Kuroda's Translation for the $\lambda\Pi$-Calculus Modulo Theory and Dedukti

Thomas Traversié
Université Paris-Saclay, CentraleSupélec, MICS
Gif-sur-Yvette, France
Université Paris-Saclay, Inria, CNRS, ENS-Paris-Saclay, LMF
Gif-sur-Yvette, France
thomas.traversie@centralesupelec.fr

Kuroda's translation embeds classical first-order logic into intuitionistic logic, through the insertion of double negations. Recently, Brown and Rizkallah extended this translation to higher-order logic. In this paper, we adapt it for theories encoded in higher-order logic in the $\lambda\Pi$-calculus modulo theory, a logical framework that extends λ-calculus with dependent types and user-defined rewrite rules. We develop a tool that implements Kuroda's translation for proofs written in DEDUKTI, a proof language based on the $\lambda\Pi$-calculus modulo theory.

1 Introduction

The $\lambda\Pi$-calculus modulo theory [6] is an extension of simply typed λ-calculus with dependent types and user-defined rewrite rules. It is a logical framework, meaning that one can express many theories in it—through the definitions of typed constants and rewrite rules. For instance, it is possible to encode Predicate Logic, Simple Type Theory and the Calculus of Constructions in the $\lambda\Pi$-calculus modulo theory [2]. In particular, theories from other proof systems can be expressed inside this logical framework [20]. The $\lambda\Pi$-calculus modulo theory has been implemented in the concrete language DEDUKTI [1, 15]. Besides automatic proof checking, DEDUKTI can be used as a common language to exchange proofs between different systems. However, if one wants to translate proofs from the *classical* proof assistant HOL LIGHT to the *intuitionistic* proof assistant COQ *via* DEDUKTI, one must transform classical proofs into intuitionistic proofs *inside* DEDUKTI.

Classical logic corresponds to intuitionistic logic extended with the principle of excluded middle $A \vee \neg A$, or equivalently the double-negation elimination $\neg\neg A \Rightarrow A$. Classical logic can be embedded into intuitionistic logic, using double-negations translations. Glivenko [12] proved that any propositional formula A is provable in classical logic if and only if its double negation $\neg\neg A$ is provable in intuitionistic logic. Kolmogorov [17], Gödel [13], Gentzen [10] and Kuroda [18] developed double-negation translations $A \mapsto A^*$, which transforms any first-order formula A such that:

(i) if A is provable in classical logic then its translation A^* is provable in intuitionistic logic,

(ii) A and A^* are classically equivalent.

More recently, Brown and Rizkallah [4] showed that Kolmogorov's and Gödel-Gentzen's translations cannot be extended to higher-order logic. They proved that, in higher-order logic, Kuroda's translation satisfies Property (i), but that it fails in the presence of functional extensionality. In fact [21], Property (i) holds in the presence of functional extensionality under some specific condition, and Property (ii) holds when assuming functional extensionality and propositional extensionality.

Rabe, Sacerdoti Coen (Eds): LFMTP 2024
EPTCS ??, 2024, pp. 35–48, doi:10.4204/EPTCS.??.3

Contribution. In this paper, we express Kuroda's translation for theories of the $\lambda\Pi$-calculus modulo theory that are encoded in higher-order logic. It is both an encoding—into a logical framework that features proofs as terms—and an extension—to a logical framework that features dependent types and user-defined rewrite rules—of Kuroda's translation. We implement such translation inside CONSTRUKTI, a tool that translates DEDUKTI files. CONSTRUKTI is tested on a benchmark of a hundred formal proofs. This tool and this benchmark are available at `https://github.com/Deducteam/Construkti`.

Outline of the paper. In Section 2, we present the $\lambda\Pi$-calculus modulo theory and we detail an encoding of higher-order logic in it. In Section 3, we define Kuroda's translation for theories of $\lambda\Pi$-calculus modulo theory that are encoded in higher-order logic, and we prove the embedding of classical logic into intuitionistic logic. In Section 4, we implement CONSTRUKTI and test it on DEDUKTI proofs.

2 Higher-Order Logic in the $\lambda\Pi$-Calculus Modulo Theory

In this section, we present the $\lambda\Pi$-calculus modulo theory, and we detail an encoding of higher-order logic in this logical framework. We characterize the theories considered in the rest of this paper—theories encoded in higher-order logic.

2.1 The $\lambda\Pi$-Calculus Modulo Theory

The Edinburgh Logical Framework [14], also called $\lambda\Pi$-calculus, is an extension of simply typed λ-calculus with dependent types. The $\lambda\Pi$-calculus modulo theory [6] corresponds to the Edinburgh Logical Framework extended with user-defined rewrite rules [7]. Its syntax is given by:

Sorts	$s ::= \mathtt{TYPE} \mid \mathtt{KIND}$
Terms	$t,u,A,B ::= c \mid x \mid s \mid \Pi x : A.\, B \mid \lambda x : A.\, t \mid t\ u$
Contexts	$\Gamma ::= \langle\rangle \mid \Gamma, x : A$
Signatures	$\Sigma ::= \langle\rangle \mid \Sigma, c : A$
Rewrite systems	$\mathscr{R} ::= \langle\rangle \mid \mathscr{R}, \ell \hookrightarrow r$

where c is a constant and x is a variable (ranging over disjoint sets). TYPE and KIND are two sorts: terms of type TYPE are called types, and terms of type KIND are called kinds. $\Pi x : A.\, B$ is a dependent product (simply written $A \to B$ if x does not occur in B), $\lambda x : A.\, t$ is an abstraction, and $t\ u$ is an application. Contexts, signatures and rewrite systems are finite sequences, and are written $\langle\rangle$ when empty. Signatures Σ are composed of typed constants $c : A$, where A is a closed term (that is a term with no free variables). Rewrite systems $\mathscr{R}$ are composed of rewrite rules $\ell \hookrightarrow r$, where the head symbol of ℓ is a constant. The $\lambda\Pi$-calculus modulo theory is a logical framework, in which Σ and $\mathscr{R}$ are fixed by the users depending on the logic they are working in. The relation $\hookrightarrow_{\beta\mathscr{R}}$ is generated by β-reduction and by the rewrite rules of $\mathscr{R}$. The conversion $\equiv_{\beta\mathscr{R}}$ is the reflexive, symmetric, and transitive closure of $\hookrightarrow_{\beta\mathscr{R}}$.

The typing rules for the $\lambda\Pi$-calculus modulo theory are given in Figure 1. We write $\vdash \Gamma$ when the context Γ is well formed, and $\Gamma \vdash t : A$ when the term t is of type A in the context Γ. For convenience, $\langle\rangle \vdash t : A$ is simply written $\vdash t : A$. The standard weakening rule is admissible.

We write $\Lambda(\Sigma)$ for the set of terms whose constants belong to Σ. We say that $(\Sigma, \mathscr{R})$ is a theory when: (i) for each rule $\ell \hookrightarrow r \in \mathscr{R}$, both ℓ and r belongs to $\Lambda(\Sigma)$, (ii) $\hookrightarrow_{\beta\mathscr{R}}$ is confluent on $\Lambda(\Sigma)$, and (iii) each

$$\frac{}{\vdash \langle\rangle}\ [\textsc{Empty}] \qquad \frac{\vdash \Gamma \quad \Gamma \vdash A : s}{\vdash \Gamma, x : A}\ [\textsc{Decl}]\ x \notin \Gamma \qquad \frac{\vdash \Gamma}{\Gamma \vdash \mathtt{TYPE} : \mathtt{KIND}}\ [\textsc{Sort}]$$

$$\frac{\vdash \Gamma \quad \vdash A : s}{\Gamma \vdash c : A}\ [\textsc{Const}]\ c : A \in \Sigma \qquad \frac{\vdash \Gamma}{\Gamma \vdash x : A}\ [\textsc{Var}]\ x : A \in \Gamma$$

$$\frac{\Gamma \vdash A : \mathtt{TYPE} \quad \Gamma, x : A \vdash B : s}{\Gamma \vdash \Pi x : A.\, B : s}\ [\textsc{Prod}] \qquad \frac{\Gamma \vdash A : \mathtt{TYPE} \quad \Gamma, x : A \vdash B : s \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.\, t : \Pi x : A.\, B}\ [\textsc{Abs}]$$

$$\frac{\Gamma \vdash t : \Pi x : A.\, B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x \leftarrow u]}\ [\textsc{App}] \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s}{\Gamma \vdash t : B}\ [\textsc{Conv}]\ A \equiv_{\beta\mathscr{R}} B$$

Figure 1: Typing rules of the $\lambda\Pi$-calculus modulo theory.

rule $\ell \hookrightarrow r \in \mathscr{R}$ preserves types (for all context Γ, substitution θ, and term $A \in \Lambda(\Sigma)$, if $\Gamma \vdash \ell\theta : A$ then $\Gamma \vdash r\theta : A$).

