

EXISTENTIAL TYPE SYSTEMS BETWEEN CHURCH AND CURRY STYLE

KEN-ETSU FUJITA AND ALEKSY SCHUBERT

Gunma University, Tenjin-cho 1-5-1, Kiryu 376-8515, Japan
e-mail address: fujita@cs.gunma-u.ac.jp

Institute of Informatics, The University of Warsaw, ul. Banacha 2, 02-097 Warsaw, Poland
e-mail address: alx@mimuw.edu.pl

ABSTRACT. We study type checking, typability, and type inference problems for type-free style and Curry style second-order existential systems where the type-free style differs from the Curry style in that the terms of the former contain information on where the existential quantifier elimination and introduction take place but omit the information on which types are involved. We show that all the problems are undecidable employing reduction of second-order unification in case of the type-free system and semiunification in case of the Curry style system. This provides a fine border between problems yielding to a reduction of second-order unification problem and the semiunification problem. In addition, we investigate the subject reduction property of the system in the Curry-style.

1. INTRODUCTION

The System F ($\lambda 2$) of Girard-Reynolds is a fundamental system for the study of polymorphism. Its polymorphic properties are seen as the theoretical basis for the functional programming languages such as ML and this resulted in thorough studies
5 e.g. [Boe85, Wel99, Sch98] and for the $\lambda 2$ subsystems [FS00, Mai90]. A system with 2nd order existential types provides a theoretical basis for abstract data types [MP88]. This ingredient is also present in languages based on ML, but has been less studied except for the domain-free style [NTKN08].

The polymorphic system has enough power to encode impredicatively other con-
10 nectives \wedge, \vee, \exists and so on in terms of \rightarrow, \forall [Pra65]. In this sense, the 2nd order existential type system λ^\exists [Fuj05] can be regarded as a subsystem of $\lambda 2$. On the

1998 ACM Subject Classification: F.4.1.

Key words and phrases: existential types, type checking, typability, undecidability, Curry style type system, Church style type system.

This paper is an extended version of the report from proceeding on TLCA'09 [eFS09].

This work was partly supported by the bilateral program (2008) between Japan Society for the Promotion of Science and Polish Academy of Sciences.

This work was partly supported by the Polish government grant no N N206 355836.

other hand, the existential system λ^\exists can serve, under CPS-translations, as a target calculus not only of $\lambda 2$ (2nd order intuitionistic logic) but also of $\lambda\mu$ -calculus (2nd order classical logic) of Parigot [Fuj05, Has06]. Moreover, a recent work [Fuj10] on an intimate connection between $\lambda 2$ and λ^\exists reveals duality not only on reduction correspondence but also on proof structures between the systems.

In this light, the expressiveness of $\lambda 2$ and λ^\exists is comparable. Still, the undecidability of the type related problems cannot be established as a corollary of the mentioned above translations. This is in principle due to the fact that the translation works smoothly for systems with adequate type information, e.g. domain-free style [BS00, NTKN08], while the additional or missing type information introduces complications which are not immediate to overcome.

In this paper, we study type checking, typability, and type inference problems of 2nd order existential type systems in the styles of type-free and Curry. First, we show that all of the type related problems are undecidable for 2nd order existential type system in the type-free style, which can be regarded as an intermediate system between Church style and Curry style. For this, 2nd order unification problem in the flat form is introduced, and this form is proved undecidable by the reduction from 2nd order unification of simple instances [Sch98]. Then, by the reduction from the flat forms, it is proved that all of the type related problems of the type-free system are undecidable.

Secondly, we show that type checking and typability problems of the system in the style of Curry are undecidable. By reductions from the semiunification problem [KTU90], the undecidability proofs are established for fragments: (\rightarrow, \exists) and (\neg, \wedge, \exists) . The first of these results answers the decidability issue for the type related problems the solutions of which were left open by Nakazawa et al. [NTKN08]. Moreover, we investigate the subject reduction property of the system in Curry-style.

The 2nd order unification is used to prove the undecidability of several type related problems for second-order calculi [Sch98, FS00, Pfe93, Boe85]. Similarly, the semiunification problem (SUP) is appropriate as a tool to prove the undecidability for other ones [Wel99, KTU93]. The systems in this paper elucidate the mutual relation between the two problems as applied to type related questions. We can conclude that the possibility to fix the number of arguments to a second-order construct plays the crucial part here. This possibility is, indeed, the only difference between the type-free system and the Curry-style system. In this light the semiunification problem can be viewed as a kind of 2nd order unification where the number and the shape of 2nd order variable arguments are not known.

The main advantage of the Curry-style type systems from the programmers' point of view is that a programmer is liberated from the burden of making the type annotations. The type-free system can be an interesting alternative here as the annotating tax is not high, but the programmer can decide where the polymorphism is employed and how much of the polymorphism is exploited. One more advantage of the type-free systems is that they correspond to the 2nd order unification which is more extensively studied than the semiunification problem.

System	TCP	TIP	TP
Curry- (\rightarrow, \exists)	No	No	?
TF- (\rightarrow, \exists)	No	No	No
DF- (\rightarrow, \exists)	no ¹		no ¹
Church- (\rightarrow, \exists)	?	?	?
Curry- (\neg, \wedge, \exists)	No	No	?
TF- (\neg, \wedge, \exists)	?	?	?
DF- (\neg, \wedge, \exists)	no ¹		no ¹
Church- (\neg, \wedge, \exists)	?	?	?

Figure 1: (Un)decidability of TCP, TIP, and TP for systems with existence
fig:undecidability

We summarise the results concerning the undecidability of the type related problems in type systems with existential quantifier in Figure 1. In the figure, DF means domain-free, TF means type-free, *no* and *No* mean undecidable and *yes* decidable. The results marked with superscript 1 are proved in [NTKN08] by the so-called CPS-translation. Remark that according to [KN09], the undecidability of TIP for DF- (\neg, \wedge, \exists) in the case of a closed term follows from that of the corresponding TP. The undecidability of TCP, TIP, TP for TF- (\neg, \wedge, \exists) and TP for Church- (\neg, \wedge, \exists) respectively can be deduced from those of $\lambda 2$ in the corresponding style by the CPS-translation. These problems will be discussed elsewhere. In this paper, we show that
10 *No* indeed means undecidable in the figure above.

2. SECOND ORDER EXISTENTIAL TYPE SYSTEMS $(\neg, \rightarrow, \wedge, \exists)$

2.1. Type-free systems. We consider a second order system consisting of \perp , \neg , \rightarrow , \wedge , and second order existential quantification \exists . The subsystem with \rightarrow and \exists is denoted by (\rightarrow, \exists) while the one with \perp , \neg , \wedge and \exists is by (\neg, \wedge, \exists) . The possible types of the full system are:

$$A ::= X \mid \perp \mid \neg A \mid (A \rightarrow A) \mid (A \wedge A) \mid \exists X.A$$

In the system, we infer judgements of the form $\Gamma \vdash M : A$ where Γ is an environment, M a term, and A is a type. The environments we deal with here are sets of pair of the form $x : A$ where x is a variable and A is a type. We write $\vdash M : A$ as a shorthand
15 for $\emptyset \vdash M : A$.

First, a system between Church and Curry styles is defined as follows, where the special symbol \exists marks the existence of a witness. We call this system *type-free system* as we erase here all the types from the terms, but we mark the locations where the types should occur in the corresponding Church-style terms.

$$M ::= x \mid (\lambda x.M) \mid (MM) \mid \langle M, M \rangle \mid \pi_i(M) \mid \langle \exists, M \rangle \mid (\mathbf{let} \langle \exists, x \rangle = M \mathbf{in} M)$$

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

$$\begin{array}{c}
\frac{\Gamma, x:A \vdash M : \perp}{\Gamma \vdash \lambda x.M : \neg A} (\neg I) \quad \frac{\Gamma \vdash M_1 : \neg A \quad \Gamma \vdash M_2 : A}{\Gamma \vdash M_1 M_2 : \perp} (\neg E) \\
\frac{\Gamma, x:A_1 \vdash M : A_2}{\Gamma \vdash \lambda x.M : A_1 \rightarrow A_2} (\rightarrow I) \quad \frac{\Gamma \vdash M_1 : A_1 \rightarrow A_2 \quad \Gamma \vdash M_2 : A_1}{\Gamma \vdash M_1 M_2 : A_2} (\rightarrow E) \\
\frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2 \rangle : A_1 \wedge A_2} (\wedge I) \quad \frac{\Gamma \vdash M : A_1 \wedge A_2}{\Gamma \vdash \pi_i(M) : A_i} (\wedge E) \\
\frac{\Gamma \vdash M : A[X := A_1]}{\Gamma \vdash \langle \exists, M \rangle : \exists X.A} (\exists I) \quad \frac{\Gamma \vdash M_1 : \exists X.A \quad \Gamma, x:A \vdash M_2 : A_1}{\Gamma \vdash \mathbf{let} \langle \exists, x \rangle = M_1 \mathbf{in} M_2 : A_1} (\exists E)^*
\end{array}$$

where $(\exists E)^*$ denotes the eigenvariable condition $X \notin \text{FV}(\Gamma, A_1)$ and Γ are contexts which are sets of pairs $x : A$ that assign types to object variables.

