Type Checking and Typability in Domain-Free Lambda Calculi

Koji Nakazawa [a,*]

[a] *Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*

Makoto Tatsuta [b]

[b] *National Institute of Informatics, Japan*

Yukiyoshi Kameyama [c]

[c] *Department of Computer Science, University of Tsukuba, Japan*

Hiroshi Nakano [d]

[d] *Department of Applied Mathematics and Informatics, Ryukoku University, Japan*

Abstract

This paper shows (1) the undecidability of the type checking and the typability problems in the domain-free lambda calculus with negation, product, and existential types, (2) the undecidability of the typability problem in the domain-free polymorphic lambda calculus, and (3) the undecidability of the type checking and the typability problems in the domain-free lambda calculus with function and existential types. The first and the third results are proved by the second result and CPS translations that reduce those problems in the domain-free polymorphic lambda calculus to those in the the domain-free lambda calculi with existential types. The key idea is the conservativity of the domain-free lambda calculi with existential types over the images of the translations.

Key words: Existential Type, Type Checking, Typability, Undecidability, CPS Translation, Domain-Free Type System

* Corresponding author.
Email address: knak@kuis.kyoto-u.ac.jp (Koji Nakazawa).

1 Introduction

The polymorphic lambda calculus has been widely investigated since the monumental works due to Girard [7] and Reynolds [17]. On the other hand, the existential type, which corresponds to the second-order existence in logic through the Curry-Howard isomorphism, has been also studied actively from the point of view of computer science since Mitchell and Plotkin [11] showed that abstract data types are existential types.

The existential types also give a suitable target calculus for continuation-passing-style (CPS) translations. Thielecke showed that the negation ($\neg$) and conjunction ($\wedge$) fragment of a lambda calculus suffices for a CPS calculus [21] as the target of various first-order calculi. Recent studies on CPS translations for polymorphic calculi have shown that the $\neg \wedge \exists$-fragment of lambda calculus is an essence of a target calculus of CPS translations for various polymorphic calculi, such as the polymorphic lambda calculus [5], the lambda-mu calculus [3,9], and delimited continuations [10]. In [10], Hasegawa showed that the $\neg \wedge \exists$-fragment is suitable as a target calculus of a CPS translation for delimited continuations such as `shift` and `reset` [2].

Domain-free type systems [1], which are in an intermediate style between Church and Curry styles, are useful for considering some extension of polymorphic calculi and for theoretical studies on CPS translations. In domain-free style lambda calculi, the type of a bound variable is not explicit in $\lambda x.M$ as in Curry style, whereas terms contain type information for second-order quantifiers as in Church style, such as a type abstraction $\lambda X.M$ for $\forall$-introduction rule, and a term $\langle A, M\rangle$ with a "witness" A for $\exists$-introduction rule. In [8], it is shown that an extension of the Damas-Milner polymorphic type assignment system, which can be seen as a Curry-style formulation, with a control operator destroys the type soundness. In [3], Fujita showed that the Curry-style lambda-mu calculus, which is an extension of the polymorphic lambda calculus, does not have the subject reduction property, and he introduced a domain-free lambda-mu calculus $\lambda_V\mu$ which has the subject reduction property. In addition, the $\neg \wedge \exists$-fragment of the domain-free lambda calculus works as a target calculus of a CPS translation for $\lambda_V\mu$.

For type systems, the following decision problems are of great importance. Type checking (TC), which is symbolically described by $\Gamma \vdash M : A?$, asks whether $\Gamma \vdash M : A$ is derivable for given Γ, M, and A. Typability (TP) asks whether there exist a context Γ and a type A such that $\Gamma \vdash M : A$ is derivable for given M. Strong typability (STP) is a generalization of TP, and it asks whether there exist an extension Γ' of Γ and a type A such that $\Gamma' \vdash M : A$ is derivable for given M and Γ. Type inhabitation (INH) asks whether there exists a term M such that $\vdash M : A$ is derivable for given A. The problem INH

corresponds to provability of the formula A.

For polymorphic lambda calculi, some of these decision problems were given answers in the literature. Wells [23] showed undecidability of TC and TP in the Curry-style polymorphic lambda calculus, which we call Curry-F in this paper. Barthe and Sørensen [1] showed undecidability of TC and STP in the domain-free polymorphic lambda calculus, which we call DF-F, and Fujita and Schubert [4] independently showed the same result. However they did not show undecidability of TP in DF-F, which is stronger than the undecidability of STP.

By contrast, these decision problems for existential types have not been studied sufficiently, and we know only the following results. INH in the second-order propositional logic with logical connectives $\neg$, $\wedge$, $\forall$, and $\exists$ was proved to be decidable in [20], and INH in the logic with $\bot$, $\to$, $\wedge$, $\vee$, and $\exists$ was proved to be undecidable in [19]. Fujita and Schubert [6] studied the problems in the lambda calculi with existential types in the Curry style and the type-free style.

This paper proves undecidability of the type checking and the typability problems in domain-free lambda calculi with polymorphic and existential types. First, we show undecidability of TP in the domain-free polymorphic lambda calculus DF-F. Secondly, we show undecidability of TC and TP in the following domain-free lambda calculi with existential types: (1) a $\neg \wedge \exists$-fragment DF-$\lambda^{\neg\wedge\exists}$, (2) the system DF-$\lambda_g^{\neg\wedge\exists}$ of the $\neg \wedge \exists$-fragment with a generalized $\wedge$-elimination rule, and (3) an $\to\exists$-fragment DF-$\lambda^{\to\exists}$. Our results show that the system DF-$\lambda^{\neg\wedge\exists}$ is interesting, because Tatsuta et al. [20] showed the decidability of INH in it, while ours shows the undecidability of its TC and TP. So far we know few type systems that have this property except for trivial cases.

In order to prove undecidability of TP in DF-F, we show that TC can be reduced to TP in DF-F by a computable translation. Hence undecidability of TP follows from that of TC, which has already been proved in [1,4].

In order to prove undecidability of TC and TP for existential types, we reduce it to undecidability of TC and TP in DF-F. For DF-$\lambda^{\neg\wedge\exists}$, we define a negative translation $(\cdot)^\bullet$ from types of DF-F to types of DF-$\lambda^{\neg\wedge\exists}$, and a translation $[\![\cdot]\!]$ from terms of DF-F to terms of DF-$\lambda^{\neg\wedge\exists}$, which is a variant of the call-by-name CPS translation in [5]. We will show that $\Gamma \vdash M : A$ is derivable in DF-F if and only if $\neg\Gamma^\bullet \vdash [\![M]\!] : \neg A^\bullet$ is derivable in DF-$\lambda^{\neg\wedge\exists}$. Hence undecidability of TC in DF-$\lambda^{\neg\wedge\exists}$ follows from the undecidability of TC in DF-F. By the same idea, we can show that undecidability of TP in DF-$\lambda^{\neg\wedge\exists}$ also follows from that in DF-F.

The key of the proof is to show that $\neg\Gamma^\bullet \vdash [\![M]\!] : \neg A^\bullet$ in DF-$\lambda^{\neg\wedge\exists}$ implies $\Gamma \vdash M : A$ in DF-F. For a DF-F-term M, a type derivation of $\neg\Gamma^\bullet \vdash [\![M]\!] : \neg A^\bullet$

Systems	TC $\Gamma \vdash M : A$?	STP $\Gamma, ? \vdash M :?$	TP $? \vdash M :?$	INH $\vdash ? : A$
Curry-F	no[23]	no	no[23]	no
DF-F	no[1,4]	no[1,4]	NO	
DF-$\lambda^{\neg\wedge\exists}$	NO	NO	NO	yes[20]
DF-$\lambda_g^{\neg\wedge\exists}$	NO	NO	NO	yes
DF-$\lambda^{\to\exists}$	NO	NO	NO	unknown

Fig. 1. Decidability of TC, STP, TP and INH

in DF-$\lambda^{\neg\wedge\exists}$ may contain a type B which is not any CPS type, where CPS types are defined as types in the form of $\neg C^\bullet$ for some type C of DF-F. If a derivation contains such a type B, it does not correspond to any derivation in DF-F. However, in fact, we can define a contraction transformation that maps types of DF-$\lambda^{\neg\wedge\exists}$ to CPS types so that, from any type derivation of $\neg\Gamma^\bullet \vdash [\![M]\!] : \neg A^\bullet$, we can construct another type derivation of the same judgment in which every type is a CPS type. By this we can pull it back to a derivation in DF-F.

We summarize related results about decidability of TC, STP, TP, and INH in several systems in Figure 1, where "yes" means that the problem is decidable and "no" means undecidable. The results of this paper are denoted by "NO". Note that provability in the $\neg \wedge \exists$-fragment DF-$\lambda_g^{\neg\wedge\exists}$ with general elimination rules is equivalent to provability in the $\neg \wedge \exists$-fragment DF-$\lambda^{\neg\wedge\exists}$ with the ordinary elimination rules, so INH in DF-$\lambda_g^{\neg\wedge\exists}$ is the same problem as INH in DF-$\lambda^{\neg\wedge\exists}$.

Section 2 introduces the domain-free lambda calculi DF-F with polymorphic types and DF-$\lambda^{\neg\wedge\exists}$ with existential types. Section 3 gives our main theorems which state undecidability of TP in DF-F, and TC and TP in DF-$\lambda^{\neg\wedge\exists}$. Section 4 proves undecidability of TP in DF-F. Section 5 proves undecidability of TC and TP in DF-$\lambda^{\neg\wedge\exists}$, and applies the idea to DF-$\lambda_g^{\neg\wedge\exists}$. Section 6 discusses CPS translations for various systems to show that DF-$\lambda^{\neg\wedge\exists}$ is suitable for a target of CPS translations. Section 7 shows undecidability of TC and TP in a domain-free typed lambda calculus DF-$\lambda^{\to\exists}$ with implication and existence.

2 Domain-Free Lambda Calculi

In this section, we introduce two domain-free lambda calculi: one is the domain-free polymorphic lambda calculus DF-F, and the other is the negation,

conjunction, and existence fragment DF-$\lambda^{\neg\wedge\exists}$ of domain-free lambda calculus.

2.1 Polymorphic Lambda Calculus

First, we introduce the domain-free variant DF-F of the polymorphic lambda calculus.

Definition 1 (DF-F) (1) The language of the system DF-F contains type variables (denoted by X, Y,...) and term variables (denoted by x, y,...). The types (denoted by A, B,..., and called $\rightarrow\forall$-types), and the terms (denoted by M, N,...) of DF-F are defined by

$$A ::= X \mid (A{\rightarrow}A) \mid (\forall X.A),$$
$$M ::= x \mid (\lambda x.M) \mid (\lambda X.M) \mid (MM) \mid (MA),$$

Outermost parentheses are often omitted. For $n \geq 3$, $A_1{\rightarrow}A_2{\rightarrow}\cdots{\rightarrow}A_n$ denotes $A_1{\rightarrow}(A_2{\rightarrow}(\cdots{\rightarrow}A_n))$. For $m, n \geq 0$, $\lambda x_1 x_2 \cdots x_m.M_1M_2M_3\cdots M_n$ denotes $(\lambda x_1.(\lambda x_2.(\cdots(\lambda x_m.((\cdots((M_1M_2)M_3)\cdots)M_n))\cdots)))$. In the type $\forall X.A$, the variable X is bound in A. In the term $\lambda x.M$, the variable x is bound in M. In the term $\Lambda X.M$, the variable X is bound in M. A variable occurrence is free if it is not bound. We use $\equiv$ to denote syntactic identity modulo renaming of bound variables. In the system DF-F, the symbol $\bot$ denotes $\forall X.X$. The ordinary capture-avoiding substitution for types is denoted by $A[X := B]$.