In the $\lambda\Pi$-calculus modulo theory, if $\Gamma \vdash t : A$ then Γ is well-formed and A is well-typed. To prove this, we use the two following properties.

Lemma 1. *If $\Gamma \vdash t : A$, then either $A = \mathtt{KIND}$ or $\Gamma \vdash A : s$ for $s = \mathtt{TYPE}$ or $s = \mathtt{KIND}$. If $\Gamma \vdash \Pi x : A.\, B : s$, then $\Gamma \vdash A : \mathtt{TYPE}$.*

2.2 An Encoding of Higher-Order Logic

It is possible to express higher-order logic in the $\lambda\Pi$-calculus modulo theory [2]. For this, we have to introduce the notions of proposition and proof. We declare the constant *Set*, which represents the universe of sorts, along with the injection *El* that maps sorts to the type of its elements. The constant *Prop* defines the universe of propositions, and the injection *Prf* maps propositions into the type of its proofs. In this encoding, we say that P of type *Prop* is a proposition, that *Prf* P is a formula and that a term of type *Prf* P is a proof of P.

$Set : \mathtt{TYPE}$	$El : Set \to \mathtt{TYPE}$	$\rightsquigarrow : Set \to Set \to Set$	$o : Set$
$Prop : \mathtt{TYPE}$	$Prf : Prop \to \mathtt{TYPE}$	$El\ (x \rightsquigarrow y) \hookrightarrow El\ x \to El\ y$	$El\ o \hookrightarrow Prop$

The arrow $\rightsquigarrow$ (written infix) is used to represent function types between terms of type *Set*. Propositions are considered as objects, using the sort o and the rewrite rule $El\ o \hookrightarrow Prop$.

Now that we have introduced the notions of proposition and proof, we can define the logical connectives and quantifiers of predicate logic.

$\Rightarrow : Prop \to Prop \to Prop$	$\top : Prop$	$\forall : \Pi x : Set.\, (El\ x \to Prop) \to Prop$
$\wedge : Prop \to Prop \to Prop$	$\bot : Prop$	$\exists : \Pi x : Set.\, (El\ x \to Prop) \to Prop$
$\vee : Prop \to Prop \to Prop$	$\neg : Prop \to Prop$	$\Leftrightarrow\ : Prop \to Prop \to Prop$

Remark that $\forall$ and $\exists$ are polymorphic quantifiers that can be applied to the sort of proposition o. Hence the higher-order feature directly derives from the rewrite rule $El\ o \hookrightarrow Prop$.

In natural deduction, each connective and quantifier comes with an introduction and an elimination inference rule. The encoding of the notions of proposition and proof is well-suited for representing inference rules: logical consequences are represented by arrow types, and parameters are represented by dependent types. For instance, the inference rule for the elimination of disjunction

$$\frac{\Gamma \vdash P \vee Q \qquad \Gamma, P \vdash R \qquad \Gamma, Q \vdash R}{\Gamma \vdash R}$$

is simply expressed by the constant $\mathsf{or_e}$ of type

$$\Pi p, q : Prop.\ Prf\ (p \vee q) \to \Pi r : Prop.\ (Prf\ p \to Prf\ r) \to (Prf\ q \to Prf\ r) \to Prf\ r$$

that can be used for any context Γ. The constants representing the natural deduction rules for the logical connectives are:

$\mathsf{imp_i} : \Pi p, q : Prop.\ (Prf\ p \to Prf\ q) \to Prf\ (p \Rightarrow q)$
$\mathsf{imp_e} : \Pi p, q : Prop.\ Prf\ (p \Rightarrow q) \to Prf\ p \to Prf\ q$
$\mathsf{and_i} : \Pi p : Prop.\ Prf\ p \to \Pi q : Prop.\ Prf\ q \to Prf\ (p \wedge q)$
$\mathsf{and_{e\ell}} : \Pi p, q : Prop.\ Prf\ (p \wedge q) \to Prf\ p$
$\mathsf{and_{er}} : \Pi p, q : Prop.\ Prf\ (p \wedge q) \to Prf\ q$
$\mathsf{or_{i\ell}} : \Pi p : Prop.\ Prf\ p \to \Pi q : Prop.\ Prf\ (p \vee q)$
$\mathsf{or_{ir}} : \Pi p, q : Prop.\ Prf\ q \to Prf\ (p \vee q)$
$\mathsf{or_e} : \Pi p, q : Prop.\ Prf\ (p \vee q) \to \Pi r : Prop.\ (Prf\ p \to Prf\ r) \to (Prf\ q \to Prf\ r) \to Prf\ r$
$\mathsf{neg_i} : \Pi p : Prop.\ (Prf\ p \to Prf\ \bot) \to Prf\ (\neg p)$
$\mathsf{neg_e} : \Pi p : Prop.\ Prf\ (\neg p) \to Prf\ p \to Prf\ \bot$

For convenience, the semantic of the logical biconditional is encoded through the rewrite rule $p \Leftrightarrow q \hookrightarrow (p \Rightarrow q) \wedge (q \Rightarrow p)$. The introduction of tautology and the elimination of contradiction are encoded by:

$\mathsf{top_i} : Prf\ \top$
$\mathsf{bot_e} : Prf\ \bot \to \Pi p : Prop.\ Prf\ p$

The natural deduction rules for the quantifiers are represented by the following constants:

$\mathsf{all_i} : \Pi a : Set.\ \Pi p : El\ a \to Prop.\ (\Pi x : El\ a.\ Prf\ (p\ x)) \to Prf\ (\forall\ a\ p)$
$\mathsf{all_e} : \Pi a : Set.\ \Pi p : El\ a \to Prop.\ Prf\ (\forall\ a\ p) \to \Pi x : El\ a.\ Prf\ (p\ x)$
$\mathsf{ex_i} : \Pi a : Set.\ \Pi p : El\ a \to Prop.\ \Pi x : El\ a.\ Prf\ (p\ x) \to Prf\ (\exists\ a\ p)$
$\mathsf{ex_e} : \Pi a : Set.\ \Pi p : El\ a \to Prop.\ Prf\ (\exists\ a\ p) \to \Pi r : Prop.\ (\Pi x : El\ a.\ Prf\ (p\ x) \to Prf\ r) \to Prf\ r$

All those constants and rewrite rules define the encoding of intuitionistic higher-order logic in the $\lambda\Pi$-calculus modulo theory. We write Σ^i_{HOL} for its constants and $\mathscr{R}_{HOL}$ for its rewrite rules. The principle of excluded middle is represented by:

$\mathsf{pem} : \Pi p : Prop.\ Prf\ (p \vee \neg p)$

Classical higher-order logic is encoded in the $\lambda\Pi$-calculus modulo theory by the constants Σ^c_{HOL} (that is Σ^i_{HOL} along with pem) and by the rewrite rules $\mathscr{R}_{HOL}$.

