Case Calculus for Classical Logic

Baba, Kensuke
Graduate School of Information Science and Electrical Engineering, Kyushu Univ.

Hirokawa, Sachio
Computing and Communications Center, Kyushu Univ.

Kashima, Ryo
Department of Mathematical and Computing Sciences, Tokyo Institute of Technology

Komori, Yuichi
Department of Mathematics, Faculty of Science, Chiba Univ.

他

出版情報：DOI Technical Report. 178, 2000-08-08. Department of Informatics, Kyushu University
バージョン：
権利関係：

DOI-TR-178

Technical Report

Case Calculus for Classical Logic

by

Kensuke Baba, Sachio Hirokawa, Ryo Kashima,
Yuichi Komori, Izumi Takeuti

August 18, 2000

Department of Informatics
Kyushu University
Fukuoka 812-81, Japan

Email: baba@i.kyushu-u.ac.jp Phone: +81-92-641-3131(8806)

Case Calculus for Classical Logic

Kensuke Baba[1], Sachio Hirokawa[2], Ryo Kashima[3], Yuichi Komori[4], and Izumi Takeuti[5]

[1] Graduate School of Information Science and Electrical Engineering, Kyushu Univ.
[2] Computing and Communications Center, Kyushu Univ.
[3] Department of Mathematical and Computing Sciences, Tokyo Institute of Technology
[4] Department of Mathematics, Faculty of Science, Chiba Univ.
[5] Department of Information Science, Kyoto Univ.

Abstract. After the work of Griffin in 1990, there have been many works to extend the Curry-Howard isomorphism for classical logic. Most of these research uses sequent calculus with multiple conclusions. In this paper, we introduce a new natural deduction formulation with one conclusion for implicational classical logic. The idea of the formulation is based on proof by "case analysis". It is natural and captures our intuition in logical reasoning. We define reduction rules and prove weak normalization. As a corollary, we have the subformula property for normal proofs. The natural deduction systems proposed so far does not have subformula property. We also explain some computational meaning of our calculus by showing the simulation of β-reduction, disjunction and catch/throw.

1 Introduction

The Curry-Howard isomorphism[6] is one of the foundation in logic and computer science. It says that normalization of proof in intuitionistic logic corresponds to β-reduction of λ-calculus[2, 3]. After the work of Griffin[1] there have been many works to extend the isomorphism for classical logic and to understand computational meaning of classical logic[8–11, 14].

Most of these research uses sequent calculus with multiple conclusion. However, in sequent calculus, we need a new concept of list of formulae in right side which is not logical. It is regarded as disjunction in semantics, but it contains the axiom $\bot \to \alpha$ which means that any formula can be inferensed from falsehood and double-negation-elimination rule implicitly. Therefore we cannot separate implication and other connectives, which prevents us to analyze the distinction

of intuitionistic part and classical part clearly. For example, in Gentzen's sequent calculus LK, the right weakning rule allows the axiom $\bot \to \alpha$ implicitly. In Parigot's $\lambda\mu$-calculus[10], the right elimination rule of falsehood are allowed implicitly. Instead of sequent calculus, we can obtain classical logic by adding the formula for intuitionistic logic. We can prove the formula in NK – Gentzen's original natural deduction system for classical logic which has the law of excluded middle $\alpha \vee \neg\alpha$. But the proof contains negation and disjunction. Griffin's type system[1] uses double-negation-elimination rule $\neg\neg\alpha \to \alpha$. It also contains negation. Sato[14] introduce a classical natural deduction system $\mathsf{NK}_{c/t}$ with the catch and the throw rules. But the conclusion of the throw rule is contains disjunction. Therefore we cannot separate implication from disjunction. Read[13] focuses disjuction and formulates a deduction system with single conclusion. But implication and disjunction cannot be separated. The distinction of classical logic and intuitionistic logic can be clarified without negation and disjunction. The formula of Peirce $((\alpha \to \beta) \to \alpha) \to \alpha$ is an example which distinguishes classical logic from intuitionistic logic. Hirokawa et al.[4, 5] introduce a formulation of classical implicational logic with this formula. But this system does not have subformula property.

Therefore, we choose natural deduction formulation with single conclusion. It captures our intuition of logical reasoning such that we infer a conclusion from a set of assumptions. We consider implicational fragment in this paper. Because, it lies at the core of the isomorphism and we can clarify the distinction of classical logic and intuitionistic logic by considering implicational fragment. As an approach to obtain computational meaning of classical logic, we introduce a formulation of classical implicational logic and its normalization. The normal form proof we obtain satisfies subformula property.