(2) A context (denoted by Γ, Γ_1,...) is defined as a finite set of type assignments in the form of $(x : A)$. We suppose that $A \equiv B$ holds if both $(x : A)$ and $(x : B)$ are in Γ. We write $\Gamma, x : A$ for $\Gamma \cup \{x : A\}$, and Γ_1, Γ_2 for $\Gamma_1 \cup \Gamma_2$. The typing rules of DF-F are the following.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (\mathsf{Ax})$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A{\rightarrow}B}\ (\rightarrow\mathsf{I}) \qquad \frac{\Gamma_1 \vdash M : A{\rightarrow}B \quad \Gamma_2 \vdash N : A}{\Gamma_1, \Gamma_2 \vdash MN : B}\ (\rightarrow\mathsf{E})$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \lambda X.M : \forall X.A}\ (\forall\mathsf{I}) \qquad \frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash MB : A[X := B]}\ (\forall\mathsf{E})$$

In the rule $(\forall\mathsf{I})$, the lower sequent must not contain X freely. We write $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ to denote that $\Gamma \vdash M : A$ is derivable in DF-F.

We define the domain-free lambda calculus DF-$\lambda^{\neg\wedge\exists}$ with negation ($\neg$), conjunction ($\wedge$), and existence ($\exists$).

Definition 2 (DF-$\lambda^{\neg\wedge\exists}$) (1) The language of DF-$\lambda^{\neg\wedge\exists}$ contains type variables (denoted by X, Y,...), a type constant $\bot$, and term variables (denoted by x, y,...). The types (denoted by A, B,..., and called $\neg\wedge\exists$-types) and the terms (denoted by M, N,...) of DF-$\lambda^{\neg\wedge\exists}$ are defined by

$$\begin{aligned}\mathsf{A} ::= {} & X \mid \bot \mid \neg\mathsf{A} \mid (\mathsf{A}\wedge\mathsf{A}) \mid (\exists X.\mathsf{A}),\\ M ::= {} & x \mid (\lambda x.M) \mid \langle M, M\rangle \mid \langle \mathsf{A}, M\rangle\\ & \mid (MM) \mid (M\pi_1) \mid (M\pi_2) \mid (M[Xx.M]).\end{aligned}$$

Outermost parentheses are often omitted. In the type $\exists X.\mathsf{A}$, the variable X is bound in A. In the term $\lambda x.M$, the variable x is bound in M. In the term $N[Xx.M]$, the variables X and x are bound in M.

(2) The definition of the contexts is similar to DF-F, and $\neg\Gamma$ is defined as $\{(x:\neg\mathsf{A})|(x:\mathsf{A})\in\Gamma\}$. The typing rules of DF-$\lambda^{\neg\wedge\exists}$ are the following.

$$\frac{}{\Gamma, x:\mathsf{A}\vdash x:\mathsf{A}}\ (\mathsf{Ax})$$

$$\frac{\Gamma, x:\mathsf{A}\vdash M:\bot}{\Gamma\vdash\lambda x.M:\neg\mathsf{A}}\ (\neg\mathsf{I}) \qquad \frac{\Gamma_1\vdash M:\neg\mathsf{A}\quad \Gamma_2\vdash N:\mathsf{A}}{\Gamma_1,\Gamma_2\vdash MN:\bot}\ (\neg\mathsf{E})$$

$$\frac{\Gamma_1\vdash M:\mathsf{A}\quad \Gamma_2\vdash N:\mathsf{B}}{\Gamma_1,\Gamma_2\vdash\langle M,N\rangle:\mathsf{A}\wedge\mathsf{B}}\ (\wedge\mathsf{I}) \qquad \frac{\Gamma\vdash N:\mathsf{A}[X:=\mathsf{B}]}{\Gamma\vdash\langle\mathsf{B},N\rangle:\exists X.\mathsf{A}}\ (\exists\mathsf{I})$$

$$\frac{\Gamma\vdash M:\mathsf{A}_1\wedge\mathsf{A}_2}{\Gamma\vdash M\pi_1:\mathsf{A}_1}\ (\wedge\mathsf{E1}) \qquad \frac{\Gamma\vdash M:\mathsf{A}_1\wedge\mathsf{A}_2}{\Gamma\vdash M\pi_2:\mathsf{A}_2}\ (\wedge\mathsf{E2})$$

$$\frac{\Gamma_1\vdash M:\exists X.\mathsf{A}\quad \Gamma_2, x:\mathsf{A}\vdash N:\mathsf{C}}{\Gamma_1,\Gamma_2\vdash M[Xx.N]:\mathsf{C}}\ (\exists\mathsf{E})$$

The ordinary capture-avoiding substitution for types is denoted by $\mathsf{A}[X:=\mathsf{B}]$. In the rule ($\exists\mathsf{E}$), the variable X must not occur in Γ_2 nor C freely. We write $\Gamma\vdash_{\lambda^{\neg\wedge\exists}} M:\mathsf{A}$ to denote that $\Gamma\vdash M:\mathsf{A}$ is derivable by the typing rules above.

In Section 6, we will show that this system is useful for a target of CPS translations. In addition, $\lambda^{\neg\wedge\exists}$ represents every function representable in the polymorphic typed lambda calculus, because there exists a CPS-translation from the polymorphic typed lambda calculus to this calculus.

3 Type Checking and Typability

In this section, we introduce some decision problems on typability of terms, and state our main theorems.

Type checking (TC) asks, for given Γ, M, and A, whether $\Gamma \vdash M : A$ is derivable. Typability (TP) asks, for given M, whether there exist a context Γ and a type A such that $\Gamma \vdash M : A$ is derivable.

Type inhabitation (INH) is another problem, which asks, for given A, whether there exists a term M such that $\vdash M : A$ is derivable. The problem INH corresponds to provability of the formula A. In [20], Tatsuta et al. proved decidability of INH in $\lambda^{\neg\wedge\exists}$, which immediately implies decidability of INH in DF-$\lambda^{\neg\wedge\exists}$.

In [1,4], undecidability of TC in DF-F is proved.

Theorem 3 ([1,4]) *Type checking is undecidable in* DF-F.

They also have proved undecidability of strong typability and type synthesis in DF-F. Strong typability is a generalization of TP, and it asks, for given M and Γ, whether there exist an extension Γ' of Γ and a type A such that $\Gamma' \vdash M : A$ is derivable. Type synthesis asks, for given M and Γ, whether there exists a type A such that $\Gamma \vdash M : A$ is derivable. The undecidability of TP in DF-F is stronger than the undecidability of the strong typability in DF-F, and their approach does not seem to apply to the undecidability of TP.

Theorem 4 *Typability is undecidable in* DF-F.

Undecidability of TC and TP in DF-$\lambda^{\neg\wedge\exists}$ is proved by showing that these problems in DF-F can be reduced to those in DF-$\lambda^{\neg\wedge\exists}$.

Theorem 5 *Type checking and typability are undecidable in* DF-$\lambda^{\neg\wedge\exists}$.

Theorem 4 will be proved in Section 4, and Theorem 5 will be proved in Section 5.

4 Undecidability of TP in DF-F

This section proves Theorem 4. It is proved by a reduction from TC to TP in DF-F. In order to obtain the reduction, we use a technique inspired by [23].

Lemma 6 *For any* $\rightarrow\forall$*-type* A *and any closed* DF-F*-term* M*, we can effec-*

tively construct a closed DF-F*-term* J *such that* $\vdash_{\mathsf{DF}\text{-}F} M : A$ *is derivable if and only if, for some* $\to\forall$*-type* B, $\vdash_{\mathsf{DF}\text{-}F} J : B$ *is derivable.*

PROOF. Let x and y be fresh variables and take J as $\lambda x.(\lambda y.x(A{\to}\bot)M)(x(\bot{\to}\bot)x)$. It is easily proved that if $\vdash M : A$ is derivable then J is typable, so we will prove the converse direction.

First, we define three partial functions on types: lvar, ldep, and lbdep. The function $\mathsf{lvar}(A)$ is the leftmost-variable occurrence of A when it is free in A, and $\mathsf{lvar}(A)$ is undefined otherwise. We define lvar by

$$\begin{aligned}
\mathsf{lvar}(X) &\equiv X, \\
\mathsf{lvar}(A{\to}B) &\equiv \mathsf{lvar}(A), \\
\mathsf{lvar}(\forall X.A) &\equiv \mathsf{undefined} && (\text{if } \mathsf{lvar}(A) \equiv X), \\
\mathsf{lvar}(\forall X.A) &\equiv \mathsf{lvar}(A) && (\text{otherwise}).
\end{aligned}$$

The left depth $\mathsf{ldep}(A)$ is the depth from the root to the leftmost-variable occurrence in the abstract syntax tree of A, and it is defined by

$$\begin{aligned}
\mathsf{ldep}(X) &= 0, \\
\mathsf{ldep}(A{\to}B) &= \mathsf{ldep}(A) + 1, \\
\mathsf{ldep}(\forall X.A) &= \mathsf{ldep}(A) + 1.
\end{aligned}$$

The leftmost-bound-variable depth $\mathsf{lbdep}(A)$ is $\mathsf{ldep}(B)$ when A contains the subexpression $\forall X.B$ and the leftmost variable of A is bound by this quantifier. We define lbdep by

$$\begin{aligned}
\mathsf{lbdep}(X) &= \mathsf{undefined}, \\
\mathsf{lbdep}(A{\to}B) &= \mathsf{lbdep}(A), \\
\mathsf{lbdep}(\forall X.A) &= \mathsf{ldep}(A) && (\text{if } \mathsf{lvar}(A) \equiv X), \\
\mathsf{lbdep}(\forall X.A) &= \mathsf{lbdep}(A) && (\text{otherwise}).
\end{aligned}$$

Then we have the following lemmas: (1) $\mathsf{ldep}(A[X := B]) \neq \mathsf{ldep}(A)$ implies $\mathsf{lvar}(A) \equiv X$, (2) $\mathsf{lvar}(A) \equiv X$ implies $\mathsf{lbdep}(A[X := B]) = \mathsf{lbdep}(B)$ and $\mathsf{lvar}(A[X := B]) \equiv \mathsf{lvar}(B)$. These are proved by induction on A straightforwardly.

Now suppose that we have a derivation D of $\vdash J : B$ for some B. By analyzing the form of J, the derivation D must be

$$\dfrac{\dfrac{\dfrac{\vdots\, D_1}{x : C, y : B_2 \vdash x(A \to \bot)M : B_1}}{x : C \vdash \lambda y.x(A \to \bot)M : B_2 \to B_1} \quad \dfrac{\vdots\, D_2}{x : C \vdash x(\bot \to \bot)x : B_2}}{\dfrac{x : C \vdash (\lambda y.x(A \to \bot)M)(x(\bot \to \bot)x) : B_1}{\vdash J : B}}$$

for some types B_1, B_2, and C such that $B \equiv C{\to}B_1$.