Remark that we have decided to encode the natural deduction rules via *typed constants*, while they are often expressed via *rewrite rules* in the $\lambda\Pi$-calculus modulo theory [2]. For instance, both the introduction and the elimination of implication can be derived from the rewrite rule $Prf\ (p \Rightarrow q) \hookrightarrow Prf\ p \to Prf\ q$. So as to perform the translation from classical logic to intuitionistic logic, the natural deduction steps must be *explicit* deduction steps, and cannot be *implicit* computation steps. That is why we encode the natural deduction rules with a deep embedding—via typed constants—instead of a shallow embedding—via rewrite rules.

2.3 Theories Encoded in Higher-Order Logic

When working with the encoding of higher-order logic in the $\lambda\Pi$-calculus modulo theory, it is possible to mix sorts, propositions and proofs—which is not expected in higher-order logic. For example, propositions can be inserted in sorts when we have a term of type $Prop \to Set$, and proofs can be inserted in propositions when we have a term of type $\Pi p : Prop.\ Prf\ p \to Prop$. To avoid such behavior, we introduce five grammars:

$$
\begin{aligned}
&\kappa_1 ::= Set \mid \kappa_1 \to \kappa_1 \\
&\kappa_2 ::= Prop \mid El\ a \mid \Pi x : \kappa_i.\ \kappa_2 \text{ with } i \in \{1,2\} \\
&\kappa_3 ::= Prf\ p \mid \kappa_3 \to \kappa_3 \mid \Pi x : \kappa_i.\ \kappa_3 \text{ with } i \in \{1,2\} \\
&\kappa_4 ::= \texttt{TYPE} \mid \Pi x : \kappa_i.\ \kappa_4 \text{ with } i \in \{1,2\} \\
&\kappa_5 ::= \texttt{KIND}
\end{aligned}
$$

The grammar κ_3 generates formulas and inference rules. The grammar κ_4 generates a subclass of kinds, and κ_5 only generates KIND. We characterize the judgments of the $\lambda\Pi$-calculus modulo theory to ensure that types and kinds are generated by one of those grammars.

Definition 1 (κ-property)**.** *The judgment $\Gamma \vdash t : A$ satisfies the κ-property when $A \in \kappa_i$ for some $i \in [\![1,5]\!]$. The judgment $\vdash \Gamma$ satisfies the κ-property when for each $(x : A) \in \Gamma$ we have $A \in \kappa_i$ for some $i \in [\![1,5]\!]$. A derivation satisfies the κ-property when each of its judgments satisfies the κ-property.*

Theories encoded in higher-order logic are theories that feature the base higher-order encoding and in which the user-defined constants satisfy the κ-property.

Definition 2 (Theory encoded in higher-order logic)**.** *Let $\mathscr{T} = (\Sigma, \mathscr{R})$ be a theory in the $\lambda\Pi$-calculus modulo theory. $\mathscr{T}$ is encoded in higher-order logic when:*

1. *$\Sigma = \Sigma^k_{HOL} \cup \Sigma_{\mathscr{T}}$ with $k \in \{i,c\}$ and $\Sigma_{HOL} \cap \Sigma_{\mathscr{T}} = \emptyset$,*
2. *$\mathscr{R} = \mathscr{R}_{HOL} \cup \mathscr{R}_{\mathscr{T}}$ with $\mathscr{R}_{HOL} \cap \mathscr{R}_{\mathscr{T}} = \emptyset$,*
3. *for every $c : A \in \Sigma_{\mathscr{T}}$, the judgment $\vdash c : A$ satisfies the κ-property,*
4. *for every $\ell \hookrightarrow r \in \mathscr{R}_{\mathscr{T}}$, ℓ is neither Prf nor $\forall$.*

The fourth condition will ensure that the translation of a rewrite rule is a well-defined rewrite rule. Theories encoded in higher-order logic extend higher-order logic with user-defined rewrite rules and inference rules. The introduction of rewrite rules is part and parcel of deduction modulo theory [8], while the introduction of inference rules has been developed in superdeduction modulo theory [3, 16].

When considering a theory encoded in higher-order logic, all the user-defined constants satisfy the κ-property. In that respect, the only way to mix sorts, propositions and proofs is through λ-abstractions. For instance, $(\lambda P : Prop.\ o)$ is a term taking as input a proposition and returning a sort. The type

$El\ ((\lambda P : Prop.\ o)\ \bot)$ mixes propositions and sorts, but it is β-convertible to $El\ o$, in which no proposition occurs. Using this principle, we can transform every derivation of a theory encoded in higher-order logic into a derivation that satisfies the κ-property, by applying β-reduction on fragments of the derivation. When a derivation satisfies the κ-property, the rewrite rules $\ell \hookrightarrow r$ with ℓ and r of type $A \in \kappa_3$ cannot be used. In the rest of this paper and without loss of generality, we only consider derivations that satisfy the κ-property and rewrite rules $\ell \hookrightarrow r$ with ℓ and r of type $A \in \kappa_i$ for $i \neq 3$.

Example 1 (Equational theory)**.** *Consider the theory* $\mathscr{T} = (\Sigma_{HOL} \cup \Sigma_{eq}, \mathscr{R}_{HOL} \cup \mathscr{R}_{eq})$*, with a polymorphic equality symbol* $=\ : \Pi a : Set.\ El\ a \to El\ a \to Prop$*, and a rewrite rule for the Leibniz principle* $Prf\ (= a\ x\ y) \hookrightarrow \Pi P : El\ a \to Prop.\ Prf\ (P\ x) \to Prf\ (P\ y)$*. This theory is encoded in higher-order logic. We can prove that the equality is reflexive, symmetric and transitive. For instance, the proof of reflexivity is given by* $\lambda a : Set.\ \mathsf{all_i}\ a\ (\lambda x : El\ a.\ = a\ x\ x)\ (\lambda x : El\ a.\ \lambda P : El\ a \to Prop.\ \lambda P_x : Prf\ (P\ x).\ P_x)$ *which is of type* $\Pi a : Set.\ Prf\ (\forall\ a\ (\lambda x : El\ a.\ = a\ x\ x))$.

3 Kuroda's Translation in the $\lambda\Pi$-Calculus Modulo Theory

In this section, we adapt Kuroda's double-negation translation to the $\lambda\Pi$-calculus modulo theory, when working in theories encoded in higher-order logic. Kuroda's translation [18] inserts a double negation in front of formulas and one after every universal quantifier. More formally, we have $A^{Ku} := \neg\neg A_{Ku}$ where A_{Ku} is defined by induction:

$$\begin{array}{lll} (A \Rightarrow B)_{Ku} := A_{Ku} \Rightarrow B_{Ku} & (\neg A)_{Ku} := \neg A_{Ku} & P_{Ku} := P \text{ if } P \text{ atomic} \\ (A \wedge B)_{Ku} := A_{Ku} \wedge B_{Ku} & \top_{Ku} := \top & (\forall x\, A)_{Ku} := \forall x\, \neg\neg A_{Ku} \\ (A \vee B)_{Ku} := A_{Ku} \vee B_{Ku} & \bot_{Ku} := \bot & (\exists x\, A)_{Ku} := \exists x\, A_{Ku} \end{array}$$

This translation embeds classical logic into intuitionistic logic, as for any first-order formula A we have $\Gamma \vdash A$ in classical logic if and only if $\Gamma^{Ku} \vdash A^{Ku}$ in intuitionistic logic.

3.1 Translation of Terms and Theories

When working inside a theory encoded in higher-order logic in the $\lambda\Pi$-calculus modulo theory, every formula has head symbol *Prf*. Inserting a double negation in front of every formula is therefore equivalent to inserting it after every *Prf* symbol. In that respect, we define a single translation $t \mapsto t^{Ku}$ by induction on the terms of the $\lambda\Pi$-calculus modulo theory. The translation of *Prf* is $\lambda p.\ Prf\ (\neg\neg p)$, and the translation of the universal quantifier $\forall$ is $\lambda a.\ \lambda p.\ \forall\ a\ (\lambda z.\ \neg\neg(p\ z))$. The translation of λ-abstraction $\lambda x : A.\ t$ is naturally given by $\lambda x : A^{Ku}.\ t^{Ku}$, the one of dependent type $\Pi x : A.\ B$ is given by $\Pi x : A^{Ku}.\ B^{Ku}$ and the one of application $t\ u$ is defined by $t^{Ku}\ u^{Ku}$.

