Type Checking and Inference for Polymorphic and Existential Types

Koji Nakazawa[1] **Makoto Tatsuta**[2]

[1]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
knak@kuis.kyoto-u.ac.jp

[2]National Institute of Informatics, Japan
tatsuta@nii.ac.jp

Abstract

This paper proves undecidability of type checking and type inference problems in some variants of typed lambda calculi with polymorphic and existential types. First, type inference in the domain-free polymorphic lambda calculus is proved to be undecidable, and then it is proved that type inference is undecidable in the negation, conjunction, and existence fragment of the domain-free typed lambda calculus. Secondly, their variants with multiple quantifier rules are introduced, and their type checking and type inference are proved to be undecidable. Finally, it is proved that we can reduce undecidability of type checking and type inference problems in the Curry-style lambda calculus in negation, conjunction, and existential fragment to undecidability of those problems in another variant of the domain-free polymorphic lambda calculus.

Keywords. type checking, type inference, polymorphic type, existential type, domain-free style

1 Introduction

The second-order universal and existential quantifiers are important from the point of view of computer science as well as logic. Girard (1972) and Reynolds (1974) have independently established the typed lambda calculus with polymorphic types, which correspond to the second-order universal quantifiers in logical systems. Since their monumental works, many papers have been devoted to investigation on the polymorphic types. The computational meaning of the second-order existential quantifiers has also been studied actively since the work of Mitchell and Plotkin (1988) on the abstract data types. In more recent years, Fujita (2005) and Hasegawa (2006) pointed out that calculi in negation, conjunction, and existence fragment can be CPS targets of polymorphic typed calculi.

Type checking and type inference problems are important for type assignment systems. *Type checking* (TC) is the problem that asks whether given typing judgment is derivable. *Type inference* (TI) asks whether given term has some type under some context. *Strong type inference* (STI), which is a generalization of TI, asks whether, for a given term and a given context, the term has some type under some extension of the given context. In the usual notation, TC asks $\Gamma \vdash M : A?$ for given Γ, M and A, TI asks $? \vdash M :?$ for given M, and STI asks $\Gamma, ? \vdash M :?$ for given Γ and M. TI is a special case of STI, so undecidability of TI directly implies undecidability of STI.

Wells (1999) proved that all these problems are undecidable in the Curry-style polymorphic lambda calculus, and it is surprising result due to Schubert (1998) that TI is undecidable even in the Church style, where we consider typability of untyped pseudo terms. Barthe and Sørensen (2000) proved that TC and STI are undecidable in the domain-free polymorphic lambda calculus. In the *domain-free style*, terms contain information of applications of quantifier rules such as $\Lambda X.M$ for the $\forall$-introduction rule and MA for the $\forall$-elimination rule, while the domain types of lambda abstractions are not indicated such as $\lambda x.M$. Fujita and Schubert (2000) independently proved the same result in a more direct way.

On the other hand, properties of lambda calculi with existential types have not been studied enough yet. The inhabitation problem in the negation, conjunction, and existence fragment was only recently proved to be decidable in Tatsuta et al. (2008). The *inhabitation* (INH) is the problem that asks $\vdash ? : A$ for given type A. It was also recently that TC and STI in its domain-free-style variant was proved to be undecidable in Nakazawa et al. (2008).

This paper studies the type checking and the type inference problems in several variants of lambda calculi with the polymorphic types and the existential types, and proves the following: (1) TI is undecidable in the domain-free polymorphic lambda calculus DF-F and the domain-free lambda calculus DF-$\lambda^{\neg\wedge\exists}$ in negation, conjunction, and existence fragment, (2) TC and TI are undecidable in their variants MWDF-F and M-$\lambda^{\neg\wedge\exists}$ which have multiple applications of quantifier rules, and MWDF means *multiple-weak-domain-free* style, (3) undecidability of TC and TI in the Curry-style $\lambda^{\neg\wedge\exists}$ can be reduced to undecidability of those in another variant WDF-F of the polymorphic lambda calculus.

As for the first result, Barthe and Sørensen (2000) and Fujita and Schubert (2000) have considered deeply on DF-F and proved that TC and STI are undecidable in DF-F, but undecidability of TI has not been proved yet.

Although TI and STI may seem to be almost the same, undecidability of TI is more difficult to prove than undecidability of STI. In many systems such as the Curry-style and the domain-free-style F, TC can be easily reduced to STI. For example, in the Curry-style F, any given TC problem $\Gamma \vdash M : A?$ can be reduced to a STI problem $\Gamma, f : A \to \bot, ? \vdash fM :?$. Hence, undecidability of STI directly follows from undecidability of TC. On the other hand, undecidability of TI in the Curry-style F is much more difficult to prove, because it needs an elaborate technique such as the proof of Wells (1999) to show that TC can be reduced to TI. This paper proves that TC can be

reduced to TI in DF-F by an elaborate technique similar to the proof of Wells (1999). This result implies undecidability of TI in DF-F, since TC in DF-F has been already proved to be undecidable.

Undecidability of TI in DF-F directly implies undecidability of TI in DF-$\lambda^{\neg\wedge\exists}$ by the result of Nakazawa et al. (2008). Nakazawa et al. (2008) claimed that they showed undecidability of TI in DF-$\lambda^{\neg\wedge\exists}$. However, it was not correct, since TI in DF-F had not been proved at that time and they used it. The undecidability result of TI in DF-F of this paper completes their proof of undecidability of TI in DF-$\lambda^{\neg\wedge\exists}$.

The systems MWDF-F and M-$\lambda^{\neg\wedge\exists}$ are variants of the Curry style calculi. M-$\lambda^{\neg\wedge\exists}$ is important since it is one the most implicitly typed systems for the existential type, that is, terms in M-$\lambda^{\neg\wedge\exists}$ have less type information than terms in other styles. Terms in MWDF-F and M-$\lambda^{\neg\wedge\exists}$ contain information of multiple applications of quantifier rules such as

$$\frac{\Gamma \vdash N : \mathsf{A}[\overline{X} := \overline{\mathsf{B}}]}{\Gamma \vdash \langle \exists^*, N\rangle : \exists \overline{X}.\mathsf{A}}\ (\exists\mathsf{I})$$

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

in M-$\lambda^{\neg\wedge\exists}$, where $\overline{X}$ denotes a finite list of type variables. We call such rules *multiple quantifier rules*. The multiple existential rules can handle mutual abstract data types even without parameters. MWDF-F and M-$\lambda^{\neg\wedge\exists}$ can be considered as an intermediate style between the domain-free style and the systems without any type annotation. This paper discusses M-$\lambda^{\neg\wedge\exists}$ in two ways. The first proof shows that TC is undecidable in M-$\lambda^{\neg\wedge\exists}$ by a direct reduction of the semi-unification problem to STI of M-$\lambda^{\neg\wedge\exists}$. The semi-unification problem has been proved to be undecidable in Kfoury et al. (1993). The second proof shows that TC and TI are undecidable in MWDF-F by reducing undecidability of TC and TI in the Curry-style F, and then it is proved that TC and TI in MWDF-F can be reduced to those in M-$\lambda^{\neg\wedge\exists}$ by the method in Nakazawa et al. (2008) with some modification.

Furthermore, this paper shows that the proof method of Nakazawa et al. (2008) can be adopted to the Curry-style $\lambda^{\neg\wedge\exists}$. The curry-style F is not suitable to be the source calculus of the *continuation-passing-style* (CPS) translation to Curry-$\lambda^{\neg\wedge\exists}$, so we introduce another variant WDF-F, which works as the source calculus. WDF means *weak-domain-free* style. It is proved that undecidability of TC and TI in Curry-$\lambda^{\neg\wedge\exists}$ is reduced to undecidability of those in WDF-F. However, undecidability of TC and TI in WDF-F is an open problem.

Figure 1 summarizes related results about decidability of TC, STI, TI, and INH, where "no" means that the problem is undecidable. (1) is proved by Wells (1999), (2) is proved by Barthe and Sørensen (2000) and Fujita and Schubert (2000), independently, (3) is proved by Tatsuta et al. (2008), and (4) is proved by Nakazawa et al. (2008). "NO" denotes the main results of this paper.