In case of the rules $(\neg E)$, $(\rightarrow E)$, and $(\exists E)^*$, we use the term *major premise* to denote the premise with the type $\neg A$, $A_1 \rightarrow A_2$, and $\exists X.A$, respectively. The other
5 premise is called *minor premise*.

Example 1. ex:derivation-to-exists Here is an example of a derivation in the type-free system (\rightarrow, \exists) for a closed term $\lambda x.x(\mathbf{let} \langle \exists, y \rangle = \langle \exists, x \rangle \mathbf{in} \langle \exists, y \rangle)$. Let $A = (\exists X.X) \rightarrow (\exists X.X)$. We can derive

$$\frac{\overline{x : A \vdash x : A}^{(var)}}{x : A \vdash \langle \exists, x \rangle : \exists X.X} (\exists I) \tag{2.1}$$

and also

$$\frac{\overline{x : A, y : Y \vdash y : Y}^{(var)}}{x : A, y : Y \vdash \langle \exists, y \rangle : \exists X.X} (\exists I) \tag{2.2}$$

These derivations can be used in the following derivation around places marked (2.1) and (2.2) respectively.

$$\frac{\overline{x : A \vdash x : A}^{(var)} \quad \frac{\frac{\overline{x : A \vdash \langle \exists, x \rangle : \exists X.X}^{(2.1)} \quad \overline{x : A, y : Y \vdash \langle \exists, y \rangle : \exists X.X}^{(2.2)}}{x : A \vdash \mathbf{let} \langle \exists, y \rangle = \langle \exists, x \rangle \mathbf{in} \langle \exists, y \rangle : \exists X.X} (\exists E)}{x : A \vdash x(\mathbf{let} \langle \exists, y \rangle = \langle \exists, x \rangle \mathbf{in} \langle \exists, y \rangle) : \exists X.X} (\rightarrow E)}{\vdash \lambda x.x(\mathbf{let} \langle \exists, y \rangle = \langle \exists, x \rangle \mathbf{in} \langle \exists, y \rangle) : A \rightarrow \exists X.X} (\rightarrow I)$$

Note that the term is a type-free counterpart of the Curry-style term $\lambda x.xx$ (see Section 2.2 for an appropriate forgetful mapping).

As a curiosity for a reader who is acquainted with the System F in Church version we provide here a term

$$\lambda x : \forall X.X.(xA'x)A', \tag{2.3}$$

where $A' = (\forall X.X) \rightarrow (\forall X.X)$, which manifests various symmetries with the term in Example 1. Note that the term is closed and its type is $(\forall X.X) \rightarrow A'$. We leave
10 it for the reader to enjoy it.

We use the term above to construct an interesting example of a derivation in the system (\neg, \wedge, \exists) . The term considered below is motivated from the term (2.3) using the CPS translation from [NTKN08, Def. 3] and [Fuj10].

Example 2. Here is an example of a derivation in the type-free system (\neg, \wedge, \exists) for a closed term

$$M \equiv \lambda k. \pi_1(k) \langle \exists, \langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle \rangle$$

Let us define the types $A_\exists = (\exists X.X)$, $B_\exists = \neg A_\exists \wedge A_\exists$, $C_\exists = \neg A_\exists \wedge B_\exists$. Consider now derivations of subterms of M . We start with a derivation for $\pi_1(k)$:

$$\frac{\overline{k : \neg A_\exists \wedge B_\exists \vdash k : \neg A_\exists \wedge B_\exists} \text{ (var)}}{k : C_\exists \vdash \pi_1(k) : \neg A_\exists} \text{ (\wedge E)} \quad (2.4)$$

The derivation above helps in providing of the following derivation for the first component of $\langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle$:

$$\frac{\frac{\frac{\overline{k : \neg A_\exists \wedge B_\exists \vdash k : \neg A_\exists \wedge B_\exists} \text{ (\wedge E)}}{k : C_\exists \vdash \pi_2(k) : B_\exists} \text{ (\exists I)}}{k : C_\exists \vdash \langle \exists, \pi_2(k) \rangle : \exists X.X} \text{ (\wedge I)}}{k : C_\exists \vdash \langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle : \neg A_\exists \wedge A_\exists} \text{ (\exists I)} \quad (2.5)$$

Now, we are ready to compose the full derivation for M :

$$\frac{\frac{\frac{\overline{k : \neg A_\exists \wedge B_\exists \vdash \pi_1(k) : \neg A_\exists} \text{ (2.4)}}{k : C_\exists \vdash \langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle : \neg A_\exists \wedge A_\exists} \text{ (\exists I)}}{k : C_\exists \vdash \langle \exists, \langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle \rangle : A_\exists} \text{ (\exists I)}}{k : \neg A_\exists \wedge B_\exists \vdash \pi_1(k) \langle \exists, \langle \pi_1(k), \langle \exists, \pi_2(k) \rangle \rangle \rangle : \perp} \text{ (\neg I)}}{\vdash M : \neg(\neg A_\exists \wedge (\neg A_\exists \wedge A_\exists))} \text{ (\neg I)}$$

The term derived here becomes $\lambda k. \pi_1(k) \langle \pi_1(k), \pi_2(k) \rangle$ under a forgetful mapping, which can be regarded as a counterpart of the Curry-style term $\lambda x. xx$.

It is worth stating explicitly that the type-free systems are syntax directed in the following sense.

5 **Proposition 2.1. (Syntax directed system)**

In the systems TF- (\rightarrow, \exists) and TF- (\neg, \wedge, \exists) , whenever $\Gamma \vdash M : A$ is derivable then

- if $M \equiv x$ then the last rule in the derivation is (var) ,
- if $M \equiv \lambda x. M_1$ and the system is TF- (\rightarrow, \exists) then the last rule in the derivation is $(\rightarrow I)$,
- 10 • if $M \equiv M_1 M_2$ and the system is TF- (\rightarrow, \exists) then the last rule in the derivation is $(\rightarrow E)$,
- if $M \equiv \lambda x. M_1$ and the system is TF- (\neg, \wedge, \exists) then the last rule in the derivation is $(\neg I)$,
- if $M \equiv M_1 M_2$ and the system is TF- (\wedge, \neg, \exists) then the last rule in the derivation is $(\neg E)$,
- 15 • if $M \equiv \langle M_1, M_2 \rangle$ then the last rule in the derivation is $(\wedge I)$,
- if $M \equiv \pi_i(M_1)$ then the last rule in the derivation is $(\wedge E)$,
- if $M \equiv \langle \exists, M_1 \rangle$ then the last rule in the derivation is $(\exists I)$, and
- if $M \equiv (\text{let } \langle \exists, x \rangle = M_1 \text{ in } M_2)$ then the last rule in the derivation is $(\exists E)$.

Proof. The proof is by routine analysis of all the possible forms of terms. We leave it to interested readers. \square

We use the proposition in the paper by referring only by recalling that the systems are syntax directed. From the syntax directed property, the type-free system enjoys the subject reduction property. Here, the reduction rules are defined as usual, and we write \rightarrow^* for the reflexive transitive closure of the compatible closure of \rightarrow .

- (1) $(\lambda x.M_1)M_2 \rightarrow M_1[x := M_2]$
- (2) $\pi_i \langle M_1, M_2 \rangle \rightarrow M_i$ ($i = 1, 2$)
- (3) **let** $\langle \exists, x \rangle = \langle \exists, M_1 \rangle$ **in** $M_2 \rightarrow M_2[x := M_1]$
- (4) $\lambda x.Mx \rightarrow M$ if $x \notin \text{FV}(M)$

Proposition 2.2. (Subject reduction property)

If $M_1 \rightarrow^* M_2$ and $\Gamma \vdash M_1 : A$, then $\Gamma \vdash M_2 : A$.

Proof. By induction on the derivation of $M_1 \rightarrow^* M_2$. \square

2.2. Curry style systems. `sec:curry-style-systems`

In definition of the Curry style system we follow [SU06]. The terms are generated using this grammar:

$$M ::= x \mid (\lambda x.M) \mid (MM) \mid \langle M, M \rangle \mid \pi_i(M)$$

The type rules which differ from the ones of the type-free system are

$$\frac{\Gamma \vdash M : A[X := A_1]}{\Gamma \vdash M : \exists X.A} (\exists I) \quad \frac{\Gamma \vdash M_1 : \exists X.A \quad \Gamma, x:A \vdash M_2 : A_1}{\Gamma \vdash M_2[x := M_1] : A_1} (\exists E)^*$$

where $(\exists E)^*$ denotes the eigenvariable condition $X \notin \text{FV}(\Gamma, A_1)$.

