta_{\mathcal{L}}, |\Gamma|^{(A\ t)}_c \vdash^c_{HO} (t_A\ t) : \mathsf{Prf}\ |A\ t|_c$. By Corollary 26 and the conversion rule, $\Delta_{\mathcal{L}}, |\Gamma|^{(A\ t)}_c \vdash^c_{HO} t_A : \mathsf{Prf}\ |(A\ t){\downarrow}|_c$. By the substitution lemma [40, Lemma 2.6.9.] applied to every variable of $fv(A\ t) \backslash fv(\Gamma, (A\ t){\downarrow})$ and witnesses of the associated simples types, $\Delta_{\mathcal{L}}, |\Gamma|^{(A\ t){\downarrow}}_c \vdash^c_{HO} (t_A\ t) : \mathsf{Prf}\ |(A\ t){\downarrow}|_c$.

Rule $\dot{\forall}$-intro: By induction hypothesis, there is a term t_A such that $\Delta_{\mathcal{L}}, |\Gamma|^{(A\ x){\downarrow}}_c \vdash^c_{HO} t_A : \mathsf{Prf}\ |(A\ x){\downarrow}|_c$. By weakening and the conversion rule, $\Delta_{\mathcal{L}}, |\Gamma|^{(A\ x)}_c \vdash^c_{HO} t_A : \mathsf{Prf}\ |A\ x|_c$. As $x \notin fv(\Gamma)$, the variable x is not declared in the context $|\Gamma|^A_c$ and we can apply the abstraction rule to the typing judgment $\Delta_{\mathcal{L}}, |\Gamma|^{A\ x}_c \vdash^c_{HO} t_A : \mathsf{Prf}\ |A\ x|_c$ to obtain a derivation of $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} \lambda x : \mathsf{El}\ |T|.\ t_A : \mathsf{Prf}\ (\forall\ |T|\ |A|_c)$. ◀

.

5.2.2 Conservativity of Constructive STT

The conservativity proof developed in this section is similar to other proofs of conservativity of higher order logics expressed in $\lambda\Pi / \equiv$ [10, 2]. Notably, we heavily rely on the normalization of Ecumenical STT; we consider normal forms to constrain the form of $\lambda\Pi / \equiv$ terms to establish the preliminary Lemmas 30–35.

▶ **Lemma 30** (Normal proof types). *If A is a HOL-λI proposition, then $(\mathsf{Prf}\ |A|_c){\downarrow}$ has the form $(x_1 : M_1) \to \ldots \to (x_n : M_n) \to \mathsf{Prf}\ M$ for some terms $M, M_1, \ldots, M_n$.*

▶ **Lemma 31** (Normal simple types). *If T is a simple type, the normal form of the term $\widetilde{T}$ is in the set inductively defined by $\mathsf{Prop} \mid \mathsf{El}\ s \mid \widetilde{T_1} \to \widetilde{T_2}$, where $s : \mathsf{Set}$ and $s \not\equiv_{\beta\mathcal{R}^c_{HO}} o$.*

▶ **Lemma 32** (Conserving hypotheses). *Let A and B be HOL-λI propositions and x a variable or constant. If $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} x : (x_1 : T_1) \to \ldots \to (x_n : T_n) \to \mathsf{Prf}\ |B|_c$, then there is $C \in \Gamma$ such that $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} x : \mathsf{Prf}\ |C|_c$.*

▶ **Lemma 33** (Conserving simple types). *Let A be a HOL-λI proposition and $u \in \Lambda(\Sigma^c_{HO})$ a normal term. If $\Delta_{\mathcal{L}}, |\Gamma|^A_c \vdash^c_{HO} u : \mathsf{Set}$, then there is a simple type T such that $u = |T|$.*