Section 2 defines the domain-free polymorphic lambda calculus DF-F, and proves that TI is undecidable in DF-F. Section 3 defines the domain-free typed lambda calculus DF-$\lambda^{\neg\wedge\exists}$ with negation, conjunction and existence, and completes the proof of undecidability of TI in DF-$\lambda^{\neg\wedge\exists}$. Section 4 defines their variants MWDF-F and M-$\lambda^{\neg\wedge\exists}$ with the multiple quantifier rules, and proves that TC and TI are undecidable in them. Section 5 discusses about the Curry-style $\lambda^{\neg\wedge\exists}$, and shows that undecidability of TC and TI in Curry-$\lambda^{\neg\wedge\exists}$ can be reduced to undecidability of those problems in the weak-domain-free-style polymorphic lambda calculus WDF-F.

2 Domain-Free Lambda Calculus with Polymorphic Types

In this section, we define the domain-free polymorphic lambda calculus DF-F, and prove undecidability of the type inference problem in DF-F by a similar technique of Wells (1999).

Definition 2.1 (DF-F) (1) The types (denoted by A, B,..., and called $\rightarrow\forall$-*types*), and the terms (denoted by M, N,...) of DF-F are defined by
$A ::= X \mid A{\rightarrow}A \mid \forall X.A$,
$M ::= x \mid \lambda x.M \mid \Lambda X.M \mid MM \mid MA$,
where X and x denote a type variable and a term variable, respectively. 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. We use $\equiv$ to denote syntactic identity modulo renaming of bound variables. A variable is said to be *free* in a term if it is not bound in the term. The free variables in types are similarly defined. $FV(M)$ is defined as the set of the variables which are free in M. A term M and a type A are said to be *closed* if no free variable occurs in M and A, respectively. We write $\forall X.A{\rightarrow}B$ for $(\forall X.A){\rightarrow}B$.

(2) Γ denotes a context, which is a finite set of type assignments in the form of $(x : A)$. We suppose that if both $(x : A)$ and $(x : B)$ are in Γ, then $A \equiv B$ holds. We write $\Gamma, x : A$ for $\Gamma \cup \{x : A\}$, and Γ_1, Γ_2 for $\Gamma_1 \cup \Gamma_2$. $\neg\Gamma$ is defined as $\{(x : \neg A) | (x : A) \in \Gamma\}$, and $\mathrm{dom}(\Gamma)$ is defined as $\{x | (x : A) \in \Gamma\}$.

(3) The typing rules of DF-F are the following.

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

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

$$\frac{\Gamma_1 \vdash M : A{\rightarrow}B \quad \Gamma_2 \vdash N : A}{\Gamma_1, \Gamma_2 \vdash MN : B}\ (\rightarrow\mathsf{E})$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda X.M : \forall X.A}\ (\forall\mathsf{I})$$

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

$A[X := B]$ is the ordinary capture-avoiding substitution for types. In the rule ($\forall$I), the lower sequent must not contain X freely. We write $\Gamma \vdash_{\text{DF-}F} M : A$ to denote that $\Gamma \vdash M : A$ is derivable by the typing rules above.

(4) $\bot$ denotes the type $\forall X.X$.

Type checking (TC) is the problem that asks whether $\Gamma \vdash M : A$ is derivable for given Γ, M, and A. *Type inference* (TI) asks whether there exist Γ and A such that $\Gamma \vdash M : A$ is derivable for given M. *Strong type inference* (STI) asks whether there exist Γ and A such that $\Gamma, \Gamma_0 \vdash M : A$ is derivable for given M and Γ_0.

Decidability problems of DF-F has been already discussed in Barthe and Sørensen (2000) and Fujita and Schubert (2000). They proved only that TC and STI are undecidable in the calculus, and undecidability of TI in DF-F has not been proved. In general, undecidability of TI is more difficult to prove than undecidability of STI. TC can be easily

F	TC	STI	TI	INH	$\lambda^{\neg\wedge\exists}$	TC	STI	TI	INH
Curry-	no$^{(1)}$	no	no$^{(1)}$	no					
MWDF-	NO	NO	NO		M-	NO	NO	NO	yes$^{(3)}$
WDF-	?	?	?		Curry-	?	?	?	
DF-	no$^{(2)}$	no$^{(2)}$	NO		DF-	no$^{(4)}$	no	NO	

Figure 1: Decidability of TC, STI, TI and INH

reduced to STI in DF-F if we consider a STI problem $\Gamma, f : A \to \bot, ? \vdash fM :?$ to solve a TC problem $\Gamma \vdash M : A$?. On the other hand, undecidability of TI is much more difficult, because it needs an elaborate technique such as the proof of Wells (1999) to show that TC can be reduced to TI in the Curry-style F.

In the rest of this section, we will prove undecidability of TI by showing that TC can be reduced to TI in DF-F by an elaborate technique similar to Wells (1999).

Lemma 2.2 For any $\to\forall$-type A and any closed DF-F-term M, there exists a closed DF-F-term J such that $\vdash_{\mathsf{DF}\text{-}F} M : A$ is derivable if and only if $\vdash_{\mathsf{DF}\text{-}F} J : B$ is derivable for some type B.

Proof. Take J as

$$J \equiv \lambda x.(\lambda y.x(A{\to}\bot)M)(x(\bot{\to}\bot)x).$$

It is easily proved that if $\vdash M : A$ holds then J is typable, so we will prove the converse direction.

The leftmost variable occurrence $\mathsf{lvar}(A)$ and the left depth $\mathsf{ldep}(A)$ of a type A are defined by

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

and then we have some lemmas: (1) $\mathsf{lvar}(A) \neq X$ implies $\mathsf{ldep}(A[X := B]) = \mathsf{ldep}(A)$, (2) if $\mathsf{ldep}(A_1) = \mathsf{ldep}(A_2)$ and $A_1[X := B] \equiv A_2{\to}C$ hold, then we have $\mathsf{lvar}(A_1) = X$. (1) is easily proved by induction on A. For (2), $\mathsf{ldep}(A_1[X := B]) = \mathsf{ldep}(A_2) + 1 = \mathsf{ldep}(A_1) + 1$ holds, so $\mathsf{lvar}(A_1)$ must be X by (1).

Now suppose that we have a derivation D of $\vdash J : B$ for some B. In the following, C_x denotes the type of the variable x in D.

Since the subterm $x(\bot{\to}\bot)x$ is typable, we have $C_x \equiv \forall X.C_1$ and $C_1[X := \bot{\to}\bot] \equiv C_x{\to}C_2$ for some C_1 and C_2. $\mathsf{ldep}(C_1) = \mathsf{ldep}(\forall X.C_1) = \mathsf{ldep}(C_x)$ holds, so we have $\mathsf{lvar}(C_1) = X$ by the lemma (2). Hence, C_1 is of the form

$$\forall\overline{Y}_{n+1}(\forall\overline{Y}_n.(\cdots(\forall\overline{Y}_2.(\forall\overline{Y}_1. X{\to}A_1){\to}A_2){\to}\cdots A_{n-1}){\to}A_n),$$

where each $\overline{Y}_i$ denotes a finite list of type variables which may be empty, and $n \geq 0$. Then $C_1[X := \bot{\to}\bot]$ is

$$\forall\overline{Y}_{n+1}(\forall\overline{Y}_n.(\cdots(\forall\overline{Y}_2.(\forall\overline{Y}_1. (\bot{\to}\bot){\to}A'_1){\to}A'_2){\to}\cdots A'_{n-1}){\to}A'_n),$$

where A'_i is $A_i[X := \bot{\to}\bot]$. Since $C_1[X := \bot{\to}\bot] \equiv C_x{\to}C_2$ holds, $\forall X.C_1{\to}C_2$ should be identical to

$$\forall\overline{Y}_{n+1}(\forall\overline{Y}_n.(\cdots(\forall\overline{Y}_2.(\forall\overline{Y}_1. (\bot{\to}\bot){\to}A'_1){\to}A'_2){\to}\cdots A'_{n-1}){\to}A'_n).$$

The leftmost variable of this type is bound immediately outside of it in $\bot \equiv \forall X.X$, so the variable $\mathsf{lvar}(\forall X.C_1{\to}C_2)$ must be bound immediately outside of it in $\forall X.C_1{\to}C_2$. Hence, n must be 0, $\overline{Y}_1$ must be empty, and $\forall X.C_1$ must be $\forall X.X$.