As usual, a forgetful mapping can be defined from the set of the type-free terms to the terms in the Curry style:

- $\|x\| = x$,
- $\|\lambda x.M\| = \lambda x.\|M\|$,
- $\|M_1M_2\| = \|M_1\|\|M_2\|$,
- $\|\langle M_1, M_2 \rangle\| = \langle \|M_1\|, \|M_2\| \rangle$,
- $\|\pi_i(M)\| = \pi_i(\|M\|)$,
- $\|\langle \exists, M \rangle\| = \|M\|$,
- $\|\mathbf{let} \langle \exists, x \rangle = M_1 \mathbf{in} M_2\| = \|M_2\|[x := \|M_1\|]$

for type-free terms M, M_1, M_2 . Then well-typed Curry terms can be lifted to well-typed type-free terms as in [Bar93].

2.3. Type related problems. The interest in type related problems comes from the studies on development of functional programming languages and partially from theorem proving. The most basic of them is the problem of type checking:

Definition 2.3. (type checking problem)

- 5 The *type checking problem* (TCP) is a problem: given a term M , a type A , and a context Γ , is the judgement $\Gamma \vdash M : A$ derivable?

This problem arises when a programming language imposes the discipline that whenever a function is defined it must be supplied with a type that describes its most basic properties. It is also relevant for applications such as theorem proving where the user provides all the three elements: the context, which is the library of theorems, the type, which is the formula to be proved, and the term which is the term that supposedly proves the formula.

Most of the functional languages, however, prefer to relieve the programmers from the burden of supplying the type of the function she or he defines. Therefore, a strongly typed language infrastructure must be able to cope with the following problem:

Definition 2.4. (type inference problem)

The *type inference problem* (TIP) is a problem: given a term M and a context Γ , is there a type A such that $\Gamma \vdash M : A$ is derivable?

20 Finally, a typability problem arises when one tries to discover wrong assumptions the programmer made on the library she or he uses:

Definition 2.5. (typability problem)

The *typability problem* (TP) is a problem: given a term M , are there a context Γ and a type A such that $\Gamma \vdash M : A$ is derivable?

25

3. UNDECIDABILITY OF RESTRICTED UNIFICATION

The proof of undecidability of type related problems for type-free system is based on the reduction of a version of the second-order unification problem to the problems. We present here the second-order unification we are interested in.

Based on the syntax of types, we define expressions for unification problems. Here, the countably infinite set of type variables is separated into two sets of variables: One is for first-order variables denoted by X, Y and another is for constants denoted by C .

An instance of the unification problem consists of a set of equations $E = \{A_1 \doteq B_1, \dots, A_k \doteq B_k\}$. We say that the instance E is solvable if there exists a substitution S such that $S(A_1) = S(B_1), \dots, S(A_k) = S(B_k)$. We write $\text{Dom}(S)$ for the domain of S .

Expressions for the *first-order unification problems* are defined from first-order variables and constants, together with \rightarrow

$$A, B ::= X \mid C \mid (A \rightarrow B).$$

Expressions for *simple instances* of the second order unification problem are defined using second order functional variables of a fixed arity denoted by F, G and terms built of constants and \rightarrow only, e.g. $FC_1 \cdots C_n$ for F of arity n . Here, free variables in expressions of unification problems are defined as follows:

$$\begin{aligned} \text{FV}_u(C) &= \emptyset; & \text{FV}_u(X) &= \{X\}; \\ \text{FV}_u(F) &= \{F\}; & \text{FV}_u(A \rightarrow B) &= \text{FV}_u(A) \cup \text{FV}_u(B). \end{aligned}$$

Definition 3.1. (Simple instances E_s)

def:simple-instance We consider the following set E_s of only two equations of simple instances, which is enough for the undecidability [Sch98].

$$\left\{ \begin{array}{l} FA_1 \dots A_n \doteq (FA'_1 \dots A'_n) \rightarrow A'_0, \\ GB_1 \dots B_m \doteq (GB'_1 \dots B'_m) \rightarrow (FA''_1 \dots A''_n) \end{array} \right\}$$

where F has the arity n and G has the arity m with $m, n \geq 1$ and free variables fulfil $\text{FV}_u(A_i, A'_i, A''_i, B_j, B'_j) = \emptyset$ for each i, j .

Theorem 3.2 (Undecidability for simple instances [Sch98]). **theorem:undecidability-simple** *The second order unification for simple instances is undecidable.*

- 5 The equations can be transformed so that the second-order variables occur only as topmost symbols (at the root of the tree they correspond to). The simple instances of Definition 3.1 can be reduced to the following equations E_{sr} :

Definition 3.3. (Simple instances with root restriction E_{sr})

def:simple-root

$$\begin{array}{ll} FA_1 \dots A_n \doteq X_{FA'_1 \dots A'_n} \rightarrow A'_0 & FA'_1 \dots A'_n \doteq X_{FA'_1 \dots A'_n} \\ FA_1 \dots A_n \doteq X_{FA''_1 \dots A''_n} & \\ GB_1 \dots B_m \doteq Y_{GB'_1 \dots B'_m} \rightarrow X_{FA''_1 \dots A''_n} & GB'_1 \dots B'_m \doteq Y_{GB'_1 \dots B'_m} \end{array}$$

We immediately have the following proposition:

Proposition 3.4. (Root restriction)

- 10 **prop:root-restriction** *The unification of simple instances is undecidable iff it is undecidable for instances defined in Definition 3.3.*

Proof. The proof is left to the interested readers. □

- The root-restricted instances can be transformed further to the next restricted form. We say that a set of unification equations E is in the *flat form* if it obeys all the restrictions as follows:

Definition 3.5. (Flat form)

def:flat-form

- (1) *Root restriction:* Second-order variables occur only at root positions.

- (2) *Monadic restriction*: If second-order variable occurs as $\mathsf{F}A_1 \cdots A_n$ then either all A_i are constants or all A_i are first-order variables. Moreover, the symbols A_i occur only in the equation where $\mathsf{F}A_1 \cdots A_n$ occurs.
- (3) *Constant restriction*: Each time a second-order variable F is applied to a vector X_1, \dots, X_n of first-order variables as $\mathsf{F}X_1 \cdots X_n \doteq A$, there is a set of pairwise distinct constants C_1, \dots, C_n such that there is exactly one equation $\mathsf{F}C_1 \cdots C_n \doteq B \in E$, where C_1, \dots, C_n occur, all C_1, \dots, C_n occur in B , and the positions where the constants occur in B exist in A .

The first equation of the simple instances with the root restriction can be transformed to the following flat form E_{sf} :

Definition 3.6. (Simple instances in flat form E_{sf} and E_{SF})
def:simple-flat

$$\begin{aligned} X_{A_1 \dots A_n} &\doteq X_{\mathsf{F}'A_1 \dots A_n} \rightarrow A'_0 \\ \mathsf{F}'X_1 \dots X_n &\doteq X_{A_1 \dots A_n} \rightarrow A_1 \rightarrow \dots \rightarrow A_n \rightarrow o \\ \mathsf{F}'C_1 \dots C_n &\doteq X'_{A_1 \dots A_n} \rightarrow C_1 \rightarrow \dots \rightarrow C_n \rightarrow o \end{aligned}$$

where F' is a fresh second order variable of arity n ; the symbols $X_i, X_{A_1 \dots A_n}, X'_{A_1 \dots A_n}, X_{\mathsf{F}'A_1 \dots A_n}$ are fresh first-order variables; C_1, \dots, C_n are subject to the constant restriction; and o is a distinguished constant.

In this way, all the equations in E_{sr} of Definition 3.3 can be transformed to the flat form. Let us call E_{SF} the set with all the results of the transformation.

Lemma 3.7. (Simple and flat)

lem:root-flat *The equations in E_{sr} from Definition 3.3 are solvable if and only if the equations in E_{SF} from Definition 3.6 are solvable.*

Proof. We here show only that the first equation in E_{sr} is solvable if and only if the equations in the corresponding E_{sf} are solvable. The other equations of the simple instances with the root restriction can be reduced to the flat form in a similar way.