We will show that C must be $\bot$. Since $x(\bot{\to}\bot)x$ is typable, we have $C \equiv \forall X.C_1$ and $C_1[X := \bot{\to}\bot] \equiv C{\to}B_2$ for some C_1. Then, $\mathsf{ldep}(C_1[X := \bot{\to}\bot])$ is equal to $\mathsf{ldep}((\forall X.C_1){\to}B_2) = \mathsf{ldep}(C_1) + 2$, which is not equal to $\mathsf{ldep}(C_1)$, so we have $\mathsf{lvar}(C_1) \equiv X$ by the lemma (1). By the lemma (2), we have $\mathsf{lbdep}(C_1[X := \bot{\to}\bot]) = \mathsf{lbdep}(\bot{\to}\bot) = 0$, so we have $\mathsf{lbdep}(C{\to}B_2) = 0$. By definition, we have $\mathsf{lbdep}(C{\to}B_2) = \mathsf{ldep}(C_1)$ since $\mathsf{lvar}(C_1) \equiv X$ holds. Therefore, $\mathsf{ldep}(C_1) = 0$ holds, which means that C_1 is a variable, and it must be X since $\mathsf{lvar}(C_1) \equiv X$.

Hence we have $C \equiv \bot$, and then D_1 must be

$$\dfrac{\dfrac{\overline{x : \bot, y : B_2 \vdash x : \bot}}{x : \bot, y : B_2 \vdash x(A \to \bot) : A \to \bot} \quad \vdots\ \vdash M : A}{x : \bot, y : B_2 \vdash x(A \to \bot)M : B_1}$$

so $\vdash M : A$ is derivable. □

Proposition 7 *Type checking can be reduced to typability in* DF-F.

PROOF. A judgment $x_1 : A_1, \cdots, x_n : A_n \vdash M : A$ is derivable if and only if $\vdash \lambda x_1.\cdots\lambda x_n.M : A_1{\to}\cdots{\to}A_n{\to}A$ is derivable, which can be reduced to a typability problem for some J by Lemma 6. □

PROOF. (Proof of Theorem 4) It immediately follows from Theorem 3 and Proposition 7. □

5 Undecidability of TC and TP in DF-$\lambda^{\neg\wedge\exists}$

This section is devoted to the proof of Theorem 5. The subsection 5.1 will define a CPS translation from DF-F to DF-$\lambda^{\neg\wedge\exists}$. We will also define an inverse CPS translation from the image DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ of the CPS translation to DF-F. The subsection 5.2 will show the main lemma, which directly implies that DF-$\lambda^{\neg\wedge\exists}$ is conservative over DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. The subsection 5.3 will finish our undecidability proof. Our method will be applied to a variant DF-$\lambda^{\neg\wedge\exists}_{g}$ with general elimination rules in the subsection 5.4.

We give a CPS translation for DF-F in this subsection. Our translation is inspired by Fujita's translation in [5], but we cannot use it directly for the undecidability proof in domain-free calculi, since his calculus is in Church style. In the following, we modify the translation appropriately.

Definition 8 (CPS Translation) (1) The negative translation from $\rightarrow\forall$-types to $\neg \wedge \exists$-types is defined by

$$\begin{aligned} X^\bullet &\equiv X, \\ (A{\rightarrow}B)^\bullet &\equiv \neg A^\bullet \wedge B^\bullet, \\ (\forall X.A)^\bullet &\equiv \exists X.A^\bullet. \end{aligned}$$

For a context Γ, we define $\Gamma^\bullet$ as $\{(x : A^\bullet) | (x : A) \in \Gamma\}$.

(2) The CPS translation from terms of DF-F to terms of DF-$\lambda^{\neg\wedge\exists}$ is defined by

$$\begin{aligned} [\![x]\!] &\equiv \lambda k.xk, \\ [\![\lambda x.M]\!] &\equiv \lambda k.(\lambda x.[\![M]\!](k\pi_2))(k\pi_1), \\ [\![\lambda X.M]\!] &\equiv \lambda k.k[Xk'.[\![M]\!]k'], \\ [\![MN]\!] &\equiv \lambda k.[\![M]\!]\langle [\![N]\!], k\rangle, \\ [\![MA]\!] &\equiv \lambda k.[\![M]\!]\langle A^\bullet, k\rangle, \end{aligned}$$

where variables k and k' are supposed to be fresh.

Note that this CPS translation preserves the β-reduction relation, and does not preserve the η-reduction.

Proposition 9 *If* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ *holds, then we have* $\neg\Gamma^\bullet \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^\bullet$.

Definition 10 (DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$) (1) The continuation types (denoted by $\mathcal{A}$, $\mathcal{B}$,...) are defined by

$$\mathcal{A} ::= X \mid \neg\mathcal{A} \wedge \mathcal{A} \mid \exists X.\mathcal{A},$$

and the CPS types are defined as types in the form $\neg\mathcal{A}$.

The set of CPS terms (denoted by P, Q,...), which is a subset of DF-$\lambda^{\neg\wedge\exists}$-terms, is inductively defined as follows: if x is a variable, P and Q are CPS terms, $\mathcal{A}$ is a continuation type, X is a type variable, and k and k' are fresh variables, then $\lambda k.xk$, $\lambda k.(\lambda x.P(k\pi_2))(k\pi_1)$, $\lambda k.k[Xk'.Pk']$, $\lambda k.P\langle P, k\rangle$, and $\lambda k.P\langle \mathcal{A}, k\rangle$ are CPS terms.

(2) We define the subsystem DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ of DF-$\lambda^{\neg\wedge\exists}$ by restricting terms and types to CPS terms and CPS types, respectively. Judgments of DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$

are in the form of $\neg\Gamma \vdash P : \neg\mathcal{A}$ for a CPS term P, a CPS type $\neg\mathcal{A}$, and a context $\neg\Gamma$ every element of which is in the form of $(x : \neg\mathcal{B})$ for a CPS type $\neg\mathcal{B}$. The typing rules of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ are the following.

$$\frac{}{\neg\Gamma, x : \neg\mathcal{A} \vdash \lambda k.xk : \neg\mathcal{A}}$$

$$\frac{\neg\Gamma, x : \neg\mathcal{A} \vdash P : \neg\mathcal{B}}{\neg\Gamma \vdash \lambda k.(\lambda x.P(k\pi_2))(k\pi_1) : \neg(\neg\mathcal{A} \wedge \mathcal{B})}$$

$$\frac{\neg\Gamma_1 \vdash P : \neg(\neg\mathcal{A} \wedge \mathcal{B}) \quad \neg\Gamma_2 \vdash Q : \neg\mathcal{A}}{\neg\Gamma_1, \neg\Gamma_2 \vdash \lambda k.P\langle Q, k\rangle : \neg\mathcal{B}}$$

$$\frac{\neg\Gamma \vdash P : \neg(\exists X.\mathcal{B})}{\neg\Gamma \vdash \lambda k.P\langle \mathcal{A}, k\rangle : \neg\mathcal{B}[X := \mathcal{A}]}$$

$$\frac{\neg\Gamma \vdash P : \neg\mathcal{B}}{\neg\Gamma \vdash \lambda k.k[Xk'.Pk'] : \neg\exists X.\mathcal{B}} \ (X \notin FV(\Gamma))$$

We write $\neg\Gamma \vdash_{\mathsf{cps}} P : \neg\mathcal{A}$ to denote that the judgment is derivable in $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$.

Lemma 11 *(1) The set of the continuation types coincides with the image of the negative translation.*
(2) The set of the CPS terms coincides with the image of the CPS translation.

PROOF. (1) It is proved by the induction on the continuation types that any continuation type is in the image of the negative translation. It is proved by the induction on the $\rightarrow\forall$-types that $A^\bullet$ is a continuation type for any $\rightarrow\forall$-type A.

(2) It is proved by the induction on the CPS terms that any CPS term is in the image of the CPS translation. It is proved by the induction on the $\mathsf{DF}\text{-}F$-terms that $[\![M]\!]$ is a CPS term for any $\mathsf{DF}\text{-}F$-term M. □

Definition 12 (Inverse CPS Translation) The inverse translation $(\cdot)^\circ$ from continuation types to $\rightarrow\forall$-types is defined by

$$X^\circ \equiv X,$$
$$(\neg\mathcal{A} \wedge \mathcal{B})^\circ \equiv \mathcal{A}^\circ \rightarrow \mathcal{B}^\circ,$$
$$(\exists X.\mathcal{A})^\circ \equiv \forall X.\mathcal{A}^\circ.$$

For a context Γ every element of which is in the form of $(x : \mathcal{A})$ for some continuation type $\mathcal{A}$, we define Γ° as $\{(x : \mathcal{A}^\circ) | (x : \mathcal{A}) \in \Gamma\}$.

The inverse translation $(\cdot)^{\#}$ from CPS terms to terms of DF-F is defined by

$$\begin{aligned}(\lambda k.xk)^{\#} &\equiv x,\\ (\lambda k.(\lambda x.P(k\pi_2))(k\pi_1))^{\#} &\equiv \lambda x.P^{\#},\\ (\lambda k.k[Xk'.Pk'])^{\#} &\equiv \lambda X.P^{\#},\\ (\lambda k.P\langle Q,k\rangle)^{\#} &\equiv P^{\#}Q^{\#},\\ (\lambda k.P\langle \mathcal{A},k\rangle)^{\#} &\equiv P^{\#}\mathcal{A}^{\circ}.\end{aligned}$$

Lemma 13 *(1) For any $\rightarrow\forall$-type A, we have $A^{\bullet\circ} \equiv A$.*
(2) For any DF-F-term M, we have $[\![M]\!]^{\#} \equiv M$.

Proposition 14 *(1) If $\neg\Gamma \vdash_{\mathsf{cps}} P : \neg\mathcal{A}$ holds, then $\Gamma^{\circ} \vdash_{\mathsf{DF}\text{-}F} P^{\#} : \mathcal{A}^{\circ}$ holds.*
(2) If $\neg\Gamma^{\bullet} \vdash_{\mathsf{cps}} [\![M]\!] : \neg A^{\bullet}$ holds, then $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ holds.

PROOF. (1) By induction on the derivation.

(2) By (1), we have $\Gamma^{\bullet\circ} \vdash_{\mathsf{DF}\text{-}F} [\![M]\!]^{\#} : A^{\bullet\circ}$. By Lemma 13, we have the claim. □

5.2 Typing for CPS Terms in DF-$\lambda^{\neg\wedge\exists}$

Proposition 9 shows that, for any typable term M in DF-F, its CPS translation $[\![M]\!]$ has a CPS type. In fact, the converse can be also proved. In order to prove that, in this subsection, we will show that DF-$\lambda^{\neg\wedge\exists}$ is conservative over DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$.