Since the type of x in D must be $\bot$, D contains a derivation of $M : A$. □

Theorem 2.3 (1) Type checking can be reduced to type inference in DF-F.

(2) Type inference is undecidable in DF-F.

Proof. (1) A judgment $x_1 : A_1, \cdots, x_n : A_n \vdash M : A$ holds if and only if $\vdash \lambda x_1.\cdots\lambda x_n.M : A_1{\to}\cdots{\to}A_n{\to}A$ holds, which can be reduced to a typability problem for some J by Lemma 2.2.

(2) Since it has been proved in Barthe and Sørensen (2000) and Fujita and Schubert (2000) that TC is undecidable in DF-F, TI is undecidable by (1). □

3 Domain-Free Lambda Calculus with Existential Types

In contrast to the polymorphic lambda calculus, the type inhabitation problem (INH) is decidable in the negation, conjunction, and existence fragment, which was proved in Tatsuta et al. (2008). So the problems such as TC and TI in $\lambda^{\neg\wedge\exists}$ may seem easier than those in F. However, TC and TI in DF-$\lambda^{\neg\wedge\exists}$ are not less difficult than those in DF-F, which was proved in Nakazawa et al. (2008) by showing that TC and TI in DF-F can be reduced to those in DF-$\lambda^{\neg\wedge\exists}$ by a continuation-passing-style (CPS) translation and type contraction translation.

In this section, combining the undecidability of TI in DF-F, which is the result of the previous section, we complete the proof of undecidability of TI in DF-$\lambda^{\neg\wedge\exists}$.

Definition 3.1 (DF-$\lambda^{\neg\wedge\exists}$) (1) 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

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

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 is bound in M.

(2) The typing rules of DF-$\lambda^{\neg\wedge\exists}$ are the following.

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

$$\frac{\Gamma, x : \mathsf{A} \vdash M : \bot}{\Gamma \vdash \lambda x.M : \neg\mathsf{A}}\ (\neg\mathsf{I})$$

$$\frac{\Gamma_1 \vdash M : \neg\mathsf{A} \quad \Gamma_2 \vdash N : \mathsf{A}}{\Gamma_1, \Gamma_2 \vdash MN : \bot}\ (\neg\mathsf{E})$$

$$\frac{\Gamma_1 \vdash M : \mathsf{A} \quad \Gamma_2 \vdash N : \mathsf{B}}{\Gamma_1, \Gamma_2 \vdash \langle M, N\rangle : \mathsf{A} \wedge \mathsf{B}}\ (\wedge\mathsf{I})$$

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

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

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

In the rule $(\exists\mathsf{E})$, Γ_2 and C must not contain X freely. We write $\Gamma \vdash_{\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}} M : \mathsf{A}$ to denote that $\Gamma \vdash M : \mathsf{A}$ is derivable by the typing rules above.

(3) $A_1 \wedge A_2 \wedge \cdots \wedge A_{n-1} \wedge A_n$ denotes $A_1 \wedge (A_2 \wedge (\cdots \wedge (A_{n-1} \wedge A_n))$, $\langle M_1, M_2, \cdots, M_{n-1}, M_n\rangle$ denotes $\langle M_1, \langle M_2, \cdots \langle M_{n-1}, M_n\rangle\rangle\rangle$, and A^n denotes $\underbrace{A \wedge \cdots \wedge A}_{n}$.

Theorem 3.2 Type inference is undecidable in DF-$\lambda^{\neg\wedge\exists}$.

Proof. By Proposition 3 of Nakazawa et al. (2008) we can reduce TI in DF-F to TI in DF-$\lambda^{\neg\wedge\exists}$. By Theorem 2.3 TI in DF-F is undecidable, so is TI in DF-$\lambda^{\neg\wedge\exists}$. □

4 Multiple Quantifier Rules

In this section, we prove that TC and TI are undecidable in another variant M-$\lambda^{\neg\wedge\exists}$ of the negation, conjunction, and existence fragment, which contains the multiple applications of quantifier rules. We call such rules *multiple quantifier rules*.

M-$\lambda^{\neg\wedge\exists}$ is important since it is one of the most implicitly typed systems for the existential type, that is, terms in M-$\lambda^{\neg\wedge\exists}$ have less type information than terms in other styles. In fact, terms in M-$\lambda^{\neg\wedge\exists}$ contain only information about possibility to eliminate quantifiers, and we cannot entirely omit the information of use of quantifier rules like the Curry style F because of the minor premise in the elimination rule for existence. M-$\lambda^{\neg\wedge\exists}$ works as a CPS target of MWDF-F, which can be considered as an intermediate style between the domain-free style and the Curry style.

We will prove the undecidability of the problems in M-$\lambda^{\neg\wedge\exists}$ in two different ways. First, we directly reduce undecidability of STI in M-$\lambda^{\neg\wedge\exists}$ to semi-unification problem, and prove undecidability of TC by using it. Secondly, we show that TC and TI are undecidable in another variant of F, which will be called MWDF-F, which means the multiple-weak-domain-free F. We prove that TC and TI in MWDF-F can be reduced to those problems in M-$\lambda^{\neg\wedge\exists}$ by the technique of Nakazawa et al. (2008).

Definition 4.1 (M-$\lambda^{\neg\wedge\exists}$) (1) We use $\overline{X}$ and $\overline{\mathsf{A}}$ to denote finite lists of type variables and types, respectively. When $\overline{X}$ denotes $(X_1, \cdots, X_n)$ and $\overline{\mathsf{B}}$ denotes $(\mathsf{B}_1, \cdots, \mathsf{B}_n)$, $\exists\overline{X}.\mathsf{A}$ denotes the type $\exists X_1 \cdots \exists X_n.\mathsf{A}$, and $\mathsf{A}[\overline{X} := \overline{\mathsf{B}}]$ denotes the simultaneous substitution $\mathsf{A}[X_1 := \mathsf{B}_1, \cdots, X_n := \mathsf{B}_n]$. $\overline{X}$ and $\overline{\mathsf{B}}$ may denote the empty list, and then both $\exists\overline{X}.\mathsf{A}$ and $\mathsf{A}[\overline{X} := \overline{\mathsf{B}}]$ denote the same type A.

(2) The types of M-$\lambda^{\neg\wedge\exists}$ are the $\neg\wedge\exists$-types. The terms (denoted by M, N,...) of M-$\lambda^{\neg\wedge\exists}$ are defined by

$$M ::= x \mid \lambda x.M \mid \langle M, M\rangle \mid \langle \exists^*, M\rangle \mid MM \mid M\pi_1 \mid M\pi_2 \mid M[x.M].$$

(3) The typing rules of M-$\lambda^{\neg\wedge\exists}$ are the same as those of DF-$\lambda^{\neg\wedge\exists}$ except for rules of existence, which are the following.

$$\frac{\Gamma \vdash N : \mathsf{A}[\overline{X} := \overline{\mathsf{B}}]}{\Gamma \vdash \langle \exists^*, N\rangle : \exists\overline{X}.\mathsf{A}}\ (\exists\mathsf{I})$$

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

In the rule $(\exists\mathsf{E})$, Γ_2 and C must not contain any type variable in $\overline{X}$ freely. We write $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} M : \mathsf{A}$ to denote that $\Gamma \vdash M : \mathsf{A}$ is derivable by the typing rules above.

4.1 Undecidability of TC in M-$\lambda^{\neg\wedge\exists}$

This subsection gives a direct reduction of the semi-unification problem to STI in M-$\lambda^{\neg\wedge\exists}$, and prove undecidability of TC by using it.