(\Rightarrow) Suppose that S is a substitution such that $S(\mathsf{F})A_1 \dots A_n = S(X_{\mathsf{F}'A_1 \dots A_n}) \rightarrow A'_0$. Then we can define a substitution S' which solves the equations in Definition 3.6 as follows:

$$\begin{aligned} S'(\mathsf{F}')Z_1 \dots Z_n &= (S(\mathsf{F})Z_1 \dots Z_n) \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o; \\ S'(X_{A_1 \dots A_n}) &= S(\mathsf{F})A_1 \dots A_n; \\ S'(X'_{A_1 \dots A_n}) &= S(\mathsf{F})C_1 \dots C_n; \\ S'(X_i) &= A_i; \text{ and} \\ S'(X_{\mathsf{F}'A_1 \dots A_n}) &= S(X_{\mathsf{F}'A_1 \dots A_n}). \end{aligned}$$

Now all the equations in Definition 3.6 are solvable. Note also that the definition of S' is universal and works the same for all equations from Definition 3.3.

(\Leftarrow) Suppose that the equations in Definition 3.6 are solvable under S . As the constants C_1, \dots, C_n are different than subtypes A_1, \dots, A_n , the substitution S has the following form: $S(\mathsf{F}')Z_1 \dots Z_n = (A \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o)$ for some A such that $S(X_{A_1 \dots A_n}) = A[Z_1 := A_1, \dots, Z_n := A_n]$ and $S(X'_{A_1 \dots A_n}) = A[Z_1 := C_1, \dots, Z_n :=$

$C_n]$. Then define a substitution S' for the first equation in Definition 3.3 as follows: $S'(\mathbf{F})Z_1 \dots Z_n = A$ and $S'(X_{\mathbf{F}A'_1 \dots A'_n}) = S(X_{\mathbf{F}'A_1 \dots A_n})$. Now the first equation in Definition 3.3 is solvable. Note that the construction is universal with regard to the equations from Definition 3.3 and gives the same result for \mathbf{F} independent of the equation that is taken into account in the procedure above. \square

Proposition 3.8. (Reduction from E_s to E_{SF})

prop:simple-flat *The simple instances of E_s are solvable if and only if the simple instances in the flat form E_{SF} are solvable.*

Proof. The equations E_s can be reduced to E_{sr} by Proposition 3.4 and then to E_{SF} by Lemma 3.7. \square

4. UNDECIDABILITY OF TYPE RELATED PROBLEMS FOR TYPE-FREE SYSTEM

Now, we embark on the reduction of the unification of equations in the flat form to the type related problems. We provide for a set of equations E , a λ -term M such that if a type derivation for M exists then a unifier S for E can be separated from it. The main idea of the construction is to ensure that the shape of a type for a variable x_A occurring in M , which corresponds to a subexpression A in E strictly corresponds to $S(A)$.

Since we have a countably infinite set of term variables of λ -calculus, we can assume one-to-one mappings between expressions of unification problems and term variables of λ -terms. Based on this, we write term variables x_A and y_A corresponding to an expression A of unification problem. For instance, we have term variables x_X and y_X from the first order variable X , and similarly term variables x_C and y_C from the constant C . In particular, from the distinguished constant o we have a term variable x_o .

We simply write $A^n \rightarrow B$ for $A \rightarrow \dots \rightarrow A \rightarrow B$ with n -occurrences of A , and MN^n for $MN \dots N$ with n -occurrences of N . For a context Γ , an updated context $\Gamma(x:A)$ is defined as usual, such that $\Gamma(x:A)(y) = A$ if $y \equiv x$, otherwise $\Gamma(x:A)(y) = \Gamma(y)$.

We define a context $\Gamma_o = \{x_o : o, y_{o_1} : o \rightarrow o, \dots, y_{o_n} : o^n \rightarrow o\}$ for a fixed $n \geq 16$, where y_{o_i} are fresh term variables and o is the distinguished constant. We often shorten y_{o_i} to y_o when this does not lead to confusion.

Definition 4.1. (Encoding of first order expressions)

def:first-encoding For an expression A of the first order unification problem, we define a λ -term M_A , as follows:

(1) Case A of X (first order variable):

$$M_X \equiv y_{o_1}(y_X x_X)$$

(2) Case A of C (constant):

$$M_C \equiv y_{o_1}(y_C x_C)$$

(3) Case A of $(A_1 \rightarrow A_2)$:

$$M_{A_1 \rightarrow A_2} \equiv y_{o_4}(y_{A_2}(x_{A_1 \rightarrow A_2} x_{A_1}))(y_{A_2} x_{A_2}) M_{A_1} M_{A_2}.$$

Note that $\text{FV}(M_A) \subseteq \text{Dom}(\Gamma_o) \cup \{x_B, y_B \mid B \text{ is subexpression of } A\}$.

The encoding in Definition 4.1 is constructed in such a way that the types of variables x_A follow the structure of the corresponding expressions A . In addition, the encoding gives enough freedom to enable the operation of first-order substitutions.

5 This is precisely expressed by the following lemma.

Lemma 4.2. (First-order equations)

lem:basic-first

- (1) For any first-order expression A , there exists a context $\Gamma \supseteq \Gamma_o$ such that $\Gamma \vdash M_A : o$ and a substitution S_Γ such that $\Gamma(x_B) = S_\Gamma(B)$ and $\Gamma(y_B) = S_\Gamma(B) \rightarrow o$ for any subtype B of A and $\text{FV}_u(A) \subseteq \text{Dom}(S_\Gamma)$.
- 10 (2) For any first-order expression A and a first-order substitution S such that $\text{FV}_u(A) \subseteq \text{Dom}(S)$, there exists a context $\Gamma_S \supseteq \Gamma_o$ such that $\Gamma_S \vdash M_A : o$ where $\Gamma_S(x_B) = S(B)$ and $\Gamma_S(y_B) = S_\Gamma(B) \rightarrow o$ for each subtype B of A .

Proof. We prove (1) and (2) together by induction on the structure of A . We present here the proofs for the case $A \equiv A_1 \rightarrow A_2$. The remaining cases are left to the reader.

- (1) For the proof of (1), we inductively construct a context Γ and a substitution S_Γ accordingly. We proceed only with demonstration that $\Gamma(x_A) = S_\Gamma(A)$ the condition with y_A is proved in the similar way.

Case $A \equiv A_1 \rightarrow A_2$: From the induction hypothesis (1), we have $\Gamma_1 \vdash M_{A_1} : o$ such that $\Gamma_1 \supseteq \Gamma_o$ and a substitution S_{Γ_1} such that $\Gamma_1(x_B) = S_{\Gamma_1}(B)$ for all subtypes B of A_1 and $\text{FV}_u(A_1) \subseteq \text{Dom}(S_{\Gamma_1})$. We obtain S'_{Γ_1} by adding to S_{Γ_1} the associations $(x_X : B_X)$ for all $X \in \text{FV}_u(A_2) \setminus \text{FV}_u(A_1)$ and arbitrary B_X . By the induction hypothesis (2) we obtain a context $\Gamma_2 \supseteq \Gamma_o$ such that $\Gamma_2 \vdash M_{A_2} : o$ and such that $\Gamma_2(x_B) = S'_{\Gamma_1}(B)$ for each subtype B of A_2 . Note that Γ_2 gives the same types as Γ_1 on variables that occur both in M_{A_1} and M_{A_2} since S_{Γ_1} coincides with S'_{Γ_1} on variables from $\text{FV}_u(A_1) \cap \text{FV}_u(A_2)$ and both Γ_1 and Γ_2 include Γ_o . Therefore, $\Gamma' = \Gamma_1 \cup \Gamma_2$ is well defined and as this is extension of Γ_i we obtain $\Gamma' \vdash M_{A_i} : o$ for $i = 1, 2$. Moreover, we have $S'_{\Gamma_1}(B) = \Gamma'(x_B)$ for all subtypes B of A_1 and A_2 . Now we define

$$\Gamma_{A_1 \rightarrow A_2} = \Gamma' \cup \{y_{A_2} : \Gamma'(x_{A_2}) \rightarrow o, x_{A_1 \rightarrow A_2} : \Gamma'(x_{A_1}) \rightarrow \Gamma'(x_{A_2})\}.$$

Finally, we establish $\Gamma_{A_1 \rightarrow A_2} \vdash M_{A_1 \rightarrow A_2} : o$. Moreover, we can obtain the remaining equality of types:

$$\begin{aligned} \Gamma_{A_1 \rightarrow A_2}(x_{A_1 \rightarrow A_2}) &= \Gamma'(x_{A_1}) \rightarrow \Gamma'(x_{A_2}) \\ &= S'_{\Gamma_1}(A_1) \rightarrow S'_{\Gamma_1}(A_2) = S'_{\Gamma_1}(A_1 \rightarrow A_2). \end{aligned}$$

Therefore, we can define $S_{\Gamma_{A_1 \rightarrow A_2}} = S_{\Gamma'}$.