As we are in the $\lambda\Pi$-calculus modulo theory with the *proofs-as-terms* paradigm, we have to translate proofs as well. Kuroda's translation relies on the fact that the translation of each natural deduction rule is admissible in intuitionistic logic. For instance, the introduction of implication allows to derive $\Gamma \vdash P \Rightarrow Q$ from $\Gamma, P \vdash Q$. In intuitionistic logic, $\Gamma^{Ku} \vdash (P \Rightarrow Q)^{Ku}$ is derivable from $\Gamma^{Ku}, P^{Ku} \vdash Q^{Ku}$. In the $\lambda\Pi$-calculus modulo theory, the *constant* $\mathsf{imp_i}$ is of type $\Pi p, q : Prop.\ (Prf\ p \to Prf\ q) \to Prf\ (p \Rightarrow q)$, and we can build a *term* $\mathsf{imp_i}^i$ of type $\Pi p, q : Prop.\ (Prf\ \neg\neg p \to Prf\ \neg\neg q) \to Prf\ \neg\neg(p \Rightarrow q)$, that only depends on the constants representing *intuitionistic* natural deduction rules. Each constant c of type A representing a natural deduction rule is translated by the term c^i of type A^{Ku}, where c^i is an intuitionistic proof term of A^{Ku}.

Definition 3 (Translation of terms)**.** *Kuroda's translation is inductively defined on the terms of the λΠ-calculus modulo theory by:*

$$
\begin{aligned}
&x^{Ku} := x\\
&c^{Ku} := \begin{cases} \lambda p.\ Prf\ (\neg\neg p) & \text{if } c = Prf\\ \lambda x.\ \lambda p.\ \forall\ x\ (\lambda z.\ \neg\neg(p\ z)) & \text{if } c = \forall\\ c^i & \text{if } c \text{ is a constant representing a natural deduction rule}\\ c & \text{otherwise}\end{cases}\\
&s^{Ku} := s\\
&(\lambda x : A.\ t)^{Ku} := \lambda x : A^{Ku}.\ t^{Ku}\\
&(\Pi x : A.\ B)^{Ku} := \Pi x : A^{Ku}.\ B^{Ku}\\
&(t\ u)^{Ku} := t^{Ku}\ u^{Ku}
\end{aligned}
$$

Proposition 1. *For every constant $c : A \in \Sigma_{HOL}$ representing a natural deduction rule, we have $\vdash c^i : A^{Ku}$ in the theory $(\Sigma^i_{HOL}, \mathscr{R}_{HOL})$.*

Proof. We have formalized the proof terms c^i in DEDUKTI[1]. For instance, $\mathsf{top_i}^i$ is given in Section 4. □

As we are not mixing sorts, propositions and proofs, we know that the symbol ∀, the symbol *Prf* and the constants representing the natural deduction rules only occur in the grammar κ_3. Therefore, any type $A \in \kappa_i$ is modified by Kuroda's translation for $i = 3$, whereas $A^{Ku} = A$ for $i \neq 3$.

We have defined the translation for terms, and we now want to define it for theories. Intuitively, we would like to translate a rewrite rule $\ell \hookrightarrow r$ by $\ell^{Ku} \hookrightarrow r^{Ku}$. However, if the head constant of ℓ is *Prf* or ∀, then the head symbol of ℓ^{Ku} is Prf^{Ku} or $\forall^{Ku}$, that is a λ-abstraction and not a constant. Hence $\ell^{Ku} \hookrightarrow r^{Ku}$ may not be a valid rewrite rule in the λΠ-calculus modulo theory. We write $\lfloor \ell^{Ku} \rfloor$ for the term obtained by β-reducing the head symbol of ℓ^{Ku} if it is Prf^{Ku} or $\forall^{Ku}$.

Definition 4. *The translation $t \mapsto t^{Ku}$ is extended to contexts, signatures and rewrite systems by:*

$$
\begin{aligned}
&\langle\rangle^{Ku} ::= \langle\rangle\\
&(\Gamma, x : A)^{Ku} := \Gamma^{Ku}, x : A^{Ku}\\
&(\Sigma, c : A)^{Ku} := \Sigma^{Ku}, c : A^{Ku}\\
&(\mathscr{R}, \ell \hookrightarrow r)^{Ku} := \mathscr{R}^{Ku}, \lfloor \ell^{Ku} \rfloor \hookrightarrow r^{Ku}
\end{aligned}
$$

When translating a theory encoded in higher-order logic, we replace Σ^c_{HOL} by Σ^i_{HOL}, and we translate the user-defined signature $\Sigma_{\mathscr{T}}$ and rewrite system $\mathscr{R}_{\mathscr{T}}$.

Definition 5 (Translation of theories)**.** *Let $\mathscr{T} = (\Sigma^c_{HOL} \cup \Sigma_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}_{\mathscr{T}})$ be a theory encoded in higher-order logic. The translation of $\mathscr{T}$ is $\mathscr{T}^{Ku} = (\Sigma^i_{HOL} \cup \Sigma^{Ku}_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}^{Ku}_{\mathscr{T}})$.*

Remark that $\mathscr{T}^{Ku}$ is a theory. Specifically, rewrite rules $\lfloor \ell^{Ku} \rfloor \hookrightarrow r^{Ku} \in \mathscr{R}^{Ku}_{\mathscr{T}}$ are always well-defined, since ℓ is neither *Prf* nor ∀, and by definition of $\lfloor \ell^{Ku} \rfloor$.

3.2 Embedding Classical Logic into Intuitionistic Logic

We aim at proving that the extension of Kuroda's translation in the λΠ-calculus modulo theory indeed embeds classical logic into intuitionistic logic. In other words, we want to show that $\Gamma \vdash t : A$ in $\mathscr{T}$ entails $\Gamma^{Ku} \vdash t^{Ku} : A^{Ku}$ in $\mathscr{T}^{Ku}$. To do so, we translate the derivations step by step. In particular, when the CONV rule is used with $A \equiv_{\beta\mathscr{R}} B$ in $\mathscr{T}$, we want to have $A^{Ku} \equiv_{\beta\mathscr{R}} B^{Ku}$ in $\mathscr{T}^{Ku}$.

[1]See `https://github.com/Deducteam/Construkti/blob/master/kuroda.dk`.

Lemma 2 (Translation of substitutions)**.** $(t[z \leftarrow w])^{Ku} = t^{Ku}[z \leftarrow w^{Ku}]$

Proof. By induction on the term t. We have $(c[z \leftarrow w])^{Ku} = c^{Ku} = c^{Ku}[z \leftarrow w^{Ku}]$ since c^{Ku} is a closed term. Similarly, $(s[z \leftarrow w])^{Ku} = s^{Ku} = s^{Ku}[z \leftarrow w^{Ku}]$. If $x \neq z$, then $(x[z \leftarrow w])^{Ku} = x^{Ku} = x^{Ku}[z \leftarrow w^{Ku}]$. If $x = z$, then $(x[z \leftarrow w])^{Ku} = w^{Ku} = x[z \leftarrow w^{Ku}] = x^{Ku}[z \leftarrow w^{Ku}]$. The cases for λ-abstractions, dependent types, and applications follow from the induction hypotheses. □

Lemma 3 (Translation of conversions)**.** *If $A \equiv_{\beta\mathscr{R}} B$ in $\mathscr{T}$, then $A^{Ku} \equiv_{\beta\mathscr{R}} B^{Ku}$ in $\mathscr{T}^{Ku}$.*

Proof. By induction on the construction of $A \equiv_{\beta\mathscr{R}} B$.

