カリー・ハワード同型と証明の形式化
Introduction to the Curry-Howard isomorphism*

Ken-etsu Fujita (Gunma University)
June 22-23, 2022

概　要

最初に，カリー・ハワード同型，Formulae-as-Type，Proofs-as-Programs など様々な呼ばれ方をしている “証明に対する見方” の基礎になっていると考えれれる BHK （Brouwer-Heyting-Kolmogorov）解釈 [TD88] を，Matrin-Löf の型理論 [NPS90] の観点から読み直してみる．この見方は，Judgement 「$a \in A$」に対して，

$$\begin{aligned} \text{「}a \text{ は型 } A \text{ の要素」} &= \text{「}a \text{ は命題 } A \text{ の証明」} \\ &= \text{「}a \text{ は問題 } A \text{ の解」} \\ &= \text{「}a \text{ は仕様 } A \text{ を満たすプログラム」} \end{aligned}$$

などの解釈を与えている．特に,「論理式 = プログラムの仕様」の解釈では，プログラムの “評価”（計算，実行）の概念と，評価結果である “値”（value，canonical な表現）の概念が不可欠であり，他の見方に比べてこれらの概念が重要となっている．値は推論規則の導入規則と密接に関係しているが，評価戦略の変更には CPS 変換の定義を変えることで対応可能になることを以降において述べる．ここで，BHK 解釈については Troelstra-vanDalen [TD88] を参考にして，Martin-Löf の型理論については Nordström-Petersson-Smith [NPS90] から引用する．

次に，値呼び評価（$\mathtt{Eval}_{\mathtt{V}}$）と名前呼び評価（$\mathtt{Eval}_{\mathtt{N}}$）の例を身近な式を使って紹介する．これらの計算戦略をプログラム変換の形式で表現したものが CPS 変換と呼ばれている．ここで，カリー・ハワード同型のもとでは

$$\text{プログラム（ラムダ式）} \quad = \quad \text{証明}$$

であり，プログラムの変換は証明の変換に対応することになる．特に，$\overline{\text{値呼び変換}}$ は，論理式 A の証明 M を黒田の埋め込み q で変換した論理式 $\neg\neg A^q$ の証明に変換している．

$$\Gamma \vdash_{\mathrm{NJ}} M : A \implies \Gamma^q \vdash_{\mathrm{Min}} \overline{M} : \neg\neg A^q$$

*This work was supported by the Research Institute for Mathematical Science, a Joint Usage/Research Center located in Kyoto University, and partly supported by Grants-in-Aid for Scientific Research KAKENHI (C) 17K05343 and 20K03711.

また，ラムダ式 M の 名前び変換 より，論理式 A の証明 M からコルモゴロフの埋め込み k で変換した論理式 A^k の証明 $\underline{M}$ が得られることになる．

$$\Gamma \vdash_{\mathrm{NJ}} M : A \implies \Gamma^k \vdash_{\mathrm{Min}} \underline{M} : A^k$$

ここでは，拙著 [F22] に従って体系 NJ の含意断片に限る．

最後に，古典論理（NK）から直観主義論理（NJ）や最小論理（Min）への埋め込みとしてグリベンコの定理，ゲーデル・ゲンツェンの変換，コルモゴロフの変換，黒田の変換が知られているが [TD88]，これらの変換 F をラムダミュウ式（$\lambda\mu$ 計算）で与える．これにより，Martin-Löf の型理論で説明されていた評価・値の概念が自然に表現されていることが読み取れる．

$$\Gamma; \Delta \vdash_{\mathrm{NK}} M : A \implies F(\Gamma); F(\Delta) \vdash_{\mathrm{NJ(Min)}} F(M) : F(A)$$

なお，ゲーデル・ゲンツェンの変換，コルモゴロフの変換（名前呼び $\lambda\mu$ 計算），黒田の変換（値呼び $\lambda\mu$ 計算）の含意断片に対する F の定義は文献 [F95] にある．そして，コルモゴロフの変換（名前呼び変換，Φ）では，名前呼び計算 $\to_{\mathtt{N}}^*$ が値呼び計算 $\to_{\mathtt{V}}^*$ により模倣されるという Plotkin の模倣定理を紹介する．

模倣定理（Plotkin [Plot75]）$\mathtt{Eval}_{\mathtt{V}}(\underline{M}\ \mathtt{id}) = \Phi(\mathtt{Eval}_{\mathtt{N}}(M))$

$$\begin{array}{ccc} M & \to_{\mathtt{N}}^* & V \\ \Downarrow & & \Downarrow \\ \underline{M}\ \mathtt{id} & \to_{\mathtt{V}}^* & \Phi(V) \end{array}$$

元々この模倣定理は型のないラムダ計算の体系で得られた結果であるが，型を付けると Kolmogorov negative translation が浮かび上がってくる．また，黒田の変換（値呼び変換，Ψ）では，値呼び計算が名前呼び計算で模倣されることになる [Plot75]．

他方では，古典論理における双対性を CPS 変換を活用して特徴付ける仕事がある [Sel01, Wad03]．ここでは，古典論理に限定しない双対性の研究 [F10] に関係する結果をまとめる．

Summary

This tutorial consists of three talks respectively devoted to the following subjects based on the references.

1. Curry-Howard-deBruijn isomorphism [TD88, NPS90]:

   The isomorphism provides an interpretation of compound formulae in terms of proofs, and we review the isomorphism from the viewpoints below.

   - Propositions-as-types and proofs-as-programs
   - BHK-interpretation (Brouwer-Heyting-Kolmogorov)
   - Programming in Martin-Löf's Type Theory

We note that following Martin-Löf's explanation [NPS90], the notions of evaluation and values (canonical expressions) are indispensable in particular to the correspondence of types-as-specifications (proofs-as-programs), compared with other correspondence. The notions of evaluation and values can be formally handled by corresponding CPS-translations.

Remarked that for formal variants of BHK, see the notion of realizability interpretation [TD88] such as Kleene: realizer for proof and partial recursive operation for construction; and Feferman: interpretation in type-free theories called APP consisting of k and s combinators.

2. Correspondence between type system (simply typed λ-calculus) and logic (minimal logic with implication):

   We review the following correspondence by using simple examples [F22].

   - λ-terms (Programs) as proofs
   - Type checking as proof checking
   - Computation rule as proof reduction
   - Program translation as proof translation
   - Call-by-Value CPS-translation as Kuroda's negative translation
   - Call-by-Name CPS-translation as Kolmogorov's negative translation

   Remarked that for more general frameworks, take the notion of PTS (Pure Type System).

3. Negative translations and CPS-translations [TD88, Plot75]:

   We summarize four embeddings and variants from classical logic into intuitionistic or minimal logic in terms of $\lambda\mu$-calculus, which reveals program translations depending on evaluation strategy.

   - Embedding: Gödel-Gentzen, Glivenko, Kolmogorov, and Kuroda
   - CPS-translation: Call-by-value and call-by-name

   For the implicational fragment of classical logic, the definitions of four embeddings are given by [F95] in terms of call-by-name and call-by-value $\lambda\mu$-calculi.

   We should note that CPS-translations are originally investigated for semantics and interpretation of programs in general, and Plotkin [Plot75] established the theorems of simulation between call-by-value and call-by-name evaluations via CPS-translations of type-free λ-terms. The CPS-translations for type-free λ-terms are still available even for typed cases, so that attached types reveal negative translations depending on evaluation strategy, i.e., program translations as proof translations. This correspondence between programs and proofs can be naturally extended to classical logic, for which $\lambda\mu$-calculus is devised to represent proof terms.

Remarked that CPS-translation is also applied to analyze duality of input-output and evaluations in classical logic by Selinger [Sel01] and Wadler [Wad03]. Moreover, we show that CPS-translations work not only for classical logic but also for intuitionistic logic to investigate duality [F10].

1 Curry-Howard-deBruijn isomorphism

The slogan is owing to Martin-Löf's Type Theory [NPS90]:

$$\text{Sets} = \text{Propositions} = \text{Specifications}$$

In his type theory the judgement

$$a \in A$$

can be read in at least the following ways:

- a is an element in the set A.
- a is a proof object for the proposition A.
- a is a solution to the problem A.
- a is a program satisfying the specification A.

We should note that the notions of evaluation, computation, execution; and value, canonical expression are indispensable to the last reading, which comes to us again formally via CPS-translations.

1.1 BHK interpretation

According to BHK interpretation (propositions as sets of proofs), a constructive explanation of propositions is given in terms of proofs as follows:

- A proof of $A \to B$ is a function (method, program) which to each proof of A gives a proof of B. The set of functions from A to B can be written by
$$(A \to B) \equiv \Pi(A, (x)B).$$
- A proof of $A \wedge B$ is a pair whose first component is a proof of A and whose second component is a proof of B. The cartesian product of A and B is now can be written by
$$(A \wedge B) \equiv (A \times B) \equiv \Sigma(A, (x)B).$$
- A proof of $A \vee B$ is either a proof of A or a proof of B with the information of which of A or B we have a proof. The disjoint union of A and B can be written by
$$(A \vee B) \equiv (A + B).$$

- For $\bot$ (Absurdity), we have no proof, so that $\bot$ can be written by the empty set. Then a proof of $\neg A$ constructs a proof of a contradiction from a hypothetical proof of A.

According to Heyting (1956), quantified propositions can be explained by sets of proofs as follows:

- A proof of $(\exists x \in A)B(x)$ is a pair whose first component a is an element in the set A and whose second component is a proof of $B(a)$. The disjoint union of a family of sets $(\Sigma x \in A)B(x)$ can be written by

$$(\exists x \in A)B(x) \equiv (\Sigma x \in A)B(x) \equiv \Sigma(A, B).$$

- A proof of $(\forall x \in A)B(x)$ is a function which to each element a in the set A gives a proof of $B(a)$. The cartesian product of a family of sets $(\Pi x \in A)B(x)$ can be written by

$$(\forall x \in A)B(x) \equiv (\Pi x \in A)B(x) \equiv \Pi(A, B).$$

Similarly, according to Kolmogorov (1932), if A and B are tasks (problems) then propositions can be explained as tasks (problems) as follows:

- $A \wedge B$ is the task of solving the tasks A and B.
- $A \vee B$ is the task of solving at least one of the tasks A and B.
- $A \to B$ is the task of solving the task B under the assumption that we have a solution of A.

Finally, if A and B are specifications of a program, then propositions can be explained as specifications of programs as follows, together with the notion of execution:

- $A \wedge B$ is a specification of programs which, when executed, yield a pair $\langle a, b \rangle$, where a is a program for the specification of A and b is a program for the specification of B.
- $A \vee B$ is a specification of programs which, when executed, either yields $\mathtt{inl}(a)$ or $\mathtt{inr}(b)$, where a is a program for the specification of A and b is a program for the specification of B.
- $A \to B$ is a specification of programs which, when executed, yields $\lambda x.b(x)$, where $b(x)$ is a program for the specification of B under the assumption that x is a program for the specification of A.
- $(\forall x \in A)B$ is a specification of programs which, when executed, yields $\lambda x.b(x)$, where $b(x)$ is a program for the specification of $B(x)$ under the assumption that x is an object of A.
- $(\exists x \in A)B$ is a specification of programs which, when executed, yields $\langle a, b \rangle$, where a is an object of A and b is a program for the specification of $B(a)$.

1.2 Programming in Martin-Löf's Type Theory

As a formal example of propositions-as-specifications, we cite Martin-Löf's Type Theory [NPS90]. The judgement forms of the theory have one of the following four forms:

1. A *set* (A is a set)

   To know that A is a set is to know how to form the *canonical* elements in the set and under what conditions two canonical elements are *equal*.

2. $A = B$ (A and B are equal sets)

   To know that two sets, A and B, are equal is to know that a canonical element in the set A is also a canonical elements in the set B and, moreover, equal canonical elements of the set A are also equal canonical elements of the set B, and vice versa.

3. $a \in A$ (a is an element in the set A)

   If A is a set then to know that $a \in A$ is to know that a, when evaluated, yields a canonical element in A as a value.

4. $a = b \in A$ (a and b are equal elements in the set A)

   To know that a and b are equal elements in the set A is to know that they yield equal canonical elements in the set A as a value.

We note that the *canonical* expressions are the values of expressions under evaluation by computation rules, called equality rules below.

We show an example of natural numbers, where four types of rules are generally defined, respectively called formation, introduction, elimination, and equality rules. Here, we use the constant $\mathbf{N}$ of arity $\mathbf{0}$; and canonical constants $\mathtt{0}$ and $\mathtt{succ}$ of arities $\mathbf{0}$ and $\mathbf{0} \twoheadrightarrow \mathbf{0}$, respectively.