- 20 (2) For the proof of (2) we proceed as follows.

Case $A \equiv A_1 \rightarrow A_2$: By induction hypothesis, we have $\Gamma_{S,i} \vdash M_{A_i} : o$ for $i = 1, 2$. Moreover, for each subtype B of both A_i we have $\Gamma_{S,i}(x_B) = S(B)$ and $\Gamma_{S,i}(y_B) = S(B) \rightarrow o$ respectively (*). W.l.o.g. we can assume that $\text{Dom}(\Gamma_{S,i}) = \text{FV}(M_{A_i}) \cup \text{Dom}(\Gamma_o)$. Now, we observe that $\text{FV}(M_B) = \{x_D, y_D \mid D \text{ is a subtype of } B\}$. Therefore, $\text{FV}(M_{A_1}) \cap \text{FV}(M_{A_2})$ contains only x_B and

y_B such that B is a subtype of both A_1 and A_2 and the variables from Γ_o . We can now derive $\Gamma_{S,1}(y_B) = S(B) \rightarrow o = \Gamma_{S,2}(y_B)$ (and similarly $\Gamma_{S,1}(x_B) = \Gamma_{S,2}(x_B)$) and obtain, therefore, that for all $z \in \text{FV}(M_{A_1}) \cap \text{FV}(M_{A_2})$ the equality $\Gamma_{S,1}(z) = \Gamma_{S,2}(z)$ holds. As the contexts coincide on the common part we can safely define $\Gamma_S = \Gamma_{S,1} \cup \Gamma_{S,2}(x_{A_1 \rightarrow A_2} : S(A_1 \rightarrow A_2))(y_{A_1 \rightarrow A_2} : S(A_1 \rightarrow A_2) \rightarrow o)$ such that $\Gamma_S \vdash M_{A_i} : o$ is derivable for $i = 1, 2$. A direct check confirms now that $\Gamma_S \vdash M_{A_1 \rightarrow A_2} : o$. $\Gamma_S(x_B) = S(B)$ and $\Gamma_S(y_B) = S(B) \rightarrow o$ for arbitrary subtype B of A we obtain using (*) above in case B is a proper subtype of A and directly from definition of S in case $B \equiv A$.

10

□

Definition 4.3. (Encoding of first order unification)

def:encoding1 Let E be a finite set of equations of first-order unification. In case $E = \emptyset$, we define $M_E = x_o$. In case $E = \{A_1 \doteq B_1\} \cup E_0$, we define M_E to be:

$$M_E \equiv y_{o_5} (y_{A_1} x_{A_1}) (y_{A_1} x_{B_1}) M_{A_1} M_{B_1} M_{E_0}.$$

In the definition above, we use M_{A_1}, M_{B_1} to encode the shape of the types A_1, B_1 respectively. This is done in such a way that x_{A_1}, x_{B_1} have the types $S(A_1), S(B_1)$ for some substitution S . We can now force them to be equal by placing x_{A_1}, x_{B_1} as arguments to the same variable y_{A_1} . The rest of the set of equations can be taken into account in the same fashion in the subterm M_{E_0} . Note that

$$\text{FV}(M_E) \subseteq \{x_B, y_B \mid B \text{ is subexpression of some expression in } E\} \cup \text{Dom}(\Gamma_o).$$

Lemma 4.4. (First-order unification and typechecking)

lem:1st-unif Let E be a finite set of equations of first-order unification and S a substitution. The substitution S solves E if and only if there is a context $\Gamma \supseteq \Gamma_o$ such that $\Gamma \vdash M_E : o$ is derivable and $\Gamma(x_B) = S(B)$, $\Gamma(y_B) = S(B) \rightarrow o$ for each subexpression B of expressions in E .

15

Proof. The proof is by induction on the size of E . The case $E = \emptyset$ holds trivially. Consider now the case $E = \{A_1 \doteq B_1\} \cup E_0$.

(\Rightarrow) A straightforward check with help of Lemma 4.2(2) to obtain derivations for M_{A_1} and M_{B_1} and the induction hypothesis to obtain one for M_E .

20

(\Leftarrow) Suppose that $\Gamma \vdash M_E : o$ for some context $\Gamma \supseteq \Gamma_o$ with $\Gamma(x_B) = S(B)$, $\Gamma(y_B) = S(B) \rightarrow o$. This means that $\Gamma \vdash M_{E_0} : o$ is derivable. The induction hypothesis gives immediately that S solves E_0 . We have also that $\Gamma \vdash M_{A_1} : o$ and $\Gamma \vdash M_{B_1} : o$ are derivable. As x_{A_1} and x_{B_1} are both applied to the same variable, we have $\Gamma(x_{A_1}) = \Gamma(x_{B_1})$. Then we can infer that $S(A_1) = \Gamma(x_{A_1}) = \Gamma(x_{B_1}) = S(B_1)$.

25

Hence all $A \doteq B \in E$ are solvable under S . □

We define a λ -term $\langle \exists^{n+1}, M \rangle \equiv \langle \exists, \langle \exists^n, M \rangle \rangle$ and $\langle \exists^1, M \rangle \equiv \langle \exists, M \rangle$, which means successive applications of $(\exists I)$. We also define a λ -term $(\text{let } \langle \exists^{n+1}, a \rangle = M \text{ in } N) \equiv (\text{let } \langle \exists, a_1 \rangle = M \text{ in } (\text{let } \langle \exists^n, a \rangle = a_1 \text{ in } N))$ and $(\text{let } \langle \exists^1, a \rangle = M \text{ in } N) \equiv (\text{let } \langle \exists, a \rangle = M \text{ in } N)$, which means successive applications of $(\exists E)$ where a_1 is fresh.

30

A reduction from E_s of Definition 3.1 to simple instances in the flat form E_{sf} provides totally 10 equations with second order variables, where two second order variables are used, to say F for four equations and G for the rest six equations. We define the following encoding for equations with F , and the case with G follows the same pattern.

Definition 4.5. (Encoding of second order unification)

def:encoding2 For a set of equations E such that the set $E \setminus E'$ consists of the equations in the flat form containing the second order variable F of arity n :

$$FX_1 \dots X_n \doteq B_1, \quad FC_1 \dots C_n \doteq B_2, \quad FY_1 \dots Y_n \doteq B_3, \quad FC'_1 \dots C'_n \doteq B_4,$$

where $B_1 \equiv (X \rightarrow A_1 \rightarrow \dots \rightarrow A_n \rightarrow o)$, $B_3 \equiv (Y \rightarrow A'_1 \rightarrow \dots \rightarrow A'_n \rightarrow o)$, $B_2 \equiv (X' \rightarrow C_1 \rightarrow \dots \rightarrow C_n \rightarrow o)$, and $B_4 \equiv (Y' \rightarrow C'_1 \rightarrow \dots \rightarrow C'_n \rightarrow o)$; we define a λ -term M_E as follows:

$$\begin{aligned} y_{o_{12}} & (y_F \langle \exists^n, x_{B_1} \rangle) (y_F \langle \exists^n, x_{B_2} \rangle) (y_F \langle \exists^n, x_{B_3} \rangle) (y_F \langle \exists^n, x_{B_4} \rangle) (y_F x_F) \\ & (y_F (\mathbf{let} \langle \exists^n, a \rangle = x_F \mathbf{in} \langle \exists^n, \lambda z z_1 \dots z_n. y_{o_{12}} (a z z_1 \dots z_n)^{12} \rangle)) \\ & M_{B_1} M_{B_2} M_{B_3} M_{B_4} M_{E'} (y_{o_{12}} x_o^{12}) \end{aligned} \quad (4.1)$$

where M_{B_1}, \dots, M_{B_4} are encodings of the first order expressions B_1, \dots, B_4 ; and $M_{E'}$ is an encoding of the set E' of equations.

Note that

$$\begin{aligned} \text{FV}(M_E) \subseteq & \{x_B, y_B \mid B \text{ is subexpression of some first-order expression in } E\} \cup \\ & \{x_F, y_F \mid F \text{ is a second-order variable in } E\} \cup \text{Dom}(\Gamma_o). \end{aligned}$$

In order to reduce the unification problem, additional care should be taken to make sure that the types of different constants are indeed different. This can be obtained with the help of the following construction.

From the definitions of the encoding, the set of free variables $\text{FV}(M_E)$ is a finite set. Let $\vec{x}_C = \{x_{C_1}, \dots, x_{C_m}\}$, where all the constants in E subject to the constant restriction are collected. Let $\vec{y} = \text{FV}(M_E) \setminus (\{x_o\} \cup \vec{x}_C)$, and \hat{M}_E be the following term:

$$\mathbf{let} \langle \exists, x_{C_1} \rangle = x_{\exists} \mathbf{in} \dots \mathbf{let} \langle \exists, x_{C_m} \rangle = x_{\exists} \mathbf{in} \mathbf{let} \langle \exists, x_o \rangle = x_{\exists} \mathbf{in} \langle \exists, \lambda \vec{y}. M_E \rangle$$

Proposition 4.6. (second-order unification)

prop:2nd-unif Let E be the equations in the flat form above. The problem E is solvable if and only if $x_{\exists} : \exists X. X \vdash \hat{M}_E : \exists X. X$.