Definition 4.2 (Semi-Unification Problem) (1) The $\wedge$-*types* are $\neg\wedge\exists$-types which contain neither $\neg$ nor $\exists$. The $\wedge$-*substitutions* are simultaneous type substitutions $\Theta = [\overline{X} := \overline{\mathsf{A}}]$ where each element A_i in $\overline{\mathsf{A}}$ is $\wedge$-type.

(2) A pair $(\mathsf{A}_1 \leq \mathsf{B}_1, \mathsf{A}_2 \leq \mathsf{B}_2)$ of inequalities between two $\wedge$-types is called an *instance of the semi-unification problem*. An instance $I = (\mathsf{A}_1 \leq \mathsf{B}_1, \mathsf{A}_2 \leq \mathsf{B}_2)$ has a solution if there exist $\wedge$-substitutions Θ, Θ_1 and Θ_2 such that $\mathsf{A}_1\Theta\Theta_1 \equiv \mathsf{B}_1\Theta$ and $\mathsf{A}_2\Theta\Theta_2 \equiv \mathsf{B}_2\Theta$.

If we replace $\wedge$ by $\rightarrow$, this is a special case of the original semi-unification problem. It is proved in Kfoury et al. (1993) that the special case of the semi-unification is undecidable.

Theorem 4.3 (Kfoury et al. (1993)) It is not possible to effectively decide whether a given instance of the semi-unification problem has a solution.

Lemma 4.4 For any instance I of the semi-unification problem, there exist Γ_0 and M such that I has a solution if and only if $\Gamma_0, \Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} M : \mathsf{A}$ is derivable for some Γ and A.

Proof. For given $I = (\mathsf{A}_1 \leq \mathsf{B}_1, \mathsf{A}_2 \leq \mathsf{B}_2)$, take Γ_0 and M as

$$\begin{aligned}\Gamma_0 = \{\ & c : \neg\exists X(X \wedge X),\\ & d : \neg((\exists\overline{X}.((\mathsf{A}_1 \wedge \mathsf{B}_1) \wedge (\mathsf{A}_2 \wedge \mathsf{B}_2))) \wedge \bot^4)\},\\ M \equiv d\langle\ & \langle \exists^*, \langle x_1, x_2\rangle\rangle, P_1, Q_1, P_2, Q_2\ \rangle,\end{aligned}$$

where P_i and Q_i for $i = 1, 2$ are defined as

$$\begin{aligned}P_i &\equiv c\langle \exists^*, \langle x_i, \langle y_i, z_i\rangle\rangle\rangle,\\ Q_i &\equiv \langle \exists^*, z_i\rangle[u.c\langle \exists^*, \langle y_i, u\rangle\rangle],\end{aligned}$$

and $\overline{X}$ denotes the free type variables in A_1, B_1, A_2, and B_2.

(I) Suppose that I has a solution $(\Theta, \Theta_1, \Theta_2)$, that is, we have $\mathsf{A}_i\Theta\Theta_i \equiv \mathsf{B}_i\Theta$ for $i = 1, 2$. Let $\Gamma_i = \{x_i : \mathsf{A}_i\Theta \wedge \mathsf{B}_i\Theta, y_i : \mathsf{A}_i\Theta, z_i : \mathsf{B}_i\Theta\}$, then we have

$$\frac{\Gamma_i \vdash \langle x_i, \langle y_i, z_i\rangle\rangle : (\mathsf{A}_i\Theta \wedge \mathsf{B}_i\Theta) \wedge (\mathsf{A}_i\Theta \wedge \mathsf{B}_i\Theta)}{\Gamma_i \vdash \langle \exists^*, \langle x_i, \langle y_i, z_i\rangle\rangle\rangle : \exists X.(X \wedge X)},$$

so we have $\Gamma_0, \Gamma_i \vdash P_i : \bot$. Similarly, we have $\Gamma_0, \Gamma_i, u : \mathsf{A}_i\Theta \vdash c\langle \exists^*, \langle y_i, u\rangle\rangle : \bot$. Moreover, since $\mathsf{A}_i\Theta\Theta_i \equiv \mathsf{B}_i\Theta$ holds, $\Gamma_i \vdash \langle \exists^*, z_i\rangle : \exists\overline{Y}.\mathsf{A}_i\Theta$ is derivable for some $\overline{Y}$. We also have $\Gamma_0, \Gamma_i, u : \mathsf{A}_i\Theta \vdash c\langle \exists^*, \langle y_i, u\rangle\rangle : \bot$, so $\Gamma_0, \Gamma_i \vdash Q_i : \bot$ holds. On the other hand, we have

$$\frac{\Gamma_1, \Gamma_2 \vdash \langle x_1, x_2\rangle : (\mathsf{A}_1\Theta \wedge \mathsf{B}_1\Theta) \wedge (\mathsf{A}_2\Theta \wedge \mathsf{B}_2\Theta)}{\Gamma_1, \Gamma_2 \vdash \langle \exists^*, \langle x_1, x_2\rangle\rangle : \exists\overline{X}.(\mathsf{A}_1 \wedge \mathsf{B}_1) \wedge (\mathsf{A}_2 \wedge \mathsf{B}_2)}$$

Hence, we have $\Gamma_0, \Gamma_1, \Gamma_2 \vdash M : \bot$.

(II) Conversely, suppose that there exist Γ and A such that $\Gamma_0, \Gamma \vdash M : \mathsf{A}$. We write Γ' for Γ_0, Γ. First, since $\Gamma' \vdash \langle \exists^*, \langle x_1, x_2\rangle\rangle : \exists\overline{X}.(\mathsf{A}_1 \wedge \mathsf{B}_1) \wedge (\mathsf{A}_2 \wedge \mathsf{B}_2)$ holds, the type of x_i in Γ must be $\mathsf{A}_i\Theta \wedge \mathsf{B}_i\Theta$ for some Θ. Note that Θ is a substitution but it may not a $\wedge$-substitution. Secondly, since $\Gamma' \vdash P_i : \bot$ holds, the type of $\langle y_i, z_i\rangle$ must be the same as the type of x_i, which is $\mathsf{A}_i\Theta \wedge \mathsf{B}_i\Theta$. So Γ must contain $y_i : \mathsf{A}_i\Theta$ and $z_i : \mathsf{B}_i\Theta$. Thirdly, the derivation of $\Gamma' \vdash Q_i : \bot$ must be of the following form.

$$\frac{\dfrac{\Gamma' \vdash z_i : \mathsf{C}_i\Theta_i}{\Gamma' \vdash \langle \exists^*, z_i\rangle : \exists\overline{Y}.\mathsf{C}_i} \quad \begin{matrix}\vdots\\ \Gamma', u : \mathsf{C}_i \vdash c\langle \exists^*, \langle y_i, u\rangle\rangle : \bot\end{matrix}}{\Gamma' \vdash Q_i : \bot}$$

Since the right premise holds, y_i and u must have the same type, so we have $\mathsf{C}_i \equiv \mathsf{A}_i\Theta$. Moreover, we have $\mathsf{B}_i\Theta \equiv \mathsf{C}_i\Theta_i$, so we have $\mathsf{A}_i\Theta\Theta_i \equiv \mathsf{B}_i\Theta$.

Finally we construct $\wedge$-substitutions from Θ, Θ_1, and Θ_2. Following the idea of Wells (1999), define the erase function with a fresh type variable Z as follows

$e(X) = X$,
$e(\mathsf{A} \wedge \mathsf{B}) = e(\mathsf{A}) \wedge e(\mathsf{B})$,
$e(\neg\mathsf{A}) = e(\mathsf{A})$,
$e(\exists X.\mathsf{A}) = e(\mathsf{A}[X := Z])$,

and we prove $e(\mathsf{A}\Theta) \equiv e(e(\mathsf{A})\Theta)$ for any $\neg \wedge \exists$-type A and any substitution Θ by induction on the size of A as follows.

(Case X) The both sides are equal to $e(X\Theta)$.

(Case $\mathsf{B} \wedge \mathsf{C}$) The left-hand side is $e(\mathsf{B}\Theta) \wedge e(\mathsf{C}\Theta)$, which is identical to $e(e(\mathsf{B})\Theta) \wedge e(e(\mathsf{C})\Theta)$ by the induction hypothesis. The right-hand side is $e((e(\mathsf{B}) \wedge e(\mathsf{C}))\Theta) \equiv e(e(\mathsf{B})\Theta) \wedge e(e(\mathsf{C})\Theta)$.