- $\mathbf{N}$-formation

$$\mathbf{N}\ \mathit{set}$$

- $\mathbf{N}$-introduction 1

$$\mathtt{0} \in \mathbf{N}$$

- $\mathbf{N}$-introduction 2

$$\frac{a \in \mathbf{N}}{\mathtt{succ}(a) \in \mathbf{N}}$$

We often write $\mathtt{1}$ for $\mathtt{succ}(0)$, $\mathtt{2}$ for $\mathtt{succ}(1)$, and so on. We should note that $\mathtt{0}$, $\mathtt{succ}(0)$, $\mathtt{succ}(1)$, $\ldots \in \mathbf{N}$ are canonical elements in $\mathbf{N}$.

We also use the constant $\mathtt{natrec}$ which has arity $\mathbf{0} \otimes \mathbf{0} \otimes (\mathbf{0} \otimes \mathbf{0} \twoheadrightarrow \mathbf{0}) \twoheadrightarrow \mathbf{0}$.

- $\mathbf{N}$-elimination

$$\frac{b \in \mathbf{N} \quad d \in C(0) \quad \begin{array}{c}[v \in \mathbf{N}] \\ C(v)\ \mathit{set}\end{array} \quad \begin{array}{c}[x \in \mathbf{N}, y \in C(x)] \\ e(x,y) \in C(\mathtt{succ}(x))\end{array}}{\mathtt{natrec}(b, d, e) \in C(b)}$$

We note that the value of $\mathtt{natrec}(b, d, e)$ is a canonical element in $C(b)$, as follows:

1. Case the value of b is $\mathtt{0}$:

   The value of $\mathtt{natrec}(b, d, e)$ is the vale of d in $C(\mathtt{0})$. The value of d is a canonical element in $C(\mathtt{0})$ by the second premise.

2. Case the value of b is $\mathtt{succ}(a)$ with $a \in \mathbf{N}$:

   The value of $\mathtt{natrec}(b, d, e)$ is the value of $e(a, \mathtt{natrec}(a, d, e))$ from $\mathtt{natrec}(a, d, e) \in C(a)$. The value of $e(a, \mathtt{natrec}(a, d, e))$ is a canonical element in $C(\mathtt{succ}(a))$ from $\mathtt{natrec}(a, d, e) \in C(a)$ by the fourth premise.

The above explanation can be justified by the following equality rules, which means the computation rule for $\mathtt{natrec}$:

- $\mathbf{N}$-equality 1

$$\frac{\begin{array}{cc}[v \in \mathbf{N}] & [x \in \mathbf{N}, y \in C(x)] \\ C(v)\ set \quad d \in C(0) & e(x, y) \in C(\mathtt{succ}(x))\end{array}}{\mathtt{natrec}(0, d, e) = d \in C(0)}$$

- $\mathbf{N}$-equality 2

$$\frac{\begin{array}{ccc}[v \in \mathbf{N}] & & [x \in \mathbf{N}, y \in C(x)] \\ C(v)\ set \quad a \in \mathbf{N} & d \in C(0) & e(x, y) \in C(\mathtt{succ}(x))\end{array}}{\mathtt{natrec}(\mathtt{succ}(a), d, e) = e(a, \mathtt{natrec}(a, d, e)) \in C(\mathtt{succ}(a))}$$

We show an example from Peano's third axiom:

$$\frac{m \in \mathbf{N} \quad n \in \mathbf{N} \quad \mathtt{succ}(m) = \mathtt{succ}(n) \in \mathbf{N}}{m = n \in \mathbf{N}}$$

First define the predecessor:

$$pred \equiv (x)\mathtt{natrec}(x, \mathtt{0}, (u, v)u).$$

Let $x \in \mathbf{N}$. Then the definition of *pred* and $\mathbf{N}$-elimination give

$$pred(x) \in \mathbf{N}. \tag{1}$$

Let $m \in \mathbf{N}$, $n \in \mathbf{N}$ and

$$\mathtt{succ}(m) = \mathtt{succ}(n) \in \mathbf{N}. \tag{2}$$

From (1), (2) and Substitution in equal elements, we have:

$$pred(\mathtt{succ}(m)) = pred(\mathtt{succ}(n)) \in \mathbf{N}.$$

The definition of *pred* and **N**-equality 2 give:

$$pred(\mathtt{succ}(m)) = m \in \mathbf{N},\ pred(\mathtt{succ}(n)) = n \in \mathbf{N}.$$

From symmetry and transitivity of judgemental equality, we finally obtain:

$$m = n \in \mathbf{N}.$$

We show another example of cartesian product of a family of sets. We use constants Π and λ, which have arities $\mathbf{0} \otimes (\mathbf{0} \twoheadrightarrow \mathbf{0}) \twoheadrightarrow \mathbf{0}$, $(\mathbf{0} \twoheadrightarrow \mathbf{0}) \twoheadrightarrow \mathbf{0}$, respectively. The formation and introduction rules are defined as follows:

- Π-formation

$$\frac{A\ set \quad \begin{array}{c}[x \in A] \\ B(x)\ set\end{array}}{\Pi(A, B)\ set}$$

- Π-introduction

$$\frac{\begin{array}{c}[x \in A] \\ b(x) \in B(x)\end{array}}{\lambda(b) \in \Pi(A, B)}$$

We should note that the canonical elements in the set $\Pi(A, B)$ are of the form $\lambda(b)$ where $b(x) \in B(x)$ with $x \in A$. For the Π-set, a non-canonical constant `funsplit` is introduced, which has arity $(\mathbf{0} \otimes ((\mathbf{0} \twoheadrightarrow \mathbf{0}) \twoheadrightarrow \mathbf{0})) \twoheadrightarrow \mathbf{0}$. The elimination and equality rules are defined as follows:

- Π-elimination

$$\frac{f \in \Pi(A, B) \quad \begin{array}{c}[v \in \Pi(A, B)] \\ C(v)\ set\end{array} \quad \begin{array}{c}[x \in A] \\ [y(x) \in B(x)] \\ d(y) \in C(\lambda(y))\end{array}}{\mathtt{funsplit}(f, d) \in C(f)}$$

- Π-equality

$$\frac{\begin{array}{c}[x \in A] \\ b(x) \in B(x)\end{array} \quad \begin{array}{c}[v \in \Pi(A, B)] \\ C(v)\ set\end{array} \quad \begin{array}{c}[w \in A] \\ [y(w) \in B(w)] \\ d(y) \in C(\lambda(y))\end{array}}{\mathtt{funsplit}(\lambda(b), d) = d(b) \in C(\lambda(b))}$$

Now `apply` can be defined by the following way:

$$\mathtt{apply}(f, a) \equiv \mathtt{funsplit}(f, (x)(x(a))) \in B(a).$$

Indeed, an example of the so-called proof detour consisting of successive $(\forall I)$ and $(\forall E)$ can be reduced as follows:

$$\mathtt{apply}(\lambda(b), a) = ((x)(x(a)))b = b(a) \in B(a).$$

2 Correspondence between type system and logic: simply typed λ-calculus and minimal logic with implication

The key point of the second talk is how to evaluate programs, and transformation of evaluation strategy of programs can be given by the so-called CPS-translations. Based on the Curry-Howard isomorphism, this means that program translation can be read as proof translation, some of which are known as logical embedding via negative translations:

- Call-by-value translation as Kuroda's negative translation
- Call-by-name translation as Kolmogorov's negative translation

Originally, CPS-translations are investigated for denotational semantics of programs in general. For instance, consider the following simple example of procedural program:

$\texttt{int}\ x, i;$
$\texttt{if}\ x = 0\ \texttt{then}\ x := 1;$
$i := x - 1;$
$\texttt{while}\ i \geq 2\ \texttt{do}\ (x := x * i;\ i := i - 1;);$

A naive translation from procedural programs to functional programs provides transformation of state transition semantics to denotational semantics, so that the following recursive program with two arguments can be obtained from the above program:

$\texttt{factorial}(x) \stackrel{\text{def}}{=} \texttt{if}\ x = 0\ \texttt{then}\ (1, 0)\ \texttt{else}\ f(x, x - 1)$
where $f(x, i) = \texttt{if}\ i \geq 2\ \texttt{then}\ f(x * i, i - 1)\ \texttt{else}\ (x, i).$

Here, we note that the recursive definition of f can be regarded as a function which is derived from the definition of factorial ! by CPS-translation:

$$\begin{aligned} 0! &= 1, \\ (n+1)! &= (n+1) * n! \end{aligned}$$

The slogan of the second part is given as follows:

- Proofs-as-programs:
  Computation rule = proof reduction,
  Program translation = proof translation.

For the core fragment, we take the system of natural deduction $\Gamma \vdash_{\mathrm{S}} M : A$, where S is NJ($\to$) or minimal logic consisting of the following inference rules:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\mathrm{Var})$$

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : (A \to B)}\ (\to I)$$

$$\frac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}\ (\to E)$$

For instance, one can derive

$$\vdash (\lambda x.\lambda y.x) : (A \to (B \to A))$$

where we adopt proof terms in the style of Curry, instead of Church-style:

$$(\lambda x{:}A.\lambda y{:}B.x) : (A \to (B \to A)).$$

Another example is that there exist no proof terms for $(((A \to B) \to A) \to A)$ under the system.

The definitions of negative translations of formulae are given as follows:

1. Kuroda's translation q:
   $X^q = X$, $(A \to B)^q = (A^q \to \neg\neg B^q)$.
2. Kolmogorov's translation k:
   $X^k = \neg\neg X$, $(A \to B)^k = \neg\neg(A^k \to \neg\neg B^k)$.

2.1 Recursion, iteration, and CPS-form

We introduce simple examples of recursive functions in distinct forms. First take the following definition of $\mathtt{fact} : \mathtt{int} \to \mathtt{int}$

$$\begin{aligned} \mathtt{fact}(0) &= 1 \\ \mathtt{fact}(n+1) &= (n+1) * \mathtt{fact}(n), \end{aligned}$$

and observe elementary computation steps including a recursive call of $\mathtt{fact}(3)$

$$\begin{aligned} \mathtt{fact}(3) &= 3 * \mathtt{fact}(2) \\ &= 3 * (2 * \mathtt{fact}(1)) \\ &= 3 * (2 * (1 * \mathtt{fact}(0))) \\ &= 3 * (2 * (1 * 1)). \end{aligned}$$

Here, each recursive call is executed under the contexts with the so-called hole $[\,]$:

$$[\,],\ 3 * [\,],\ 3 * (2 * [\,])),\ 3 * (2 * (1 * [\,])).$$

The contexts in which `fact` is called are often refereed to as evaluation contexts for the computation, and also as continuations with respect to each recursive call. Then the definition of `fact` can be rewritten with an extra argument, such

that the evaluation contexts are handled as one of the arguments of `fact-it` : $\texttt{int} \times \texttt{int} \to \texttt{int}$ as follows:

$$\begin{array}{rcl} \texttt{fact-it}(0, k) & = & k, \\ \texttt{fact-it}(n+1, k) & = & \texttt{fact-it}(n, k * (n+1)). \end{array}$$

This form of recursive definitions is often called iteration. The evaluation context with a hole can be regarded as a function waiting for a returned value, and then we can associate evaluation contexts (continuations) to λ-terms, for instance, $(n+1) * [\,]$ to $\lambda v.(n+1) * v$. In this way, the iterative form can be redefined in the functional form `fact-cps` : $\texttt{int} \times (\texttt{int} \to \texttt{int}) \to \texttt{int}$ as follows:

$$\begin{array}{rcl} \texttt{fact-cps}(0, f) & = & f(1), \\ \texttt{fact-cps}(n+1, f) & = & \texttt{fact-cps}(n, \lambda v.f((n+1) * v)). \end{array}$$

For any natural number n we have the equation:

$$\texttt{fact-cps}(n, f) = f(\texttt{fact}(n)),$$

and then $\texttt{fact}(n)$ can be defined by $\texttt{fact-cps}(n, \lambda x.x)$ where the identity function $\lambda x.x$ means the evaluation context $[\,]$, i.e., no task remains. Now, observe the following proof of the equation by induction on n where case n of $k+1$:

$$\begin{array}{rcl} \texttt{fact-cps}(k+1, f) & = & \texttt{fact-cps}(k, \lambda v.f((k+1) * v)) \\ & =_{ih} & (\lambda v.f((k+1) * v))(\texttt{fact}(k)) \\ & \to_{\beta} & f((k+1) * \texttt{fact}(k)) \\ & = & f(\texttt{fact}(k+1)). \end{array}$$

Here, we can read the elementary computation steps such that the function `fact-cps` sends the value of $\texttt{fact}(k+1)$ to its continuation f, and in other words, the continuation (function) f is waiting for and accepting the returned value of $\texttt{fact}(k+1)$. And the value of $\texttt{fact}(k)$ is sent to its continuation $\lambda v.f((k+1) * v)$ as well, and the continuation can be read as accepting the value of $\texttt{fact}(k)$. That is, the cps-form sends an evaluated value to its continuation, and the continuation accepts the expected value.