A type derivation of a CPS term in DF-$\lambda^{\neg\wedge\exists}$ may contain a type which is not in DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ even if its conclusion is in DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. For example, let P be $\lambda k.P_1\langle P_2,k\rangle$, where $P_1 \equiv \lambda l.(\lambda y.(\lambda l'.zl')(l\pi_2))(l\pi_1)$ and $P_2 \equiv \lambda k.(\lambda x.(\lambda k'.xk')(k\pi_2))(k\pi_1)$. Then both $z : \neg\mathcal{B} \vdash P_1 : \neg(\neg(\neg\mathsf{A}\wedge\mathsf{A})\wedge\mathcal{B})$ and $\vdash P_2 : \neg(\neg\mathsf{A}\wedge\mathsf{A})$ are derivable in DF-$\lambda^{\neg\wedge\exists}$ for any type A and any continuation type $\mathcal{B}$. Therefore the following is a derivation in DF-$\lambda^{\neg\wedge\exists}$.

$$\dfrac{\dfrac{z : \neg\mathcal{B} \vdash P_1 : \neg(\neg(\neg\mathsf{A}\wedge\mathsf{A})\wedge\mathcal{B}) \quad \dfrac{\vdots}{\vdash P_2 : \neg(\neg\mathsf{A}\wedge\mathsf{A})} \quad \overline{k : \mathcal{B} \vdash k : \mathcal{B}}}{k : \mathcal{B} \vdash \langle P_2,k\rangle : \neg(\neg(\neg\mathsf{A}\wedge\mathsf{A})\wedge\mathcal{B})}}{\dfrac{z : \neg\mathcal{B}, k : \mathcal{B} \vdash P_1\langle P_2,k\rangle : \bot}{z : \neg\mathcal{B} \vdash P : \neg\mathcal{B}}}$$

The conclusion $z : \neg\mathcal{B} \vdash P : \neg\mathcal{B}$ is a judgment of DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ since P is a CPS term and $\neg\mathcal{B}$ is a CPS type, but the derivation is not in DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ when A is not a continuation type. However, as we can see in the example, such a type A

can be replaced by arbitrary type without changing the form of the derivation. In the case of the example, if we replace A in the derivation by a continuation type, we can obtain a derivation in $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ of the same conclusion.

In general, we can define a translation $(\cdot)^{\mathsf{c}}$ from $\neg\wedge\exists$-types to CPS types such that, for any CPS term P and any type derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \mathsf{A}$, we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \mathsf{A}^{\mathsf{c}}$. We call the translation $(\cdot)^{\mathsf{c}}$ the contraction translation. Moreover, we have $(\neg A^{\bullet})^{\mathsf{c}} \equiv \neg A^{\bullet}$, so the property of the contraction translation implies that $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$ is conservative over $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$.

Definition 15 (Contraction Translation) Let $\mathcal{S}$ be a fixed closed continuation type, such as $\exists X.X$. The contraction translation $(\cdot)^{\mathsf{c}}$ from $\neg\wedge\exists$-types to CPS types is mutually defined with the auxiliary translation $(\cdot)^{\mathsf{d}}$ from $\neg\wedge\exists$-types to continuation types by

$$\begin{aligned}
(\neg\mathsf{A})^{\mathsf{c}} &\equiv \neg\mathsf{A}^{\mathsf{d}}, \\
\mathsf{A}^{\mathsf{c}} &\equiv \neg\mathcal{S} && (\mathsf{A} \text{ is not a negation}), \\
X^{\mathsf{d}} &\equiv X, \\
\bot^{\mathsf{d}} &\equiv \mathcal{S}, \\
(\neg\mathsf{A})^{\mathsf{d}} &\equiv \mathcal{S}, \\
(\mathsf{A}\wedge\mathsf{B})^{\mathsf{d}} &\equiv \mathsf{A}^{\mathsf{c}}\wedge\mathsf{B}^{\mathsf{d}}, \\
(\exists X.\mathsf{A})^{\mathsf{d}} &\equiv \exists X.\mathsf{A}^{\mathsf{d}}.
\end{aligned}$$

For a context Γ, we define Γ^{c} as $\{(x : \mathsf{A}^{\mathsf{c}}) | (x : \mathsf{A}) \in \Gamma\}$.

Lemma 16 *(1) For any continuation type $\mathcal{A}$, we have $(\neg\mathcal{A})^{\mathsf{c}} \equiv \neg\mathcal{A}$ and $\mathcal{A}^{\mathsf{d}} \equiv \mathcal{A}$.*

(2) For any continuation type $\mathcal{A}$ and any $\neg\wedge\exists$-type B, we have $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[X := \mathcal{A}]$ and $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[X := \mathcal{A}]$.

PROOF. (1) By induction on $\mathcal{A}$.

(2) By induction on B. Note that any continuation type $\mathcal{A}$ is not a negation, so we have $\mathcal{A}^{\mathsf{c}} \equiv \neg\mathcal{S}$. □

Lemma 17 (Main Lemma) *If P is a CPS term, then $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : A$ implies $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : A^{\mathsf{c}}$.*

PROOF. By induction on P. Note that any type of P is a negation, since any CPS term is a λ-abstraction. So we will show that $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ implies $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\mathsf{A}^{\mathsf{d}}$.

Case $P \equiv \lambda k.xk$. Any derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form.

$$\dfrac{\dfrac{}{\Gamma \vdash x : \neg\mathsf{A}} \quad \dfrac{}{k : \mathsf{A} \vdash k : \mathsf{A}}}{\dfrac{\Gamma, k : \mathsf{A} \vdash xk : \bot}{\Gamma \vdash \lambda k.xk : \neg\mathsf{A}}}$$

Since $(x : \neg\mathsf{A}) \in \Gamma$ holds, we have $(x : \neg\mathsf{A}^{\mathsf{d}}) \in \Gamma^{\mathsf{c}}$, so $\Gamma^{\mathsf{c}} \vdash P : \neg\mathsf{A}^{\mathsf{d}}$ is derivable.

Case $P \equiv \lambda k.(\lambda x.Q(k\pi_2))(k\pi_1)$. Any derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form, where A must be $\mathsf{B} \wedge \mathsf{C}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x : \mathsf{B} \vdash Q : \neg\mathsf{C}} \quad \dfrac{\dfrac{}{\Gamma, k : \mathsf{A} \vdash k : \mathsf{A}}}{\Gamma, k : \mathsf{A} \vdash k\pi_2 : \mathsf{C}}}{\Gamma, k : \mathsf{A}, x : \mathsf{B} \vdash Q(k\pi_2) : \bot}}{\Gamma, k : \mathsf{A} \vdash \lambda x.Q(k\pi_2) : \neg\mathsf{B}} \quad \dfrac{\dfrac{}{\Gamma, k : \mathsf{A} \vdash k : \mathsf{A}}}{\Gamma, k : \mathsf{A} \vdash k\pi_1 : \mathsf{B}}}{\Gamma, k : \mathsf{A} \vdash (\lambda x.Q(k\pi_2))(k\pi_1) : \bot}}{\Gamma \vdash \lambda k.(\lambda x.Q(k\pi_2))(k\pi_1) : \neg\mathsf{A}}$$

By the induction hypothesis, we have $\Gamma^{\mathsf{c}}, x : \mathsf{B}^{\mathsf{c}} \vdash_{\mathsf{cps}} Q : \neg\mathsf{C}^{\mathsf{d}}$, so we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg(\mathsf{B}^{\mathsf{c}} \wedge \mathsf{C}^{\mathsf{d}})$, where $\mathsf{A}^{\mathsf{d}} \equiv (\mathsf{B} \wedge \mathsf{C})^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{c}} \wedge \mathsf{C}^{\mathsf{d}}$.

Case $P \equiv \lambda k.Q\langle R, k\rangle$. Any derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form.

$$\dfrac{\dfrac{}{\Gamma \vdash Q : \neg(\mathsf{B} \wedge \mathsf{A})} \quad \dfrac{\dfrac{}{\Gamma \vdash R : \mathsf{B}} \quad \dfrac{}{k : \mathsf{A} \vdash k : \mathsf{A}}}{\Gamma, k : \mathsf{A} \vdash \langle R, k\rangle : \mathsf{B} \wedge \mathsf{A}}}{\dfrac{\Gamma, k : \mathsf{A} \vdash Q\langle R, k\rangle : \bot}{\Gamma \vdash \lambda k.Q\langle R, k\rangle : \neg\mathsf{A}}}$$

By the induction hypotheses, we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} Q : \neg(\mathsf{B}^{\mathsf{c}} \wedge \mathsf{A}^{\mathsf{d}})$ and $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} R : \mathsf{B}^{\mathsf{c}}$, so we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\mathsf{A}^{\mathsf{d}}$.

Case $P \equiv \lambda k.k[Xk'.Qk']$. Any derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form, where A must be $\exists X.\mathsf{B}$, and Γ must not contain a free type variable X.

$$\dfrac{\dfrac{}{k : \mathsf{A} \vdash k : \mathsf{A}} \quad \dfrac{\dfrac{}{\Gamma \vdash Q : \neg\mathsf{B}} \quad \dfrac{}{k' : \mathsf{B} \vdash k' : \mathsf{B}}}{\Gamma, k' : \mathsf{B} \vdash Qk' : \bot}}{\dfrac{\Gamma, k : \mathsf{A} \vdash k[Xk'.Qk'] : \bot}{\Gamma \vdash \lambda k.k[Xk'.Qk'] : \neg\mathsf{A}}}$$

By the induction hypothesis, we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} Q : \neg\mathsf{B}^{\mathsf{d}}$. Since Γ^{c} does not contain X freely, we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\exists X.\mathsf{B}^{\mathsf{d}}$, where $\exists X.\mathsf{B}^{\mathsf{d}} \equiv (\exists X.\mathsf{B})^{\mathsf{d}}$.

Case $P \equiv \lambda k.Q\langle \mathcal{B}, k\rangle$. Any derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following

form, where A must be $\mathsf{C}[X := \mathcal{B}]$.

$$\dfrac{\dfrac{\Gamma \vdash Q : \neg\exists X.\mathsf{C} \quad \dfrac{\overline{k : \mathsf{A} \vdash k : \mathsf{A}}}{k : \mathsf{A} \vdash \langle \mathcal{B}, k\rangle : \exists X.\mathsf{C}}}{\Gamma, k : \mathsf{A} \vdash Q\langle \mathcal{B}, k\rangle : \bot}}{\Gamma \vdash \lambda k.Q\langle \mathcal{B}, k\rangle : \neg\mathsf{A}}$$

By the induction hypothesis, $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} Q : \neg\exists X.\mathsf{C}^{\mathsf{d}}$ holds, so we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\mathsf{C}^{\mathsf{d}}[X := \mathcal{B}]$ by letting $k : \mathsf{C}^{\mathsf{d}}[X := \mathcal{B}]$, where $\mathsf{C}^{\mathsf{d}}[X := \mathcal{B}]$ is identical to $(\mathsf{C}[X := \mathcal{B}])^{\mathsf{d}}$ by Lemma 16 (2). □

5.3 Proof of Undecidability

By the main lemma, we can reduce TC and TP of DF-F to those of DF-$\lambda^{\neg\wedge\exists}$, and then conclude undecidability of TC and TP in DF-$\lambda^{\neg\wedge\exists}$.

Proposition 18 *(1) We have* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ *if and only if we have* $\neg\Gamma^{\bullet} \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^{\bullet}$.

(2) Let M *be a* DF-F*-term. We have* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ *for some* Γ *and* A *if and only if we have* $\Gamma' \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \mathsf{A}'$ *for some* Γ' *and* A'.

PROOF. (1) The only-if part is Proposition 9, so we will show the if part. If $\neg\Gamma^{\bullet} \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^{\bullet}$ holds, by Lemma 17, we have $(\neg\Gamma^{\bullet})^{\mathsf{c}} \vdash_{\mathsf{cps}} [\![M]\!] : (\neg A^{\bullet})^{\mathsf{c}}$, from which $\neg\Gamma^{\bullet} \vdash_{\mathsf{cps}} [\![M]\!] : \neg A^{\bullet}$ follows by Lemma 16 (1). By Proposition 14 (2), we have $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$.