We begin by Gentzen's natural deduction system for intuitionistic logic NJ [12]. Since we consider implicational fragment, there are only two inference rules I and E. It is known, e.g., [4], that classical implicational logic is obtained by adding Peirce's formula as an axiom or by adding the inference rule P.

$$\frac{\begin{array}{c}[\alpha]\\ \vdots\\ \beta\end{array}}{\alpha \to \beta}\ I \qquad \frac{\alpha \to \beta \quad \alpha}{\beta}\ E \qquad \frac{\begin{array}{c}[\alpha \to \beta]\\ \vdots\\ \alpha\end{array}}{\alpha}\ P$$

We can consider this inference P as an implicational representation of the classical absurdity rule. In this paper, we introduce the following rule C instead of P.

$$\frac{\begin{array}{cc}[\alpha \to \beta] & [\alpha]\\ \vdots & \vdots\\ \gamma & \gamma\end{array}}{\gamma}\ C$$

This reasoning is not an intuitionistic proof, because we do not know if α holds or not. The inferences P and C are equivalent in the sense that one is derived from another. P is derived from C as follows.

$$\frac{\begin{array}{cc}[\alpha \to \beta] & \\ \vdots & \\ \alpha & [\alpha]\end{array}}{\alpha}\ C$$

C is derived from P as follows.

$$\dfrac{\dfrac{\dfrac{\begin{array}{cc}[\gamma \to \beta]^2 & [\alpha]^1\\ & \vdots\\ & \gamma\end{array}}{\beta}\ E}{\alpha \to \beta}\ I^1}{\begin{array}{c}\vdots\\ \gamma\end{array}}{\gamma}\ P^2$$

Therefore the three rules I,E and C is a formulation of classical logic.

The inference C is a formulation of proof by "case analysis". Typical case is when $\beta = \bot$. Imagine that we are confirming some conjecture γ. We begin considering a typical case first, then other cases later. This is a proof by case analysis. We pay our attention to some property α for typical case. Then we analyse the situation into two cases – case 1 where α holds and case 2 where α does not hold, i.e., $\alpha \to \bot$ holds. If we succeed to confirm our conjecture γ in both cases, we have a proof for γ.

For technical reason, we modify the I-rule such that the assumption α is not discharged. We name such a modified rule as K.

$$\frac{\begin{array}{c}\vdots\\ \beta\end{array}}{\alpha \to \beta}\ K$$

The original I-rule is derived by K and C as follows.

$$\frac{[\alpha \to \beta] \quad \dfrac{\begin{array}{c}[\alpha]\\ \vdots\\ \beta\end{array}}{\alpha \to \beta}\ K}{\alpha \to \beta}\ C$$

Finally, we use K, E and C as our formulation of classical logic. Note that only C-rule discharges the assumption.

2 Terms for Case Calculus and Type Assignment

In this section, we introduce the terms to represent proof figures with the inference rules K, E and C and define the type assignment system.

The set of terms for our case calculus is constructed from variables and the constant **K** with application and "case"-construction.

Definition 1 (term).

1. *A variable x is a term.*
2. *If M is a term, then $\mathbf{K}M$ is a term.*
3. *If M and N are terms, then MN is a term.*
4. *If M and N are terms and x is a variable, then* **case** $x.M + N$ *is a term.*

A term which is constructed by 1, 2 and 3 (namely, contains no **case**) is called *case free.*

The set of formulas is constructed from type variables $a, b, c, \cdots$ with implication "$\to$". We use the word "types" and "formuas" with the same meaning in this paper.

A type assignment formula is an expression of the form $M : \alpha$ where M is a term and α is a type. The following is the type assignment formulation for the inference rules K, E and C.

Definition 2 (type assignment).

$$\frac{\begin{array}{c}\vdots\\ M:\alpha\end{array}}{\mathbf{K}M:\beta\to\alpha}\ K \qquad \frac{\begin{array}{cc}\vdots & \vdots\\ M:\alpha\to\beta & N:\alpha\end{array}}{MN:\beta}\ E \qquad \frac{\begin{array}{cc}x^-:\alpha\to\beta & x^+:\alpha\\ \vdots & \vdots\\ M:\gamma & N:\gamma\end{array}}{\mathbf{case}\ x.M+N:\gamma}\ C$$

Note that the variable x in the case rule C has two kinds of occurrences in M and in N. The occurrences in M is signed negative $(-)$ and the occurrences in N is signed positive $(+)$. A variable signed by case rule is called *bound variable*, otherwise called *free variable*. The variable x is assigned differently according to its sign. The same variable x does not appear free with positive and negative at the same time.

Our main concern is to find appropriate reduction rules for classical logic and to find a criterium to decide which reduction rules for proofs we should accept. In the present paper, we consider the subformula property as a criterium.