Under this kind of reading, next we consider examples of translations of a function to its cps-form. First, take an example of an unary function F, and write $\overline{F}$ for its cps-form with an extra argument. In the following, we take Curried functions, e.g., $\overline{\texttt{fact}} : \texttt{int} \to (\texttt{int} \to \texttt{int}) \to \texttt{int}$ for $\texttt{fact} : \texttt{int} \to \texttt{int}$, instead of $\texttt{fact-cps} : \texttt{int} \times (\texttt{int} \to \texttt{int}) \to \texttt{int}$.

- $\overline{F(m)+n}\ K \ \Rightarrow\ \overline{F}\ m\ (\lambda v.K(v+n))$

  1. Consider the case where $F(m)+n$ is evaluated under the continuation K.
  2. First, evaluate the value of $F(m)$, to say v, which is executed under its continuation $\lambda v.K(v+n)$.

3. $\overline{F}$ with two arguments sends the value v of $F(m)$ to the continuation $\lambda v.K(v+n)$, so that the continuation K accepts the sum of v and n.

The second example has two unary functions F and G, and then write $\overline{F}$ and $\overline{G}$ for their cps-forms with two arguments, respectively.

- $\overline{F(G(m))+n}\ K\ \Rightarrow\ \overline{G}\ m\ (\lambda v_1.\overline{F}\ v_1\ (\lambda v_2.K(v_2+n)))$

1. Evaluate $F(G(m))+n$ under the continuation K.
2. First, evaluate the value of $G(m)$, to say v_1, and next evaluate the value of $F(v_1)$, to say v_2, which is executed under its continuation $\lambda v_2.K(v_2+n)$.
3. $\overline{G}$ with two arguments sends the value v_1 of $G(m)$ to its continuation $\lambda v_1.\overline{F}\ v_1\ (\lambda v_2.K(v_2+n))$.
4. $\overline{F}$ accepts the value v_1, and sends the value v_2 to its continuation $\lambda v_2.K(v_2+n)$, so that the continuation K accepts the sum of v_2 and n, which is the expected computation.

In this way, we evaluate λ-terms with β-reduction:

$$(\lambda x.M_1)M_2 \rightarrow_\beta M_1[x := M_2].$$

Two reduction strategies are well-known as call-by-name and call-by-value.

- Call-by-name evaluation of $(M\ N)$:

1. First evaluate M, and then let us call the result $\lambda x.M'$.
2. Next execute β-reduction $M'[x := N]$, and continue this evaluation.

- Call-by-value evaluation of $(M\ N)$:

1. First evaluate M, and then let us call the result $\lambda x.M'$.
2. Next evaluate N, and then call the result V.
3. Then execute β-reduction $M'[x := V]$, and continue this evaluation.

Observe cps-form $\overline{MN}$ of an application term (MN) under the call-by-value evaluation.

- $\overline{M\ N}\ K\ \Rightarrow\ \overline{M}\ (\lambda v_1.\overline{N}\ (\lambda v_2.v_1 v_2\ K))$

(1) Evaluate (MN) under the continuation K.

(2) First evaluate the value of M, to say v_1, under its continuation $(\lambda v_1.\overline{N}\ (\lambda v_2.v_1 v_2\ K))$

(3) Next evaluate the value of N, to say v_2, under its continuation $(\lambda v_2.v_1 v_2\ K)$.

(4) $\overline{M}$ sends the value v_1 of M to the continuation $\lambda v_1.\overline{N}\ (\lambda v_2.v_1 v_2\ K)$.

(5) The continuation $\lambda v_1.\overline{N}\ (\lambda v_2.v_1 v_2\ K)$ accepts the value of M.

(6) $\overline{N}$ sends the value v_2 of N to the continuation $\lambda v_2.v_1 v_2\ K$.

(7) The continuation $\lambda v_2.v_1 v_2\ K$ accepts the value of N, from which M and N are evaluated under the continuation K by call-by-value.

We refer to the explanation numbers $(1), (2), \cdots, (7)$ for the formal computation steps in the next subsection.

2.2 Call-by-value translation as Kuroda's embedding, Call-by-name translation as Kolmogorov's embedding

In the previous subsection, we observed λ-terms transformed under the strategy of call-by-value. These kinds of cps-forms are originally investigated in type-free λ-calculi [Plot75]. Under call-by-value, it is commonly considered as values that the terms in the form of variables or λ-abstractions, so that the continuation K can accept variables and λ-abstraction directly. Now the obtained cps-form can be rewritten by using λ-abstraction over the variable k as follows. Then putting type information on terms reveals the embedding of Kuroda.

- Call-by-value CPS translation:

  1. $\overline{x} = \lambda k.kx$
  2. $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$
  3. $\overline{MN} = \lambda k.\overline{M}(\lambda m.\overline{N}(\lambda n.mnk))$

  where we write $\overline{V} = \lambda k.k\Psi(V)$ such that $\Psi(x) = x$ and $\Psi(\lambda x.M) = \lambda x.\overline{M}$ for a value V which is either x or $\lambda x.M$.

- Kuroda's negative translation q:
  If $\Gamma \vdash_{\mathrm{NJ}(\to)} M : A$ then $\Gamma^q \vdash_{\mathrm{Min}} \overline{M} : \neg\neg A^q$.

Now observe the computation of call-by-value β_V-reduction:

$$(\lambda x.M)V \to_{\beta_V} M[x := V],$$

which can be simulated by this translation with continuation K, as follows:

$$\begin{array}{lcll}
\overline{(\lambda x.M)V}\ K & \to_\beta & \overline{\lambda x.M}(\lambda m.\overline{V}(\lambda n.mnK)) & (1)\\
& = & (\lambda k.k(\lambda x.\overline{M}))(\lambda m.\overline{V}(\lambda n.mnK)) & (2),\ (3)\\
& \to_\beta & (\lambda m.\overline{V}(\lambda n.mnK))(\lambda x.\overline{M}) & (4)\\
& \to_\beta & \overline{V}(\lambda n.(\lambda x.\overline{M})nK) & (5)\\
& = & (\lambda k.k(\Psi(V)))(\lambda n.(\lambda x.\overline{M})nK) & \text{by definition of } \Psi\\
& \to_\beta & (\lambda n.(\lambda x.\overline{M})nK)(\Psi(V)) & (6)\\
& \to_\beta & (\lambda x.\overline{M})(\Psi(V))K & (7)\\
& \to_\beta & \overline{M}[x := \Psi(V)]K & \\
& = & \overline{M[x := V]}K & \text{by simple induction.}
\end{array}$$

The computation steps are indeed justified by explanations $(1), (2), \ldots, (7)$ given in the previous subsection.

We should remark that the following variant is available for the translation as well

$$\overline{MN} = \lambda k.\overline{N}(\lambda n.\overline{M}(\lambda m.mnk)),$$

which means that the notion of CPS-translation is so sensitive on evaluation order.

Similarly, the call-by-name evaluation can be rewritten by the following translation, which together with type information, provides the embedding of Kolmogorov.

- Call-by-name CPS translation:
  1. $\underline{x} = \lambda k.xk$
  2. $\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$
  3. $\underline{MN} = \lambda k.\underline{M}(\lambda m.m\underline{N}k)$
- Kolmogorov's negative translation k:
  If $\Gamma \vdash_{\mathrm{NJ}(\to)} M : A$ then $\Gamma^k \vdash_{\mathrm{Min}} \underline{M} : A^k$.

Now the reduction $(\lambda x.M)N \to M[x := N]$ can be simulated by this call-by-name CPS translation as follows:

$$\begin{array}{lll}
\underline{(\lambda x.M)N} & = & \lambda k.(\lambda k'.k'(\lambda x.\underline{M}))(\lambda m.m\overline{N}k) \\
& \to_\beta & \lambda k.(\lambda m.m\underline{N}k)(\lambda x.\underline{M}) \\
& \to_\beta & \lambda k.(\lambda x.\underline{M})\underline{N}k \\
& \to_\beta & \lambda k.\underline{M}[x := \underline{N}]k \\
& \to_\beta^* & \lambda k.\underline{M[x := N]}k \\
& \to_\beta & \underline{M[x := N]},
\end{array}$$

where each computation step can be justified by the call-by-value β_V-reduction, which leads to the theorem of simulation of call-by-name computation under call-by-value computation.

2.3 Overview of Curry-Howard correspondence

The corresponding notions between programs and proofs are summarized in the following Table 1 [F22].

Moreover, we can adopt extended λ-calculi and 2nd order logics so that polymorphic functions are formalized as proofs of 2nd order $\forall$-formulae and abstract data types are as proofs of 2nd order $\exists$-formulae [F10].

Simply Typed λ-calculus	$\mathbf{NJ}(\to)$
type	formula
function space	implication
λ-term	proof figure
bound variable	discharged assumption
free variable	open assumption
function abstraction	introduction rule $(\to I)$
function application	elimination rule $(\to E)$
redex	cut formula
reduction (computation rule)	proof reduction
normal form	normal proof
type check	checking proof of formula
type inference	generating formula from proofs
CPS-translation (program translation) (Call-by-Name/Call-by-Value CPS trans.)	embedding (negative translation) (Kolmogorov/Kuroda embedding)

Table 1: Corresponding notions between programs and proofs [F22]

3 Embedding from classical logic into intuitionistic or minimal logic

The third part is devoted to four embeddings and their variants from classical logic into intuitionistic or minimal logic in terms of $\lambda\mu$-calculus. For this, we introduce the system of $\lambda\mu$-calculus with call-by-name reduction and call-by-value reduction rules.

- Inference rules of $\lambda\mu$-calculus

$$\frac{x{:}A \in \Gamma}{\Gamma; \neg\Delta \vdash x : A}\ (\text{Var})$$

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma; \neg\Delta \vdash \lambda x.M : (A \to B)}\ (\to I) \qquad \frac{\Gamma \vdash M : (A \to B) \quad \Gamma; \neg\Delta \vdash N : A}{\Gamma; \neg\Delta \vdash (MN) : B}\ (\to E)$$

$$\frac{\Gamma; \neg\Delta \vdash M : A \quad \Gamma; \neg\Delta \vdash N : B}{\Gamma; \neg\Delta \vdash \langle M, N\rangle : A \wedge B}\ (\wedge I) \qquad \frac{\Gamma; \neg\Delta \vdash M : A_1 \wedge A_2}{\Gamma; \neg\Delta \vdash \mathtt{proj}_i(M) : A_i}\ (\wedge E)$$

where we may write `fst` and `snd` instead of $\mathtt{proj}_1$ and $\mathtt{proj}_2$, respectively.

$$\frac{\Gamma; \neg\Delta \vdash M : A_i}{\Gamma; \neg\Delta \vdash \mathtt{inj}_i(M) : A_1 \vee A_2}\ (\vee I)$$

where we may write `inl` and `inr` instead of $\mathtt{inj}_1$ and $\mathtt{inj}_2$, respectively.

$$\frac{\Gamma; \neg\Delta \vdash M : A \vee B \quad \Gamma, x{:}A; \neg\Delta \vdash N_1 : C \quad \Gamma, y{:}B; \neg\Delta \vdash N_2 : C}{\Gamma; \neg\Delta \vdash \mathtt{when}(M, x.N_1, y.N_2) : C}\ (\vee E)$$

$$\frac{\Gamma; \neg\Delta \vdash M : A}{\Gamma; \neg\Delta \vdash \lambda \mathtt{x}.M : \forall x.A}\ (\forall I)^* \qquad \frac{\Gamma; \neg\Delta \vdash M : \forall x.A}{\Gamma; \neg\Delta \vdash Mt : A[x := t]}\ (\forall E)$$

where $(\forall I)^*$ denotes the eigenvariable condition $x \notin \mathrm{FV}(\Gamma, \Delta)$.

$$\frac{\Gamma; \neg\Delta \vdash M : A[x := t]}{\Gamma; \neg\Delta \vdash (t, M) : \exists x.A}\ (\exists I)$$

$$\frac{\Gamma; \neg\Delta \vdash M : \exists x.A \quad \Gamma, z : A; \neg\Delta \vdash N : C}{\Gamma; \neg\Delta \vdash \mathtt{split}(M, z.N) : C}\ (\exists E)$$

where $(\exists E)^*$ denotes the eigenvariable condition $z \notin \mathrm{FV}(\Gamma, \Delta, C)$.