Proof. (\Rightarrow) Suppose that E is solvable under S . Then, from the constant restriction on C_i, C'_i , the substitution $S(F)Z_1 \dots Z_n$ has the form of $(A \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o)$ for some A . On the one hand, from the solution S restricted to first order variables, there exists a context $\Gamma_S \supseteq \Gamma_o$ such that $\Gamma_S \vdash M_{B_i} : o$ together with $\Gamma_S(x_{B_i}) = S(B_i)$ for $i = 1, 2, 3, 4$. Let

$$\Gamma = \Gamma_S \cup \{y_F : (\exists Z_1 \dots Z_n. S(F)Z_1 \dots Z_n) \rightarrow o, x_F : \exists Z_1 \dots Z_n. S(F)Z_1 \dots Z_n\}.$$

From $\Gamma \vdash x_{B_1} : S(B_1)$ and $S(F)S(X_1) \dots S(X_n) = S(B_1)$, we have $\Gamma \vdash \langle \exists^n, x_{B_1} \rangle : \exists Z_1 \dots Z_n. S(F)Z_1 \dots Z_n$ by n -times application of $(\exists I)$. Similarly, we

also have $\Gamma \vdash \langle \exists^n, x_{B_2} \rangle : \exists Z_1 \dots Z_n. S(\mathbf{F})Z_1 \dots Z_n$. Hence, all of the terms $(y_{\mathbf{F}} \langle \exists^n, x_{B_i} \rangle)$ and $(y_{\mathbf{F}} x_{\mathbf{F}})$ are well-typed. Since

$\Gamma \vdash x_{\mathbf{F}} : \exists Z_1 \dots Z_n. S(\mathbf{F})Z_1 \dots Z_n$ and the form of $S(\mathbf{F})Z_1 \dots Z_n$ is $(A \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o)$ for some A , we have $\Gamma \vdash$

- 5 $(\mathbf{let} \langle \exists^n, a \rangle = x_{\mathbf{F}} \mathbf{in} \langle \exists^n, \lambda z z_1 \dots z_n. y_o(a z z_1 \dots z_n)^{12} \rangle) : \exists Z_1 \dots Z_n. S(\mathbf{F})Z_1 \dots Z_n$ by successive applications of $(\exists I)$ and $(\exists E)$. Thus, the term $(y_{\mathbf{F}}(\mathbf{let} \langle \exists^n, a \rangle = x_{\mathbf{F}} \mathbf{in} \langle \exists^n, \lambda z z_1 \dots z_n. y_o(a z z_1 \dots z_n)^{12} \rangle))$ is also well-typed. Hence, we have $\Gamma \vdash M_E : o$ and then $\Gamma \upharpoonright (\{x_o\} \cup \vec{x}_C) \vdash \langle \exists, \lambda \vec{y}. M_E \rangle : \exists X. X$, where the domain of Γ is restricted to $\{x_o\} \cup \vec{x}_C$. Therefore, from the constant restriction, we have
- 10 $x_{\exists} : \exists X. X \vdash \hat{M}_E : \exists X. X$ by successive applications of $(\exists E)$.

- (\Leftarrow) Suppose $x_{\exists} : \exists X. X \vdash \hat{M}_E : \exists X. X$. Then, from the eigenvariable condition, we have $x_{C_1} : C_1, \dots, x_{C_m} : C_m, x_o : o \vdash \langle \exists, \lambda \vec{y}. M_E \rangle : B'$ for some B' , where C_1, \dots, C_m, o are pairwise distinct type variables. Hence, from the coding of M_E , we also have
- 15 $\Gamma \vdash M_E : o$ for some $\Gamma \supseteq \Gamma_o \cup \{x_{C_1} : C_1, \dots, x_{C_m} : C_m, x_o : o\}$. Then for all $i = 1, 2, 3, 4$, we have $\Gamma \vdash M_{B_i} : o$ as well and $\Gamma(x_{B_i}) = S(B_i)$ for some substitution S restricted to first order variables.

- Since $(\mathbf{let} \langle \exists^n, a \rangle = x_{\mathbf{F}} \mathbf{in} \langle \exists^n, \lambda z z_1 \dots z_n. y_o(a z z_1 \dots z_n)^{12} \rangle)$ is well-typed by n -times application of $(\exists E)$ and $(\exists I)$, type of $x_{\mathbf{F}}$ has the form of $\exists Z_1 \dots Z_n. B$ for some
- 20 B , such that $\lambda z z_1 \dots z_n. y_o(a z z_1 \dots z_n)^{12}$ and a have the same type B in the form of $(A \rightarrow \square_1 \rightarrow \dots \rightarrow \square_n \rightarrow o)$ for some types A, \square_i .

- From $\Gamma \vdash x_{B_i} : S(B_i)$ for all $i = 1, 2, 3, 4$, we must derive the same type $\exists Z_1 \dots, Z_n. B$ such that $\Gamma \vdash \langle \exists^n, x_{B_i} \rangle : \exists Z_1 \dots, Z_n. B$ by n -times application of $(\exists I)$. Since C_i and C'_i are pairwise distinct, these variables (constants) are to be abstracted
- 25 one by one, and moreover the head parts of $S(B_i); S(X), S(X'), S(Y), S(Y')$ are also to be abstracted by $(\exists I)$. This means that $B \equiv (A \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o)$ with $S(X) = A[Z_1 := A_1, \dots, Z_n := A_n]; S(X') = A[Z_1 := C_1, \dots, Z_n := C_n]; S(Y) = A[Z_1 := A'_1, \dots, Z_n := A'_n]; S(Y') = A[Z_1 := C'_1, \dots, Z_n := C'_n]$. Now we can extend the substitution S to \mathbf{F} , such that $S(\mathbf{F})Z_1 \dots Z_n = (A \rightarrow Z_1 \rightarrow \dots \rightarrow Z_n \rightarrow o)$,
- 30 and that $S(X_i) = A_i$ and $S(Y_i) = A'_i$ for fresh first order variables X_i, Y_i . Then we establish that

$$\begin{aligned} S(\mathbf{F})S(X_1) \dots S(X_n) &= S(\mathbf{F})A_1 \dots A_n \\ &= A[Z_1 := A_1, \dots, Z_n := A_n] \rightarrow A_1 \rightarrow \dots \rightarrow A_n \rightarrow o \\ &= S(X) \rightarrow A_1 \rightarrow \dots \rightarrow A_n \rightarrow o = S(B_1); \text{ and} \\ 35 \quad S(\mathbf{F})C_1 \dots C_n &= A[Z_1 := C_1, \dots, Z_n := C_n] \rightarrow C_1 \rightarrow \dots \rightarrow C_n \rightarrow o \\ &= S(X') \rightarrow C_1 \rightarrow \dots \rightarrow C_n \rightarrow o = S(B_2). \end{aligned}$$

The other equations can be solved as well. Thus E is solvable. \square

Lemma 4.7. (Term which enforces \exists)

- lem:exists** Let $M_{\exists} = z_1(z_2 z_{\exists})(z_2 \langle \exists, \lambda x. x \rangle)(z_2 \langle \exists^2, z_3 \rangle)$, where $z_1, z_2, z_3, z_{\exists}$ are fresh
- 40 term variables. Then M_{\exists} is typable if and only if there are some Γ and A such that $\Gamma(z_{\exists}) = \exists X. X$ and $\Gamma \vdash M_{\exists} : A$.

Proof. Take $\Gamma_{\exists} = \{z_{\exists} : \exists X.X, z_1 : B_1^3 \rightarrow A, z_2 : \exists X.X \rightarrow B_1, z_3 : B_3\}$ where A, B_1, B_3 are arbitrary types, and observe that only the possible form of $\Gamma(z_{\exists})$ is $\exists X.X$ if M_{\exists} is typable. \square

Theorem 4.8 (TP, TIP, TCP (\rightarrow, \exists)). **th:undecidable-typefree** TP, TIP, and
5 TCP are undecidable for the system (\rightarrow, \exists) .

Proof. The flat form E is solvable iff $x_{\exists} : \exists X.X \vdash \hat{M}_E : \exists X.X$ by Proposition 4.6 iff $x_{\exists} : \exists X.X, z : \exists X.X \rightarrow Z \vdash z\hat{M}_E : B$ for some B , where z and Z are fresh variables. So the undecidability of TCP and TIP follows.