It is known that the formulation with I,E and P does not satisfy the subformula property.[1] The following is a normal form proof for $\alpha = ((a\to b)\to c)\to((b\to a)\to c)\to c$. The P-rule is shown as a different abstraction $\lambda\overline{x}.M$.

$$\dfrac{\dfrac{u:(a\to b)\to c \qquad \dfrac{\dfrac{z:c\to b \qquad \dfrac{y:(b\to a)\to c \qquad \dfrac{x:a}{\lambda w.x:b\to a}}{y(\lambda w.x):c}}{z(y(\lambda w.x)):b}}{\lambda x.z(y(\lambda w.x)):a\to b}}{u(\lambda x.z(y(\lambda w.x))):c}\ P}{\lambda\overline{z}.u(\lambda x.z(y(\lambda w.x))):c}\ I^*}{\lambda u\overline{y}\overline{z}.u(\lambda x.z(y(\lambda w.x))):\alpha}$$

Note that $z : c\to b$ is not a subformula of α. Detailed analysis of possible proofs reveals that any normal proof of α with I,E and P requires $c\to b$. [2]

[1] This fact might not be known widely.

[2] The authors thanks for the comment from Prof. Uesu and Prof. Shiraishi.

According to the equivalence, explained in the previous section, between $I+E+P$ and $K+E+C$, we have the following proof for α in $K+E+C$. We are applying the "case" rule for $z^- : c \to b$ and $z^+ : c$. The variable $z : c \to b$ occurs as an assumption before the case inference, but it does not remain in the conclusion of the case inference. This is where subformula property fails. If there is no such an occurrence of negative variable z of the form zM, such a negative variable is applied to some non-case variable, i.e., to the variables of λ-abstraction. Therefore a type of such negative variables is a subtype of usual λ-term. So, our goal is to delete such occurrences of negative variables. Note, here, that the argument $y(\mathbf{K}x) : c$ has the type c and that it is derived without using the negative assumption $z^- : c \to b$. If we substitute this proof $y(\mathbf{K}x) : c$ on the top $z^+ : c$ of proof for positive case, we have a proof without the case inference with respect to z. This process erases a negative occurrence case variable z^- of the form zM. In this way, the subformula property of the proof is derived from that of normal form proof of intuitionistic natural deduction system.

$$\dfrac{\dfrac{u : (a \to b) \to c \qquad \dfrac{\dfrac{\dfrac{x^- : a \to b \qquad \dfrac{\dfrac{z^- : c \to b \qquad \dfrac{y : (b \to a) \to c \qquad \dfrac{x^+ : a}{\mathbf{K}x : b \to a}}{y(\mathbf{K}x) : c}}{z(y(\mathbf{K}x)) : b}}{\mathbf{K}(z(y(\mathbf{K}x))) : a \to b}}{\mathbf{case}\ x.x + \mathbf{K}(z(y(\mathbf{K}x))) : a \to b}}{u(\mathbf{case}\ x.x + \mathbf{K}(z(y(\mathbf{K}x)))) : c} \qquad z^+ : c}{\mathbf{case}\ z.u(\mathbf{case}\ x.x + \mathbf{K}(z(y(\mathbf{K}x)))) + z) : c}\ C}{\lambda uy.(\mathbf{case}\ z.u(\mathbf{case}\ x.x + \mathbf{K}(z(y(\mathbf{K}x)))) + z) : \alpha}\ I*$$

3 Weak Normalization and Subformula Property

3.1 Case postponement

We formalise the intuitive idea of normalization explained in the previous section. We prove weak normalization. As a corollary, we have the subformula property for the normal form proofs.

The first reduction rules we introduce is to postpone the application of case rule. By repeating the postponement, a thread in a proof separates into intu-

itionistic part and classical part. Note that all the case rules have the same type in the result of postponement.

Definition 3. (case postpone rules)
kp: $\mathbf{K}(\mathbf{case}\ x.M + N) \to_c \mathbf{case}\ x.\mathbf{K}M + \mathbf{K}N$
lp: $L(\mathbf{case}\ x.M + N) \to_c \mathbf{case}\ x.LM + LN$
rp: $(\mathbf{case}\ x.M + N)L \to_c \mathbf{case}\ x.ML + NL$

The one step reduction relation $\to$ is the compatible closure of this reduction relation $\to_c$, and reduction relation $\Rightarrow$ is the reflexive and transitive closure of one step reduction relation $\to$.

A term is called a *case postponed form* (*cp-form*) iff no K-rule or E-rule appear below a C-rule.