$$\frac{\Gamma; \neg\Delta, \alpha : \neg A \vdash M : \bot}{\Gamma; \neg\Delta \vdash \mu\alpha.M : A}\ (\bot_C) \qquad \frac{\Gamma; \neg\Delta \vdash M : A}{\Gamma; \neg\Delta, \alpha : \neg A \vdash [\alpha]M : \bot}\ (\mathrm{Name})$$

We may sometimes use another form of $(\wedge E)$:

$$\frac{\Gamma; \neg\Delta \vdash N : A \wedge B \quad \Gamma, x : A, y : B; \neg\Delta \vdash M : C}{\Gamma; \neg\Delta \vdash \mathtt{let}\ \langle x, y\rangle = N\ \mathtt{in}\ M : C}\ (\wedge E)$$

We also adopt a dual form of implication $\leftarrow$ in particular for an intuitionistic fragment:

$$\frac{\Gamma; \neg\Delta \vdash M : B \quad \Gamma, x : A; \neg\Delta \vdash N : \bot}{\Gamma; \neg\Delta \vdash \mathtt{maL}(M, x.N) : (A \leftarrow B)}\ (\leftarrow I)$$

$$\frac{\Gamma; \neg\Delta \vdash M : (A \leftarrow B) \quad \Gamma, y : B; \neg\Delta \vdash P : A}{\Gamma; \neg\Delta \vdash \mathtt{ppA}(M, y.P) : \bot}\ (\leftarrow E)$$

- Call-by-name reduction rules of $\lambda\mu$-calculus

1. $(\lambda x.M)N \to M[x := N]$
2. $\mathtt{proj}_i\langle M_1, M_2\rangle \to M_i\ (i = 1, 2)$
3. $\mathtt{when}(\mathtt{inj}_i(M), x_1.N_1, x_2.N_2) \to N_i[x_i := M]\ (i = 1, 2)$
4. $(\lambda \mathtt{x}.M)t \to M[x := t]$
5. $\mathtt{split}((M, t), z.N) \to N[z := M]$
6. $(\mu\alpha.M)N \to \mu\alpha.M[\alpha \Leftarrow N]$, where $M[\alpha \Leftarrow N]$ denotes a term such that each subterm in the form of $[\alpha]N'$ in M is replaced with $[\alpha](N'N)$.
7. $\mu\alpha.[\alpha]M \to M$ with $\alpha \notin FV(M)$
8. $[\beta](\mu\alpha.M) \to M[\alpha := \beta]$
9. $\mathtt{let}\ \langle x, y\rangle = \langle N_1, N_2\rangle\ \mathtt{in}\ M \to M[x := N_1, y := N_2]$
10. $\mathtt{ppA}(\mathtt{maL}(M, x.N), y.P) \to N[x := P[y := M]]$

- Call-by-value reduction rules of $\lambda\mu$-calculus

1. $(\lambda x.M)V \to M[x := V]$
2. $\mathtt{ppA}(\mathtt{maL}(V_1, x.N), y.V_2) \to N[x := V_2][y := V_1]$
3. $V(\mu\alpha.M) \to \mu\alpha.M[V \Rightarrow \alpha]$, where $M[V \Rightarrow \alpha]$ denotes a term such that each subterm in the form of $[\alpha]N$ in M is replaced with $[\alpha](VN)$.
4. Call-by-name reduction rules on which value restrictions are imposed, where the notion of values may depend on concrete systems.

In this section, we summarize the following embeddings from the viewpoint of reduction strategy, equational correspondence, reduction correspondence, and/or duality of formulae.

1. Glivenko
2. Gödel-Gentzen
3. Kolmogorov
4. Kuroda

3.1 Glivenko's theorem *Gl*

For propositional logic, it is well-known that Glivenko's theorem holds:

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma \vdash_{\mathrm{NJ}} \neg\neg A.$$

In order to establish the same statement for predicate logic, one may add the so-called DNS (Glivenko, Gödel) $\forall x\neg\neg A \to \neg\neg\forall x A$ to NJ:

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma \vdash_{\mathrm{NJ+DNS}} \neg\neg A.$$

However, due to DeMorgan's law $\forall x A \leftrightarrow \neg\exists x\neg A$, one can replace each $\forall x A$ with $\neg\exists x\neg A$. Then we have the following statement even for predicate logic:

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma \vdash_{\mathrm{NJ}} \neg\neg A.$$

We note that this form of Glivenko's theorem holds as well for 2nd order classical propositional logic and 2nd order intuitionistic propositional logic. This embedding can be applied to obtain decidable fragments of 2nd order intuitionistic propositional logic, where 2nd order intuitionistic propositional logic is in general undecidable for the fragment of $(\forall^2, \to)$. On the other hand, it is known that 2nd order intuitionistic propositional logic becomes decidable for the fragment of $(\forall^2, \exists^2, \neg, \wedge)$.

Based on DeMorgan's law, we show Glivenko's theorem for predicate logic.

Theorem 1 (Glivenko)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$ *then* $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NJ}} Gl(M) : \neg\neg A$.

Here, $Gl(M)$ is given as a translation from classical logic into the full intuitionistic logic in terms of $\lambda\mu$-terms as follows:

1. $Gl(x) = \lambda k.kx$
2. $Gl(\lambda x.M) = \lambda k.k(\lambda x.\mu\beta.Gl(M)(\lambda m.k(\lambda v.m)))$;
   $Gl(MN) = \lambda k.Gl(M)(\lambda m.Gl(N)(\lambda n.k(mn)))$
3. $Gl(\langle M, N\rangle) = \lambda k.Gl(M)(\lambda m.Gl(N)(\lambda n.k\langle m, n\rangle))$;
   $Gl(\mathtt{fst}(M)) = \lambda k.Gl(M)(\lambda m.k(\mathtt{fst}(m)))$
4. $Gl(\mathtt{inl}(M)) = \lambda k.Gl(M)(\lambda m.k(\mathtt{inl}(m)))$;
   $Gl(\mathtt{when}(M, x.N_1, y.N_2)) = \lambda k.Gl(M)(\lambda m.\mathtt{when}(m, x.Gl(N_1), y.Gl(N_2))k)$
5. $Gl(\mu\alpha.M) = \lambda\alpha.Gl(M)\mathtt{id}; \quad Gl([\alpha]M) = \lambda k.k(Gl(M)\alpha)$
6. $Gl((M, t)) = \lambda k.Gl(M)(\lambda m.k(m, t))$;
   $Gl(\mathtt{split}(M, x.N) = \lambda k.Gl(M)(\lambda m.\mathtt{split}(m, x.Gl(N))k)$.

Note that the rule of $\bot_I$ (ex falso sequitur quodlibet) is indispensable to establish the theorem, see the case of $Gl(\lambda x.M)$ above where $\mu\beta$ is applied vacuously. Note also that $Gl(M)$ may be quite similar to the forthcoming call-by-value translation by Kuroda's embedding, however it is not straightforward to interpret the case of $Gl(\lambda x.M)$ along this line where vacuous λ and μ abstractions are applied.

3.2 Gödel-Gentzen g

The second embedding is known as Gödel-Gentzen from classical logic into minimal logic.

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma^g \vdash_{\mathrm{Min}} A^g,$$

where g is defined as follows:

1. $P^g = \neg\neg P \quad (\bot^g = \bot)$,
2. $(A \to B)^g = A^g \to B^g$,
3. $(A \wedge B)^g = A^g \wedge B^g$,
4. $(A \vee B)^g = \neg(\neg A^g \wedge \neg B^g)$,
5. $(\forall x A)^g = \forall x A^g$,
6. $(\exists x A)^g = \neg\forall x \neg A^g$.

Note that we have the property of double negation elimination for A^g in minimal logic:

$$\vdash_{\mathrm{Min}} \neg\neg A^g \leftrightarrow A^g$$

from the elementary facts such that

- $\neg\neg\neg A \leftrightarrow \neg A$; $\neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$;
- $\neg\neg(A \to B) \to (A \to \neg\neg B)$; $\neg\neg\forall x A \to \forall x \neg\neg A$.

Now we show the translation of Gödel-Gentzen from classical logic into minimal logic.

Theorem 2 (Gödel-Gentzen)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$ *then* $\vec{x}{:}\Gamma^g, \vec{\alpha}{:}\neg\Delta^g \vdash_{\mathrm{Min}} G(M) : A^g$,

where the translation G is defined as a translation from $\lambda\mu$-terms to λ-terms as follows:

1. $G(x) = x$
2. $G(\lambda x.M) = \lambda x.G(M)$; $\quad G(MN) = G(M)G(N)$
3. $G(\langle M, N\rangle) = \langle G(M), G(N)\rangle$; $\quad G(\mathtt{fst}(M)) = \mathtt{fst}(G(M))$
4. $G(\mathtt{inl}(M)) = \lambda k.\mathtt{fst}(k)G(M)$;
   $G(\mathtt{when}(M, x.N_1, y.N_2)) = Dn(\lambda k.G(M)\langle\lambda x.k(G(N_1)), \lambda y.k(G(N_2))\rangle)$
5. $G(\lambda x.M) = \lambda x.G(M)$; $G(Mt) = G(M)t$
6. $G((M, t)) = \lambda k.kt(G(M))$;
   $G(\mathtt{split}(M, z.N)) = Dn(\lambda k.G(M)(\lambda x\lambda z.k(G(N))))$
7. $G(\mu\alpha.M) = Dn(\lambda\alpha.G(M)\mathtt{id})$; $\quad G([\alpha]M) = \lambda k.k(\alpha(G(M)))$.

Here, we write Dn for the double negation elimination for A^g:

$$x{:}\neg\neg A^g \vdash_{\mathrm{Min}} Dn(x) : A^g,$$

which is applied essentially in classical reasoning of the definition of $G(M)$.

We note that the following formulae are equivalent in minimal logic:

- $\vdash_{\mathrm{Min}} \neg(\neg A^g \wedge \neg B^g) \ \leftrightarrow \neg\neg(A^g \vee B^g)$
- $\vdash_{\mathrm{Min}} \neg\forall x \neg A^g \ \leftrightarrow \neg\neg\exists x A^g$.

This means that we have yet another translation, a variant of Gödel-Gentzen, for the cases of $\vee$ and $\exists$, as follows:

1. $G(\mathtt{inl}(M)) = \lambda k.k(\mathtt{inl}(G(M)))$;
   $G(\mathtt{when}(M, x.N_1, y.N_2))$
   $= Dn(\lambda k.G(M)(\lambda m.\mathtt{when}(m, x.k(G(N_1)), y.k(G(N_2)))))$
2. $G((M, t)) = \lambda k.k(G(M), t)$;
   $G(\mathtt{split}(M, z.N)) = Dn(\lambda k.G(M)(\lambda m.\mathtt{split}(m, z.k(G(N)))))$.

The case of $G(\mathtt{inl}(M))$ can be read computationally such that first evaluate M, and then obtain $G(M)$, and next send the value $\mathtt{inl}(G(M))$ to the continuation k. The case of $G(\mathtt{when}(M, x.N_1, y.N_2))$ can be read similarly such that evaluate M, and then depending on the value, to say m, we continue the computation $\mathtt{when}(m, x.k(G(N_1)), y.(G(N_2)))$. The cases can be verified formally as follows:

$$G(\mathtt{when}(\mathtt{inl}(M), x.N_1, y.N_2)) \to^* Dn(\lambda k.k(G(N_1[x := M]))).$$

The case of $\exists$ can be handled similarly. For the case of $G(\mu\alpha.M)$, see also [F95].

3.3 Kolmogorov k

The third embedding is known as Kolmogorov from classical logic into minimal logic.

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma^k \vdash_{\mathrm{Min}} A^k,$$

where k is defined as follows:

1. $P^k = \neg\neg P$
2. $(A \to B)^k = \neg\neg(A^k \to B^k)$
3. $(A \wedge B)^k = \neg\neg(A^k \wedge B^k)$
4. $(A \vee B)^k = \neg\neg(A^k \vee B^k)$
5. $(\forall x A)^k = \neg\neg\forall x A^k$
6. $(\exists x A)^k = \neg\neg\exists x A^k$.

Note that in minimal logic, A^g and A^k are equivalent each other:

$$\vdash_{\mathrm{Min}} A^g \leftrightarrow A^k.$$

We write A^* for the formula obtained by removing $\neg\neg$ from A^k such that $A^k = \neg\neg A^*$. Now we show the translation of Kolmogorov from classical logic into minimal logic.