(Case $\neg\mathsf{B}$) The left-hand side is $e(\mathsf{B}\Theta)$, which is identical to $e(e(\mathsf{B})\Theta)$ by the induction hypothesis. The right-hand side is $e(e(\neg\mathsf{B})\Theta) \equiv e(e(\mathsf{B})\Theta)$.

(Case $\exists X.\mathsf{B}$) By renaming bound variables, we suppose that X does not occur in Θ. The left-hand side is $e(\mathsf{B}\Theta[X := Z])$, which is identical to $e(\mathsf{B}[X := Z]\Theta)$ since Z is fresh. By the induction hypothesis, this type is identical to $e(e(\mathsf{B}[X := Z])\Theta)$, which is equal to the right-hand side.

Take the $\wedge$-substitutions (Ξ, Ξ_1, Ξ_2) as $X\Xi = e(X\Theta)$ and $X\Xi_i = e(X\Theta_i)$. Since $\mathsf{A}_i\Theta\Theta_i \equiv \mathsf{B}_i\Theta$ holds, we have $e(\mathsf{A}_i\Theta\Theta_i) \equiv e(\mathsf{B}_i\Theta) \equiv \Xi(\mathsf{B}_i)$. By the claim proved above, we have $e(\mathsf{A}_i\Theta\Theta_i) \equiv e(e(\mathsf{A}_i\Theta)\Theta_i) \equiv \mathsf{A}_i\Xi\Xi_i$, so the triple (Ξ, Ξ_1, Ξ_2) is a solution of I. □

Proposition 4.5 Strong type inference is undecidable in M-$\lambda^{\neg\wedge\exists}$.

Proof. By Theorem 4.3 and Lemma 4.4. □

Theorem 4.6 Type checking is undecidable in M-$\lambda^{\neg\wedge\exists}$.

Proof. Each STI problem $\Gamma, ? \vdash M :?$ is reduced to a TC problem $\Gamma, c : \mathsf{A} \vdash (\lambda x.c)(\lambda y_1.\cdots\lambda y_n.M) : \mathsf{A}?$, where $\{y_1, \cdots, y_n\} = FV(M) - \mathrm{dom}(\Gamma)$. In fact, if $\Gamma, c : \mathsf{A} \vdash (\lambda x.c)(\lambda y_1.\cdots\lambda y_n.M) : \mathsf{A}$ holds, then M must have some type under Γ, Δ for some context Δ for the variables $y_1, \cdots, y_n$. Conversely, if $\Gamma, \Delta \vdash M : \mathsf{B}$ holds for some Δ and B, then we have $\Gamma, c : \mathsf{A} \vdash (\lambda x.c)(\lambda y_1.\cdots\lambda y_n.M) : \mathsf{A}$ since $FV(M) - \{y_1, \cdots, y_n\} \subseteq \mathrm{dom}(\Gamma)$ holds.

Hence, undecidability of TC in M-$\lambda^{\neg\wedge\exists}$ is reduced to that of STI, and it has been proved in Proposition 4.5. □

We cannot easily prove undecidability of TI from the proposition. We will adopt another method to prove undecidability of TI in M-$\lambda^{\neg\wedge\exists}$.

4.2 Polymorphic Lambda Calculus with Multiple Quantifier Rules

The rest of this section will devoted to proof of undecidability of TI in M-$\lambda^{\neg\wedge\exists}$. It also gives another proof of undecidability of TC in M-$\lambda^{\neg\wedge\exists}$.

The proof consists of two parts. First, we introduce another polymorphic lambda calculus MWDF-F with multiple quantifier rules, and prove that TC and TI are undecidable in MWDF-F. Secondly, we prove that undecidability of TC and TI in M-$\lambda^{\neg\wedge\exists}$ can be reduced to undecidability of TC and TI in MWDF-F by the method of Nakazawa et al. (2008).

In this subsection, we prove that TC and TI are undecidable in MWDF-F by showing that it can be reduced to undecidability of those problems in the Curry-style polymorphic lambda calculus Curry-F, which has been proved in Wells (1999).

Definition 4.7 (MWDF-F) (1) The types of MWDF-F are the $\rightarrow\forall$-types. The terms of MWDF-F are defined by

$M ::= x \mid \lambda x.M \mid \Lambda^*.M \mid MM \mid M\bullet^*$.

(2) The typing rules of MWDF-F are the same as those of DF-F except for the rules for the universal quantifiers given as follows.

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda^*.M : \forall\overline{X}.A}\ (\forall\mathsf{I}) \qquad \frac{\Gamma \vdash M : \forall\overline{X}.A}{\Gamma \vdash M\bullet^* : A[\overline{X} := \overline{B}]}\ (\forall\mathsf{E})$$

The Curry-style F is defined as follows.

Definition 4.8 (Curry-F) Curry-F is the Curry-style polymorphic lambda calculus, whose types are $\rightarrow\forall$-types, and whose terms are defined as

$M ::= x \mid \lambda x.M \mid MM$.

The typing rules of Curry-F are the same as those of DF-F except for the rules for the universal quantifiers given as follows.

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

Proposition 4.9 Type checking and type inference are undecidable in MWDF-F.

Proof. Define the map $\lfloor\cdot\rfloor$ from Curry-F-terms to MWDF-F-terms by

$\lfloor x \rfloor \equiv \Lambda^*.x\bullet^*$,
$\lfloor \lambda x.M \rfloor \equiv \Lambda^*.\lambda x.\lfloor M \rfloor$,
$\lfloor MN \rfloor \equiv \Lambda^*.\lfloor M \rfloor \lfloor N \rfloor\bullet^*$,

and we have that $\Gamma \vdash M : A$ is derivable in the Curry style if and only if $\Gamma \vdash \lfloor M \rfloor : A$ is derivable in MWDF-F. It is easily proved that $\Gamma \vdash_{\text{MWDF-}F} \lfloor M \rfloor : A$ implies $\Gamma \vdash_{\text{Curry-}F} M : A$, so we will prove the converse direction.

We prove the claim by induction on M. Suppose that we have a derivation of $\Gamma \vdash_{\text{Curry-}F} M : A$ holds. We can assume that no premise of any use of $(\forall E)$ rule is never a conclusion of a use of $(\forall I)$ rule in the derivation, that is, the derivation enjoys the INST-before-GEN property in Wells (1999).

(Case $M \equiv x$) The derivation of $\Gamma \vdash_{\text{Curry-}F} x : A$ is of the following form.

$$\begin{array}{l} \Gamma', x : B \vdash x : B \\ \quad\vdots\ (\forall\mathsf{E}) \text{ (zero or more times)} \\ \quad\vdots\ (\forall\mathsf{I}) \text{ (zero or more times)} \\ \Gamma', x : B \vdash x : A \end{array}$$

So we have $\Gamma', x : B \vdash_{\text{MWDF-}F} \Lambda^*.x\bullet^* : A$.

(Case $M \equiv \lambda x.N$) The derivation of $\Gamma \vdash_{\text{Curry-}F} \lambda x.N : A$ is of the following form.

$$\begin{array}{l} \dfrac{\Gamma, x : B \vdash N : C}{\Gamma \vdash \lambda x.N : B{\rightarrow}C} \\ \quad\vdots\ (\forall\mathsf{I}) \text{ (zero or more times)} \\ \Gamma \vdash \lambda x.N : A \end{array}$$

By the induction hypothesis, we have $\Gamma, x : B \vdash_{\text{MWDF-}F} \lfloor N \rfloor : C$, so we have $\Gamma \vdash_{\text{MWDF-}F} \lambda x.\lfloor N \rfloor : B{\rightarrow}C$. Hence, we have $\Gamma \vdash_{\text{MWDF-}F} \Lambda^*.\lambda x.\lfloor N \rfloor : A$.