Next, suppose that $\Gamma' \vdash \lambda v.vM_{\exists}\hat{M}_E : B'$ for some Γ' and B' . Then, from Lemma
10 4.7, we have $\Gamma_{\exists} \vdash M_{\exists} : B''$ and $\Gamma_{\exists} \vdash \hat{M}_E : B'''$ for some B'' and B''' , where $\text{FV}(\hat{M}_E) = \{x_{\exists}\}$ and $\Gamma_{\exists}(x_{\exists}) = \exists X.X$. Hence, we also have $x_{\exists} : \exists X.X \vdash \hat{M}_E : \exists X.X$. The inverse direction is clear as well. Thus, the flat form E is solvable iff $\Gamma' \vdash \lambda v.vM_{\exists}\hat{M}_E : B'$ for some Γ' and B' .

Therefore, all of TCP, TIP, TP are undecidable for (\rightarrow, \exists) . \square

15 5. SIMPLE DERIVATIONS OF THE SYSTEM IN THE CURRY-STYLE

We analyse the structure of derivations in the full Curry-style system. This is further applied to TCP and TIP, and to investigate the subject reduction property. The results obtained here apply both to the system (\rightarrow, \exists) and (\neg, \wedge, \exists) .

We say that a quantifier \exists is vacuous in a type $\exists X.A$ when $X \notin \text{FV}(A)$. For
20 successive applications of substitutions, we write $A[Y_1 := A_1] \dots [Y_m := A_m]$ where $Y_i \notin \text{FV}(A_j)$ for $i \neq j$.

Definition 5.1. (Removing vacuous \exists)

- (1) $|X| = X$;
- 25 (2) $|\perp| = \perp$;
- (3) $|\neg A| = \neg|A|$;
- (4) $|A_1 \diamond A_2| = |A_1| \diamond |A_2|$ for $\diamond \in \{\rightarrow, \wedge\}$;
- (5) $|\exists X.A| = \exists X.|A|$ if $X \in \text{FV}(A)$;
- (6) $|\exists X.A| = |A|$ if $X \notin \text{FV}(A)$.

30 **Lemma 5.2.** (Basic lemma)

lemma:basic-lemma For the Curry-style systems the following statements hold.

- (1) **(Weakening):** $\Gamma, x:A \vdash N : A_1$ is derivable, where $x \notin \text{FV}(N)$, if and only if $\Gamma \vdash N : A_1$ is. Moreover, $\Gamma, x:A \vdash N : A_1$ has a derivation with no vacuous \exists , if and only if $\Gamma \vdash N : A_1$ has.
- 35 (2) **(Substitution):** If $\Gamma \vdash M : A$, then $\Gamma[X := A_1] \vdash M : A[X := A_1]$.
- (3) **(Cut):** If $\Gamma \vdash M : A$ and $\Gamma, x:A \vdash N : A_1$, then $\Gamma \vdash N[x := M] : A_1$.
- (4) **(Homomorphism):** $|A[X := B]| \equiv |A|[X := |B|]$.
- (5) **(Non-vacuous \exists):** If $\Gamma \vdash M : A$ is derivable, then $|\Gamma| \vdash M : |A|$ is derivable with no vacuous quantification in types throughout the whole derivation.

(6) (Permutation): $\Gamma \vdash M : \exists X.\exists Y.A$ if and only if $\Gamma \vdash M : \exists Y.\exists X.A$.

Proof. The proof of (1) is in both directions by induction on the derivation that exists by the assumption. The proof of (2) is by induction of the derivation that is assumed to exist. The proof of (3) is by induction on the derivation of the second judgement.

5 The proof of (4) is by induction on the structure of A .

As for the proof of (6), we can combine the following derivation:

$$\frac{\frac{\text{(assumption)}}{\Gamma \vdash M : \exists X.\exists Y.A} \quad \frac{\frac{\Gamma' \vdash x : \exists Y.A \quad \Gamma'' \vdash y : \exists Y.\exists X.A}{\Gamma' \vdash y[y := x] : \exists Y.\exists X.A} \text{(}\exists E\text{)}}{\Gamma \vdash x[x := M] : \exists Y.\exists X.A} \text{(}\exists E\text{)}}{\Gamma \vdash x[x := M] : \exists Y.\exists X.A} \text{(}\exists E\text{)}$$

where $\Gamma' = \Gamma, x : \exists Y.A$ and $\Gamma'' = \Gamma, x : \exists Y.A, y : A$, with the derivation

$$\frac{\frac{\Gamma'' \vdash y : A}{\Gamma'' \vdash y : \exists X.A} \text{(}\exists I\text{)}}{\Gamma'' \vdash y : \exists Y.\exists X.A} \text{(}\exists I\text{)}$$

to obtain the remaining derivation for $\Gamma'' \vdash y : \exists Y.\exists X.A$.

For the proof of (5) we proceed by induction on the derivation of $\Gamma \vdash A$ by cases on the last rule used. The cases (var) , $(\rightarrow E)$ and $(\rightarrow I)$ go by routine decomposition using induction hypothesis when necessary.

10 In case the last rule is $(\exists I)$, we have $|\Gamma| \vdash M : |A_1[X := A_2]|$, by induction hypothesis. The case (4) of the current lemma gives that $|A_1[X := A_2]| \equiv |A_1|[X := |A_2]|$. If $X \in \text{FV}(A_1)$ then $|\exists X.A_1| = \exists X.|A_1|$ and also $X \in \text{FV}(|A_1|)$. In that case we can use directly the $(\exists I)$ rule and obtain $|\Gamma| \vdash M : \exists X.|A_1|$ which is equal to $|\Gamma| \vdash M : |\exists X.A_1|$. In case $X \notin \text{FV}(A_1)$, we have that $|\exists X.A_1| = |A_1|$ and already
15 the result of induction hypothesis gives the derivation $|\Gamma| \vdash M : |A_1[X := A_2]|$ as $|A_1[X := A_2]| \equiv |A_1|$.

In case the last rule is $(\exists E)$, we have $|\Gamma| \vdash M_1 : |\exists X.A|$ and $|\Gamma|, x : |A| \vdash M_2 : |B|$ where $M = M_2[x := M_1]$ by induction hypothesis. If $X \in \text{FV}(A)$ then $|\exists X.A| \equiv \exists X.|A|$ and we are able to apply $(\exists E)$ using $|\Gamma| \vdash M_1 : \exists X.|A|$ and
20 $|\Gamma|, x : |A| \vdash M_2 : |B|$ as premises to obtain the required $|\Gamma| \vdash M_2[x := M_1] : |B|$.

If $X \notin \text{FV}(A)$ then $|\exists X.A| \equiv |A|$. This allows us to apply the already proved case (3) of the current lemma to obtain $|\Gamma| \vdash N[x := M] : |B|$. \square

Now, we are able to introduce a certain normal form of derivations for a fixed term M , i.e. simple derivations. We show below that once a term has a derivation it
25 has also a simple one. In this way we obtain a tool to restrict the number of possible derivations for a term to a manageable size.

Definition 5.3. (Simple derivations)

df:simple-derivations A derivation \mathcal{D} of $\Gamma \vdash M : A$ in the Curry-style system is called simple, if \mathcal{D} complies to all the restrictions:

30 (1) For each existential type $\exists X.A$ in \mathcal{D} , we have $X \in \text{FV}(A)$.

- (2) (No redundant $(\exists E)$) For each application of $(\exists E)$ in \mathcal{D} , we have $x \in \text{FV}(N)$ in:

$$\frac{\Gamma \vdash M : \exists X.A \quad \Gamma, x:A \vdash N : A_1}{\Gamma \vdash N[x := M] : A_1} (\exists E)$$

- (3) The derivation \mathcal{D} contains no application of rules $(\exists I)$, $(\exists E)$ such as:

$$\frac{\frac{\Gamma \vdash M : A[X := A_1]}{\Gamma \vdash M : \exists X.A} (\exists I) \quad \Gamma, x:A \vdash N : A_2}{\Gamma \vdash N[x := M] : A_2} (\exists E)$$

where the major premise of $(\exists E)$ is a consequence of $(\exists I)$.

- (4) The derivation \mathcal{D} contains no application of rules $(\exists E)$ such as:

$$\frac{\frac{\Gamma \vdash M : \exists X_1.A_1 \quad \Gamma, x_1:A_1 \vdash N_1 : \exists X_2.A_2}{\Gamma \vdash N_1[x_1 := M] : \exists X_2.A_2} (\exists E) \quad \Gamma, x_2:A_2 \vdash N_2 : A}{\Gamma \vdash N_2[x_2 := N_1[x_1 := M]] : A} (\exists E)$$

where the major premise of lower $(\exists E)$ is a consequence of upper $(\exists E)$.