Theorem 3 (Kolmogorov)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^k; \vec{\alpha}{:}\neg\Delta^* \vdash_{\mathrm{Min}} \underline{M} : A^k$,

where the under-lined translation is defined as a translation from $\lambda\mu$-terms to λ-terms, as follows:

1. $\underline{x} = \lambda k.xk$
2. $\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$; $\quad \underline{MN} = \lambda k.\underline{M}(\lambda m.m\underline{N}k)$
3. $\underline{\langle M, N\rangle} = \lambda k.k\langle\underline{M}, \underline{N}\rangle$; $\quad \underline{\mathtt{fst}(M)} = \lambda k.\underline{M}(\lambda m.\mathtt{fst}(m)k)$
4. $\underline{\mathtt{inj}(M)} = \lambda k.k(\mathtt{inj}(\underline{M}))$;
   $\underline{\mathtt{when}(M, x.N_1, y.N_2)} = \lambda k.\underline{M}(\lambda m.\mathtt{when}(m, x.\underline{N_1}, y.\underline{N_2})k)$

5. $\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$; $\underline{Mt} = \lambda k.\underline{M}(\lambda m.mtk)$
6. $\underline{(M,t)} = \lambda k.k(\underline{M},t)$;
   $\underline{\mathtt{split}(M,z.N)} = \lambda k.\underline{M}(\lambda m.\mathtt{split}(m,z.\underline{N})k)$
7. $\underline{\mu\alpha.M} = \lambda\alpha.\underline{M}\mathtt{id}$; $\underline{[\alpha]M} = \lambda k.k(\underline{M}\alpha)$.

We should note that following Martin-Löf's explanation [NPS90], each introduction rule can be read as providing the notion of values V under the computation of call-by-name, such that

$$V ::= \lambda x.M \mid \langle M,N\rangle \mid \mathtt{proj}_i(M) \mid (M,t).$$

Then one can define the auxiliary function Φ as follows:

1. $\Phi(x) = x\ \mathtt{id}$
2. $\Phi(\lambda x.M) = \lambda x.\underline{M}$
3. $\Phi(\langle M,N\rangle) = \langle\underline{M},\underline{N}\rangle$
4. $\Phi(\mathtt{inj}(M)) = \mathtt{inj}(\underline{M})$,

where we have $\underline{V} = \lambda k.k\Phi(V)$ for $V \not\equiv x$. The reduction correspondence is obtained at least for the intuitionistic fragment NJ.

Proposition 1 (Reduction correspondence)

1. *If* $\vec{x}{:}\Gamma \vdash_{\mathrm{NJ}} M : A$, *then* $\vec{x}{:}\Gamma^k \vdash_{\mathrm{Min}} \underline{M} : A^k$.
2. *If* $M \to_{\mathtt{N}} N$ *under call-by-name, then* $\underline{M} \to_{\mathtt{V}}^{+} \underline{N}$ *under call-by-value.*

Here, the reduction rule $\to_{\mathtt{N}}$ of call-by-name is defined by the following rules:

1. $(\lambda x.M)N \to M[x := N]$,
2. $\mathtt{proj}_i\langle M_1,M_2\rangle \to M_i$,
3. $\mathtt{when}(\mathtt{inj}_i(M),x_1.N_1,x_2.N_2) \to N_i[x_i := M]$,
4. $\mathtt{split}((M,t),x.N) \to N[x := M]$,
5. $(\mu\alpha.M)N \to \mu\alpha.M$ with $\alpha \notin FV(M)$.

Now the simulation theorem of call-by-name via call-by-value can be established from the reduction correspondence such that if $M \to_{\mathtt{N}}^{*} V$ then $\underline{M}\ \mathtt{id} \to_{\mathtt{V}}^{+} \Phi(V)$.

Theorem 4 (Simulation [Plot75]) $\mathtt{Eval}_{\mathtt{V}}(\underline{M}\ \mathtt{id}) = \Phi(\mathtt{Eval}_{\mathtt{N}}(M))$

For the full system NK, it is straightforward to prove the equational correspondence between $\lambda\mu$-terms and λ-terms.

Proposition 2 (Equational correspondence) *If* $M =_{\mu} N$ *then* $\underline{M} =_{\beta} \underline{N}$.

Here, the equality $=_\mu$ is defined by the following reduction rules.

1. $(\lambda x.M)N \to M[x := N]$,
2. $\mathtt{proj}_i\langle M_1, M_2\rangle \to M_i$,
3. $\mathtt{when}(\mathtt{inj}_i(M), x_1.N_1, x_2.N_2) \to N_i[x_i := M]$,
4. $\mathtt{split}((M,t), x.N) \to N[x := M]$,
5. $(\mu\alpha.M)N \to \mu\alpha.M[\alpha \Leftarrow N]$,
6. $\mu\alpha.[\alpha]M \to M$ with $\alpha \notin FV(M)$.

Note that the fifth rule above is justified by the substitution property such that $\underline{M[\alpha \Leftarrow N]} \to^* \underline{M}[\alpha := \lambda m.m\underline{N}\alpha]$, see also [F95] for the details.

3.4 Kuroda q

The fourth embedding is known as Kuroda from classical logic into minimal or intuitionistic logic.

$$\Gamma \vdash_{\mathrm{NK}} A \text{ iff } \Gamma^q \vdash_{\mathrm{Min}} \neg\neg A^q,$$

where q is defined as follows:

1. $P^q = P$
2. $(A \to B)^q = A^q \to \neg\neg B^q$
3. $(A \wedge B)^q = A^q \wedge B^q$
4. $(A \vee B)^q = A^q \vee B^q$
5. $(\forall x A)^q = \forall x \neg\neg A^q$
6. $(\exists x A)^q = \exists x A^q$.

We should note that the original embedding of Kuroda (1951) is defined as q' such that $(A \to B)^{q'} = A^{q'} \to B^{q'}$ for the second case above. Then this embedding provides a translation from classical logic into the full intuitionistic logic, see also 3.1 Glivenko's theorem. The original embedding can be considered as Glivenko's theorem for the fragment of $\forall$-free systems.

Note also that in minimal logic, we have the following equivalence:

$$\vdash_{\mathrm{Min}} A^k \leftrightarrow \neg\neg A^q$$

Now we show the translation of Kuroda from classical logic into minimal logic.

Theorem 5 (Kuroda)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^q; \vec{\alpha}{:}\neg\Delta^q \vdash_{\mathrm{Min}} \overline{M} : \neg\neg A^q$,

where the over-lined translation is defined as a translation from $\lambda\mu$-terms to λ-terms, as follows:

1. $\overline{x} = \lambda k.kx$
2. $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$; $\quad \overline{MN} = \lambda k.\overline{M}(\lambda m.\overline{N}(\lambda n.mnk))$
3. $\overline{\langle M, N\rangle} = \lambda k.\overline{M}(\lambda m.\overline{N}(\lambda n.k\langle m, n\rangle))$; $\quad \overline{\mathtt{fst}(M)} = \lambda k.\overline{M}(\lambda m.k(\mathtt{fst}(m))$
4. $\overline{\mathtt{inj}(M)} = \lambda k.\overline{M}(\lambda m.k(\mathtt{inj}(m)))$;
   $\overline{\mathtt{when}(M, x.N_1, y.N_2)} = \lambda k.\overline{M}(\lambda m.\mathtt{when}(m, x.\overline{N_1}, y.\overline{N_2})k)$
5. $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$; $\quad \overline{Mt} = \lambda k.\overline{M}(\lambda m.mtk)$
6. $\overline{(M, t)} = \lambda k.\overline{M}(\lambda m.k(m, t))$;
   $\overline{\mathtt{split}(M, x.N)} = \lambda k.\overline{M}(\lambda m.\mathtt{split}(m, x.\overline{N})k)$
7. $\overline{\mu\alpha.M} = \lambda\alpha.\overline{M}\mathtt{id}$; $\quad \overline{[\alpha]M} = \lambda k.k(\overline{M}\alpha)$.

We should remark that the translation (interpretation) above provides the notion of values V under the computation of call-by-value, such that

$$V ::= x \mid \lambda x.M \mid \mathtt{proj}(V) \mid \langle V, V\rangle \mid \mathtt{inj}(V) \mid (V, t).$$

Then one can define an auxiliary function Ψ as follows:

1. $\Psi(x) = x$
2. $\Psi(\lambda x.M) = \lambda x.\overline{M}$
3. $\Psi(\mathtt{proj}(V)) = \mathtt{proj}(\Psi(V))$
4. $\Psi(\langle V_1, V_2\rangle) = \langle\Psi(V_1), \Psi(V_2)\rangle$
5. $\Psi(\mathtt{inj}(V)) = \mathtt{inj}(\Psi(V))$
6. $\Psi(V, t) = (\Psi(V), t)$,

where we have the following reduction properties:

- $\overline{V} \to^* \lambda k.k\Psi(V)$
- $\overline{M[x := V]} \to^* \overline{M}[x := \Psi(V)]$.

Now the equational correspondence holds for the intuitionistic fragment NJ.

Proposition 3 (Equational correspondence)

1. *If* $\vec{x}{:}\Gamma \vdash_{\mathrm{NJ}} M : A$, *then* $\vec{x}{:}\Gamma^q \vdash_{\mathrm{Min}} \neg\neg\overline{M} : A^q$.
2. *If* $M \to_V N$, *then* $\overline{M} =_\beta \overline{N}$.

Here, the reduction rule of call-by-value is defined by the following rules.

1. $(\lambda x.M)V \to_V M[x := V]$
2. $\mathtt{proj}_i\langle V_1, V_2\rangle \to_V V_i$

3. $\mathtt{when}(\mathtt{inj}_i(V), x_1.N_1, x_2.N_2) \to_V N_i[x_i := V]$

4. $\mathtt{split}((V,t), z.N) \to_V N[z := V]$.

For the full system NK, we have the equational correspondence as well, see also [F95] for the implicational fragment of $\lambda\mu$-calculus.

Proposition 4 (Equational correspondence)

1. *If* $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^q; \vec{\alpha}{:}\neg\Delta^q \vdash_{\mathrm{Min}} \overline{M} : \neg\neg A^q$.

2. *If* $M =_{\mu_V} N$, *then* $\overline{M} =_\beta \overline{N}$,

where the equality $=_{\mu_V}$ is defined by the reduction rule $\to_{\mu_V}$ of call-by-value, consisting of the following rules:

1. $(\lambda x.M)V \to M[x := V]$,

2. $\mathtt{proj}_i\langle V_1, V_2\rangle \to V_i$,

3. $\mathtt{when}(\mathtt{inj}_i(V), x_1.N_1, x_2.N_2) \to N_i[x_i := V]$,

4. $\mathtt{split}((V,t), x.N) \to N[x := V]$,

5. $(\mu\alpha.M)N \to \mu\alpha.M[\alpha \Leftarrow N]$,

6. $V(\mu\alpha.M) \to \mu\alpha.M[V \Rightarrow \alpha]$,

7. $\mu\alpha.[\alpha]M \to M$ with $\alpha \notin FV(M)$.

Note that we have the following reduction properties:

- $\overline{M[\alpha \Leftarrow N]} \to^* \overline{M}[\alpha := \lambda m.\overline{N}(\lambda n.mn\alpha)]$
- $\overline{M[V \Rightarrow \alpha]} \to^* \overline{M}[\alpha := \lambda n.\Psi(V)n\alpha)]$.

3.5 Modified Kuroda q_2

In order to establish the reduction correspondence for the full intuitionistic fragment, we modified Kuroda's embedding, naturally leading to the simulation theorem of call-by-value under call-by-name.

Theorem 6 (Modified Kuroda q_2)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^{q_2}; \vec{\alpha}{:}\neg\Delta^{q_2} \vdash_{\mathrm{Min}} \overline{M} : \neg\neg A^{q_2}$,

where the modified embedding q_2 is defined as follows:

1. $P^{q_2} = P$;

2. $(A \to B)^{q_2} = A^{q_2} \to \neg\neg B^{q_2}$;

3. $(A \wedge B)^{q_2} = \neg\neg A^{q_2} \wedge \neg\neg B^{q_2}$;

4. $(A \vee B)^{q_2} = \neg\neg A^{q_2} \vee \neg\neg B^{q_2}$;

5. $(\forall x A)^{q_2} = \forall x \neg\neg A^{q_2}$;
6. $(\exists x A)^{q_2} = \exists x \neg\neg A^{q_2}$.