▶ **Lemma 34** (Conserving objects). *Let A be a HOL-λI proposition, T simple type, and u a normal term in Constructive STT. If $\Delta_{\mathscr{L}}, |\Gamma|_c^A \vdash_{HO}^c u : \mathsf{El}\ |T|$, then there is a normal HOL-λI term v of type T such that $u \equiv_{\beta\mathscr{R}_{HO}^c} |v|_c$.*

▶ **Lemma 35** (Weak conservativity). *Let A be a normal HOL-λI proposition. If there is a normal term t such that $|\Gamma|_c^A \vdash_{HO}^c t : \mathsf{Prf}\ |A|_c$, then $\Gamma \vdash A$ is provable in HOL-λI.*

Proof. By induction on term t. As t is normal, there are only two cases to consider.

If $u = x\ u_1 \dots u_n$, we prove by induction on k that for any $k \in \{0, \dots, n\}$, there is a normal HOL-λI proposition A_k such that $|\Gamma|_c^A \vdash_{HO}^c x\ u_1 \dots u_k : \mathsf{Prf}\ |A_k|_c$ and $\Gamma \vdash A_k$ in HOL-λI.

If $k = 0$, then there are terms $M_1, \dots, M_n$ such that $\Delta_{\mathscr{L}}, |\Gamma|_c^A \vdash_{HO}^c x : (x_1 : M_1) \to \dots \to (x_1 : M_n) \to \mathsf{Prf}\ |A|_c$. By Lemma 32, there is $C \in \Gamma$ such that $x : \mathsf{Prf}\ |C|_c$, and as a consequence $\Delta_{\mathscr{L}}, |\Gamma|_c^A \vdash_{HO}^c x : \mathsf{Prf}\ |C|_c$ and $\Gamma \vdash C$ in HOL-λI by the axiom rule.

If $0 < k \leq n$, by induction hypothesis there is a HOL-λI proposition A_{k-1} such that $x\ u_1 \dots u_{k-1}$ has type $\mathsf{Prf}\ |A_{k-1}|_c$ in context $\Delta_{\mathscr{L}}, |\Gamma|_c^A$ and $\Gamma \vdash A_{k-1}$ in HOL-λI. By inversion, there are M_k and N_k such that $\mathsf{Prf}\ |A_{k-1}|_c \equiv_{\beta\mathscr{R}_{HO}^c} (y : M_k) \to N_k$ and u_k has type M_k in context $\Delta_{\mathscr{L}}, |\Gamma|_c^A$.
Let $\widetilde{A}_{k-1}$ be the $\beta\mathscr{R}_{HO}^c$-normal form of $|A_{k-1}|_c$. Given the form of $\mathsf{Prf}\ \widetilde{A}_{k-1}$, we know that $\widetilde{A}_{k-1}$ has a head connective; as $|\cdot|_c$ is shallow, A_{k-1} also has a head connective. We proceed by case disjunction on the head connective of A_{k-1}. Here, we develop the case of the universal quantifier, *i.e.* $A_{k-1} = \dot{\forall}_T B_k$ and $\mathsf{Prf}\ |A_{k-1}|_c \equiv_{\beta\mathscr{R}_{HO}^c} (y : \mathsf{El}\ |T|) \to \mathsf{Prf}\ (|B_k|_c\ y)$. By product compatibility [40, Definition 2.4.5], $M_k \equiv_{\beta\mathscr{R}_{HO}^c} \mathsf{El}\ |T|$ and $N_k \equiv_{\beta\mathscr{R}_{HO}^c} \mathsf{Prf}\ (|B_k|_c\ y)$. By Lemma 34, there is a HOL-λI term v_k of type T such that $u_k \equiv_{\beta\mathscr{R}_{HO}^c} |v_k|_c$. Using Corollary 26 and conversion, $x\ u_1 \dots u_k$ has type $\mathsf{Prf}\ (|(B_k\ v_k){\downarrow}\,|_c)$ in context $\Delta_{\mathscr{L}}, |\Gamma|_c^A$. Finally, $(B_k\ v_k){\downarrow}$ is provable in context Γ by an application of $\dot{\forall}$-elim.