(2) The only-if part follows from the only-if part of (1). The if part follows from Lemma 17 and Proposition 14 (2). □

PROOF. (Proof of Theorem 5) Undecidability of TC and TP in DF-$\lambda^{\neg\wedge\exists}$ follows from Theorems 3 and 4 and Proposition 18. □

5.4 Undecidability of TC and TP in DF-$\lambda_g^{\neg\wedge\exists}$

We consider another variant of $\wedge$-elimination rules called general elimination rules, which are introduced to study the relationship between natural deductions and sequent calculi [22,13]. The discussion for DF-$\lambda^{\neg\wedge\exists}$ in the previous subsections can be applied to a variant DF-$\lambda_g^{\neg\wedge\exists}$ with the general elimination rules by defining a suitable CPS translation from DF-F to DF-$\lambda_g^{\neg\wedge\exists}$.

Definition 19 (DF-$\lambda_g^{\neg\wedge\exists}$) The terms of DF-$\lambda_g^{\neg\wedge\exists}$ are defined by

$$M ::= x \mid \lambda x.M \mid \langle M, M\rangle \mid \langle \mathsf{A}, M\rangle \mid MM \mid M[xx.M] \mid M[Xx.M].$$

The typing rules of DF-$\lambda_g^{\neg\wedge\exists}$ are the same as DF-$\lambda^{\neg\wedge\exists}$ except for replacing ($\wedge$E1) and ($\wedge$E2) by the following rule.

$$\frac{\Gamma_1 \vdash M : \mathsf{A}\wedge\mathsf{B} \quad \Gamma_2, x:\mathsf{A}, y:\mathsf{B} \vdash N : \mathsf{C}}{\Gamma_1,\Gamma_2 \vdash M[xy.N] : \mathsf{C}}\ (\wedge\mathsf{E})$$

We write $\Gamma \vdash_{\lambda_g^{\neg\wedge\exists}} M : \mathsf{A}$ to denote that $\Gamma \vdash M : \mathsf{A}$ is derivable in DF-$\lambda_g^{\neg\wedge\exists}$.

Definition 20 The definition of CPS terms are the same as that for DF-$\lambda_{\mathsf{cps}}^{\neg\wedge\exists}$ except $\lambda k.k[xk'.Pk']$ is a CPS term instead of $\lambda k.(\lambda x.P(k\pi_2))(k\pi_1)$.

The CPS translation $[\![\cdot]\!]$ of DF-$\lambda_g^{\neg\wedge\exists}$ and its inverse $(\cdot)^{\#}$ are the same as those of DF-$\lambda^{\neg\wedge\exists}$ except for the cases of λ-abstractions. These are defined by $[\![\lambda x.M]\!] \equiv \lambda k.k[xk'.[\![M]\!]k']$, and $(\lambda k.k[xk'.Pk'])^{\#} \equiv \lambda x.P^{\#}$. We write $\neg\Gamma \vdash_{\mathsf{g\text{-}cps}} P : \neg\mathcal{A}$ to denote that the judgment is derivable in the image of the CPS translation, which is defined similarly to DF-$\lambda^{\neg\wedge\exists}$.

Lemma 21 *If P is a CPS term, $\Gamma \vdash_{\lambda_g^{\neg\wedge\exists}} P : \mathsf{A}$ implies $\Gamma^{\mathsf{c}} \vdash_{\mathsf{g\text{-}cps}} P : \mathsf{A}^{\mathsf{c}}$.*

Proposition 22 *(1) We have $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ if and only if we have $\neg\Gamma^{\bullet} \vdash_{\lambda_g^{\neg\wedge\exists}} [\![M]\!] : \neg A^{\bullet}$.*

(2) Let M be a DF-F-term. We have $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ for some Γ and A if and only if we have $\Gamma' \vdash_{\lambda_g^{\neg\wedge\exists}} [\![M]\!] : \mathsf{A}'$ for some Γ' and A'.

Theorem 23 *Type checking and typability are undecidable in* DF-$\lambda_g^{\neg\wedge\exists}$.

PROOF. It follows from Theorems 3 and 4 and Proposition 22. □

6 DF-$\lambda^{\neg\wedge\exists}$ as Target of CPS Translations

In this section, we discuss that DF-$\lambda^{\neg\wedge\exists}$ is suitable as a target calculus of CPS translations by showing it works well as a CPS target for the call-by-value computational lambda calculus, the call-by-value lambda-mu calculus, and delimited continuations. At first sight, $\lambda^{\neg\wedge\exists}$ may look weak as a computational system, but it suffices as a target calculus of several CPS translations [5,9]. Moreover, the domain-free style calculus with existence works also as a CPS target of the domain-free call-by-value lambda-mu calculus $\lambda_V\mu$ [3].

First, we define the reduction relation in DF-$\lambda^{\neg\wedge\exists}$. We omit η-rules, but the results in this section can be extended straightforwardly to η-rules.

Definition 24 The reduction rules of DF-$\lambda^{\neg\wedge\exists}$ are the following:

$(\beta_{\to})$ $(\lambda x.M)N \to M[x := N]$,
$(\beta_{\wedge})$ $\langle M_1, M_2\rangle \pi_i \to M_i \qquad (i = 1 \text{ or } 2)$,
$(\beta_{\exists})$ $\langle \mathsf{A}, M\rangle [Xx.N] \to N[X := \mathsf{A}, x := M]$.

The relation $\to_{\lambda^{\neg\wedge\exists}}$ is the compatible closure of the above rules, and the relation $\to^*_{\lambda^{\neg\wedge\exists}}$ is its reflexive transitive closure.

6.1 Computational Lambda Calculus

In [18], Sabry and Wadler gave a call-by-value CPS translation from the computational lambda calculus λ_c [12] to a CPS calculus λ_{cps}, which is a subsystem of the ordinary lambda calculus. Furthermore, they gave an inverse translation from λ_{cps} to λ_c, and showed that those translations form a reflection of λ_{cps} in λ_c.

The system DF-$\lambda^{\neg\wedge\exists}$ can be a target of a CPS translation for λ_c with polymorphic types. In this subsection, we define DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ as a subsystem of DF-$\lambda^{\neg\wedge\exists}$, and show that we have a reflection of DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ in λ_c with polymorphic types.

Definition 25 (DF-$\lambda^{\forall}_c$) The system DF-$\lambda^{\forall}_c$ is an extension of DF-F by adding `let`-expressions with the typing rule for them as follows.

$$\frac{\Gamma_1 \vdash M : A \quad \Gamma_2, x : A \vdash N : B}{\Gamma_1, \Gamma_2 \vdash \texttt{let } x = M \texttt{ in } N : B}\ (\mathsf{let})$$

The values are defined by $V ::= x \mid \lambda x.M \mid \lambda X.M$. We use P, Q,... to denote terms that are not values. The call-by-value reduction is defined by the following rules:

$(\beta.v)$ $(\lambda x.M)V \to M[x := V]$,
$(\beta.t)$ $(\lambda X.M)A \to M[X := A]$,
$(\beta.\texttt{let})$ $\texttt{let } x = V \texttt{ in } M \to M[x := V]$,
(ass) $\texttt{let } y = (\texttt{let } x = L \texttt{ in } M) \texttt{ in } N \to \texttt{let } x = L \texttt{ in } (\texttt{let } y = M \texttt{ in } N)$,
$(\texttt{let}.1)$ $PM \to \texttt{let } x = P \texttt{ in } xM$,
$(\texttt{let}.2)$ $VP \to \texttt{let } x = P \texttt{ in } Vx$,
$(\texttt{let}.3)$ $PA \to \texttt{let } x = P \texttt{ in } xA$.

In (ass), N must not contain x freely.

Definition 26 (DF-$\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$) (1) Let k be a fixed term variable. The value types (denoted by $\mathcal{A}$, $\mathcal{B}$,...) are defined by

$$\mathcal{A} ::= X \mid \neg(\mathcal{A} \wedge \neg\mathcal{A}) \mid \neg\exists X.\neg\mathcal{A}.$$

The terms (denoted by M, N,...), the values (denoted by V, W,...), and the continuations (denoted by K,...) of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ are mutually defined as follows. If V is a value, K is a continuation, and $\mathcal{A}$ is a value type, then KV, $V\langle V, K\rangle$, and $V\langle\mathcal{A}, K\rangle$ are terms. If x is a variable, X is a type variable, c is a fresh variable, and M is a term, then x, $\lambda c.(\lambda x.(\lambda k.M)(c\pi_2))(c\pi_1)$, and $\lambda c.c[Xk.M]$ are values. If x is a variable and M is a term, then k and $\lambda x.M$ are continuations.

(2) The reduction rules of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ are the following.

$(\beta.v)$ $(\lambda\langle x, k\rangle.M)\langle V, K\rangle \to M[x := V, k := K]$,
$(\beta.t)$ $(\lambda\langle X, k\rangle.M)\langle\mathcal{A}, K\rangle \to M[x := \mathcal{A}, k := K]$,
$(\beta.\mathtt{let})$ $(\lambda x.M)V \to M[x := V]$.

Note that $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ is a subsystem of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$, and closed under the reduction. The first-order fragment of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ is isomorphic to λ_{cps} in [18].

Definition 27 (1) The negative translation $(\cdot)^{\triangle}$ from $\to\forall$-types to value types and its inverse $(\cdot)^{\triangledown}$ are defined by

$$\begin{aligned} X^{\triangle} &\equiv X, & X^{\triangledown} &\equiv X, \\ (A{\to}B)^{\triangle} &\equiv \neg(A^{\triangle} \wedge \neg B^{\triangle}), & (\neg(\mathcal{A} \wedge \neg\mathcal{B}))^{\triangledown} &\equiv \mathcal{A}^{\triangledown}{\to}\mathcal{B}^{\triangledown}, \\ (\forall X.A)^{\triangle} &\equiv \neg\exists X.\neg A^{\triangle}, & (\neg\exists X.\neg\mathcal{A})^{\triangledown} &\equiv \forall X.\mathcal{A}^{\triangledown}. \end{aligned}$$

(2) For a $\mathsf{DF}\text{-}\lambda^{\forall}_{c}$-term M, the CPS translation $[\![M]\!]$ is mutually defined with the auxiliary translations $M \bullet K$ and $\Phi(V)$ as follows:

$$\begin{aligned} [\![M]\!] &\equiv M \bullet k, \\ V \bullet K &\equiv K\Phi(V), \\ VW \bullet K &\equiv \Phi(V)\langle\Phi(W), K\rangle, \\ PW \bullet K &\equiv P \bullet \lambda m.m\langle\Phi(W), K\rangle, \\ VQ \bullet K &\equiv Q \bullet \lambda n.\Phi(V)\langle n, K\rangle, \\ PQ \bullet K &\equiv P \bullet \lambda m.(Q : \lambda n.m\langle n, K\rangle), \\ VA \bullet K &\equiv \Phi(V)\langle A^{\triangle}, K\rangle, \\ PA \bullet K &\equiv P \bullet \lambda m.m\langle A^{\triangle}, K\rangle, \\ (\mathtt{let}\ x = M\ \mathtt{in}\ N) \bullet K &\equiv M \bullet \lambda x.(N : K), \\ \Phi(x) &\equiv x, \\ \Phi(\lambda x.M) &\equiv \lambda\langle x, k\rangle.[\![M]\!], \\ \Phi(\lambda X.M) &\equiv \lambda\langle X, k\rangle.[\![M]\!], \end{aligned}$$

where m and n are fresh variables.