If we consider the terms $\mathbf{K}M$ and PQ as product and $\mathbf{case}\ x.S+T$ as sum, the case postpone rules represent the distributive laws $(P+Q)(R+S) = PR+PS+QR+QS$. We can describe the case postpone rules in more general distributive rules in terms of contexts.

A context is a term with a hole [ ]. The set of contexts is defined as follows.

Definition 4. (context)

1. [ ] *is a context;*
2. $\mathbf{K}C[\,]$ *is a context, if* $\mathbf{K}$ *is a constant and* $C[\,]$ *is a context;*
3. $MC[\,]$ *and* $C[\,]M$ *are contexts, if* M *is a term and* $C[\,]$ *is a context;*
4. $\mathbf{case}\ x.M + C[\,]$ *and* $\mathbf{case}\ x.C[\,] + M$ *are contexts, if* x *is a variable,* M *is a terms and* $C[\,]$ *is a context;*

A context constructed by 1, 2 and 3 is called an *applicative context*, and a context constructed by 1 and 4 is called a *case context*. They are denoted by $C_a[\,]$ and $C_c[\,]$, respectively.

A cp-form M is constructed from case free terms $N_1, \cdots, N_n$ by case rules and can be represented as $M = C_c[N_1, \cdots, N_n]$. Now we have a general form of case postponement. If it is surrounded by an applicative context $C_a[\,]$, we can move case rules $C_c[\,, \cdots, \,]$ out of applicative context $C_a[\,]$.

$$C_a[C_c[N_1, \cdots, N_n]] \Rightarrow C_c[C_a[N_1]], \cdots, C_a[N_n]]$$

Then we have the following case-postponement theorem as an easy consequence.

Theorem 1. (cp-form) *Any term has a cp-form, i.e., for any term M, there exists a cp-form M' such that M reduces to M'.*

Proof. By induction on the structure of M.

1. $M = x$.

Since x is a cp-form, Thorem holds for $M' = x$.

2. $M = \mathbf{K}M_1$.

By induction hypothesis, we have a cp-form M_1' such that $M_1 \Rightarrow M_1'$. Since M_1' is a cp-form, M_1' has the form $M_1' = C_c[P_1, \cdots, P_n]$ for some case free term $P_1, \cdots, P_n$ and a case context $C_c[\ , \cdots, \ ]$. Then we have $M = \mathbf{K}M_1 \Rightarrow \mathbf{K}M_1' = \mathbf{K}C_c[P_1, \cdots, P_n] \Rightarrow C_c[\mathbf{K}P_1, \cdots, \mathbf{K}P_n]$. Since $P_1, \cdots, P_n$ are case free, $\mathbf{K}P_1, \cdots, \mathbf{K}P_n$ are case free. Therefore, $M' = C_c[\mathbf{K}P_1, \cdots, \mathbf{K}P_n]$ is a cp-form.

3. $M = M_1M_2$

By induction hypothesis, we have cp-form M_1' and M_2' such that $M_1 \Rightarrow M_1'$ and $M_2 \Rightarrow M_2'$. Since M_1' and M_2' are cp-form, M_1' and M_2' have the form $M_1' = C_c[P_1, \cdots, P_n]$ and $M_2' = C_c[Q_1, \cdots, Q_m]$ for some case free term $P_1, \cdots, P_n$ and $Q_1, \cdots, Q_m$ and a case context $C_c[\ , \cdots, \ ]$. Then we have $M = M_1M_2 \Rightarrow M_1'M_2' = C_c[P_1, \cdots, P_n]C_c[Q_1, \cdots, Q_m] \Rightarrow C_c[P_1C_c[Q_1, \cdots, Q_m], \cdots, P_nC_c[Q_1, \cdots, Q_m]] \Rightarrow C_c[C_c[P_1Q_1, \cdots, P_1Q_m], \cdots, C_c[P_nQ_1, \cdots, P_nQ_m]]$. Since $P_1, \cdots, P_n$ and $Q_1, \cdots, Q_m$ are case free, $P_1Q_1, \cdots, P_1Q_m, \cdots, P_nQ_1, \cdots, P_nQ_m$ are case free. Therefore, $M' = C_c[C_c[P_1Q_1, \cdots, P_1Q_m], \cdots, C_c[P_nQ_1, \cdots, P_nQ_m]]$ is a cp-form.

4. $M = \mathbf{case}\ x.M_1 + M_2$

By induction hypothesis, we have cp-forms M_1' and M_2' such that $M_1 \Rightarrow M_1', M_2 \Rightarrow M_2'$. Therefore, $M' = \mathbf{case}\ x.M_1' + M_2'$ is a cp-form and we have $M = \mathbf{case}\ x.M_1 + M_2 \Rightarrow \mathbf{case}\ x.M_1' + M_2'$. Thus Theorem holds for $M' = \mathbf{case}\ x.M_1' + M_2'$. QED

3.2 Weak Normalization

In addition to case postponement rules, we introduce the following three reduction rules. A term M is called *normal form* iff no reduction rules can be applied to M.