By induction, there is a normal HOL-λI proposition A_n such that $\Delta_{\mathscr{L}}, |\Gamma|_c^A \vdash_{HO}^c t : \mathsf{Prf}\ |A_n|_c$ and $\Gamma \vdash A$. By the uniqueness of types [40, Theorem 2.6.25.], $|A_n|_c \equiv_{\beta\mathscr{R}_{HO}^c} |A|_c$. However, by Lemma 27, $|A|_c$ and $|A_n|_c$ are normal. By confluence, $|A_n|_c = |A|_c$. The embedding $|.|_c$ is injective, so $A_n = A$ and we can conclude that $\Gamma \vdash A$ in HOL-λI.

If $u = \lambda x : M.\, u_0$, then $\mathsf{Prf}\ |A|_c$ is convertible to a product $(x : M_1) \to M_2$. Using the same reasoning as in the previous case, we deduce that proposition A has a head connective and once again develop the case of the universal quantification, *i.e.* $A = \dot{\forall}_T B$ and $\mathsf{Prf}\ |A|_c \equiv_{\beta\mathscr{R}_{HO}^c} (x : \mathsf{El}\ |T|) \to \mathsf{Prf}\ (|B|_c\ y)$. By product compatibility [40, Definition 2.4.5], $M_1 \equiv_{\beta\mathscr{R}_{HO}^c} \mathsf{El}\ |T|$ and $M_2 \equiv_{\beta\mathscr{R}_{HO}^c} \mathsf{Prf}\ (|B|_c\ x)$. By inversion, $\Delta_{\mathscr{L}}, |\Gamma|_c^A, x : \mathsf{El}\ |T| \vdash_{HO}^c u_0 : \mathsf{Prf}\ (|B|_c\ x)$ with $x \notin fv(\Gamma, A)$. By Corollary 26, conversion, and substitution, $\Delta_{\mathscr{L}}, |\Gamma|_c^{(B\ x)\downarrow} \vdash_{HO}^c u_0 : \mathsf{Prf}\ (|(B\ x){\downarrow}\,|_c)$. By induction hypothesis, $\Gamma \vdash (B\ x){\downarrow}$. Finally, A is provable in context Γ using $\dot{\forall}$-intro. ◀

Using the weak normalization of Constructive STT we finally establish conservativity.

▶ **Corollary 36** (Conservativity). *Let A be a normal HOL-λI proposition. If there is a term t such that $\Delta_{\mathscr{L}}, |\Gamma|_c^A \vdash_{HO}^c t : \mathsf{Prf}\ |A|_c$, then $\Gamma \vdash A$ is provable in HOL-λI.*

5.3 Soundness and conservativity of Ecumenical STT

Here, we show that the Ecumenical STT has a similar expressivity to the HOL-λ system. Similarly to the first order case, we use the soundness and conservativity of Constructive STT with respect to HOL-λI and the properties of the translations by double negation.

$$\begin{array}{r} |\dot{\bowtie}|_e = \bowtie_c \\ |\dot{Q}|_e = Q_c\ |T|_c \\ |t|_e = |t|_c \text{ else} \end{array} \qquad \begin{array}{rcl} x^\perp & = & x \\ (t\ u)^\perp & = & t^\perp\ u^\perp \\ (\lambda x.\ t)^\perp & = & \lambda x.\ t^\perp \end{array} \qquad \begin{array}{rcl} \dot{\bowtie}^\perp & = & \lambda x_1.\ \lambda x_2.\ (\dot{\neg}\dot{\neg}x_1)\dot{\bowtie}(\dot{\neg}\dot{\neg}x_2) \\ \dot{Q}^\perp & = & \lambda f.\ \dot{Q}(\lambda y.\ \dot{\neg}\dot{\neg}(f\ y)) \\ t^\perp & = & t \text{ in all other cases} \end{array}$$

(a) Defining $|\cdot|_e$. **(b)** Higher order $\perp$-translation.