Accordingly, the over-lined translation is modified as well as follows:

1. $\overline{x} = \lambda k.kx$
2. $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$; $\overline{MN} = \lambda k.\overline{M}(\lambda m.\overline{N}(\lambda n.mnk))$
3. $\overline{\langle M, N\rangle} = \lambda k.k\langle\overline{M}, \overline{N}\rangle$; $\overline{\mathtt{fst}(M)} = \lambda k.\overline{M}(\lambda m.\mathtt{fst}(m)k)$
4. $\overline{\mathtt{inj}(M)} = \lambda k.k(\mathtt{inj}(\overline{M}))$;
   $\overline{\mathtt{when}(M, x.N_1, y.N_2)} = \lambda k.\overline{M}(\lambda m.\mathtt{when}(m, u.u(\lambda x.\overline{N_1}k), v.v(\lambda y.\overline{N_2}k)))$
5. $\overline{\lambda x.M} = \lambda k.k(\lambda x.\overline{M})$; $\overline{Mt} = \lambda k.\overline{M}(\lambda m.mtk)$
6. $\overline{(M, t)} = \lambda k.k(\overline{M}, t)$;
   $\overline{\mathtt{split}(M, z.N)} = \lambda k.\overline{M}(\lambda m.\mathtt{split}(m, u.u(\lambda z.\overline{N}k)))$
7. $\overline{\mu\alpha.M} = \lambda\alpha.\overline{M}\mathtt{id}$; $\overline{[\alpha]M} = \lambda k.k(\overline{M}\alpha)$.

Then we also modify the notion of values V for call-by-value, together with the modified auxiliary function Ψ, respectively as follows:

$$V ::= x \mid \lambda x.M \mid \langle M, M\rangle \mid \mathtt{inj}(M) \mid (M, t)$$

1. $\Psi(x) = x$
2. $\Psi(\lambda x.M) = \lambda x.\overline{M}$
3. $\Psi(\langle M, N\rangle) = \langle\overline{M}, \overline{N}\rangle$
4. $\Psi(\mathtt{inj}(M)) = \mathtt{inj}(\overline{M})$
5. $\Psi(M, t) = (\overline{M}, t)$,

where the expected properties hold for values and substitution as well:

- $\overline{V} = \lambda k.k\Psi(V)$
- $\overline{M}[x := \Psi(V)] = \overline{M[x := V]}$.

Now the reduction correspondence is established for the full intuitionistic fragment NJ, where the definition of $\to_{\mathtt{V}}$ is the exactly same as that of $\to_V$ in the previous subsection. Note that the definition itself has the same form, however the notion of values and Ψ are modified from those in the previous subsection.

Proposition 5 (Reduction correspondence q_2)
If $M \to_{\mathtt{V}} N$ under call-by-value, then $\overline{M} \to^+_{\mathtt{N}} \overline{N}$ under call-by-name.

Then the simulation theorem of call-by-value via call-by-name can be established from the reduction correspondence such that if $M \to^*_{\mathtt{V}} V$ then $\overline{M}\ \mathtt{id} \to^+_{\mathtt{N}} \Psi(V)$.

Theorem 7 (Simulation [Plot75]) $\mathtt{Eval}_{\mathtt{N}}(\overline{M}\ \mathtt{id}) = \Psi(\mathtt{Eval}_{\mathtt{V}}(M))$

For the full system NK, we still have the equational correspondence just like in the previous subsection.

Proposition 6 (Equational correspondence) *If $M =_{\mu_V} N$, then $\overline{M} =_\beta \overline{N}$.*

3.6 Modified Kolmogorov k_2

From the viewpoint of DeMorgan's duality such as $\wedge$-$\vee$ and $\forall$-$\exists$, we modify kolmogorov's embedding such that $\wedge$ is interpreted by $\vee$, $\forall$ is by $\exists$, and vice versa under the modified k_2. The definition of the embedding k_2 is given as follows:

1. $P^{k_2} = \neg\neg P$;
2. $(A \to B)^{k_2} = \neg\neg(A^{k_2} \to B^{k_2})$;
3. $(A \vee B)^{k_2} = \neg(\neg A^{k_2} \wedge \neg B^{k_2})$;
4. $(A \wedge B)^{k_2} = \neg(A^* \vee B^*)$;
5. $(\forall x A)^{k_2} = \neg\exists x A^*$;
6. $(\exists x A)^{k_2} = \neg\forall x \neg A^{k_2}$,

where we write A^* for the formula obtained by removing $\neg$ from A^{k_2} such that $A^{k_2} = \neg A^*$.

Theorem 8 (Modified Kolmogorov k_2)
If $\vec{x}{:}\Gamma, \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^{k_2}; \vec{\alpha}{:}\Delta^* \vdash_{\mathrm{Min}} \underline{M} : A^{k_2}$,

where the under-lined translation is defined as a translation from $\lambda\mu$-terms to λ-terms as follows:

1. $\underline{x} = \lambda k.xk$
2. $\underline{\lambda x.M} = \lambda k.k(\lambda x.\underline{M})$; $\quad \underline{MN} = \lambda k.\underline{M}(\lambda m.m\underline{N}k)$
3. $\underline{\langle M, N\rangle} = \lambda k.\mathtt{when}(k, x.\underline{M}x, y.\underline{N}y)$;
   $\underline{\mathtt{fst}(M)} = \lambda k.\underline{M}(\mathtt{inl}(k))$
4. $\underline{\mathtt{inl}(M)} = \lambda k.\mathtt{fst}(k)(\underline{M})$;
   $\underline{\mathtt{when}(M, x.N_1, y.N_2)} = \lambda k.\underline{M}\langle\lambda x.\underline{N_1}k, \lambda y.\underline{N_2}k\rangle$
5. $\underline{\lambda \mathtt{x}.M} = \lambda k.\mathtt{split}(k, x.\underline{M}x)$; $\quad \underline{Mt} = \lambda k.\underline{M}(k, t)$
6. $\underline{(M, t)} = \lambda k.kt\underline{M}$; $\quad \underline{\mathtt{split}(M, z.N)} = \lambda k.\underline{M}(\lambda x.\lambda z.\underline{N}k)$
7. $\underline{\mu\alpha.M} = \lambda\alpha.\underline{M}\mathtt{id}$; $\quad \underline{[\alpha]M} = \lambda k.k(\underline{M}\alpha)$.

In particular, the full intuitionistic fragment NJ enjoys the reduction correspondence, where the reduction rules are defined as usual.

Proposition 7 (Reduction correspondence k_2)

1. *If* $\vec{x}{:}\Gamma \vdash_{\mathrm{NJ}} M : A$, *then* $\vec{x}{:}\Gamma^{k_2} \vdash_{\mathrm{Min}} \underline{M} : A^{k_2}$.
2. *If* $M \to N$ *then* $\underline{M} \to^+ \underline{N}$.

We remark that the second case $\underline{MN}$ of the under-lined translation can be read as a coding of the pair $\lambda k.\underline{M}\langle\underline{N}, k\rangle$. In the next subsection, this reading makes it possible to interpret implication $\to$ by conjunction $\wedge$, where we use another form of $(\wedge E)$ with the expression of `let`.

3.7 Modified Kolmogorov k_3 in dual

We introduce yet another modification of Kolmogorov, that is to say k_3 as follows:

1. $P^{k_3} = \neg P$;
2. $(A \to B)^{k_3} = (\neg A^{k_3} \wedge B^{k_3})$;
3. $(A \wedge B)^{k_3} = (A^{k_3} \vee B^{k_3})$;
4. $(A \vee B)^{k_3} = \neg\neg A^{k_3} \wedge \neg\neg B^{k_3}$;
5. $(\forall x A)^{k_3} = \exists x A^{k_3}$;
6. $(\exists x A)^{k_3} = \forall x \neg\neg A^{k_3}$.

Note that the second case interprets $\to$ by $\wedge$, where $(A \to B)$ can be read as $(\neg A \vee B)$ under the dual translation k_3 between $\vee$ and $\wedge$. We show that this dual embedding k_3 works as a translation from classical logic into minimal logic.

Theorem 9 (Modified Kolmogorov k_3)
If $\vec{x}{:}\Gamma, \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\neg\Gamma^{k_3}, \vec{\alpha}{:}\Delta^{k_3} \vdash_{\mathrm{Min}} \underline{M} : \neg A^{k_3}$,

where the under-lined translation is given as a translation from $\lambda\mu$-terms to λ-terms as follows:

1. $\underline{x} = \lambda k.xk$
2. $\underline{\lambda x.M} = \lambda k.(\mathtt{let}\ \langle x, k\rangle = k\ \mathtt{in}\ \underline{M}k)$;
   $\underline{MN} = \lambda k.\underline{M}\langle \lambda k.\underline{N}, k\rangle$
3. $\underline{\langle M, N\rangle} = \lambda k.\mathtt{when}(k, k.\underline{M}k, k.\underline{N}k)$;
   $\underline{(\mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N)}$
   $= \lambda k.((\lambda y.(\lambda x.\underline{N}k)(\lambda k.\underline{M}(\mathtt{inl}(k))))(\lambda k.\underline{M}(\mathtt{inr}(k))))$
4. $\underline{\mathtt{inj}_i(M)} = \lambda k.(\mathtt{let}\ \langle x_1, x_2\rangle = k\ \mathtt{in}\ x_i\underline{M})$;
   $\underline{\mathtt{when}(M, x.N_1, y.N_2)} = \lambda k.\underline{M}\langle \lambda x.\underline{N_1}k, \lambda y.\underline{N_2}k\rangle$
5. $\underline{\lambda x.M} = \lambda k.\mathtt{split}(k, x.\underline{M}x)$; $\underline{Mt} = \lambda k.\underline{M}(k, t)$
6. $\underline{(M, t)} = \lambda k.kt\underline{M}$; $\underline{\mathtt{split}(M, z.N)} = \lambda k.\underline{M}(\lambda x.\lambda z.\underline{N}k)$
7. $\underline{\mu\alpha.M} = \lambda\alpha.\underline{M}\mathtt{id}$; $\underline{[\alpha]M} = \lambda k.k(\underline{M}\alpha)$.

For the third case of $\mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N$, instead one can take the substituted form $\underline{N}[x := \lambda k.\underline{M}(\overline{\mathtt{inl}(k)}), y := \overline{\lambda k.\underline{M}(\mathtt{inr}(k))}]$ by contracting the redexes. Observe that $(\to I)$ is interpreted by $(\wedge E)$, $(\to E)$ is by $(\wedge I)$, and each introduction rule is by the corresponding elimination rule in dual, and vice versa. In particular, for the full intuitionistic fragment NJ, the reduction correspondence is obtained, where the reduction rules are defined as usual.

Proposition 8 (Reduction correspondence k_3)
If $M \to N$, then $\underline{M} \to^+ \underline{N}$.

We should note that this form of dual embedding is applied to 2nd order propositional logic, so called Girard's System F with extensional rules. Then the reduction correspondence induces the fundamental properties such as Church-Rosser, Normalization, polymorphic functions, and abstract data types between source and target calculi, see CPS-translation as adjoint [F10] for the details.

3.8 Modified Kolmogorov k_4 in dual

We introduce another and yet another embedding k_4 which handles both implication $\to$ and its dual implication $\leftarrow$, rather than conjunction with negation. For convenience, recall the inference rules and reduction rule here.

- Inference rules:

$$\frac{\Gamma \vdash M : B \quad \Gamma, x{:}A \vdash N : \bot}{\Gamma \vdash \mathtt{maL}(M, x.N) : (A \leftarrow B)} \ (\leftarrow I)$$

$$\frac{\Gamma \vdash M : (A \leftarrow B) \quad \Gamma, y{:}B \vdash P : A}{\Gamma \vdash \mathtt{ppA}(M, y.P) : \bot} \ (\leftarrow E)$$

- Reduction rule:

$$\mathtt{ppA}(\mathtt{maL}(M, x.N), y.P) \to N[x := P[y := M]]$$

The dual embedding k_4 is defined as follows.

1. $P^{k_4} = \neg P$;
2. $(A \to B)^{k_4} = (\neg\neg A^{k_4} \leftarrow B^{k_4})$;
3. $(A \leftarrow B)^{k_4} = (A^{k_4} \to \neg\neg B^{k_4})$;
4. $(A \wedge B)^{k_4} = (A^{k_4} \vee B^{k_4})$;
5. $(A \vee B)^{k_4} = \neg\neg A^{k_4} \wedge \neg\neg B^{k_4}$;
6. $(\forall x A)^{k_4} = \exists x A^{k_4}$;
7. $(\exists x A)^{k_4} = \forall x \neg\neg A^{k_4}$.

We show that the embedding k_4 works as a translation from classical logic into minimal logic as follows.