Definition 5. (reduction rules)

K:	$\mathbf{K}MN \to_c M$
γ:	$\mathbf{case}\ x.C_c[C_a[x^- M]] + N \to_c \mathbf{case}\ x.C_c[N[x^+ := M]] + N$
erase:	$\mathbf{case}\ x.M + N \to_c M \quad$ *if* $x^- \notin FV(M)$

The reduction rules represent the following normalization steps of proof figures.

$$\dfrac{\dfrac{\vdots}{M : \alpha}}{\dfrac{\mathbf{K}M : \beta \to \alpha \quad N : \beta}{\mathbf{K}MN : \alpha}} \quad \to_c \quad \begin{array}{c}\vdots\\ M : \alpha\end{array}$$

$$\dfrac{\dfrac{x^- : \alpha \to \beta \quad \begin{array}{c}\vdots\\ M : \alpha\end{array}}{xM : \beta}}{\dfrac{\begin{array}{c}\vdots\\ C_a[xM] : \gamma\end{array} \qquad \begin{array}{c}x^+ : \alpha\\ \vdots\\ N : \gamma\end{array}}{\mathbf{case}\ x.C_c[C_a[xM]] + N : \gamma}} \quad \to_c \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ M : \alpha\end{array}}{\begin{array}{c}\vdots\\ N[x^+ := M] : \gamma\end{array}} \qquad \begin{array}{c}x^+ : \alpha\\ \vdots\\ N : \gamma\end{array}}{C_c[N[x^+ := M]] : \gamma \quad N : \gamma \over \mathbf{case}\ x.C_c[N[x^+ := M] + N] : \gamma}$$

$$\dfrac{\begin{array}{c}\vdots\\ M : \gamma\end{array} \quad \begin{array}{c}\vdots\\ N : \gamma\end{array}}{\mathbf{case}\ x.M + N : \gamma} \quad \to_c \quad \begin{array}{c}\vdots\\ M : \gamma\end{array}$$

A term of form $\mathbf{K}MN$ is called *K-redex*, and a term of form $x^- M$ in $\mathbf{case}\ x.C_c[C_a[x^- M]] + N$ is called γ*-redex*. We call both of them *redex*.

The weak normlisation holds for the reduction rules.

Lemma 1. (normal form) *Any cp-form has a normal form, i.e., for any cp-form M, there exists a normal form M' such that M reduces to M'.*

Proof. First note that a term obtained by applying K or γ reduction to a cp-form is a cp-form. We define the degree $deg(R)$ of a redex R by $deg((\mathbf{K}(M^{\alpha}))^{\beta\to\alpha}N^{\beta}) = |\beta\to\alpha|$ and $deg(x^{-\alpha\to\beta}M^{\alpha}) = |\alpha\to\beta|$, where $|\alpha|$ is the number of occurrences of $\to$ in α. Then we can show that the degree of a new redex is smaller than that of the redex which creates such a new redex.

In the following two cases, a K reduction makes a new redex.

1. $(\mathbf{K}(\mathbf{K}P)^{\beta\to\alpha})^{\gamma\to\beta\to\alpha}N^{\gamma}Q^{\beta} \Rightarrow (\mathbf{K}P)^{\beta\to\alpha}Q^{\beta}$. Then, we have $deg((\mathbf{K}(\mathbf{K}P))^{\gamma\to\beta\to\alpha}N) > deg((\mathbf{K}P)^{\beta\to\alpha}Q)$.

2. $(\mathbf{K}x^{-\alpha\to\beta})^{\gamma\to(\alpha\to\beta)}N^{\gamma}Q^{\alpha} \Rightarrow x^{-\alpha\to\beta}Q^{\alpha}$ Then, we have $deg((\mathbf{K}x^{-})^{\gamma\to(\alpha\to\beta)}N) > deg(x^{-\alpha\to\beta}Q)$.

In the following two cases, a γ reduction makes new redexes.

3. **case** $x.C_c[C_a[x^{-(\beta\to\alpha)\to\gamma}(\mathbf{K}P)^{\beta\to\alpha}]]+N \Rightarrow$ **case** $x.C_c[N[x^{+\beta\to\alpha} := (\mathbf{K}P)^{\beta\to\alpha}]]+N$, where x^{+} takes the form of $x^{+\beta\to\alpha}Q^{\beta}$ in N. Then the result contains a K redex of the form $(\mathbf{K}P)^{\beta\to\alpha}Q'$. However, we have $deg(x^{-(\beta\to\alpha)\to\gamma}(\mathbf{K}P)) > deg((\mathbf{K}P)^{\beta\to\alpha}Q')$.