- If $\ell \hookrightarrow r$ in $\mathscr{T}$, then we show $(\ell\theta)^{Ku} \equiv_{\beta\mathscr{R}} (r\theta)^{Ku}$ in $\mathscr{T}^{Ku}$ for any substitution θ. For $\ell \hookrightarrow r \in \mathscr{R}_{HOL}$, we have $\ell^{Ku} = \ell$ and $r^{Ku} = r$, and we use Lemma 2. For $\ell \hookrightarrow r \in \mathscr{R}_{\mathscr{T}}$, we have $\lfloor \ell^{Ku} \rfloor \hookrightarrow r^{Ku} \in \mathscr{R}_{\mathscr{T}}^{Ku}$, which entails that $(\ell\theta)^{Ku} = \ell^{Ku}\theta^{Ku} \equiv_{\beta\mathscr{R}} \lfloor \ell^{Ku} \rfloor \theta^{Ku} \equiv_{\beta\mathscr{R}} r^{Ku}\theta^{Ku} = (r\theta)^{Ku}$ by Lemma 2.
- If $(\lambda x : A.\ t)\ u \hookrightarrow t[x \leftarrow u]$ in $\mathscr{T}$, then we have $((\lambda x : A.\ t)\ u)^{Ku} = (\lambda x : A^{Ku}.\ t^{Ku})\ u^{Ku}$, which β-reduces to $t^{Ku}[x \leftarrow u^{Ku}]$, that is $(t[x \leftarrow u])^{Ku}$ using Lemma 2.
- Closure by context, reflexivity, symmetry, and transitivity are immediate.

□

Theorem 1 (Translation of judgments)**.** *Let $\mathscr{T}$ be a theory encoded in higher-order logic.*

- *If $\vdash \Gamma$ in $\mathscr{T}$ then $\vdash \Gamma^{Ku}$ in $\mathscr{T}^{Ku}$.*
- *If $\Gamma \vdash t : A$ in $\mathscr{T}$ then $\Gamma^{Ku} \vdash t^{Ku} : A^{Ku}$ in $\mathscr{T}^{Ku}$.*

Proof. We proceed by induction on the derivation. We present the most interesting cases, the others follow the definition and the induction hypotheses.

- <u>CONST</u>: By induction we have $\vdash \Gamma^{Ku}$ and $\Gamma^{Ku} \vdash A^{Ku} : s^{Ku}$ in $\mathscr{T}^{Ku}$.

  If $c : A \in \Sigma_{\mathscr{T}}$, then $c : A^{Ku} \in \Sigma_{\mathscr{T}}^{Ku}$ and we derive $\Gamma^{Ku} \vdash c : A^{Ku}$ using CONST.

  Suppose that $c = Prf$. We simply derive $\Gamma^{Ku} \vdash \lambda p.\ Prf\ (\neg\neg p) : Prop \to \texttt{TYPE}$, that is $\Gamma^{Ku} \vdash Prf^{Ku} : (Prop \to \texttt{TYPE})^{Ku}$, in $\mathscr{T}^{Ku}$.

  Suppose that $c = \forall$. We simply derive $\Gamma^{Ku} \vdash \lambda x.\ \lambda p.\ \forall\ x\ (\lambda z.\ \neg\neg(p\ z)) : \Pi x : Set.\ (El\ x \to Prop) \to Prop$, that is $\Gamma^{Ku} \vdash \forall^{Ku} : (\Pi x : Set.\ (El\ x \to Prop) \to Prop)^{Ku}$, in $\mathscr{T}^{Ku}$.

  Suppose that c is a constant representing a natural deduction rule. Using Proposition 1, we have $\Gamma^{Ku} \vdash c^i : A^{Ku}$ in $\mathscr{T}^{Ku}$, that is $\Gamma^{Ku} \vdash c^{Ku} : A^{Ku}$. In particular, we replace the classical axiom $\mathsf{pem} : \Pi p : Prop.\ Prf\ (p \vee \neg p)$ by the intuitionistic term $\mathsf{pem}^i : \Pi p : Prop.\ Prf\ (\neg\neg(p \vee \neg p))$.

  Otherwise, $c : A \in \Sigma_{HOL}$ but is not Prf, not $\forall$, and not a constant representing a natural deduction rule. Then A does not contain Prf and $\forall$, so $A^{Ku} = A$. We derive $\Gamma^{Ku} \vdash c : A^{Ku}$ using CONST.
- <u>CONV</u>: By induction we have $\Gamma^{Ku} \vdash t^{Ku} : A^{Ku}$ in $\mathscr{T}^{Ku}$ and $\Gamma^{Ku} \vdash B^{Ku} : s^{Ku}$ in $\mathscr{T}^{Ku}$. From Lemma 3, we know that $A^{Ku} \equiv_{\beta\mathscr{R}} B^{Ku}$, and we conclude that $\Gamma^{Ku} \vdash t^{Ku} : B^{Ku}$ in $\mathscr{T}^{Ku}$ using CONV.

□

Example 2 (Translated equational theory)**.** *The translation of the theory $\mathscr{T} = (\Sigma_{HOL} \cup \Sigma_{eq}, \mathscr{R}_{HOL} \cup \mathscr{R}_{eq})$ of Example 1 is obtained by taking the equality symbol $= \ : \Pi a : Set.\ El\ a \to El\ a \to Prop$ (which remains unchanged), and by transforming the rewrite rule $Prf\ (=\ a\ x\ y) \hookrightarrow \Pi P : El\ a \to Prop.\ Prf\ (P\ x) \to Prf\ (P\ y)$ into $Prf\ (\neg\neg(=\ a\ x\ y)) \hookrightarrow \Pi P : El\ a \to Prop.\ Prf\ (\neg\neg(P\ x)) \to Prf\ (\neg\neg(P\ y))$. The proof of reflexivity is now given by $\lambda a : Set.\ \mathsf{all}_\mathsf{i}^i\ a\ (\lambda x : El\ a.\ =\ a\ x\ x)\ (\lambda x : El\ a.\ \lambda P : El\ a \to Prop.\ \lambda P_x : Prf\ (\neg\neg(P\ x)).\ P_x)$ which is of type $\Pi a : Set.\ Prf\ (\neg\neg(\forall\ a\ (\lambda x : El\ a.\ \neg\neg(=\ a\ x\ x))))$.*

3.3 Back to the Original Theory

We have shown that, in the $\lambda\Pi$-calculus modulo theory, $\Gamma \vdash t : A$ in $\mathscr{T}$ implies $\Gamma^{Ku} \vdash t^{Ku} : A^{Ku}$ in $\mathscr{T}^{Ku}$. We now want to prove the reverse implication: if there exists an intuitionistic proof of A^{Ku} in $\mathscr{T}^{Ku}$, then there exists a classical proof of A in $\mathscr{T}$. To do so, we reason in two steps: first we show that it is possible to build a proof of A from a proof of A^{Ku} in classical logic, and then we show that any result in $\mathscr{T}^{Ku}$ can also be derived in $\mathscr{T}$.

The first step consists in proving that, for any $A \in \kappa_3$, it is possible to derive A^{Ku} from A. For this, we show that any proposition P and its translation P^{Ku} are classically equivalent. Such a result is not necessarily true in higher-order logic. We assume some property, called the Kuroda equivalence.

Definition 6 (Kuroda equivalence)**.** *Let Γ be a context, t be a constant or a variable such that $\Gamma \vdash t : T_1 \to \ldots \to T_n \to Prop$, and $u_1, \ldots, u_n$ be terms such that $\Gamma \vdash u_i : T_i$. There exists some p such that $\Gamma \vdash p : Prf\ ((t\ u_1\ \ldots\ u_n)^{Ku} \Leftrightarrow t\ u_1\ \ldots\ u_n)$.*

The Kuroda equivalence property is derivable from functional extensionality and propositional extensionality in classical logic [21]. Remark that it is satisfied for the usual logical connectives and quantifiers. For instance, we have $A_{Ku} \wedge B_{Ku} \Leftrightarrow A \wedge B$ and $\forall x\ \neg\neg A_{Ku} \Leftrightarrow \forall x\ A$ in classical logic. In the rest of this paper, we work assuming the Kuroda equivalence.