(Case $M \equiv N_1N_2$) The derivation of $\Gamma \vdash_{\text{Curry-}F} N_1N_2 : A$ is of the following form.

$$\begin{array}{l} \dfrac{\Gamma \vdash N_1 : C{\rightarrow}B \quad \Gamma \vdash N_2 : C}{\Gamma \vdash N_1N_2 : B} \\ \quad\vdots\ (\forall\mathsf{E}) \text{ (zero or more times)} \\ \quad\vdots\ (\forall\mathsf{I}) \text{ (zero or more times)} \\ \Gamma \vdash N_1N_2 : A \end{array}$$

By the induction hypotheses, we have $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} \lfloor N_1 \rfloor : C{\to}B$ and $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} \lfloor N_2 \rfloor : C$, so we have $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} \lfloor N_1 \rfloor \lfloor N_2 \rfloor : B$. Hence, we have $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} \Lambda^*.\lfloor N_1 \rfloor \lfloor N_2 \rfloor \bullet^* : A$.

Therefore undecidability of TC and TI in MWDF-F is reduced to undecidability of TC and TI in Curry-F, which has been proved in Wells (1999). □

4.3 Undecidability of TI in M-$\lambda^{\neg\wedge\exists}$

We will prove that TC and TI in M-$\lambda^{\neg\wedge\exists}$ can be reduced to those in MWDF-F. We borrow the idea of the contraction translation on types from Nakazawa et al. (2008). However, we cannot directly apply the original definition of the translation, and some modification is needed.

In line with the proof method of Nakazawa et al. (2008), we define a negative translation $(\cdot)^\bullet$ from $\to\forall$-types to $\neg\wedge\exists$-types, a CPS translation $[\![\cdot]\!]$ from MWDF-F to M-$\lambda^{\neg\wedge\exists}$, and a subsystem M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ of M-$\lambda^{\neg\wedge\exists}$ which is the image of the CPS translation. Then we prove that M has a type A in MWDF-F if and only if $[\![M]\!]$ has the type $\neg A^\bullet$ in M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. Furthermore, we show that M-$\lambda^{\neg\wedge\exists}$ is a conservative extension of M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$, that is, for any term of the form $[\![M]\!]$, if $[\![M]\!]$ has a type $\neg A^\bullet$ in M-$\lambda^{\neg\wedge\exists}$, then $[\![M]\!]$ has the type $\neg A^\bullet$ in M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. By these facts, we can conclude that M has a type A in MWDF-F if and only if $[\![M]\!]$ has the type $\neg A^\bullet$ in M-$\lambda^{\neg\wedge\exists}$, so TC in MWDF-F can be reduced to that in M-$\lambda^{\neg\wedge\exists}$.

Definition 4.10 (CPS Translation) (1) The *negative translation* from $\to\forall$-types to $\neg\wedge\exists$-types is defined by

$X^\bullet \equiv X$,
$(A{\to}B)^\bullet \equiv \neg A^\bullet \wedge B^\bullet$,
$(\forall X.A)^\bullet \equiv \exists X.A^\bullet$.

$\Gamma^\bullet$ is defined as $\{(x : \mathsf{A}^\bullet) | (x : \mathsf{A}) \in \Gamma\}$.

(2) The *CPS translation* from terms in MWDF-F to terms in M-$\lambda^{\neg\wedge\exists}$ is defined by

$[\![x]\!] \equiv \lambda k.xk$,
$[\![\lambda x.M]\!] \equiv \lambda k.(\lambda x.[\![M]\!](k\pi_2))(k\pi_1)$,
$[\![MN]\!] \equiv \lambda k.[\![M]\!]\langle [\![N]\!], k\rangle$,
$[\![\Lambda^*.M]\!] \equiv \lambda k.k[k'.[\![M]\!]k']$,
$[\![M\bullet^*]\!] \equiv \lambda k.[\![M]\!]\langle \exists^*, k\rangle$,

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

Proposition 4.11 $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$ implies $\neg\Gamma^\bullet \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^\bullet$.

Proof. By induction on the derivation of $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$. □

Definition 4.12 (M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$) (1) The *continuation types* (denoted by $\mathcal{A}, \mathcal{B}, \ldots$) are defined by

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

The *CPS types* are the types of the form $\neg\mathcal{A}$. The *CPS terms* (denoted by $P, Q, \ldots$) are defined by

$$P ::= \lambda k.xk \mid \lambda k.(\lambda x.P(k\pi_2)(k\pi_1)) \mid \lambda k.P\langle Q, k\rangle \mid \lambda k.P\langle \exists^*, k\rangle \mid \lambda k.k[k'.Pk'],$$

where occurrences of k and k' denote those of the same variable, for example, $\lambda k.xk$ denotes $\lambda k_1.xk_1$ but does not denote $\lambda k_1.xk_2$ for $k_1 \equiv k_2$. We define the subsystem M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ of M-$\lambda^{\neg\wedge\exists}$ by restricting terms and types to CPS terms and CPS types, respectively. The judgments of M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ are restricted to those of the form $\neg\Gamma \vdash P : \neg\mathcal{A}$. The typing rules of M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$ are the following.

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

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

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

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

$$\frac{\neg\Gamma \vdash P : \neg\mathcal{B}}{\neg\Gamma \vdash \lambda k.k[k'.Pk'] : \neg\exists\overline{X}.\mathcal{B}}$$

In the last rule, Γ must not contain any type variable in $\overline{X}$ freely. We write $\neg\Gamma \vdash_{\mathsf{cps}} P : \neg\mathcal{A}$ to denote that the judgment is derivable by the rules above.

(2)The inverse translation $(\cdot)^\circ$ from continuation types to $\to\forall$-types is defined by

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

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

$(\lambda k.xk)^\# \equiv x$,
$(\lambda k.(\lambda x.P(k\pi_2))(k\pi_1))^\# \equiv \lambda x.P^\#$,
$(\lambda k.k[k'.Pk'])^\# \equiv \Lambda^*.P^\#$,
$(\lambda k.P\langle Q, k\rangle)^\# \equiv P^\# Q^\#$,
$(\lambda k.P\langle \exists^*, k\rangle)^\# \equiv P^\# \bullet^*$.

Lemma 4.13 (1) For any $\to\forall$-type A, $A^\bullet$ is a continuation type, and $A^{\bullet\circ} \equiv A$ holds.

(2) For any MWDF-F-term M, $[\![M]\!]$ is a CPS term, and $[\![M]\!]^\# \equiv M$ holds.

Proof. (1) By induction on A.

(2) By induction on M. □

Proposition 4.14 (1) If $\neg\Gamma \vdash_{\mathsf{cps}} P : \neg\mathcal{A}$ holds, then $\Gamma^\circ \vdash_{\mathsf{MWDF}\text{-}F} P^\# : \mathcal{A}^\circ$ holds.

(2) If $\neg\Gamma^\bullet \vdash_{\mathsf{cps}} [\![M]\!] : \neg A^\bullet$, then $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$ holds.

Proof. (1) By induction on the derivation of $\neg\Gamma \vdash_{\mathsf{cps}} P : \neg\mathcal{A}$.

(2) By (1), we have $\Gamma^{\bullet\circ} \vdash_{\mathsf{MWDF}\text{-}F} [\![M]\!]^\# : A^{\bullet\circ}$. By Lemma 4.3, we have the claim. □

In order to prove the conservativeness of M-$\lambda^{\neg\wedge\exists}$ over M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$, that is, $\neg\Gamma^\bullet \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^\bullet$ implies $\neg\Gamma^\bullet \vdash_{\mathsf{cps}} [\![M]\!] : \neg A^\bullet$, we define the contraction translation on types, which has been introduced in Nakazawa et al. (2008). A type derivation of a CPS term in $\lambda^{\neg\wedge\exists}$ may contain a non CPS type. For example, a CPS term $Q \equiv \lambda k'.xk'$ can have an arbitrary negation type $\neg\mathsf{A}$ under a context $\{x : \neg\mathsf{A}\}$, and then $P \equiv \lambda k.(\lambda x.Q(k\pi_2))(k\pi_1)$ has a type $\neg(\neg\mathsf{A} \wedge \mathsf{A})$ under the empty context. If A is not a continuation type, the type derivation of $P : \neg(\neg\mathsf{A} \wedge \mathsf{A})$ is not in M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. However, such a type A which is not a continuation type cannot be consumed in the type derivation of a CPS term, so we can replace A by a continuation type without changing the form of the derivation. The contraction translation formally realizes this replacement, and derivations in M-$\lambda^{\neg\wedge\exists}$ are translated by it to those in M-$\lambda^{\neg\wedge\exists}_{\mathsf{cps}}$. The contraction translation is defined as $(\cdot)^{\mathsf{c}}$ as follows, and we show that $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \mathsf{A}^{\mathsf{c}}$ holds for any CPS term P and any type derivation of $\Gamma \vdash_{\lambda^{\neg\wedge\exists}} P : \mathsf{A}$.