4. **case** $x.C_c[C_a[x^{-(\alpha\to\beta)\to\gamma}y^{-\alpha\to\beta}]]+N \Rightarrow$ **case** $x.C_c[N[x^{+\alpha\to\beta} := y^{-\alpha\to\beta}]])+N$, where x^{+} takes the form of $x^{+\alpha\to\beta}Q^{\alpha}$ in N. Then the result contains a γ redex of the form $y^{-\alpha\to\beta}Q'$. However, we have $deg(x^{-(\alpha\to\beta)\to\gamma}y^{-}) > deg(y^{-\alpha\to\beta}Q')$.

Thus the degree of a newly created redex is smaller than that of the old redex. Therefore, we can decrease the maximal degree of redexes or the number of occurrences of redexes with maximal degree, by reducing the rightmost redex. Eventually, we have a term which has no redex. Since an erase reduction makes no redex, we have normal form by applying erase reductions. QED

Theorem 2. (weak normalization) *Any term has normal form.*

Proof. By Theorem 1, any term can be reduced to a cp-form. Then by Lemma 1, the cp-form can be reduced to a normal form. QED

3.3 Subformula Property

The normal form proof satisfies the subformula property.

Theorem 3. (subformula property) *Let M be a normal form and Π be a type assignment to $M : \xi$. Then any type which occurs in Π is either a subtype of an assumption or a subtype of the conclusion ξ.*

Proof. Let Γ be the assumption set of the conclusion of Π.

1. M is case free. We prove Theorem by induction on the structure of Π. Since M is case free, Π is an axiom or the last inference rule of Π is K or E.

1.1. Π is axiom. The type assignment Π has the following form.

$$x : \xi \vdash x : \xi$$

Thus ξ is a subtype of the conclusion.

1.2. The last inference rule of Π is K. Then the type assignment Π has the following form.

$$\frac{\begin{array}{c}\vdots \\ \Gamma \vdash M_1 : \alpha\end{array}}{\Gamma \vdash \mathbf{K}M_1 : \beta \to \alpha\ (= \xi)}\ K$$

By induction hypothesis, any type in Π except for the last line is either a subtype of an assumption in Γ or a subtype of the conclusion α. Since α is subtype of $\beta \to \alpha$, any type in Π is either a subtype of an assumption in Γ or a subtype of the conclusion ξ.

1.3. The last inference rule of Π is E. Since M is a normal form, M has the form $M = xM_1 \cdots M_n$ and Π has the following form.

$$\frac{x : \alpha_1 \to \cdots \to \alpha_n \to \xi \vdash x : \alpha_1 \to \cdots \to \alpha_n \to \xi \quad \begin{array}{c}\vdots\ \Pi_1 \\ \Gamma_1 \vdash M_1 : \alpha_1\end{array} \quad \cdots \quad \begin{array}{c}\vdots\ \Pi_n \\ \Gamma_n \vdash M_n : \alpha_n\end{array}}{\{x : \alpha_1 \to \cdots \to \alpha_n \to \xi\} \cup \Gamma_1 \cup \cdots \cup \Gamma_n \vdash xM_1 \cdots M_n : \xi}\ E*$$

Let Π_i be the type assignment for $\Gamma_i \vdash M_i : \alpha_i$ $(1 \leq i \leq n)$. By induction hypothesis, any type in Π_i is either a subtype of an assumption in Γ_i or a subtype of the conclusion α_i. It is clear that α_i and $\alpha_i \to \cdots \to \alpha_n \to \xi$ are subtypes of $\alpha_1 \to \cdots \to \alpha_n \to \xi$. Since $\Gamma = \{x : \alpha_1 \to \cdots \to \alpha_n \to \xi\} \cup \Gamma_1 \cup \cdots \cup \Gamma_n$, any type in Π is a subtype of an assumption in Γ. Therefore, Theorem holds.

2. M is not case free. Let P be case free and maximal subterm of M, the type assignment Π has the following form.

$$\begin{array}{c} \vdots \\ \Gamma_1 \vdash P : \xi \\ \vdots\ (C) \\ \Gamma \vdash M : \xi \end{array}$$

By the definition of C, the type of P is same as one of M. The assumption set Γ equals to the assumption set Γ_1 minus assumptions which are discharged in C. Then, it suffices to show that the type of variable which discharged in a C rule is either a subtype of an assumption in Γ or a subtype of the conclusion ξ. We only have to show for the type which is maximal among such types. Let x be an occurrence of variable which has such maximal type. Since M is normal form (for erase), x is negative occurrence of the variable. We prove Theorem by case analysis of the occurrence of x in M.