Lemma 4. *Any proposition P is β-convertible to a variable x, a constant c, or an application $t\ u_1 \ldots\ u_n$ where t is a constant or a variable of type $T_1 \to \ldots \to T_n \to Prop$ and $u_1, \ldots, u_n$ are terms of type $T_1, \ldots, T_n$.*

The constant c may be $\top$ or $\bot$, and the head symbol of the application may be any connective, quantifier or predicate.

Proposition 2. *Let $\Gamma \vdash P : Prop$. In the theory $(\Sigma^c_{HOL} \cup \Sigma, \mathscr{R}_{HOL} \cup \mathscr{R})$, there exists some proof term m_P such that $\Gamma \vdash m_P : Prf\ (P^{Ku} \Leftrightarrow P)$.*

Proof. We distinguish cases thanks to Lemma 4.

- Suppose that P is β-convertible to a variable x. We have $x^{Ku} = x$ so we build some m_x such that $\Gamma \vdash m_x : Prf\ (x^{Ku} \Leftrightarrow x)$. Since P is β-convertible to x, P^{Ku} is β-convertible to x^{Ku} (see Lemma 3) and we conclude that $\Gamma \vdash m_x : Prf\ (P^{Ku} \Leftrightarrow P)$.
- If P is β-convertible to a constant c, then we are in the case where $c^{Ku} = c$ and we proceed similarly.
- Suppose that P is β-convertible to an application $t\ u_1 \ldots\ u_n$ where t is a constant or a variable. P^{Ku} is β-convertible to $(t\ u_1 \ldots\ u_n)^{Ku}$ and we conclude using the Kuroda equivalence.

□

Lemma 5. *Let $A \in \kappa_3$ and ℓ be a strict subterm of A. In the theory $(\Sigma^c_{HOL} \cup \Sigma, \mathscr{R}_{HOL} \cup \mathscr{R})$, for any context Γ, there exists some t such that $\Gamma \vdash t : A[\ell]$ if and only if there exists some t' such that $\Gamma \vdash t' : A[\ell^{Ku}]$.*

Proof. We proceed by induction on the term A using the fact that A is generated by κ_3.

- Suppose that $A = Prf\ P$. If $\forall$ does not occur in ℓ, then $\ell^{Ku} = \ell$ and $P[\ell^{Ku}] = P[\ell]$, so we directly conclude. Otherwise, we use Proposition 2 on the right proposition.
- Suppose that $A = \Pi x : B.\ C$ with $B \in \kappa_1$ or $B \in \kappa_2$. If ℓ occurs in B, then by definition $B[\ell^{Ku}] = B[\ell]$, so $\ell^{Ku} = \ell$ and we directly conclude. Suppose that ℓ only occurs in C and that there exists some t such that $\Gamma \vdash t : \Pi x : B.\ C[\ell]$. By induction on C with $\Gamma, x : B \vdash t\ x : C[\ell]$ (obtained by weakening), we get some t'_C such that $\Gamma, x : B \vdash t'_C : C[\ell^{Ku}]$. Therefore, we have $\Gamma \vdash \lambda x : B.\ t'_C : \Pi x : B.\ C[\ell^{Ku}]$. We proceed similarly for the reverse implication.

- Suppose that $A = B \to C$ with $B, C \in \kappa_3$. Suppose that we have $\Gamma \vdash t : B[\ell] \to C[\ell]$. By induction on B with $\Gamma, x : B[\ell^{Ku}] \vdash x : B[\ell^{Ku}]$, we get some t_B such that $\Gamma, x : B[\ell^{Ku}] \vdash t_B : B[\ell]$. By induction on C with $\Gamma, x : B[\ell^{Ku}] \vdash t\ t_B : C[\ell]$, we get some t'_C such that $\Gamma, x : B[\ell^{Ku}] \vdash t'_C : C[\ell^{Ku}]$. We conclude that $\Gamma \vdash \lambda x : B[\ell^{Ku}].\ t'_C : B[\ell^{Ku}] \to C[\ell^{Ku}]$. We proceed similarly for the reverse implication.

□

Lemma 6. *Let $A \in \kappa_3$. In the theory $(\Sigma^c_{HOL} \cup \Sigma, \mathscr{R}_{HOL} \cup \mathscr{R})$, for any context Γ, there exists some t such that $\Gamma \vdash t : A$ if and only if there exists some t' such that $\Gamma \vdash t' : A^{Ku}$.*

Proof. We proceed by induction on the term A using the fact that A is generated by κ_3. We use Lemma 5 and the double-negation elimination. □

We have shown that it is possible to build a proof of A in $\mathscr{T}^{Ku}$ using a proof of A^{Ku} and the principle of excluded middle. The next step is to derive a proof of A in the original theory $\mathscr{T}$. In particular, it requires to replace each use of $\lfloor \ell^{Ku} \rfloor \hookrightarrow r^{Ku} \in \mathscr{R}^{Ku}_{\mathscr{T}}$ by a use of $\ell \hookrightarrow r \in \mathscr{R}_{\mathscr{T}}$.

Lemma 7. *Let $A \in \kappa_3$ such that $\Gamma \vdash t : A[\ell^{Ku}]$. Using $\ell \hookrightarrow r$, there exists some t' such that $\Gamma \vdash t' : A[r^{Ku}]$.*

Proof. Using Lemma 5, there exists some t' such that $\Gamma \vdash t' : A[\ell]$. Using $\ell \hookrightarrow r$, we have $\Gamma \vdash t' : A[r]$. We use Lemma 5 to obtain some t'' such that $\Gamma \vdash t'' : A[r^{Ku}]$. □

Lemma 8. *Let $(\Sigma^c_{HOL} \cup \Sigma, \mathscr{R}_{HOL} \cup \mathscr{R}^{Ku})$ and $(\Sigma^c_{HOL} \cup \Sigma, \mathscr{R}_{HOL} \cup \mathscr{R})$ be two theories, abbreviated $\mathscr{R}^{Ku}$ and $\mathscr{R}$.*

- *If $\vdash \Gamma$ in $\mathscr{R}^{Ku}$ then $\vdash \Gamma$ in $\mathscr{R}$.*
- *If $\Gamma \vdash t : A$ in $\mathscr{R}^{Ku}$ and $A \in \kappa_i$ with $i \in \{1,2,4,5\}$, then $\Gamma \vdash t : A$ in $\mathscr{R}$.*
- *If $\Gamma \vdash t : A$ in $\mathscr{R}^{Ku}$ and $A \in \kappa_3$, then there exists some t' such that $\Gamma \vdash t' : A$ in $\mathscr{R}$.*

Proof. We proceed by induction on the typing derivation. We only present the relevant cases.

- ABS: Suppose that $\Gamma \vdash A : \texttt{TYPE}$ and $\Gamma, x : A \vdash B : s$ and $\Gamma, x : A \vdash t : B$ in $\mathscr{R}^{Ku}$. By induction we have $\Gamma \vdash A : \texttt{TYPE}$ and $\Gamma, x : A \vdash B : s$ in $\mathscr{R}$.

  If $B \in \kappa_i$ with $i \in \{1,2\}$, then by induction we have $\Gamma, x : A \vdash t : B$ in $\mathscr{R}$, and we derive $\Gamma \vdash \lambda x : A.\ t : \Pi x : A.\ B$ in $\mathscr{R}$.

  If $B \in \kappa_3$, then by induction we have $\Gamma, x : A \vdash t' : B$ in $\mathscr{R}$. We derive $\Gamma \vdash \lambda x : A.\ t' : \Pi x : A.\ B$ in $\mathscr{R}$.