Figure 12 Translations of HOL-λ terms where $\bowtie \in \{\wedge, \vee, \Rightarrow\}$, $Q \in \{\forall, \exists\}$, and x_1, x_2, f, and y are HOL-λ variables of respective types o, o, $T \to o$, and T.

Formally, we define the embedding $|.|_e$ inductively over HOL-λ terms as shown in Figure 12a. Similarly to the constructive case, transformation $|.|_e$ preserves convertibility and types, and is extended to HOL-λ contexts. However, this transformation does not map normal terms to normal terms: for example $(z\dot{\wedge}z)^\perp = [\lambda x_1.\ \lambda x_2.\ (\dot{\neg}\dot{\neg}x_1)\dot{\wedge}(\dot{\neg}\dot{\neg}x_2)]\ z\ z$ which is a β-redex. The proof requires a straightforward extension of the double negation translation $.^\perp$, represented on Figure 12b and established by Lemma 38.

▶ **Lemma 37.** *If A is convertible to B in HOL-λ, then $A^\perp$ is convertible to $B^\perp$ in HOL-λ.*

Proof. The only rule in HOL-λ is β-reduction and transformation $.^\perp$ acts as a morphism over abstractions and applications. ◀

▶ **Lemma 38.** *$\Gamma \vdash A$ in HOL-λ if and only if $\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow} \vdash \dot{\neg}\dot{\neg}A^\perp{\downarrow}$ in HOL-λI.*

Proof. The forward implication is proven by induction on the derivation of $\Gamma \vdash A$. The backwards implication holds immediately, as every normal proposition A is provably equivalent to $\dot{\neg}\dot{\neg}A$ in HOL-λ, and HOL-λ is an extension of HOL-λI. ◀

▶ **Lemma 39** (Soundness). *If $\Gamma \vdash A$ is provable in HOL-λI, then there is a term t such that $|\Gamma|_e^A \vdash_{HO}^e t : Prf_c\ |A|_e$ is derivable.*

Proof. By Lemmas 29 and 38, $\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow} \vdash \dot{\neg}\dot{\neg}A^\perp{\downarrow}$ in HOL-λI and there is t such that $\Delta_{\mathscr{L}}, |\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow}|_c^{\dot{\neg}\dot{\neg}A^\perp{\downarrow}} \vdash_{HO}^e t : \mathrm{Prf}\ |\dot{\neg}\dot{\neg}A^\perp{\downarrow}|_c$. By conversion, $\Delta_{\mathscr{L}}, |\Gamma|_e^A \vdash_{HO}^e t : \mathrm{Prf}_c\ |A|_e$. ◀

▶ **Lemma 40** (Conservativity). *If there is a term t such that $\Delta_{\mathscr{L}}, |\Gamma|_e^A \vdash_{HO}^e t : Prf_c\ |A|_e$, then $\Gamma \vdash A$ is provable in HOL-λ.*

Proof. If there is a term t such that $|\Gamma|_e^A \vdash_{HO}^e t : \mathrm{Prf}_c\ |A|_e$ where A is a normal HOL-λ formula, then by conversion and weakening, $|\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow}\ |_c^{\dot{\neg}\dot{\neg}A^\perp{\downarrow}} \vdash_{HO}^e t : \mathrm{Prf}\ |\dot{\neg}\dot{\neg}A^\perp{\downarrow}\ |_c$. As t and $|\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow}\ |_c^{\dot{\neg}\dot{\neg}A^\perp{\downarrow}}$ are respectively a term and a typing context of Constructive STT, we conclude by Corollary 3 that $|\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow}\ |_c^{\dot{\neg}\dot{\neg}A^\perp{\downarrow}} \vdash_{HO}^e t : \mathrm{Prf}\ |\dot{\neg}\dot{\neg}A^\perp{\downarrow}\ |_c$. By conservativity of Constructive STT, $\dot{\neg}\dot{\neg}\Gamma^\perp{\downarrow} \vdash \dot{\neg}\dot{\neg}A^\perp{\downarrow}$ in HOL-λI. By Lemma 38, $\Gamma \vdash A$ in HOL-λ. ◀