(3) The inverse translation $(\cdot)^{\#}$ from $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ to $\mathsf{DF}\text{-}\lambda^{\forall}_{c}$ is defined by

$$\begin{aligned}
(KV)^{\#} &\equiv K^{\flat}[V^{\natural}], & x^{\natural} &\equiv x,\\
(V\langle W, K\rangle)^{\#} &\equiv K^{\flat}[V^{\natural}W^{\natural}], & (\lambda\langle x, k\rangle.M)^{\natural} &\equiv \lambda x.M^{\#},\\
(V\langle \mathcal{A}, K\rangle)^{\#} &\equiv K^{\flat}[\mathcal{A}^{\triangledown}W^{\natural}], & (\lambda\langle X, k\rangle.M)^{\natural} &\equiv \lambda X.M^{\#},\\
k^{\flat} &\equiv [], & &\\
(\lambda x.M)^{\flat} &\equiv \mathtt{let}\ x = [] \ \mathtt{in}\ M^{\#}. & &
\end{aligned}$$

Note that the translation $M \bullet K$ is based on the same idea with Plotkin's colon translation [16], but we use another notation $M \bullet K$ to avoid confusion with the typing relation $M : A$.

Proposition 28 *(1) If* $\Gamma \vdash_{\lambda^{\forall}_{c}} M : A$ *holds, then we have* $\Gamma^{\triangle}, k : \neg A^{\triangle} \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \bot$.

(2) The translations $[\![\cdot]\!]$ *and* $(\cdot)^{\#}$ *form a reflection of* $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$ *in* $\mathsf{DF}\text{-}\lambda^{\forall}_{c}$*, that is, the following hold.*

(a) The translations $[\![\cdot]\!]$ *and* $(\cdot)^{\#}$ *preserve reduction relation* $\to^{*}$.

(b) We have $[\![M^{\#}]\!] \equiv M$ *for any term* M *of* $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}_{\mathsf{cps/v}}$.

(c) We have $M \to^{*} [\![M]\!]^{\#}$ *for any term* M *of* $\mathsf{DF}\text{-}\lambda^{\forall}_{c}$.

PROOF. The claims can be proved straightforwardly. □

6.2 Call-by-Value Lambda-Mu Calculus

The lambda-mu calculus was introduced by Parigot in [15] as an extension of lambda calculus, and it corresponds to the classical natural deduction for second-order propositional logic by the Curry-Howard isomorphism. In [3], Fujita pointed out that the Curry-style call-by-value lambda-mu calculus does not enjoy the subject reduction property, so he introduced a domain-free call-by-value lambda-mu calculus $\lambda_{V}\mu$ to avoid the problem. In this subsection, we show that $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$ works as a target calculus of a CPS translation for $\lambda_{V}\mu$.

Definition 29 ($\lambda_{V}\mu$) (1) The system $\lambda_{V}\mu$ contains μ-variables (denoted by α, β,...), which are another kind of variables. The types of $\lambda_{V}\mu$ are the $\to\forall$-types. The terms (denoted by M, N,...), and the values (denoted by V, W,...) of $\lambda_{V}\mu$ are defined by

$$\begin{aligned}
M &::= V \mid (MM) \mid (MA) \mid (\mu\alpha.[\alpha]M),\\
V &::= x \mid (\lambda x.M) \mid (\lambda X.M).
\end{aligned}$$

Outermost parentheses are often omitted.

(2) The typing rules of $\lambda_V\mu$ are the following.

$$\frac{}{\Gamma, x : A \vdash x : A; \Delta}\ (\mathsf{Ax}) \qquad \frac{\Gamma \vdash M : B; \Delta}{\Gamma \vdash \mu\alpha.[\beta]M : A; (\Delta, \beta : B) - \{\alpha : A\}}\ (\mu)$$

$$\frac{\Gamma, x : A \vdash M : B; \Delta}{\Gamma \vdash \lambda x.M : A{\to}B; \Delta}\ (\to\mathsf{I}) \qquad \frac{\Gamma_1 \vdash M : A{\to}B; \Delta_1 \quad \Gamma_2 \vdash N : A; \Delta_2}{\Gamma_1, \Gamma_2 \vdash MN : B; \Delta_1, \Delta_2}\ (\to\mathsf{E})$$

$$\frac{\Gamma \vdash M : A; \Delta}{\Gamma \vdash \lambda X.M : \forall X.A; \Delta}\ (\forall\mathsf{I}) \qquad \frac{\Gamma \vdash M : \forall X.A; \Delta}{\Gamma \vdash MB : A[X := B]; \Delta}\ (\forall\mathsf{E})$$

We use Γ to denote a context similarly to $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$, and Δ to denote a μ-context, which is a finite set of type assignments for μ-variables in the form of $(\alpha : A)$. In the rule $(\forall\mathsf{I})$, the lower sequent must not contain X freely.

(3) The singular contexts are defined by $\mathcal{C} ::= []M \mid V[] \mid []A$. The term $\mathcal{C}[M]$ is obtained from $\mathcal{C}$ by replacing $[]$ by M. The structural substitution $M[\alpha{\Leftarrow}\mathcal{C}]$ is obtained from M by replacing each subterm $[\alpha]L$ by $[\alpha]\mathcal{C}[L[\alpha{\Leftarrow}\mathcal{C}]]$. The reduction rules of $\lambda_V\mu$ are the following:
   (β_{tm}) $(\lambda x.M)N \to M[x := N]$,
   (β_{tp}) $(\lambda X.M)A \to M[X := A]$,
   (μ) $\mathcal{C}[\mu\alpha.M] \to \mu\alpha.M[\alpha{\Leftarrow}\mathcal{C}]$.

Definition 30 We suppose that $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$ contains a term variable x_α for each μ-variable α. The negative translation $(\cdot)^\triangle$ and the CPS translation $[\![\cdot]\!]$ from $\lambda_V\mu$ to $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$ are the same as Definition 27, except for replacing the definition for `let`-expression by $(\mu\alpha.[\beta]M) \bullet K \equiv (M \bullet x_\beta)[x_\alpha := K]$.

Proposition 31 *(1) If* $\Gamma \vdash_{\lambda_V\mu} M : A; \Delta$ *holds, then we have* $\Gamma^\triangle, \neg\Delta^\triangle, k : \neg A^\triangle \vdash_{\lambda^{\neg\wedge\exists}} [\![M]\!] : \bot$.
(2) If $M \to^*_{\lambda_V\mu} N$ *holds, then we have* $[\![M]\!] \to^*_{\lambda^{\neg\wedge\exists}} [\![N]\!]$.

6.3 Delimited Continuations

The $\neg \wedge \exists$-fragments are also important as a target of a CPS translation for delimited continuations such as `shift` and `reset` [2]. For calculi with delimited continuations, we consider multi-staged CPS translations, and we need call-by-value calculi as intermediate CPS calculi. However, in order to have a sound CPS translation to an $\to\exists$-fragment, the calculus has to have not only the call-by-value η-reduction, but also the full η-reduction. On the other hand, as it was shown in [10], we can define a sound CPS translation from a calculus with `shift` and `reset` to a call-by-value $\neg \wedge \exists$-fragment without full η-reduction.

7 Undecidability in Implicational Fragment

Our method by means of CPS translations can be used for the domain-free typed lambda calculus DF-$\lambda^{\to\exists}$ with implication and existence. In this section, we define DF-$\lambda^{\to\exists}$ and prove undecidability of TC and TP in DF-$\lambda^{\to\exists}$, which is another main result of this paper. In order to prove that, we give a CPS translation from DF-F to DF-$\lambda^{\to\exists}$, by which TC and TP of DF-F are reduced to those of DF-$\lambda^{\to\exists}$.

Definition 32 (DF-$\lambda^{\to\exists}$) The language of DF-$\lambda^{\to\exists}$ contains type variables (denoted by X, Y,...), a type constant $\bot$, and term variables (denoted by x, y,...). The types (called $\to\exists$-types) and the terms of DF-$\lambda^{\to\exists}$ are defined by

$$\begin{aligned} \mathsf{A} &::= X \mid \bot \mid (\mathsf{A}\to\mathsf{A}) \mid (\exists X.\mathsf{A}), \\ M &::= x \mid (\lambda x.M) \mid \langle \mathsf{A}, M\rangle \mid (MM) \mid (M[Xx.M]). \end{aligned}$$

Outermost parentheses are often omitted. In DF-$\lambda^{\to\exists}$, the type $\mathsf{A}\to\bot$ is denoted by $\neg\mathsf{A}$. The typing rules of DF-$\lambda^{\to\exists}$ are (Ax), ($\exists$I), ($\exists$E) of DF-$\lambda^{\neg\wedge\exists}$ and

$$\frac{\Gamma, x:\mathsf{A}\vdash M:\mathsf{B}}{\Gamma\vdash\lambda x.M:\mathsf{A}\to\mathsf{B}}\ (\to\mathsf{I}), \qquad \frac{\Gamma_1\vdash M:\mathsf{A}\to\mathsf{B}\quad \Gamma_2\vdash N:\mathsf{A}}{\Gamma_1,\Gamma_2\vdash MN:\mathsf{B}}\ (\to\mathsf{E}).$$

Note that $\bot$ can be considered as a distinguished type variable which is not supposed to be bound by the existential quantifiers, that is, the underlying logic of DF-$\lambda^{\to\exists}$ is the minimal logic with logical connectives $\to$ and $\exists$.

Definition 33 (CPS translation) (1) The continuation types are defined by

$$\mathcal{A} ::= X \mid \neg(\neg\mathcal{A}\to\neg\mathcal{A}) \mid \exists X.\mathcal{A}.$$

The negative translation $(\cdot)^\bullet$ from $\to\forall$-types to $\to\exists$-types and its inverse $(\cdot)^\circ$ from continuation types to $\to\forall$-types are defined by

$$\begin{aligned} X^\bullet &\equiv X, & X^\circ &\equiv X, \\ (A\to B)^\bullet &\equiv \neg(\neg A^\bullet\to\neg B^\bullet), & (\neg(\neg\mathcal{A}\to\neg\mathcal{B}))^\circ &\equiv \mathcal{A}^\circ\to\mathcal{B}^\circ, \\ (\forall X.A)^\bullet &\equiv \exists X.A^\bullet, & (\exists X.\mathcal{A})^\circ &\equiv \forall X.\mathcal{A}^\circ. \end{aligned}$$

The CPS types are defined as types in the form of $\neg\mathcal{A}$ for a continuation type $\mathcal{A}$.

(2) The CPS terms are inductively defined as follows: if x is a variable, P and Q are CPS terms, $\mathcal{A}$ is a continuation type, X is a type variable, and k, k', and m are fresh variables, then $\lambda k.xk$, $\lambda k.k(\lambda x.P)$, $\lambda k.k[Xk'.Pk']$, $\lambda k.P(\lambda m.mPk)$, and $\lambda k.P\langle\mathcal{A}, k\rangle$ are CPS terms.