- APP: Suppose that $\Gamma \vdash t : \Pi x : A.\ B$ and $\Gamma \vdash u : A$ in $\mathscr{R}^{Ku}$.

  If $\Pi x : A.\ B \in \kappa_i$ with $i \in \{1,2,4\}$, then by induction we have $\Gamma \vdash t : \Pi x : A.\ B$ and $\Gamma \vdash u : A$ in $\mathscr{R}$. We derive $\Gamma \vdash t\ u : B[x \leftarrow u]$ in $\mathscr{R}$.

  If $\Pi x : A.\ B \in \kappa_3$, then by induction we have $\Gamma \vdash t' : \Pi x : A.\ B$ in $\mathscr{R}$. If $A \in \kappa_i$ with $i \in \{1,2\}$, then by induction we have $\Gamma \vdash u : A$ in $\mathscr{R}$, and we derive $\Gamma \vdash t'\ u : B[x \leftarrow u]$ in $\mathscr{R}$. If $A \in \kappa_3$ (x does not occur in B), then by induction we have $\Gamma \vdash u' : A$ in $\mathscr{R}$, and we conclude that $\Gamma \vdash_c t'\ u' : B$.

- CONV: If $A \equiv_{\beta\mathscr{R}} B$ is obtained using β-conversion or the rewrite rules of $\mathscr{R}_{HOL}$, then we conclude using the induction hypothesis and the CONV rule. Otherwise, and without loss of generality, we consider that we only use one rewrite rule of $\mathscr{R}^{Ku}$ per CONV rule.

  Suppose that $A \equiv_{\beta\mathscr{R}} B$ is obtained using the rewrite rule $\ell^{Ku} \hookrightarrow r^{Ku} \in \mathscr{R}^{Ku}$. In that case, we have $A = C[\ell^{Ku}]$ and $B = C[r^{Ku}]$ (the case $A = C[r^{Ku}]$ and $B = C[\ell^{Ku}]$ is treated similarly). By assumption, we have $\Gamma \vdash t : C[\ell^{Ku}]$ and $\Gamma \vdash C[r^{Ku}] : s$ in R^{Ku}.

If $A, B \in \kappa_i$ with $i \in \{1,2,4,5\}$, then $\ell^{Ku} = \ell$ and $r^{Ku} = r$. By induction we have $\Gamma \vdash t : C[\ell^{Ku}]$ and $\Gamma \vdash C[r^{Ku}] : s$ in $\mathscr{R}$. We apply CONV with $C[\ell] \equiv_{\beta\mathscr{R}} C[r]$.

If $A, B \in \kappa_3$, then by induction we have $\Gamma \vdash t' : C[\ell^{Ku}]$ and $\Gamma \vdash C[r^{Ku}] : s$ in $\mathscr{R}$. We conclude using Lemma 7.

□

We now have all the tools to show that, for any intuitionistic proof of A^{Ku} in the translated theory $\mathscr{T}^{Ku}$, there exists a classical proof of A in the original theory $\mathscr{T}$.

Theorem 2. *Let $\mathscr{T}$ be a theory encoded in higher-order logic and $A \in \kappa_3$. If $\Gamma^{Ku} \vdash t : A^{Ku}$ in $\mathscr{T}^{Ku}$, then under the Kuroda equivalence there exists some term t' such that $\Gamma \vdash t' : A$ in $\mathscr{T}$.*

Proof. We directly have $\Gamma^{Ku} \vdash t : A^{Ku}$ in $(\Sigma^c_{HOL} \cup \Sigma^{Ku}_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}^{Ku}_{\mathscr{T}})$.

- By Lemma 6, there exists some t' such that $\Gamma^{Ku} \vdash t' : A$ in $(\Sigma^c_{HOL} \cup \Sigma^{Ku}_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}^{Ku}_{\mathscr{T}})$ and under the Kuroda equivalence.
- Using Lemma 8, there exists some t'' such that $\Gamma^{Ku} \vdash t'' : A$ in $(\Sigma^c_{HOL} \cup \Sigma^{Ku}_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}_{\mathscr{T}})$.
- We replace the signature $\Sigma^{Ku}_{\mathscr{T}}$ by $\Sigma_{\mathscr{T}}$. For each constant $c : C \in \Sigma_{\mathscr{T}}$ with $C \in \kappa_3$, we replace c by t_c (provided by Lemma 6) in t''. We obtain $\Gamma^{Ku} \vdash t''[c \leftarrow t_c] : A$ in $(\Sigma^c_{HOL} \cup \Sigma_{\mathscr{T}}, \mathscr{R}_{HOL} \cup \mathscr{R}_{\mathscr{T}})$, that is in $\mathscr{T}$. These substitutions work since c cannot occur in a dependent type.
- We replace the context Γ^{Ku} by Γ. For each variable $x : B \in \Gamma$ with $B \in \kappa_3$, we replace x by t_x (provided by Lemma 6) in $t''[c \leftarrow t_c]$. We obtain $\Gamma \vdash t''[c \leftarrow t_c][x \leftarrow t_x] : A$ in $\mathscr{T}$, which achieves the proof.

□

The extension of Kuroda's translation to the λΠ-calculus modulo theory is a generalization of Brown and Rizkallah's translation for simple type theory [4]. Indeed, if $\mathscr{R}_{\mathscr{T}} = \langle\rangle$, then we obtain the result in higher-order logic, at the only difference that proofs are represented by terms.

4 Construkti, an Implementation for Dedukti Proofs

Dedukti. The λΠ-calculus modulo theory has been implemented in the DEDUKTI proof language. Abstractions $\lambda x : A.\ t$ are represented by `x : A => t`, and dependent types $\Pi x : A.\ B$ are represented by `x : A -> B`. Constants $c : A$ are specified by `c : A`, prefixed with the keyword `def` if the constant can be defined using rewrite rules. Rewrite rules $\ell \hookrightarrow r$, where x and y are the free variables of ℓ and r, are represented by `[x,y] l --> r`. For instance, using the encoding of the notions of proposition and proof, we can encode the addition on natural numbers via rewrite rules.

```
nat : Set.
0 : El nat.
S : El nat -> El nat.
def add : El nat -> El nat -> El nat.
[x] add x 0 --> x.
[x, y] add x (S y) --> S (add x y).
```

Theorems are represented by `thm n : T := p`, where `n` is its name, `T` its statement and `p` its proof term. For checking that `p` is indeed a proof of `T`, we can use one of the type checkers of DEDUKTI, for instance DKCHECK [19] or LAMBDAPI [15].

Construkti. We have implemented CONSTRUKTI[2], a tool that performs Kuroda's translation on DEDUKTI proofs. CONSTRUKTI takes as input a DEDUKTI file containing the specification of a user-defined theory encoded in higher-order logic, as well as proofs in this theory. It returns a DEDUKTI file containing the specification of the translated theory, as well as the translated proofs.

In this implementation, we insert one double negation after every *Prf* and $\forall$ symbols, and we replace the constants c representing natural deduction rules by the terms c^i. For instance, the constant $\mathsf{top_i}$ of type $Prf\ \top$, representing the introduction of tautology, is replaced in the formal proofs by the term $\mathsf{top_i}^i$ of type $Prf\ (\neg\neg\top)$. The proof term $\mathsf{top_i}^i$ relies on the proof of $\Pi p : Prop.\ Prf\ (p \Rightarrow \neg\neg p)$.

```
top_i : Prf top.

thm prop_double_neg : p : Prop -> Prf (imp p (not (not p)))
:= p => imp_i p (not (not p))
  (pP => neg_i (not p) (pNP => neg_e p pNP pP)).

thm top_i_i : Prf (not (not top))
:= imp_e top (not (not top)) (prop_double_neg top) top_i.
```

So as to obtain readable theorems, we directly β-reduce every application of Prf^{Ku} and $\forall^{Ku}$.