▶ **Corollary 41** (Consistency). *There is no derivation of $\vdash_{HO}^e \mathrm{Prf}\ \perp$ in Ecumenical STT.*

6 Conclusion

In this paper, we have studied the logical fragments of theory $\mathcal{U}$ and established their normalization, consistency, decidability of type-checking, soundness and conservativity. These results comfort the enterprise of using theory $\mathcal{U}$ to store, recheck, translate, and hybridize proofs from various proof assistants. The extension of these results, notably normalization and consistency, to the entirety of theory $\mathcal{U}$ is still an open question.

Another open question is the behaviour of hybrid propositions and proofs in Ecumenical STT, *i.e.* propositions and proofs mixing constructive and classical connectives. Given the distinct design choices made in Ecumenical STT and preexisting ecumenical systems, hybrid objects may not behave similarly. Some results over hybrid objects can however already be deduced by normalization and properties of Constructive STT, as the normal forms of hybrid objects are in this fragment.

Finally, the implementation of constructivization algorithms in theory $\mathcal{U}$ could further improve the interoperability between classical and constructive proofs. A first candidate would be the standard library of HOL Light, already translated in theory $\mathcal{U}$ [12], which could be partially exported to constructive proof assistants such as Coq, Agda, or Matita.

References

1 Claus Appel, Vincent van Oostrom, and Jakob Grue Simonsen. Higher-order (non-)modularity. In Christopher Lynch, editor, *Proceedings of the 21st International Conference on Rewriting Techniques and Applications, RTA 2010, July 11-13, 2010, Edinburgh, Scottland, UK*, volume 6 of *LIPIcs*, pages 17–32. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.RTA.2010.17.

2 Ali Assaf. *A framework for defining computational higher-order logics. (Un cadre de définition de logiques calculatoires d'ordre supérieur)*. PhD thesis, École Polytechnique, Palaiseau, France, 2015. URL: https://tel.archives-ouvertes.fr/tel-01235303.

3 Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti : a Logical Framework based on the $\lambda\Pi$-Calculus Modulo Theory. Manuscript, 2016.

4 Frédéric Blanqui. *Théorie des types et réécriture. (Type theory and rewriting)*. PhD thesis, University of Paris-Sud, Orsay, France, 2001. URL: https://tel.archives-ouvertes.fr/tel-00105522.

5 Frédéric Blanqui. Size-based termination of higher-order rewriting. *Journal of Functional Programming*, 28:e11, 2018. doi:10.1017/S0956796818000072.

6 Frédéric Blanqui, Gilles Dowek, Émilie Grienenberger, Gabriel Hondet, and François Thiré. Some axioms for mathematics. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 20:1–20:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.20.

7 Frédéric Blanqui, Guillaume Genestier, and Olivier Hermant. Dependency pairs termination in dependent type theory modulo rewriting. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, volume 131 of *LIPIcs*, pages 9:1–9:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSCD.2019.9.

8 Raphaël Cauderlier and Catherine Dubois. ML pattern-matching, recursion, and rewriting: From focalize to dedukti. In Augusto Sampaio and Farn Wang, editors, *Theoretical Aspects of Computing – ICTAC 2016 – 13th International Colloquium, Taipei, Taiwan, ROC, October 24-31, 2016, Proceedings*, volume 9965 of *Lecture Notes in Computer Science*, pages 459–468, 2016. doi:10.1007/978-3-319-46750-4_26.