The CPS translation $[\![\cdot]\!]$ from terms of DF-F to terms of DF-$\lambda^{\to\exists}$ and its inverse $(\cdot)^{\#}$ from CPS terms to terms of DF-F are defined by

$$\begin{aligned}
[\![x]\!] &\equiv \lambda k.xk, & (\lambda k.xk)^{\#} &\equiv x,\\
[\![\lambda x.M]\!] &\equiv \lambda k.k(\lambda x.[\![M]\!]), & (\lambda k.k(\lambda x.P))^{\#} &\equiv \lambda x.P^{\#},\\
[\![\lambda X.M]\!] &\equiv \lambda k.k[Xk'.[\![M]\!]k'], & (\lambda k.k[Xk'.Pk'])^{\#} &\equiv \lambda X.P^{\#},\\
[\![MN]\!] &\equiv \lambda k.[\![M]\!](\lambda m.m[\![N]\!]k), & (\lambda k.P(\lambda m.mQk))^{\#} &\equiv P^{\#}Q^{\#},\\
[\![MA]\!] &\equiv \lambda k.[\![M]\!]\langle A^{\bullet}, k\rangle, & (\lambda k.P\langle \mathcal{A}, k\rangle)^{\#} &\equiv P^{\#}\mathcal{A}^{\circ}.
\end{aligned}$$

(3) The system DF-$\lambda^{\to\exists}_{\mathsf{cps}}$ is defined as a subsystem of DF-$\lambda^{\to\exists}$ by restricting terms and types to CPS terms and CPS types, respectively. The typing rules of DF-$\lambda^{\to\exists}_{\mathsf{cps}}$ are the following.

$$\frac{}{\neg\Gamma, x : \neg\mathcal{A} \vdash \lambda k.xk : \neg\mathcal{A}}$$

$$\frac{\neg\Gamma, x : \neg\mathcal{A} \vdash P : \neg\mathcal{B}}{\neg\Gamma \vdash \lambda k.k(\lambda x.P) : \neg\neg(\neg\mathcal{A} \to \neg\mathcal{B})}$$

$$\frac{\neg\Gamma_1 \vdash P : \neg\neg(\neg\mathcal{A} \to \neg\mathcal{B}) \quad \neg\Gamma_2 \vdash Q : \neg\mathcal{A}}{\neg\Gamma_1, \neg\Gamma_2 \vdash \lambda k.P(\lambda m.mQk) : \neg\mathcal{B}}$$

$$\frac{\neg\Gamma \vdash P : \neg\exists X.\mathcal{B}}{\neg\Gamma \vdash \lambda k.P\langle \mathcal{A}, k\rangle : \neg\mathcal{B}[X := \mathcal{A}]}$$

$$\frac{\neg\Gamma \vdash P : \neg\mathcal{B}}{\neg\Gamma \vdash \lambda k.k[Xk'.Pk'] : \neg\exists X.\mathcal{B}}\ (X \notin FV(\Gamma))$$

We write $\neg\Gamma \vdash_{\to\exists\mathsf{cps}} P : \neg\mathcal{A}$ to denote that the judgment is derivable in DF-$\lambda^{\to\exists}_{\mathsf{cps}}$.

Lemmas 34 and 35 show that the translations $(\cdot)^{\circ}$ and $(\cdot)^{\#}$ are actually the inverses of $(\cdot)^{\bullet}$ and $[\![\cdot]\!]$ respectively.

Lemma 34 *(1) The set of the continuation types coincides with the image of the negative translation.*
(2) The set of the CPS terms coincides with the image of the CPS translation.

PROOF. They are proved in a way similar to Lemma 11. □

Lemma 35 *(1) For any $\to\forall$-type A, we have $A^{\bullet\circ} \equiv A$.*
(2) For any DF-F-term M, we have $[\![M]\!]^{\#} \equiv M$.

Proposition 36 *(1) If $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ holds, then we have $\neg\Gamma^{\bullet} \vdash_{\lambda^{\to\exists}} [\![M]\!] : \neg A^{\bullet}$.*

(2) If $\neg\Gamma \vdash_{\to\exists\mathsf{cps}} P : \neg\mathcal{A}$ *holds, then we have* $\Gamma^\circ \vdash_{\mathsf{DF}\text{-}F} P^\# : \mathcal{A}^\circ$.
(3) If $\neg\Gamma^\bullet \vdash_{\to\exists\mathsf{cps}} [\![M]\!] : \neg A^\bullet$ *holds, then we have* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$.

Definition 37 (Contraction Translation) Let $\mathcal{S}$ be a fixed closed continuation type, such as $\exists X.X$. The contraction translation $(\cdot)^{\mathsf{c}}$ from $\to\exists$-types to CPS types is mutually defined with the auxiliary translation $(\cdot)^{\mathsf{d}}$ from $\to\exists$-types to continuation types by

$$
\begin{aligned}
(\mathsf{A}\to\mathsf{B})^{\mathsf{c}} &\equiv \neg\mathsf{A}^{\mathsf{d}}, \\
\mathsf{A}^{\mathsf{c}} &\equiv \neg\mathcal{S} && (\mathsf{A} \text{ is not an implication}), \\
X^{\mathsf{d}} &\equiv X, \\
\bot^{\mathsf{d}} &\equiv \mathcal{S}, \\
((\mathsf{A}\to\mathsf{B}\to\mathsf{C})\to\mathsf{D})^{\mathsf{d}} &\equiv \neg(\mathsf{A}^{\mathsf{c}}\to\neg\mathsf{B}^{\mathsf{d}}), \\
((\mathsf{A}\to\mathsf{B})\to\mathsf{D})^{\mathsf{d}} &\equiv \neg(\mathsf{A}^{\mathsf{c}}\to\neg\mathcal{S}) && (\mathsf{B} \text{ is neither an implication nor } \bot), \\
(\mathsf{A}\to\mathsf{D})^{\mathsf{d}} &\equiv \mathcal{S} && (\text{otherwise}), \\
(\exists X.\mathsf{A})^{\mathsf{d}} &\equiv \exists X.\mathsf{A}^{\mathsf{d}}.
\end{aligned}
$$

Lemma 38 *(1) For any continuation type* $\mathcal{A}$*, we have* $(\neg\mathcal{A})^{\mathsf{c}} \equiv \neg\mathcal{A}$ *and* $\mathcal{A}^{\mathsf{d}} \equiv \mathcal{A}$.
(2) For any continuation type $\mathcal{A}$ *and any* $\to\exists$*-type* B*, we have* $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[X := \mathcal{A}]$ *and* $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[X := \mathcal{A}]$.

PROOF. (1) The claim is straightforwardly proved by induction on $\mathcal{A}$.

(2) First, we show (a) $\mathcal{A}^{\mathsf{c}} \equiv \neg\mathcal{S}$, and (b) $(\mathcal{A} \to \mathsf{D})^{\mathsf{d}} \equiv \mathcal{S}$ hold for any continuation type $\mathcal{A}$.

(a) If $\mathcal{A}$ is either a type variable or an existence, it is trivial, so we will prove $(\neg(\neg\mathcal{B}\to\neg\mathcal{C}))^{\mathsf{c}} \equiv \neg\mathcal{S}$. We have $(\neg(\neg\mathcal{B}\to\neg\mathcal{C}))^{\mathsf{c}} \equiv \neg(\neg\mathcal{B}\to\neg\mathcal{C})^{\mathsf{d}} \equiv \neg((\mathcal{B}\to\bot)\to\neg\mathcal{C})^{\mathsf{d}}$, so it is identical to $\neg\mathcal{S}$ by the definition.

(b) Similarly, we will prove only the case $\mathcal{A} \equiv \neg(\neg\mathcal{B}\to\neg\mathcal{C})$. We have $(\mathcal{A}\to\mathsf{D})^{\mathsf{d}} \equiv (((\neg\mathcal{B}\to\neg\mathcal{C})\to\bot)\to\mathsf{D})^{\mathsf{d}}$, so it is identical to $\mathcal{S}$ by the definition.

Then we show $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[X := \mathcal{A}]$ and $(\mathsf{B}[X := \mathcal{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[X := \mathcal{A}]$ simultaneously by induction on B. In the following, we write $\mathsf{B}[\mathcal{A}]$ for $\mathsf{B}[X := \mathcal{A}]$.

We will show $(\mathsf{B}[\mathcal{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[\mathcal{A}]$ by induction on B. Cases are considered according to B.

Case X. We have $(X[\mathcal{A}])^{\mathsf{c}} \equiv \mathcal{A}^{\mathsf{c}}$, which is identical to $\neg\mathcal{S}$ by (a). On the other hand, $X^{\mathsf{c}}[\mathcal{A}] \equiv \neg\mathcal{S}[\mathcal{A}] \equiv \neg\mathcal{S}$ holds.

Case $Y(\not\equiv X)$, $\bot$ or $\exists Y.\mathsf{B}$. Both sides are identical to $\neg\mathcal{S}$.

Case $\mathsf{B}{\to}\mathsf{C}$. We have $((\mathsf{B}{\to}\mathsf{C})[\mathcal{A}])^{\mathsf{c}} \equiv (\mathsf{B}[\mathcal{A}]{\to}\mathsf{C}[\mathcal{A}])^{\mathsf{c}} \equiv \neg(\mathsf{B}[\mathcal{A}])^{\mathsf{d}}$, which is identical to $\neg\mathsf{B}^{\mathsf{d}}[\mathcal{A}]$ by the induction hypothesis. On the other hand, $(\mathsf{B}{\to}\mathsf{C})^{\mathsf{c}}[\mathcal{A}] \equiv \neg\mathsf{B}^{\mathsf{d}}[\mathcal{A}]$ holds.

Next, we will show $(\mathsf{B}[\mathcal{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[\mathcal{A}]$ by induction on B. Cases are considered according to B.

Case X. Both sides are identical to $\mathcal{A}$, since $\mathcal{A}^{\mathsf{d}} \equiv \mathcal{A}$ by (1).

Case $Y(\not\equiv X)$. Both sides are identical to Y.

Case $\bot$. Both sides are identical to $\mathcal{S}$.

Case $\exists Y.\mathsf{B}$. We have $((\exists Y.\mathsf{B})[\mathcal{A}])^{\mathsf{d}} \equiv (\exists Y.\mathsf{B}[\mathcal{A}])^{\mathsf{d}} \equiv \exists Y.(\mathsf{B}[\mathcal{A}])^{\mathsf{d}}$. By the induction hypothesis, it is identical to $\exists Y.\mathsf{B}^{\mathsf{d}}[\mathcal{A}] \equiv (\exists Y.\mathsf{B}^{\mathsf{d}})[\mathcal{A}] \equiv (\exists Y.\mathsf{B})^{\mathsf{d}}[\mathcal{A}]$.

Case $X{\to}\mathsf{D}$. We have $((X{\to}\mathsf{D})[\mathcal{A}])^{\mathsf{d}} \equiv (\mathcal{A}{\to}\mathsf{D}[\mathcal{A}])^{\mathsf{d}}$, which is identical to $\mathcal{S}$ by (b). On the other hand, $(X{\to}\mathsf{D})^{\mathsf{d}}[\mathcal{A}] \equiv \mathcal{S}[\mathcal{A}] \equiv \mathcal{S}$ holds.

Case $Y{\to}\mathsf{D}$ $(Y \not\equiv X)$, $\bot{\to}\mathsf{D}$, or $(\exists Y.\mathsf{A}){\to}\mathsf{D}$. Both sides are identical to $\mathcal{S}$.