2.1. **case** $y.x + N$. Then $P = x$ and Π has the following form.

$$\begin{array}{c} \begin{array}{cc} \vdots & \vdots \\ \Gamma_1 \vdash x : \xi & \Gamma_2 \vdash N : \xi \end{array} \\ \hline \Gamma_1 \cup \Gamma_2 \vdash \textbf{case}\ y.x + N : \xi \end{array} C$$

$$\begin{array}{c} \vdots\ (C) \\ \Gamma \vdash M : \xi \end{array}$$

Since the type of x is ξ, it is a subtype of the conclusion.

2.2. **case** $y.N + x$. By the same way as the case 2.1, we have that the type of x is ξ.

2.3. $\mathbf{K}x$. Then $\mathbf{K}x$ is a subterm of P and Π has the following form.

$$\begin{array}{c} \vdots \\ \dfrac{\Gamma_2 \vdash x : \alpha}{\Gamma_2 \vdash \mathbf{K}x : \beta \to \alpha}\ K \\ \vdots \\ \Gamma_1 \vdash P : \xi \\ \vdots\ (C) \\ \Gamma \vdash M : \xi \end{array}$$

Since P is case free, $\beta \to \alpha$ is either a subtype of an assumption in Γ_2 or a subtype of ξ. Since α is maxmal among types which are discharged by C, $\beta \to \alpha$

is not a subtype of type which is discharged. Therefore, $\beta \to \alpha$ is either a subtype of an assumption in Γ or a subtype of ξ, hence so is α.

2.4. xN. Since M is a normal form, M dose not contain a γ-redex. Since M contains a case-rule with respect to the variable x, M dose not contain such an occurrence xN.

2.5. Nx. Let the type of x be α. Then the type of N is $\alpha \to \beta$. By the same way as the case 2.3, we have that α is either a subtype of an assumption or a subtype of ξ. QED.

4 Computation in Case Calculus

4.1 Simulation of β-reduction

Case calculus with K+E+C may look strange as an extension of λ-calculus, because it does not contain I-rule, i.e., λ-abstraction. But as we explained in introduction, the λ-abstraction can be described by K and C with the following type assignment in case calculus.

$$\dfrac{[x^- : \alpha \to b] \quad \dfrac{\begin{array}{c}[x^+ : a] \\ \vdots \\ M : b\end{array}}{\mathbf{K}M : \alpha \to b}\, K}{\mathbf{case}\ x.x + \mathbf{K}M : \alpha \to b}\, C$$

We can define $\lambda x.M = \mathbf{case}\ x.x + \mathbf{K}M$. Moreover, we can simulate β-reduction by the reduction rules of case calculus.

$$\begin{array}{lll}
(\lambda x.M)N = (\mathbf{case}\ x.x + \mathbf{K}M)N & & \\
\quad \to \mathbf{case}\ x.xN + \mathbf{K}MN & \text{by } rp \\
\quad \to \mathbf{case}\ x.xN + M & \text{by } K \\
\quad \to \mathbf{case}\ x.M[x^+ = N] + M & \text{by } \gamma \\
\quad \to M[x^+ = N] & \text{by } erase
\end{array}$$

4.2 Simulation of Disjunction

It is known that disjunction can be defined by implication in classical logic as $\alpha \vee \beta = (\alpha \to \beta) \to \beta$. The following is a formulation of calculus for disjunction.

$$\frac{M:\alpha}{lM:\alpha\vee\beta} \qquad \frac{M:\beta}{rM:\alpha\vee\beta} \qquad \frac{P:\alpha\vee\beta \quad \begin{array}{c}[x:\alpha]\\ \vdots\\ M:\gamma\end{array} \quad \begin{array}{c}[y:\beta]\\ \vdots\\ N:\gamma\end{array}}{j(P,x.M,y.N):\gamma}$$

There are two reduction rules for this calculus.

$$j(lP,x.M,y.N)\rightarrow M[x:=P]$$

$$j(rP,x.M,y.N)\rightarrow N[y:=P]$$

We can translate this disjunction calculus into case calculus by $lM = \mathbf{case}\ x.x+\mathbf{K}(xM)$,$rM = \mathbf{K}M$ and $j(P,x.M,y.N) = \mathbf{case}\ x.N[y:=Px^-]+M$. This translation preserves the types and the two reduction can be simulated with the reduction of case calculus.