9 Raphaël Cauderlier and Catherine Dubois. Focalize and dedukti to the rescue for proof interoperability. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *Interactive Theorem Proving – 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*, pages 131–147. Springer, 2017. doi:10.1007/978-3-319-66107-0_9.

10 Denis Cousineau and Gilles Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In Simona Ronchi Della Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2007. doi:10.1007/978-3-540-73228-0_9.

11 Dag Prawitz. *Natural deduction, a proof-theoretical study.* PhD thesis, Stockolm: Almqvist & Wicksell, 1965.

12 Deducteam. Nubo, repository of interoperable formal proofs. `https://deducteam.gitlabpages.inria.fr/nubo/index.html`, 2020. Accessed: 2022-29-11.

13 Alexis Dorra. Équivalence de Curry-Howard entre le lambda-Pi-calcul et la logique intuitionniste. Bachelor internship report, LIX École Polytechnique, 2011.

14 Gilles Dowek. On the definition of the classical connectives and quantifiers. *CoRR*, 2015. `arXiv:1601.01782`.

15 Gilles Dowek. Models and termination of proof reduction in the lambda pi-calculus modulo theory. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 109:1–109:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPIcs.ICALP.2017.109`.

16 Gilles Dowek, Thérèse Hardin, and Claude Kirchner. HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11(1):1–25, 2001.

17 Gilles Dowek and Benjamin Werner. Proof normalization modulo. *J. Symb. Log.*, 68(4):1289–1316, 2003. `doi:10.2178/jsl/1067620188`.

18 Thiago Felicissimo. Adequate and computational encodings in the logical framework dedukti. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 25:1–25:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.FSCD.2022.25`.

19 Guillaume Genestier. Encoding Agda Programs Using Rewriting. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*, volume 167 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.FSCD.2020.31`.

20 Herman Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In Peter Dybjer, Bengt Nordström, and Jan M. Smith, editors, *Types for Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers*, volume 996 of *Lecture Notes in Computer Science*, pages 14–38. Springer, 1994. `doi:10.1007/3-540-60579-7_2`.

21 Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings*, volume 3452 of *Lecture Notes in Computer Science*, pages 301–331. Springer, 2004. `doi:10.1007/978-3-540-32275-7_21`.

22 Jean-Yves Girard. On the unity of logic. *Ann. Pure Appl. Logic*, 59(3):201–217, 1993. `doi:10.1016/0168-0072(93)90093-S`.

23 Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types.* Cambridge Tracts in Theoretical Computer Science, 7, 1989.

24 Mohamed Yacine El Haddad, Guillaume Burel, and Frédéric Blanqui. EKSTRAKTO A tool to reconstruct dedukti proofs from TSTP files (extended abstract). In Giselle Reis and Haniel Barbosa, editors, *Proceedings Sixth Workshop on Proof eXchange for Theorem Proving, PxTP 2019, Natal, Brazil, August 26, 2019*, volume 301 of *EPTCS*, pages 27–35, 2019. `doi:10.4204/EPTCS.301.5`.

25 Robert Harper, Furio Honsell, and Gordon D. Plotkin. A framework for defining logics. In *Proceedings of the Symposium on Logic in Computer Science (LICS '87), Ithaca, New York, USA, June 22-25, 1987*, pages 194–204. IEEE Computer Society, 1987.

26 Gabriel Hondet and Frédéric Blanqui. Encoding of predicate subtyping with proof irrelevance in the λΠ-calculus modulo theory. In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs, TYPES 2020, March 2-5, 2020, University of Turin, Italy*, volume 188 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.TYPES.2020.6`.

27 Alfred Horn. Logic with truth values in a linearly ordered heyting algebra. *J. Symb. Log.*, 34(3):395–408, 1969. `doi:10.2307/2270905`.

28 Jean-Pierre Jouannaud and Jianqi Li. Termination of Dependently Typed Rewrite Rules. In Thorsten Altenkirch, editor, *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)*, volume 38 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 257–272, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.TLCA.2015.257`.