Definition 4.15 (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 and the auxiliary translation $(\cdot)^{\mathsf{d}}$ from $\neg \wedge \exists$-types to continuation types are defined by

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

Γ^{c} is defined as $\{(x : \mathsf{A}^{\mathsf{c}}) | (x : \mathsf{A}) \in \Gamma\}$.

The definition of A^{c} for non-negation type A is changed from Nakazawa et al. (2008) because of the following reason. The contraction translation is expected to have the following property: $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : \mathsf{A}$ implies $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \mathsf{A}^{\mathsf{c}}$ for any CPS term P. In the case of $P \equiv \lambda k.Q\langle \exists^*, k\rangle$, Q must have a type of the form $\neg\exists\overline{X}.\mathsf{C}$, and then P has a type $\neg\mathsf{C}[\overline{X} := \overline{\mathsf{B}}]$ for some $\overline{\mathsf{B}}$, so we must show that P has the type $(\neg\mathsf{C}[\overline{X} := \overline{\mathsf{B}}])^{\mathsf{c}} \equiv \neg(\mathsf{C}[\overline{X} := \overline{\mathsf{B}}])^{\mathsf{d}}$ from that Q has the type $(\neg\exists\overline{X}.\mathsf{C})^{\mathsf{c}} \equiv \neg\exists\overline{X}.\mathsf{C}^{\mathsf{d}}$. That requires the commutativity of the contraction and the substitution, that is, $(\mathsf{C}[\overline{X} := \overline{\mathsf{B}}])^{\mathsf{d}} \equiv \mathsf{C}^{\mathsf{d}}[\overline{X} := \overline{\mathsf{B}}^{\mathsf{d}}]$, but the original contraction does not have this property.

It should be noted that, in the case of $\mathsf{DF}\text{-}\lambda^{\neg\wedge\exists}$, we need only the restricted form of commutativity, that is $(\mathsf{C}[X := \mathcal{B}])^{\mathsf{d}} \equiv \mathsf{C}^{\mathsf{d}}[X := \mathcal{B}]$, because the CPS term corresponding to P above is $\lambda k.Q\langle \mathcal{B}, k\rangle$, where the type abstracted by the $\exists$-introduction is restricted to the continuation type $\mathcal{B}$. The restricted commutativity can be proved more easily, since any continuation type is not a negation.

Lemma 4.16 (1) For any continuation type $\mathcal{A}$, $(\neg\mathcal{A})^{\mathsf{c}} \equiv \neg\mathcal{A}$ and $\mathcal{A}^{\mathsf{d}} \equiv \mathcal{A}$ hold.

(2) For $\neg \wedge \exists$-types A and B, $(\mathsf{B}[X := \mathsf{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[X := \mathsf{A}^{\mathsf{d}}]$ and $(\mathsf{B}[X := \mathsf{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[X := \mathsf{A}^{\mathsf{d}}]$ hold.

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

(2) By induction on B. In the following, we write $\mathsf{B}[\mathsf{A}]$ for $\mathsf{B}[X := \mathsf{A}]$, so we will prove (i) $(\mathsf{B}[\mathsf{A}])^{\mathsf{c}} \equiv \mathsf{B}^{\mathsf{c}}[\mathsf{A}^{\mathsf{d}}]$ and (ii) $(\mathsf{B}[\mathsf{A}])^{\mathsf{d}} \equiv \mathsf{B}^{\mathsf{d}}[\mathsf{A}^{\mathsf{d}}]$.

(i) Case $\mathsf{B} \equiv X$. We have $(X[\mathsf{A}])^{\mathsf{c}} \equiv \mathsf{A}^{\mathsf{c}}$ and $X^{\mathsf{c}}[\mathsf{A}^{\mathsf{d}}] \equiv \neg\mathsf{A}^{\mathsf{d}}$. If A is not a negation, these are the same type by the definition. Otherwise, let $\mathsf{A} \equiv \neg\mathsf{C}$, and then we have $(\neg\mathsf{C})^{\mathsf{c}} \equiv \neg\mathsf{C}^{\mathsf{d}}$ and $\neg(\neg\mathsf{C})^{\mathsf{d}} \equiv \neg\mathsf{C}^{\mathsf{d}}$.

Other cases are proved by (ii).

(ii) Case $\mathsf{B} \equiv X$. In this case, the both sides are A^{d}.

Case $\mathsf{B} \equiv Y$ ($Y \not\equiv X$). The both sides are Y.

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

Case $\mathsf{B} \equiv \mathsf{C} \wedge \mathsf{D}$. If C is a negation, $\mathsf{C}[\mathsf{A}]$ is a negation, so this case is easily proved by the induction hypothesis. If $\mathsf{C}[\mathsf{A}]$ is not a negation, C is not a negation either, so this case is also easily proved by the induction hypothesis. The rest is the case where C is not a negation and $\mathsf{C}[\mathsf{A}]$ is a negation, that is, the case of $\mathsf{C} \equiv X$ and $\mathsf{A} \equiv \neg\mathsf{E}$. Then we have $(X[\neg\mathsf{E}] \wedge \mathsf{B}[\neg\mathsf{E}])^{\mathsf{d}} \equiv \neg\mathsf{E}^{\mathsf{d}} \wedge (\mathsf{B}[\neg\mathsf{E}])^{\mathsf{d}}$, which is identical to $\neg\mathsf{E}^{\mathsf{d}} \wedge \mathsf{B}^{\mathsf{d}}[\mathsf{E}^{\mathsf{d}}]$ by the induction hypothesis. On the other hand, we have $(X \wedge \mathsf{B})^{\mathsf{d}}[(\neg\mathsf{E})^{\mathsf{d}}] \equiv (\neg X \wedge \mathsf{B}^{\mathsf{d}})[\mathsf{E}^{\mathsf{d}}]$, so the both sides are the same type.

Case $\mathsf{B} \equiv \exists Y.\mathsf{C}$. This case is proved by the induction hypothesis. □

Lemma 4.17 For any CPS term P, $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : A$ 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 $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ implies $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\mathsf{A}^{\mathsf{d}}$.

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

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

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

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 $\mathsf{B} \wedge \mathsf{C}$.

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

where $\Gamma' = \Gamma, k : \mathsf{A}$. 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 $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form.

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

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 $P \equiv \lambda k.k[k'.Qk']$. Any derivation of $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form, where A must be $\exists\overline{X}.\mathsf{B}$, and Γ must not contain any type variable in $\overline{X}$ freely.

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

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

Case $P \equiv \lambda k.Q\langle \exists^*, k\rangle$. Any derivation of $\Gamma \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} P : \neg\mathsf{A}$ has the following form for some list of $\neg \wedge \exists$-types $\overline{\mathsf{B}}$, where A must be $\mathsf{C}[\overline{X} := \overline{\mathsf{B}}]$.

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

By the induction hypothesis, $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} Q : \neg\exists\overline{X}.\mathsf{C}^{\mathsf{d}}$ holds, so we have $\Gamma^{\mathsf{c}} \vdash_{\mathsf{cps}} P : \neg\mathsf{C}^{\mathsf{d}}[\overline{X} := \overline{\mathsf{B}}^{\mathsf{d}}]$ by letting $k : \mathsf{C}^{\mathsf{d}}[X := \overline{\mathsf{B}}^{\mathsf{d}}]$, where $\mathsf{C}^{\mathsf{d}}[X := \overline{\mathsf{B}}^{\mathsf{d}}]$ is identical to $(\mathsf{C}[\overline{X} := \overline{\mathsf{B}}])^{\mathsf{d}}$ by Lemma 4.16 (2). □

By Lemma 4.17, we can reduce TC and TI of MWDF-F to those of M-$\lambda^{\neg\wedge\exists}$, and then conclude undecidability of TC and TI in M-$\lambda^{\neg\wedge\exists}$.