Theorem 10 (Kolmogorov k_4)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, then $\vec{x}{:}\neg\Gamma^{k_4}, \vec{\alpha}{:}\Delta^{k_4}, k{:}A^{k_4} \vdash_{\mathrm{Min}} \underline{\underline{M}} : \bot$,

where the double-underlined translation is given as a translation from $\lambda\mu$-terms to λ-terms as follows:

1. $\underline{\underline{x}} = xk$
2. $\underline{\underline{\lambda x.M}} = \mathtt{ppA}(k, k.(\lambda x.\underline{\underline{M}}))$;
   $\underline{\underline{MN}} = \underline{\underline{M}}[k := \mathtt{maL}(k, k.k(\lambda k.\underline{\underline{N}}))]$
3. $\underline{\underline{\mathtt{maL}(M, x.N)}} = \underline{\underline{N}}[k := \mathtt{id}][x := \lambda a.ka(\lambda k.\underline{\underline{M}})]$;
   $\underline{\underline{\mathtt{ppA}(M, y.P)}} = \underline{\underline{M}}[k := \lambda k.\lambda y.\underline{\underline{P}}]$
4. $\underline{\underline{\langle M, N\rangle}} = \mathtt{when}(k, k.\underline{\underline{M}}, k.\underline{\underline{N}})$;
   $\underline{\underline{(\mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N)}}$
   $= \underline{\underline{N}}[x := \lambda k.\underline{\underline{M}}[k := \mathtt{inl}(k)], y := \lambda k.\underline{\underline{M}}[k := \mathtt{inr}(k)]]$;
   $\underline{\underline{\mathtt{proj}_i(M)}} = \underline{\underline{M}}[k := \mathtt{inj}_i(k)]$
5. $\underline{\underline{\mathtt{inj}_i(M)}} = (\mathtt{let}\ \langle x_1, x_2\rangle = k\ \mathtt{in}\ x_i(\lambda k.\underline{\underline{M}}))$;
   $\underline{\underline{\mathtt{inj}_i(M)}} = \mathtt{proj}_i(k)(\lambda k.\underline{\underline{M}})$;
   $\underline{\underline{\mathtt{when}(M, x.N_1, y.N_2)}} = \underline{\underline{M}}[k := \langle \lambda x.\underline{\underline{N_1}}, \lambda y.\underline{\underline{N_2}}\rangle]$
6. $\underline{\underline{\lambda x.M}} = \mathtt{split}(k, k.\underline{\underline{M}})$; $\underline{\underline{Mt}} = \underline{\underline{M}}[k := (k, t)]$
7. $\underline{\underline{(M, t)}} = kt(\lambda k.\underline{\underline{M}})$; $\underline{\underline{\mathtt{split}(M, z.N)}} = \underline{\underline{M}}[k := \lambda x.\lambda z.\underline{\underline{N}}]$
8. $\underline{\underline{\mu\alpha.M}} = \underline{\underline{M}}[k := \mathtt{id}][\alpha := k]$; $\underline{\underline{[\alpha]M}} = k(\underline{\underline{M}}[k := \alpha])$.

We note that the following substitution properties hold here.

- $\underline{\underline{M}}[x := \lambda k.\underline{\underline{N}}] \to^* \underline{\underline{M[x := N]}}$;
- $\underline{\underline{M[\alpha \Leftarrow N]}} = \underline{\underline{M}}[\alpha := \mathtt{maL}(\alpha, k.k(\lambda k.\underline{\underline{N}}))]$.

Now we obtain the reduction correspondence for the full system NK in the following sense.

Proposition 9 (Reduction correspondence k_4)
If $M \to N$, then $\underline{\underline{M}} \to^* \underline{\underline{N}}$.

Remarked that for a term N with type $\bot$, we may consider an extra substitution $[k := \mathtt{id}]$, such that $\underline{\underline{\mathtt{ppA}(\mathtt{maL}(M, x.N), y.P)}} \to^*_\beta \underline{\underline{N[x := P[y := M]]}}[k := \mathtt{id}]$, and for the same phenomena, see the definition of Φ for the simulation theorem Theorem 4 in 3.3 Kolmogorov k.

The following elementary properties hold where $\neg A := (A \to \bot)$.

- $\vdash_{\mathrm{Min}} \neg A \leftrightarrow (\neg\neg A \leftarrow \neg\bot)$, and $\vdash_{\mathrm{Min}} (\neg A)^{k_4} \leftrightarrow \neg A^{k_4}$
- $\vdash_{\mathrm{Min}} \neg\neg(\neg\neg A \leftarrow \neg\neg B) \leftrightarrow \neg\neg(\neg\neg A \leftarrow B) \leftrightarrow (\neg\neg A \leftarrow \neg\neg B)$

Hence, we have the following proposition for the original Kolmogorov k.

Proposition 10 $\vdash_{\mathrm{Min}} A^k \leftrightarrow \neg\neg(A^{k_4})^{k_4}$.

Here, we define $(A \leftarrow B)^k = \neg\neg(A^k \leftarrow B^k)$. For Glivenko's theorem, we note that $\vdash_{\mathrm{NJ}} \neg\neg(A^{k_4})^{k_4} \leftrightarrow \neg\neg A$ for the $\forall$-free fragment, since $\vdash_{\mathrm{NJ}} \neg\neg(A \leftarrow B) \leftrightarrow (\neg\neg A \leftarrow \neg\neg B)$.

3.9 Kuroda q_3 revised in dual/contrapositive

From the viewpoint of duality and contraposition, we consider Kuroda's embedding under call-by-value computation. The embedding q_3 is defined as follows:

1. $P^{q_3} = P$;
2. $(A \to B)^{q_3} = \neg(A^{q_3} \wedge \neg B^{q_3})$;
3. $(A \wedge B)^{q_3} = \neg(\neg A^{q_3} \vee \neg B^{q_3})$;
4. $(A \vee B)^{q_3} = \neg(\neg A^{q_3} \wedge \neg B^{q_3})$
5. $(\forall x A)^{q_3} = \neg\exists x \neg A^{q_3}$;
6. $(\exists x A)^{q_3} = \neg\forall x \neg A^{q_3}$.

We show that the embedding q_3 works as a translation from classical logic into minimal logic.

Theorem 11 (Kuroda q_3)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^{q_3}, \vec{\alpha}{:}\neg\Delta^{q_3}, k{:}\neg A^{q_3} \vdash_{\mathrm{Min}} \overline{M} : \bot$,

where the over-lined translation is given as a translation from $\lambda\mu$-terms to λ-terms mutually together with an auxiliary function Ψ in the following:

1. $\overline{x} = kx$
2. $\overline{\lambda x.M} = k(\lambda k.\mathtt{let}\ \langle x, k\rangle = k\ \mathtt{in}\ \overline{M})$;
   (a) $\overline{VN} = \overline{N}[k := \lambda n.\Psi(V)\langle n, k\rangle]$ for value V,
   (b) $\overline{MN} = \overline{M}[k := \lambda m.\overline{N}[k := \lambda n.m\langle n, k\rangle]]$ for non-value M
3. $\overline{\langle M, N\rangle} = k(\lambda k.\mathtt{when}(k, k.\overline{M}, k.\overline{N}))$;
   $\overline{(\mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N)} = \overline{M}[k := \lambda k.k(\mathtt{inj}_2(\lambda y.k(\mathtt{inj}_1(\lambda x.\overline{N}))))]$
4. $\overline{\mathtt{inj}_i(M)} = k(\lambda k.\mathtt{let}\ \langle k_1, k_2\rangle = k\ \mathtt{in}\ \overline{M}[k := k_i])$;
   $\overline{\mathtt{when}(M, x.N_1, y.N_2)} = \overline{M}[k := \lambda m.m\langle \lambda x.\overline{N_1}, \lambda y.\overline{N_2}\rangle]$
5. $\overline{\lambda \mathtt{x}.M} = k(\lambda k.\mathtt{split}(k, k.\overline{M}))$; $\overline{Mt} = \overline{M}[k := \lambda m.m(k, t)]$
6. $\overline{(M, t)} = k(\lambda m.\overline{M}[k := mt])$;
   $\overline{\mathtt{split}(M, z.N)} = \overline{M}[k := \lambda m.m(\lambda x.\lambda z.\overline{N})]$
7. $\overline{\mu\alpha.M} = \overline{M}[k := \mathtt{id}][\alpha := k]$; $\overline{[\alpha]M} = k(\overline{M}[k := \alpha])$,

where the auxiliary function Ψ for V with $\overline{V} = k\Psi(V)$ is defined as follows:

$$V ::= x \mid \lambda x.M \mid \langle M, N\rangle \mid \mathtt{inj}(M) \mid \lambda \mathtt{x}.M \mid (M, t) \mid [\alpha]M$$

1. $\Psi(x) = x$
2. $\Psi(\lambda x.M) = \lambda k.(\mathtt{let}\ \langle x, k\rangle = k\ \mathtt{in}\ \overline{M})$
3. $\Psi(\langle M, N\rangle) = \lambda k.\mathtt{when}(k, \overline{M}, \overline{N})$
4. $\Psi(\mathtt{inj}_i) = \lambda k.(\mathtt{let}\ \langle k_1, k_2\rangle = k\ \mathtt{in}\ \overline{M}[k := k_i])$
5. $\Psi(\lambda \mathtt{x}.M) = \lambda k.\mathtt{split}(k, k.\overline{M})$
6. $\Psi((M, t)) = \lambda k.\overline{M}[k := kt]$
7. $\Psi([\alpha]M) = \overline{M}[k := \alpha]$.

Note that for the case (b) of $\overline{MN}$, one can take another form such that $\overline{MN} = \overline{N}[k := \lambda n.\overline{M}[k := \lambda m.m\langle n, k\rangle]]$ as well.

We define reduction for call-by-value $\to_V$ consisting of the following rules:

1. $(\lambda x.M)V \to M[x := V]$
2. $\mathtt{let}\ \langle x, y\rangle = \langle V_1, V_2\rangle\ \mathtt{in}\ M \to M[x := V_1, y := V_2]$
3. $\mathtt{when}(\mathtt{inj}_i(V), x_1.N_1, x_2.N_2) \to N_i[x_i := V]$
4. $(\lambda \mathtt{x}.M)t \to M$
5. $\mathtt{split}((V, t), z.N) \to N[z := V]$
6. $(\mu\alpha.M)N \to \mu\alpha.M[\alpha \Leftarrow N]$
7. $V(\mu\alpha.M) \to \mu\alpha.M[V \Rightarrow \alpha]$
8. $\mu\alpha.[\alpha]M \to M$ if $\alpha \notin FV(M)$
9. $[\beta](\mu\alpha.V) \to V[\alpha := \beta]$.

Note that the following substitution properties hold.

- $\overline{M}[x := \Psi(V)] = \overline{M[x := V]}$; $\Psi(V_1)[x := \Psi(V_2)] = \Psi(V_1[x := V_2])$
- 1. $\Psi(V)[\alpha := \lambda m.\overline{N}[k := \lambda m.m\langle n, \alpha\rangle]] \to_\beta^* \Psi(V[\alpha \Leftarrow N])$ for value V,
  2. $\overline{V}[\alpha := \lambda m.\overline{N}[k := \lambda m.m\langle n, \alpha\rangle]] \to_\beta^* \overline{V[\alpha \Leftarrow N]}$ for value V,
  3. $\overline{M[\alpha \Leftarrow N]} = \overline{M}[\alpha := \lambda m.\overline{N}[k := \lambda n.m\langle n, \alpha\rangle]]$ for non-value M,
  4. $\overline{M[V \Rightarrow \alpha]} = \overline{M}[\alpha := \lambda n.\Psi(V)\langle n, \alpha\rangle]]$ for value V.

For the full intuitionistic logic NJ, we have the reduction correspondence.

Proposition 11 (Reduction correspondence)

1. *If* $\vec{x}{:}\Gamma \vdash_{\mathrm{NJ}} M : A$, *then* $\vec{x}{:}\Gamma^{q_3}, k{:}\neg A^{q_3} \vdash_{\mathrm{Min}} \overline{M} : \bot$.
2. *If* $M \to_V N$ *then* $\overline{M} \to^+ \overline{N}$.

For the full system NK, the reduction correspondence holds together with the following facts.

1. $\overline{\mu\alpha.[\alpha]M} \to \overline{M}$ if $\alpha \notin FV(M)$,
2. $\overline{[\beta](\mu\alpha.V)} \to \overline{V[\alpha := \beta]}$ for value V,
3. $\overline{(\mu\alpha.M)N} = \overline{\mu\alpha.M[\alpha \Leftarrow N]}$ for non-value M,
4. $\overline{(\mu\alpha.M)V} \to^*_\beta \overline{\mu\alpha.M[\alpha \Leftarrow N]}$ for value V ,
5. $\overline{\mu\alpha.M[V \Rightarrow \alpha]} = \overline{V(\mu\alpha.M)}$ for value V.

Proposition 12 (Reduction correspondence)
If $M \to_V N$, *then* $\overline{M} \to^*_\beta \overline{N}$.

We should remark that a similar embedding is also investigated for 2nd order $\lambda\mu$-calculus of call-by-value with extensional rules, see [F00] for the details.

3.10 Kuroda q_4 yet modified

We introduce yet another embedding q_4 which handles both implication $\to$ and its dual implication $\leftarrow$.