29 Andrei Nikolaevich Kolmogorov. On the principle of the excluded middle. In *Matematicheskij Sbornik*, volume 32, pages 646–667, 1925.

30 C.L.M. Kop. *Higher Order Termination: Automatable Techniques for Proving Termination of Higher-Order Term Rewriting Systems*. PhD thesis, Vrije Universiteit Amsterdam, 2012. Naam instelling promotie: VU Vrije Universiteit Naam instelling onderzoek: VU Vrije Universiteit.

31 Keiichirou Kusakari and Masahiko Sakai. Enhancing dependency pair method using strong computability in simply-typed term rewriting. *Appl. Algebra Eng., Commun. Comput.*, 18(5):407–431, October 2007.

32 Chin Lee, Neil Jones, and Amir Ben-Amram. The size-change principle for program termination. *ACM SIGPLAN Notices*, 36, January 2001. `doi:10.1145/360204.360210`.

33 Chuck Liang and Dale Miller. Unifying classical and intuitionistic logics for computational control. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 283–292. IEEE Computer Society, 2013. `doi:10.1109/LICS.2013.34`.

34 Paqui Lucio. Structured sequent calculi for combining intuitionistic and classical first-order logic. In Hélène Kirchner and Christophe Ringeissen, editors, *Frontiers of Combining Systems, Third International Workshop, FroCoS 2000, Nancy, France, March 22-24, 2000, Proceedings*, volume 1794 of *Lecture Notes in Computer Science*, pages 88–104. Springer, 2000. `doi:10.1007/10720084_7`.

35 Paul-André Melliès and Benjamin Werner. A generic normalisation proof for pure type systems. In Eduardo Giménez and Christine Paulin-Mohring, editors, *Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers*, volume 1512 of *Lecture Notes in Computer Science*, pages 254–276. Springer, 1996. `doi:10.1007/BFb0097796`.

36 Luiz Carlos Pereira and Ricardo Oscar Rodriguez. Normalization, soundness and completeness for the propositional fragment of prawitz' ecumenical system. In *Revista Portuguesa De Filosofia 73, no. 3/4*, pages 1153–1168, 2017.

37 Elaine Pimentel, Luiz Carlos Pereira, and Valeria de Paiva. An ecumenical notion of entailment. *Synthese*, pages 1–23, 2019. `doi:10.1007/s11229-019-02226-5`.

38 Dag Prawitz. Classical versus intuitionnistic logic. In Edward Hermann Haeusler, Wagner de Campos Sanz, Bruno Lopes, and College Publications, editors, *Why is this a proof ? Festschrift for Luiz Carlos Pereira*, pages 15–32, 2015.

39 Raphaël Cauderlier. Sigmaid. `https://gitlab.math.univ-paris-diderot.fr/cauderlier/sigmaid`, 2014. Accessed: 2022-29-11.

40 Ronan Saillard. *Typechecking in the lambda-Pi-Calculus Modulo : Theory and Practice. (Vérification de typage pour le lambda-Pi-Calcul Modulo : théorie et pratique)*. PhD thesis, Mines ParisTech, France, 2015. URL: `https://tel.archives-ouvertes.fr/tel-01299180`.

41 William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, 32(2):198–212, 1967. `doi:10.2307/2271658`.

42 Terese. *Term rewriting systems*, volume 55 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2003.

43 François Thiré. Sharing a Library between Proof Assistants: Reaching out to the HOL Family. *Electronic Proceedings in Theoretical Computer Science*, 274:57–71, July 2018. `doi:10.4204/EPTCS.274.5`.

44 François Thiré. *Interoperability between proof systems using the logical framework Dedukti. (Interopérabilité entre systèmes de preuve en utilisant le cadre logique Dedukti)*. PhD thesis, École normale supérieure Paris-Saclay, Cachan, France, 2020. URL: `https://tel.archives-ouvertes.fr/tel-03224039`.