4.3 Failure of Church-Rosser and Simulation of Catch/Throw

Consider the the reduction rule γ for the case context $C_c[\ ] = [\ ]$, $N = x$. Then we have the following reduction.

$$\mathbf{case}\ x.C_a[x^-M]+x^+\rightarrow \mathbf{case}\ x.x^+[x^+:=M]+x^+\rightarrow \mathbf{case}\ x.M+x^+\rightarrow M$$

Therefore, if the negative variable x^- has multiple occurrence, say, xM_1 and xM_2, the term reduces to M_1 and M_2. Thus, case calculus does not have Church-Rosser property.

We see a similarity in the above reduction sequence with global exit control in conventional programming. The expression xM represents an exceptional exit with the value M. On the other hand, $\mathbf{case}\ x.C_a[\ ]$ captures this control. This intuition leads to simulation of catch/throw.

The following is a simple example of catch/throw from [7].

$$\begin{aligned} prd(l) &= catch(z, prd2(l))\\ prd2(l) &= if\ null(l)\ then\ 1\\ &\quad\ else\ if\ hd(l)=0\ then\ throw(z,0)\\ &\quad\ else\ hd(l)*prd2(tl(l)) \end{aligned}$$

We have the following computation.

$$\begin{aligned} & prd[2,3,0,4,5] \\ \Longrightarrow\ & catch(z, prd2[2,3,0,4,5]) \\ \Longrightarrow\ & catch(z, 2 * prd2[3,0,4,5]) \\ \Longrightarrow\ & catch(z, 2 * 3 * prd2[0,4,5]) \\ \Longrightarrow\ & catch(z, 2 * 3 * throw(z,0)) \\ \Longrightarrow\ & 0 \end{aligned}$$

We can define catch and throw by

$$catch(x,M) = \mathbf{case}\ x.M + x, \quad throw(x,M) = xM.$$

Then, the last step of the computation of $prd[2,3,0,4,5]$ can be simulated by $catch(z, 2 * 3 * throw(z,0)) = \mathbf{case}\ z.2 * 3 * z0 + z \to \mathbf{case}\ z.0 + z \to 0$.

More generally, we have the following.

$$\begin{aligned} catch(x, C[throw(x,M)]) &= \mathbf{case}\ x.C[xM] + x \\ &\to \mathbf{case}\ x.M + x \qquad \text{by } \gamma \\ &\to M \qquad\qquad\qquad \text{by } erase \end{aligned}$$

References

1. T. Griffin: "A Formulae-as-Types Notion of Control", Conference Record of 17th ACM Symposium on Principles of Programming Languages, pp. 47 – 58, 1990.
2. J. R. Hindley: "The Basic Type Theory", Cambridge University Press, 1996.
3. J. R. Hindley and J. P. Seldin: "Introduction to Combinators and λ-Calculus", Cambridge University Press, 1986.
4. S. Hirokawa, Y. Komori and I. Takeuti: "A Reduction Rule for Peirce Formula", Studia Logica, 56, 3, pp. 419 – 426, 1996.
5. S. Hirokawa: "Right weakning and right contraction in LK", Proc. CATS'96, Australian Computer Science Communications, 18, 3, pp. 168 – 174, 1996.
6. W. A. Howard: "The Formulae-as-Types Notion of Construction", reprinted in *The Curry-Howard Isomorphism* (de Groote, Ph. ed.), Academia, 1995.
7. Y. Kameyama and M. Sato: "Strong Normalizability of the Catch/Throw Calculi", Theoretical Computer Science, to appear.
8. C. Murthy: "An Evaluation Semantics for Classical Proofs", Proc. 6th Annual IEEE Symposium on Logic in Computer Science, pp. 96 – 107, 1991.
9. C. -H. L. Ong and C. A. Stewart: "A Curry-Howard Foundation for Functional Computation with Control", 24th ACM Symposium on Principles of Programming Languages, 1997.

10. M. Parigot: "$\lambda\mu$-calculus: An Algorithmic Interpretation of Classical Natural Deduction", Lecture Notes in Artificial Intelligence, 624, pp. 190 – 201, 1992.
11. M. Parigot: "Classical Proofs as Programs", Lecture Notes in Computer Science, 713, pp. 263 – 276, 1993.
12. D. Prawitz: "Natural Deduction, A Proof-Theoretical Study", Almqvist and Wiksell, Stockholm, 1965.
13. S. Read: "The Harmonious Classicist", manuscript, 1994.
14. M. Sato: "Intuitionistic and Classical Natural Deduction Systems with the Catch and the Throw Rules", Theoretical Computer Science, 175(1), pp. 75-92, 1997.