Benchmark. We have tested CONSTRUKTI on a benchmark of 101 DEDUKTI proofs, available in the file `hol-lib.dk`. These proofs encompass results related to connectives and quantifiers, classical formulas, De Morgan's laws, polymorphic equality, and basic arithmetic. The proofs are expressed in propositional, first-order and higher-order logics. This library of proofs includes user-defined rewrite rules—a feature of the $\lambda\Pi$-calculus modulo theory—and inference rules—thanks to the encoding of the notions of proposition and proof. We compare in Table 1 the different characteristics of the library: the number of proofs, the number of classical proofs, the number of results expressed in higher-order logic, and the number of results that are expressed via admissible inference rules.

Content of	Number of ...			
the library	proofs	classical proofs	higher-order results	admissible inference rules
Basic logic	38	0	15	26
Classical results	12	12	9	3
De Morgan	8	6	4	8
Equality	10	0	6	4
Arithmetic	33	0	0	16
All	101	18	34	57

Table 1: Comparison of the different libraries.

After running CONSTRUKTI, all the translated proofs of the translated theorems typecheck, and are expressed in intuitionistic logic.

[2] Available at `https://github.com/Deducteam/Construkti`.

5 Conclusion

In this paper, we have extended Kuroda's translation to the theories encoded in higher-logic in the $\lambda\Pi$-calculus modulo theory, that is λ-calculus extended with dependent types and user-defined rewrite rules. In this logical framework, proofs are terms following the Curry-Howard correspondence, and have to be effectively translated. Due to the encoding of the notions of proposition and proof in the $\lambda\Pi$-calculus modulo theory, we can assume, prove, and translate inference rules. We have implemented CONSTRUKTI, a tool that transforms DEDUKTI proofs following Kuroda's translation. Both DEDUKTI and CONSTRUKTI pave the way for interoperability between classical proof systems—such as HOL LIGHT or MIZAR—and intuitionistic proof systems—such as COQ, LEAN or AGDA.

Future work. There exist large libraries of proofs in higher-order logic, for instance the HOL LIGHT standard library. Blanqui [9] recently translated it to COQ via DEDUKTI, taking the excluded middle as an axiom. Future work would be to obtain an intuitionistic version of the HOL LIGHT standard library, by applying Kuroda's translation and CONSTRUKTI.

Related work. Double-negation translations aim at embedding classical logic into intuitionistic logic. As such, double-negation translations *always* transform classical proofs into intuitionistic ones, but they modify the formulas during the process. Proof constructivization aims at transforming classical proofs into intuitionistic ones *without* translating the formulas, but such a process does not necessarily succeed. Cauderlier [5] developed heuristics to constructivize proofs in DEDUKTI, via rewrite systems that try to remove instances of the principle of excluded middle or of the double-negation elimination. Gilbert [11] designed a constructivization algorithm for first-order logic, that was tested in DEDUKTI and works in practice for large fragments of first-order logic.

Acknowledgments

The author would like to thank Marc Aiguier, Gilles Dowek and Olivier Hermant for helpful discussions and valuable remarks about this work.

References

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

[2] Frédéric Blanqui, Gilles Dowek, Émilie Grienenberger, Gabriel Hondet & François Thiré (2023): *A modular construction of type theories*. Logical Methods in Computer Science Volume 19, Issue 1, doi:10.46298/lmcs-19(1:12)2023. Available at https://lmcs.episciences.org/10959.

[3] Paul Brauner, Clement Houtmann & Claude Kirchner (2007): *Principles of Superdeduction*. In: LICS 2007 - 22nd Annual IEEE Symposium on Logic in Computer Science, Wroclaw, Poland, pp. 41–50, doi:10.1109/LICS.2007.37. Available at https://ieeexplore.ieee.org/abstract/document/4276550. ISSN: 1043-6871.

[4] Chad E. Brown & Christine Rizkallah (2014): *Glivenko and Kuroda for simple type theory*. The Journal of Symbolic Logic 79(2), pp. 485–495, doi:10.1017/jsl.2013.10. Available at http://www.jstor.org/stable/43303744.

[5] Raphaël Cauderlier (2016): *A Rewrite System for Proof Constructivization*. In: *LFMTP 2016 - International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, Porto, Portugal, pp. 1 – 7, doi:10.1145/2966268.2966270. Available at https://inria.hal.science/hal-01420634.

[6] Denis Cousineau & Gilles Dowek (2007): *Embedding Pure Type Systems in the Lambda-Pi-Calculus Modulo*. In Simona Ronchi Della Rocca, editor: *Typed Lambda Calculi and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 102–117, doi:10.1007/978-3-540-73228-0_9.

[7] Nachum Dershowitz & Jean-Pierre Jouannaud (1991): *Rewrite Systems*. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, doi:10.1016/B978-0-444-88074-1.50011-1.

[8] Gilles Dowek, Thérèse Hardin & Claude Kirchner (2003): *Theorem Proving Modulo*. *Journal of Automated Reasoning* 31, pp. 33–72, doi:10.1023/A:1027357912519.

[9] Frédéric Blanqui (2024): *Translating HOL-Light proofs to Coq*. In: *LPAR 2024 - 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Balaclava, Mauritius, pp. 1–18, doi:10.29007/6k4x. Available at https://easychair.org/publications/paper/mtFT.

[10] Gerhard Gentzen (1936): *Die Widerspruchsfreiheit der Reinen Zahlentheorie*. *Mathematische Annalen* 112, pp. 493–565, doi:10.1007/BF01565428.

[11] Frédéric Gilbert (2017): *Automated Constructivization of Proofs*. In: *FOSSACS 2017 - 20th International Conference on Foundations of Software Science and Computation Structures*, Uppsala, Sweden, pp. 480–495, doi:10.1007/978-3-662-54458-7_28.

[12] Valery Glivenko (1928): *Sur quelques points de la logique de M. Brouwer*. *Bulletins de la classe des sciences* 15, p. 183–188.

[13] Kurt Gödel (1933): *Zur intuitionistischen Arithmetik und Zahlentheorie*. *Ergebnisse eines Mathematischen Kolloquiums* 4, p. 34–38.

[14] Robert Harper, Furio Honsell & Gordon Plotkin (1993): *A Framework for Defining Logics*. *Journal of the ACM* 40(1), p. 143–184, doi:10.1145/138027.138060.

[15] Gabriel Hondet & Frédéric Blanqui (2020): *The New Rewriting Engine of Dedukti*. In: *FSCD 2020 - 5th International Conference on Formal Structures for Computation and Deduction*, 167, Paris, France, p. 16, doi:10.4230/LIPIcs.FSCD.2020.35. Available at https://inria.hal.science/hal-02981561.

[16] Clément Houtmann (2010): *Représentation et interaction des preuves en superdéduction modulo*. Ph.D. thesis, Université Henri Poincaré - Nancy I. Available at https://theses.hal.science/tel-00553219.

[17] Andrey Nikolaevich Kolmogorov (1925): *O principe tertium non datur*. *Matematicheskiĭ Sbornik* 32, p. 646–667.

[18] Sigekatu Kuroda (1951): *Intuitionistische Untersuchungen der formalistischen Logik*. *Nagoya Mathematical Journal* 2, p. 35–47, doi:10.1017/S0027763000010023.

[19] Ronan Saillard (2015): *Typechecking in the lambda-Pi-Calculus Modulo : Theory and Practice*. Ph.D. thesis, Ecole Nationale Supérieure des Mines de Paris. Available at https://pastel.hal.science/tel-01299180.

[20] François Thiré (2020): *Interoperability between proof systems using the logical framework Dedukti*. Ph.D. thesis, Université Paris-Saclay. Available at https://hal.science/tel-03224039.

[21] Thomas Traversié (2024): *Kuroda's translation for higher-order logic*. Available at https://hal.science/hal-04561757. Manuscript.