Proposition 4.18 (1) $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$ holds if and only if $\neg\Gamma^{\bullet} \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^{\bullet}$ holds.

(2) $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$ holds for some Γ and A if and only if $\Gamma' \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \mathsf{A}'$ holds for some Γ' and A'.

Proof. (1) The only-if part is Proposition 4.11, so we will show the if part. If $\neg\Gamma^{\bullet} \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \neg A^{\bullet}$ holds, by Lemma 4.17, 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 4.16 (1). By Proposition 4.14 (2), $\Gamma \vdash_{\mathsf{MWDF}\text{-}F} M : A$ holds.

(2) The only-if part follows from the only-if part of (1). For the if part, suppose $\Gamma' \vdash_{\mathsf{M}\text{-}\lambda^{\neg\wedge\exists}} [\![M]\!] : \mathsf{A}'$ holds for some Γ' and A'. By Lemma 4.17, we have $\Gamma'^{\mathsf{c}} \vdash_{\mathsf{cps}} [\![M]\!] : \mathsf{A}'^{\mathsf{c}}$, so $[\![M]\!]^{\#}$ is typable by Proposition 4.14 (2), where $[\![M]\!]^{\#}$ is identical to M by (2). □

Theorem 4.19 Type inference is undecidable in M-$\lambda^{\neg\wedge\exists}$.

Proof. By Proposition 4.9 and 4.18 (2). □

Proof. (Another proof of Theorem 4.6.) By Proposition 4.9 and 4.18 (1). □

5 TC and TI in Curry-Style $\lambda^{\neg\wedge\exists}$

The negation, conjunction, and existence fragment Curry-$\lambda^{\neg\wedge\exists}$ in the Curry style is defined as follows. Curry-$\lambda^{\neg\wedge\exists}$ is the same as the system introduced in Tatsuta et al. (2008), and a subsystem of the system in Tatsuta (2007).

Definition 5.1 (Curry-$\lambda^{\neg\wedge\exists}$) (1) The types of Curry-$\lambda^{\neg\wedge\exists}$ are $\neg\wedge\exists$-types, and the terms (denoted by M, N,...) of Curry-$\lambda^{\neg\wedge\exists}$ are defined by

$$M ::= x \mid \lambda x.M \mid \langle M, M\rangle \mid \langle \exists, M\rangle \\ \mid MM \mid M\pi_1 \mid M\pi_2 \mid M[x.M],$$

In the term $\lambda x.M$, the variable x is bound in M. In the term $N[x.M]$, the variable x is bound in M.

(2) The typing rules of Curry-$\lambda^{\neg\wedge\exists}$ are the same as those of DF-$\lambda^{\neg\wedge\exists}$ except for the rules of existence, which are the following.

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

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

In the rule ($\exists$E), Γ_2 and C must not contain X freely.

The proof method in the previous section for M-$\lambda^{\neg\wedge\exists}$ can be adopted to Curry-$\lambda^{\neg\wedge\exists}$, and we can prove that undecidability of TC and TI in Curry-$\lambda^{\neg\wedge\exists}$ can be reduced to undecidability of a variant of F. It should be noted that Curry-F is not suitable to be the source calculus of the CPS translation to Curry-$\lambda^{\neg\wedge\exists}$, because terms in Curry-$\lambda^{\neg\wedge\exists}$ does not contain any information of applications of ($\forall$I). So we define another variant of polymorphic lambda calculus, which we will call the weak-domain-free-style polymorphic lambda calculus WDF-F. The proof is almost the same as that in the previous section, so here we give the definition of WDF-F and the CPS translation only.

Definition 5.2 (WDF-F) (1) The types of WDF-F are the $\rightarrow\forall$-types. The terms of WDF-F are defined by

$M ::= x \mid \lambda x.M \mid \Lambda.M \mid MM \mid M\bullet$.

(2) The typing rules of WDF-F are the same as those of DF-F except for the rules of universal quantifiers.

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

Definition 5.3 (CPS Translation) The *CPS translation* from WDF-F-terms to Curry-$\lambda^{\neg\wedge\exists}$-terms is defined by

$[\![x]\!] \equiv \lambda k.xk$,
$[\![\lambda x.M]\!] \equiv \lambda k.(\lambda x.[\![M]\!](k\pi_2))(k\pi_1)$,
$[\![MN]\!] \equiv \lambda k.[\![M]\!]\langle [\![N]\!], k\rangle$,
$[\![\Lambda.M]\!] \equiv \lambda k.k[k'.[\![M]\!]k']$,
$[\![M\bullet]\!] \equiv \lambda k.[\![M]\!]\langle \exists, k\rangle$,

Theorem 5.4 (1) If type checking is undecidable in WDF-F, then type checking is undecidable in Curry-$\lambda^{\neg\wedge\exists}$.

(2) If type inference is undecidable in WDF-F, then type inference is undecidable in Curry-$\lambda^{\neg\wedge\exists}$.

6 Concluding Remarks

In this paper, we proved undecidability of the following problems: (1) type inference in DF-F and DF-$\lambda^{\neg\wedge\exists}$, (2) type checking and inference in MWDF-F and M-$\lambda^{\neg\wedge\exists}$. Moreover, we proved that undecidability of the type checking and inference in Curry-$\lambda^{\neg\wedge\exists}$ can be reduced to undecidability of those problems in WDF-F.

It is an important problem whether type checking and inference in Curry-$\lambda^{\neg\wedge\exists}$ are decidable or not. As it is proved in this paper, their undecidability directly follows from undecidability of those problems in WDF-F. Although WDF-F is similar to MWDF-F and DF-F, in which type checking and inference have been proved to be undecidable, we cannot adopt the existing proof methods to WDF-F. For example, the proofs in Wells (1999) and Section 4.1 of this paper use the undecidability of the semi-unification problem. It is essential for this approach that the terms in the systems do not contain any information of the number of use of the quantifier rules. We have to find another approach to prove undecidability of type checking and inference in Curry-$\lambda^{\neg\wedge\exists}$ and WDF-F, and it is future work.

Acknowledgments The authors would like to thank Professor James Noble and the anonymous referees for their helpful comments.

References

Barthe, G., & Sørensen, M.H. (2000), Domain-free pure type systems, *J. Functional Programming* 10:412–452.

Fujita, K. & Schubert, A. (2000), Partially Typed Terms between Church-Style and Curry-Style, In *International Conference IFIP TCS 2000*, LNCS 1872, pp. 505–520.

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

Girard, J.Y. (1972), Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur, Thèse de doctorat d'Etat, Université de Paris VII.

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

Kfoury, A.J., Tiuryn, J. & Urzyczyn, P. (1993), The Undecidability of the Semi-unification Problem, *Information and Computation* 102:83–101.

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

Nakazawa, K., Tatsuta, M., Kameyama, Y. & Nakano, H. (2008), 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.

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

Schubert, A. (1998), Second-order unification and type inference for Church-style polymorphism, In *the 25th Annual ACM Symposium on Principles of Programming Languages (POPL '98)*, pp.279–288.

Tatsuta, M. (2007), Simple saturated sets for disjunction and second-order existential quantification, In *Proceedings of 8th International Conference on Typed Lambda Calculi and Applications (TLCA 2007)*, LNCS 4583, pp. 366–380.

Tatsuta, M., Fujita, K., Hasegawa, R. & Nakano, H. (2008), Inhabitation of Existential Types is Decidable in Negation-Product Fragment, In *Proceedings of 2nd International Workshop on Classical Logic and Computation (CLC2008)*.

Wells, J.B. (1999), Typability and Type Checking in System F Are Equivalent and Undecidable, In *Annals of Pure and Applied Logic* 98:111–156.