1. $P^{q_4} = P$;
2. $(A \to B)^{q_4} = \neg(\neg\neg B^{q_4} \leftarrow A^{q_4})$;
3. $(A \wedge B)^{q_4} = \neg(\neg A^{q_4} \vee \neg B^{q_4})$;
4. $(A \leftarrow B)^{q_4} = \neg(B^{q_4} \to \neg\neg A^{q_4})$;
5. $(A \vee B)^{q_4} = \neg(\neg A^{q_4} \wedge \neg B^{q_4})$
6. $(\forall x A)^{q_4} = \neg\exists x \neg A^{q_4}$;
7. $(\exists x A)^{q_4} = \neg\forall x \neg A^{q_4}$.

We show that the embedding q_4 works as a translation from classical logic into Minimal logic.

Theorem 12 (Kuroda q_4)
If $\vec{x}{:}\Gamma; \vec{\alpha}{:}\neg\Delta \vdash_{\mathrm{NK}} M : A$, *then* $\vec{x}{:}\Gamma^{q_4}, \vec{\alpha}{:}\neg\Delta^{q_4}, k{:}\neg A^{q_4} \vdash_{\mathrm{Min}} \overline{\overline{M}} : \bot$,

where the double-overlined translation is given as a translation from $\lambda\mu$-terms to λ-terms mutually with an auxiliary function Ψ as follows:

1. $\overline{\overline{x}} = kx$

2. $\overline{\overline{\lambda x.M}} = k(\lambda k.\mathtt{ppA}(k, x.\lambda k.\overline{\overline{M}}))$;

   (a) $\overline{\overline{VN}} = \overline{\overline{N}}[k := \lambda n.\Psi(V)\,\mathtt{maL}(n, m.mk)]$ for value V,

   (b) $\overline{\overline{MN}} = \overline{\overline{M}}[k := \lambda z.\overline{\overline{N}}[k := \lambda n.z\,\mathtt{maL}(n, m.mk)]]$ for non-value M

3. $\overline{\overline{\mathtt{maL}(M, x.N)}} = k(\lambda k.\overline{\overline{M}}[k := \lambda b.kb(\lambda x.\overline{\overline{N}}[k := \mathtt{id}])])$;
   $\overline{\overline{\mathtt{ppA}(M, y.P)}} = \overline{\overline{M}}[k := \lambda k.k(\lambda y.\lambda k.\overline{\overline{P}})]$

4. $\overline{\overline{\langle M, N\rangle}} = k(\lambda k.\mathtt{when}(k, k.\overline{\overline{M}}, k.\overline{\overline{N}}))$;
   $\overline{\overline{(\mathtt{let}\ \langle x, y\rangle = M\ \mathtt{in}\ N)}} = \overline{\overline{M}}[k := \lambda k.k(\mathtt{inj}_2(\lambda y.k(\mathtt{inj}_1(\lambda x.\overline{\overline{N}}))))]$

5. $\overline{\overline{\mathtt{inj}_i(M)}} = k(\lambda k.\mathtt{let}\ \langle k_1, k_2\rangle = k\ \mathtt{in}\ \overline{\overline{M}}[k := k_i])$;
   $\overline{\overline{\mathtt{when}(M, x.N_1, y.N_2)}} = \overline{\overline{M}}[k := \lambda z.z\langle\lambda x.\overline{\overline{N_1}}, \lambda y.\overline{\overline{N_2}}\rangle]$

6. $\overline{\overline{\lambda \mathtt{x}.M}} = k(\lambda k.\mathtt{split}(k, k.\overline{\overline{M}}))$; $\overline{\overline{Mt}} = \overline{\overline{M}}[k := \lambda m.m(k, t)]$

7. $\overline{\overline{(M, t)}} = k(\lambda m.\overline{\overline{M}}[k := mt])$;
   $\overline{\overline{\mathtt{split}(M, z.N)}} = \overline{\overline{M}}[k := \lambda m.m(\lambda x.\lambda z.\overline{\overline{N}})]$

8. $\overline{\overline{\mu\alpha.M}} = \overline{\overline{M}}[k := \mathtt{id}][\alpha := k]$; $\overline{\overline{[\alpha]M}} = k(\overline{\overline{M}}[k := \alpha])$,

where the auxiliary function Ψ for V with $\overline{\overline{V}} = k\Psi(V)$ is defined as follows:

$$V ::= x \mid \lambda x.M \mid \mathtt{maL}(M, x.N) \mid \langle M, N\rangle \mid \mathtt{inj}(M) \mid \lambda \mathtt{x}.M \mid (M, t) \mid [\alpha]M$$

1. $\Psi(x) = x$
2. $\Psi(\lambda x.M) = \lambda k.\mathtt{ppA}(k, x.\lambda k.\overline{\overline{M}})$
3. $\Psi(\mathtt{maL}(M, x.N)) = \lambda k.\overline{\overline{M}}[k := \lambda b.kb(\lambda x.\overline{\overline{N}}[k := \mathtt{id}])]$
4. $\Psi(\langle M, N\rangle) = \lambda k.\mathtt{when}(k, k.\overline{\overline{M}}, k.\overline{\overline{N}})$
5. $\Psi(\mathtt{inj}_i(M)) = \lambda k.\mathtt{let}\ \langle k_1, k_2\rangle = k\ \mathtt{in}\ \overline{\overline{M}}[k := k_i]$
6. $\Psi(\lambda \mathtt{x}.M) = \lambda k.\mathtt{split}(k, k.\overline{\overline{M}})$
7. $\Psi((M, t)) = \lambda k.\overline{\overline{M}}[k := kt]$
8. $\Psi([\alpha]M) = \overline{\overline{M}}[k := \alpha]$.

Note that instead one can take $\overline{\overline{MN}} = \overline{\overline{N}}[k := \lambda n.\overline{\overline{M}}[k := \lambda z.z(\mathtt{maL}(n, m.mk))]]$ for the case (b) above.

We define reduction for call-by-value $\to_V$ consisting of the following rules:

1. $(\lambda x.M)V \to M[x := V]$

2. $\mathtt{ppA}(\mathtt{maL}(V_1, x.N), y.V_2) \to N[x := V_2][y := V_1]$
3. $\mathtt{let}\ \langle x, y\rangle = \langle V_1, V_2\rangle\ \mathtt{in}\ M \to M[x := V_1, y := V_2]$
4. $\mathtt{when}(\mathtt{inj}_i(V), x_1.N_1, x_2.N_2) \to N_i[x_i := V]$
5. $(\lambda \mathtt{x}.M)t \to M$
6. $\mathtt{split}((V, t), z.N) \to N[z := V]$
7. $(\mu\alpha.M)N \to \mu\alpha.M[\alpha \Leftarrow N]$
8. $V(\mu\alpha.M) \to \mu\alpha.M[V \Rightarrow \alpha]$
9. $\mu\alpha.[\alpha]M \to M$ if $\alpha \notin FV(M)$
10. $[\beta](\mu\alpha.V) \to V[\alpha := \beta]$.

We note that the following substitution properties hold.

- $\overline{\overline{M}}[x := \Psi(V)] = \overline{\overline{M[x := V]}}$; $\Psi(V_1)[x := \Psi(V_2)] = \Psi(V_1[x := V_2])$
- $\Psi(V)[\alpha := \lambda z.\overline{\overline{N}}[k := \lambda n.z\,\mathtt{maL}(n, m.m\alpha)]] \to^*_\beta \Psi(V[\alpha \Leftarrow N])$ for value V,
- $\overline{\overline{V}}[\alpha := \lambda z.\overline{\overline{N}}[k := \lambda n.z\,\mathtt{maL}(n, m.m\alpha)]] \to^*_\beta \overline{\overline{V[\alpha \Leftarrow N]}}$ for value V,
- $\overline{\overline{M[\alpha \Leftarrow N]}} = \overline{\overline{M}}[\alpha := \lambda z.\overline{\overline{N}}[k := \lambda n.z\,\mathtt{maL}(n, m.m\alpha)]]$ for non-value M,
- $\overline{\overline{M[V \Rightarrow \alpha]}} = \overline{\overline{M}}[\alpha := \lambda n.\Psi(V)\,\mathtt{maL}(n, m.m\alpha)]$ for value V.

Accordingly, the reduction properties hold as follows.

- $\overline{\overline{(\mu\alpha.M)N}} = \overline{\overline{\mu\alpha.M[\alpha \Leftarrow N]}}$ for non-value M,
- $\overline{\overline{(\mu\alpha.V)N}} \to^*_\beta \overline{\overline{\mu\alpha.M[\alpha \Leftarrow N]}}$ for value V,
- $\overline{\overline{V(\mu\alpha.M)}} \to^*_\beta \overline{\overline{\mu\alpha.M[V \Rightarrow \alpha]}}$ for value V.

Now for the full system NK, the reduction correspondence can be established.

Proposition 13 (Reduction correspondence)
If $M \to^+_V V$ then $\overline{\overline{M}}[k := \mathtt{id}] \to^+_\beta \Psi(V)$.

We also have the following elementary properties where $\neg A := (A \to \bot)$:

- $\vdash_{\mathrm{Min}} \neg A \leftrightarrow \neg(\neg\neg\bot \leftarrow A)$, and then $\vdash_{\mathrm{Min}} (\neg A)^{q_4} \leftrightarrow \neg A^{q_4}$.
- $\vdash_{\mathrm{NJ}} (A \to \neg\neg B) \leftrightarrow \neg\neg(A \to B)$ and $\vdash_{\mathrm{NJ}} \neg\neg(A \leftarrow B) \leftrightarrow (\neg\neg A \leftarrow \neg\neg B)$.

Hence, the following proposition is obtained for the original Kuroda q, where define $(A \leftarrow B)^q = \neg\neg A^q \leftarrow B^q$.

Proposition 14 $\vdash_{\mathrm{Min}} \neg(A^{q_4})^{q_4} \leftrightarrow \neg A^q$.

In addition, ${q_4}^2$ works just as Glivenko's theorem for the $\forall$-free fragment of predicate logic, see 3.1 Glivenko's theorem.

Proposition 15 $\vdash_{\mathrm{NJ}} \neg\neg(A^{q_4})^{q_4} \leftrightarrow \neg\neg A$ *for the* $\forall$*-free fragment.*

In other words, the effect of the double-negation of Glivenko's theorem is separated into two steps of q_4, which works in minimal logic. Note also that $\forall xA := \neg\exists x\neg A$ provides $\vdash_{\mathrm{Min}} ((\forall xA)^{q_4})^{q_4} \leftrightarrow \neg\exists x\neg(A^{q_4})^{q_4}$.

In summary, four embeddings and their variants enjoy the following relations.

- $\vdash_{\mathrm{Min}} A^g \leftrightarrow A^k \leftrightarrow \neg\neg A^q \leftrightarrow \neg\neg(A^{q_4})^{q_4} \leftrightarrow \neg\neg(A^{k_4})^{k_4}$,
- $\vdash_{\mathrm{NJ}} \neg\neg(A^{q_4})^{q_4} \leftrightarrow \neg\neg A \leftrightarrow \neg\neg(A^{k_4})^{k_4}$ for the $\forall$-free fragment.

As concluding remarks, more detailed analysis will appear in a forthcoming paper.

References

[Plot75] G. Plotkin: Call-by-name, call-by-value and the λ-calculus, Theoretical Computer Science Vol. 1, pp. 125–159, 1975.

[TD88] A.S. Troelstra, D. van Dalen: Constructivism in Mathematics: An Introduction, Vol. I, II, North-Holland, 1988.

[NPS90] B. Nordström, K. Petersson, J.M. Smith: Programming in Martin-Löf's Type Theory, Oxford University Press, 1990.

[F95] K. Fujita: On embedding of classical substructural logics, Kyoto University RIMS Kôkyûroku Vol. 918, pp. 178–195, 1995.

[F00] K. Fujita: Domain-free $\lambda\mu$-calculus, Theoretical Informatics and Applications Vol. 34, pp. 433–466, 2000.

[Sel01] P. Selinger: Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus, Math. Struct. in Comp. Science Vol. 11, pp. 207–260, 2001.

[Wad03] Ph. Wadler: Call-by-name is dual to call-by-value, International Conference on Functional Programming, Sweden, 25–29 August 2003.

[F10] K. Fujita: CPS-translation as adjoint, Theoretical Computer Science Vol. 411, pp. 324–340, 2010.

[F22] 藤田憲悦：数理パズルで楽しく学べる論理学，コロナ社，2022.
`https://www.coronasha.co.jp/np/isbn/9784339029239/`

Fujita Kenetsu

Department of Computer Science, Gunma University

Kiryu, Gunma 376-8515, JAPAN

fujita@gunma-u.ac.jp