Case $(\mathsf{A}{\to}X){\to}\mathsf{D}$. We have $((\mathsf{A}{\to}X){\to}\mathsf{D})^{\mathsf{d}}[\mathcal{A}] \equiv (\neg(\mathsf{A}^{\mathsf{c}}{\to}\neg\mathcal{S}))[\mathcal{A}] \equiv \neg(\mathsf{A}^{\mathsf{c}}[\mathcal{A}]{\to}\neg\mathcal{S})$, which is identical to $\neg((\mathsf{A}[\mathcal{A}])^{\mathsf{c}}{\to}\neg\mathcal{S})$ by the induction hypothesis. On the other hand, we have $(((\mathsf{A}{\to}X){\to}\mathsf{D})[\mathcal{A}])^{\mathsf{d}} \equiv ((\mathsf{A}[\mathcal{A}]{\to}\mathcal{A}){\to}\mathsf{D}[\mathcal{A}])^{\mathsf{d}}$. If $\mathcal{A}$ is either a type variable or an existential type, it is identical to $\neg((\mathsf{A}[\mathcal{A}])^{\mathsf{c}}{\to}\neg\mathcal{S})$. Otherwise, $\mathcal{A}$ is in the form of $\neg(\neg\mathcal{B}{\to}\neg\mathcal{C})$, so it is identical to $\neg((\mathsf{A}[\mathcal{A}])^{\mathsf{c}}{\to}\neg(\neg\mathcal{B}{\to}\neg\mathcal{C})^{\mathsf{d}})$, which is identical to $\neg((\mathsf{A}[\mathcal{A}])^{\mathsf{c}}{\to}\neg\mathcal{S})$, since $(\neg\mathcal{B}{\to}\neg\mathcal{C})^{\mathsf{d}} \equiv \mathcal{S}$ by the definition.

Case $(\mathsf{A}{\to}Y){\to}\mathsf{D}$ $(Y \not\equiv X)$ or $(\mathsf{A}{\to}\exists Y.\mathsf{B}){\to}\mathsf{D}$. By the induction hypothesis, both sides are identical to $\neg(\mathsf{A}^{\mathsf{c}}[\mathcal{A}]{\to}\neg\mathcal{S})$.

Case $(\mathsf{A}{\to}\bot){\to}\mathsf{D}$. Both sides are identical to $\mathcal{S}$.

Case $(\mathsf{A}{\to}(\mathsf{B}{\to}\mathsf{C})){\to}\mathsf{D}$. By the induction hypotheses, the both sides are identical to $\neg(\mathsf{A}^{\mathsf{c}}[\mathcal{A}]{\to}\neg\mathsf{B}^{\mathsf{d}}[\mathcal{A}])$. □

Similarly to $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$, we have the following key lemma.

Lemma 39 *If P is a CPS term, $\Gamma \vdash_{\lambda^{\to\exists}} P : \mathsf{A}$ implies $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} P : \mathsf{A}^{\mathsf{c}}$.*

PROOF. By induction on P. Note that any typable CPS term must have an implicational type, so we will show that $\Gamma \vdash_{\lambda^{\to\exists}} P : \mathsf{A}_1{\to}\mathsf{A}_2$ implies $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}}$

$P : \neg\mathsf{A}_1^{\mathsf{d}}$. We will show only non-trivial cases, and the other cases are proved similarly to $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$.

Case $P \equiv \lambda k.k(\lambda x.Q)$. Any derivation of $\Gamma \vdash_{\lambda^{\to\exists}} P : \mathsf{A}_1{\to}\mathsf{A}_2$ has the following form, where A_1 must be $(\mathsf{B}_1{\to}\mathsf{B}_2){\to}\mathsf{A}_2$.

$$\dfrac{\dfrac{}{k : \mathsf{A}_1 \vdash k : \mathsf{A}_1} \quad \dfrac{\Gamma, x : \mathsf{B}_1 \vdash Q : \mathsf{B}_2}{\Gamma \vdash \lambda x.Q : \mathsf{B}_1{\to}\mathsf{B}_2}}{\dfrac{\Gamma, k : \mathsf{A}_1 \vdash k(\lambda x.Q) : \mathsf{A}_2}{\Gamma \vdash \lambda k.k(\lambda x.Q) : \mathsf{A}_1{\to}\mathsf{A}_2}}$$

Note that B_2 is an implication since it is a type of a CPS term Q, so we have $\mathsf{A}_1^{\mathsf{d}} \equiv ((\mathsf{B}_1{\to}\mathsf{B}_2){\to}\mathsf{A}_2)^{\mathsf{d}} \equiv \neg(\mathsf{B}_1^{\mathsf{c}}{\to}\mathsf{B}_2^{\mathsf{c}})$ by the definition. By the induction hypothesis, we have $\Gamma^{\mathsf{c}}, x : \mathsf{B}_1^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} Q : \mathsf{B}_2^{\mathsf{c}}$, so $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} P : \neg\neg(\mathsf{B}_1^{\mathsf{c}}{\to}\mathsf{B}_2^{\mathsf{c}})$ holds.

Case $P \equiv \lambda k.Q(\lambda m.mRk)$. Any derivation of $\Gamma \vdash_{\lambda^{\to\exists}} P : \mathsf{A}_1{\to}\mathsf{A}_2$ has the following form, where B denotes $(\mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D}){\to}\mathsf{D}$.

$$\dfrac{\Gamma \vdash Q : \mathsf{B}{\to}\mathsf{A}_2 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{m : \mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D} \vdash m : \mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D}} \quad \Gamma \vdash R : C}{\Gamma, m : \mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D} \vdash mR : \mathsf{A}_1{\to}\mathsf{D}} \quad \dfrac{}{k : \mathsf{A}_1 \vdash k : \mathsf{A}_1}}{\Gamma, k : \mathsf{A}_1, m : \mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D} \vdash mRk : \mathsf{D}}}{\Gamma, k : \mathsf{A}_1 \vdash \lambda m.mRk : (\mathsf{C}{\to}\mathsf{A}_1{\to}\mathsf{D}){\to}\mathsf{D}}}{\dfrac{\Gamma, k : \mathsf{A}_1 \vdash Q(\lambda m.mRk) : \mathsf{A}_2}{\Gamma \vdash \lambda k.Q(\lambda m.mRk) : \mathsf{A}_1{\to}\mathsf{A}_2}}$$

By the induction hypotheses, we have $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} Q : \neg\mathsf{B}^{\mathsf{d}}$ and $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} R : \mathsf{C}^{\mathsf{c}}$, where B^{d} is identical to $\neg(\mathsf{C}^{\mathsf{c}}{\to}\neg\mathsf{A}_1^{\mathsf{d}})$. So we have $\Gamma^{\mathsf{c}} \vdash_{\to\exists\mathsf{cps}} P : \neg\mathsf{A}_1^{\mathsf{d}}$. □

Proposition 40 *(1) We have* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ *if and only if we have* $\neg\Gamma^{\bullet} \vdash_{\lambda^{\to\exists}} [\![M]\!] : \neg A^{\bullet}$.

(2) Let M *be a* DF-F*-term. We have* $\Gamma \vdash_{\mathsf{DF}\text{-}F} M : A$ *for some* Γ *and* A *if and only if we have* $\Gamma' \vdash_{\lambda^{\to\exists}} [\![M]\!] : \mathsf{A}'$ *for some* Γ' *and* A'.

Theorem 41 *Type checking and typability are undecidable in* $\mathsf{DF}\text{-}\lambda^{\to\exists}$.

PROOF. It follows from Theorems 3 and 4, and Proposition 40. □

Acknowledgments The authors are grateful to Professor Ken-etsu Fujita for his helpful comments, and Professor Masahito Hasegawa for a copy of his draft [10]. They would also like to thank anonymous reviewers for their comments to this paper and its preceding version [14]. The first author was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B) 18700008.

References

[1] G. Barthe and M.H. Sørensen, Domain-free pure type systems. *J. Functional Programming* 10:412–452, 2000.

[2] O. Danvy and A. Filinski, Representing control: a study of the CPS translation. *Mathematical Structures in Computer Science* 2(4):361–391, 1992.

[3] K. Fujita, Explicitly typed $\lambda\mu$-calculus for polymorphism and call-by-value. In *Proceedings of 4th International Conference on Typed Lambda Calculi and Applications (TLCA 1999)*, LNCS 1581, pp. 162–177, 1999.

[4] K. Fujita, and A. Schubert, Partially typed terms between Church-style and Curry-style. In *International Conference IFIP TCS 2000*, LNCS 1872, pp. 505–520, 2000.

[5] K. Fujita, Galois embedding from polymorphic types into existential types. In *Proceedings of 7th International Conference on Typed Lambda Calculi and Applications (TLCA 2005)*, LNCS 3461, pp. 194–208, 2005.

[6] K. Fujita and A. Schubert, Existential type systems with no types in terms. In *Proceedings of 9th International Conference on Typed Lambda Calculi and Applications (TLCA 2009)*, LNCS 5608, pp. 112–126, 2009.

[7] J.Y. Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Thèse de doctorat d'Etat, Université de Paris VII, 1972.

[8] R. Harper and M. Lillibridge, Polymorphic type assignment and CPS conversion. *Lisp and Symbolic Computation*, 6:361–380, 1993.

[9] M. Hasegawa, Relational parametricity and control. *Logical Methods in Computer Science*, 2(3:3):1–22, 2006.

[10] M. Hasegawa, Unpublished manuscript, 2007.

[11] J.C. Mitchell and G.D. Plotkin, Abstract types have existential type. *ACM Transactions on Programming Languages and Systems* 10(3):470–502, 1988.

[12] E. Moggi, Computational lambda-calculus and monads. In *Proceedings of 4th Annual Symposium on Logic in Computer Science (LICS 1989)*, pp. 14–23, 1989.

[13] K. Nakazawa, An isomorphism between cut-elimination procedure and proof reduction. In *Proceedings of 8th International Conference on Typed Lambda Calculi and Applications (TLCA 2007)*, LNCS 4583, pp.336–350, 2007.

[14] K. Nakazawa, M. Tatsuta, Y. Kameyama, and H. Nakano, Undecidability of type-checking in domain-free typed lambda-calculi with existence. In *the 17th EACSL Annual Conference on Computer Science Logic (CSL 2008)*, LNCS 5213, pp. 477–491, 2008.

[15] M. Parigot, $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proc. the International Conference on Logic Programming and Automated Reasoning*, LNCS 624, pp.190–201, 1992.

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

[17] J.C. Reynolds, Towards a theory of type structure. In *Symposium on Programming*, LNCS 19, pp.408–425, 1974.

[18] A. Sabry and P. Wadler, A reflection on call-by-value. *ACM Transactions on Programming Languages and Systems*, 19(6):916–941, 1997.

[19] M.H. Sørensen and P. Urzyczyn, A syntactic embedding of predicate logic into second-order propositional logic. *Notre Dame Journal of Formal Logic* 51(4):457–473, 2010.

[20] M. Tatsuta, K. Fujita, R. Hasegawa, and H. Nakano, Inhabitation of polymorphic and existential types. *Annals of Pure and Applied Logic* 161:1390–1399, 2010.

[21] H. Thielecke, Categorical structure of continuation passing style. Ph.D. Thesis, University of Edinburgh, 1997.

[22] J. von Plato, Natural deduction with general elimination rules. *Archive for Mathematical Logic* 40:541–567, 2001.

[23] J.B. Wells, Typability and type checking in system F are equivalent and undecidable. *Annals of Pure and Applied Logic* 98:111–156, 1999.