- (5) The derivation \mathcal{D} contains no application of rules $(\exists E)$, $(\exists I)$ such as:

$$\frac{\frac{\Gamma \vdash M : \exists X.A \quad \Gamma, x:A \vdash N : A_1[Y := A_2]}{\Gamma \vdash N[x := M] : A_1[Y := A_2]} (\exists E) \quad \Gamma, x:A \vdash N : A_1[Y := A_2]}{\Gamma \vdash N[x := M] : \exists Y.A_1} (\exists I)$$

where the premise of $(\exists I)$ is a consequence of $(\exists E)$.

Proposition 5.4. (Simple derivations)

- 5 **prop:simple** For any derivation \mathcal{D} of $\Gamma \vdash M : A$ in the Curry-style system, we have a simple derivation \mathcal{D}' of $\Gamma' \vdash M : A'$ with the same term M .

Proof. First step: We obtain the condition (1) of Definition 5.3 by Lemma 5.2(5). We can now remove one by one cases forbidden in the condition (2) by using the right premise of the wrong rule $(\exists E)$ and then removing the superfluous assumption $x : A$ with help of Lemma 5.2(1).
10

Second step: We can prove that for any derivation \mathcal{D} of $\Gamma \vdash M : A$ that obeys restrictions (1) and (2), there exists a derivation \mathcal{D}' of $\Gamma \vdash M : A$ with the same term, which obeys restrictions (1)–(4), following [Pra65, And95] (or more directly by induction on the term N or N_2 respectively).

Third step: Suppose that we have a derivation which obeys (1)–(4). Then the application of rules as in case (5) of Definition 5.3 can be permuted as follows:

$$\frac{\Gamma, x:A \vdash N : A_1[Y := A_2]}{\Gamma \vdash M : \exists X.A \quad \Gamma, x:A \vdash N : \exists Y.A_1} (\exists I) \quad \Gamma, x:A \vdash N : A_1[Y := A_2]}{\Gamma \vdash N[x := M] : \exists Y.A_1} (\exists E)$$

- 15 Thus we can show the existence of simple derivations. The details are left to interested readers. \square

We introduce a notion of skeleton to ease the representation of derivations. A *skeleton of a derivation* for $\Gamma \vdash x : A$ in the Curry-style system can be represented as a type-free term: $M \equiv (\text{let } \langle \exists, a_1 \rangle = x \text{ in } \cdots \text{let } \langle \exists, a_n \rangle = a_{n-1} \text{ in } \langle \exists, \dots, \langle \exists, a_n \rangle \rangle)$ with $\|M\| = x$. This can be generalised to all terms by Curry-Howard isomorphism together with a natural type erasure $\|\cdot\|$.

Now, we can conclude with a detailed description on how the applications of $(\exists E)$ and (var) in simple derivations look like.

Corollary 5.5. (Application of $(\exists E)$, (var))
cor:corollary1

(1) *On the application of $(\exists E)$ in a simple derivation:*

$$\frac{\Gamma \vdash M : \exists X.A \quad \Gamma, x:A \vdash N : A_1}{\Gamma \vdash N[x := M] : A_1} (\exists E)$$

10 *the judgement $\Gamma \vdash M : \exists X.A$ is a consequence of an application of (var) , $(\rightarrow E)$ or $(\wedge E)$. Thus the case of $M \equiv y$ (variable) gives $\Gamma(y) = \exists X.A$, and the term M cannot be in the form of $\lambda y.M_1$ or $\langle M_1, M_2 \rangle$ for some M_1, M_2 .*

(2) *If $\Gamma \vdash x : A$, then there exists $x : \exists X_1 \dots X_n.B \in \Gamma$ ($n \geq 0$), such that $B \equiv B_0[Y_1 := B_1] \dots [Y_m := B_m]$ and $A \equiv \exists Y_1 \dots Y_m.B_0$ for some B_0, B_1, \dots, B_m ($m \geq 0$). That is, the skeleton of the derivation for $\Gamma \vdash x : A$ in the Curry-style system can be represented as a type-free term:*

$$M \equiv (\text{let } \langle \exists, a_1 \rangle = x \text{ in } \cdots \text{let } \langle \exists, a_n \rangle = a_{n-1} \text{ in } \langle \exists, \dots, \langle \exists, a_n \rangle \rangle)$$

with $\|M\| = x$.

Proof. Straightforward analysis of possible derivation shapes in case a derivation is simple. The details are left to an interested reader. \square

5.1. Immediate \exists elimination. The simple derivations must be further restricted. The point of the following restriction is to gather all the possible $(\exists E)$ that result from a variable with a type that starts with \exists right after the variable is introduced into the environment.

25 Note that we could not include the definition of immediate \exists elimination in the definition of simple derivation as we need the latter in order to properly formulate the former.

Definition 5.6 (Immediate \exists elimination). We say that a derivation has immediate \exists elimination when for each occurrence of a $(\rightarrow I)$ rule such as

$$\frac{\Gamma, x:\exists X.A_1 \vdash M : A_2}{\Gamma \vdash \lambda x.M : (\exists X.A_1) \rightarrow A_2} (\rightarrow I)$$

or a $(\exists E)$ rule such as

$$\frac{\Gamma \vdash M_1 : \exists Y.\exists X.A_1 \quad \Gamma, x:\exists X.A_1 \vdash M : A_2}{\Gamma \vdash \text{let } \langle Y, x \rangle = M_1 \text{ in } M : A_2} (\exists E)^*$$

the derivation for $\Gamma, x:\exists X.A_1 \vdash M : A_2$ contains an application of a rule of the form

$$\frac{\Gamma' \vdash x : \exists X.A_1 \quad \Gamma', x':A_1 \vdash M'' : B}{\Gamma' \vdash M' : B} (\exists E)^* \quad (5.1)$$

only immediately to $\Gamma, x:\exists X.A_1 \vdash M : A_2$.

Observe that the variable x can occur in the major premise of a $(\exists E)^*$ rule in a simple derivation only in the form (5.1), i.e. only typed to the type which is assigned by the environment Γ' . Indeed, Corollary 5.5(1) implies that only the (var) rule can be used to obtain $\Gamma' \vdash x : \exists X.A_1$ there.

We can now enjoy the following lemma:

Lemma 5.7 (Immediate \exists elimination in derivations). **lemma:immediate-exists**
For each simple derivation in the full system of a judgement $\Gamma \vdash M : A$ there is its derivation with immediate \exists elimination.

Proof. The proof here is by induction on the length of the derivation. The cases of the induction amount to the check that the rule (5.1) can be permuted with all the inference rules as long as the variable x is not bound in the conclusion of the rule to permute with. The permuted derivations are still simple. In case we have multiple rules of the form (5.1), we can use Corollary 5.5(1) to arrive at the conclusion that they use the same $x : \exists X.A_1$ so we can collapse them all into a single $(\exists E)^*$ rule. \square

In order to understand better the following proofs, we consider here how the notion of immediate \exists elimination simplifies bottom-up analysis of derivations. Examine a simple derivation snippet below, where the discharged assumption on the application of $(\rightarrow I)$ or $(\exists E)^*$ has an existential type $\exists X_1 \dots X_n.A_1$:

$$\frac{[\Gamma \vdash M_1 : \exists Y.\exists X_1 \dots \exists X_n.A_1] \quad \Gamma, x:\exists X_1 \dots \overset{\vdots}{\exists X_n}.A_1 \vdash M_2 : A_2}{\Gamma \vdash M_2' : A_2'} (\rightarrow I), (\exists E)^* \quad (5.2)$$

(the part in square brackets occurs only in the rule $(\exists E)^*$, the dots represent a further part of the derivation). In certain leafs of the whole derivation tree, we have to apply the rule (var) to introduce the variable x . This is so as $x \in \text{FV}(M_2)$ in simple derivations. Then from Corollary 5.5(2), the (var) rule is in general applied as in the following derivation fragment, where $A_1 \equiv B_0[Y_1 := B_1] \dots [Y_m := B_m]$ for some B_0, B_1, \dots, B_m ($m \geq 0$). Here we can assume that A_1 is not in the form of $\exists Y.A_1'$ for any A_1' . If A_1 were existential then we could apply more $(\exists E)$ and $(\exists I)$ to exhaust \exists

from A_1 :

$$\begin{array}{c}
\frac{\Gamma'', b_n : A_1 \vdash b_n : B_0[Y_1 := B_1] \cdots [Y_m := B_m]}{\Gamma'', b_n : A_1 \vdash b_n : \exists Y_m. B_0[Y_1 := B_1] \cdots [Y_{m-1} := B_{m-1}]} (\exists I) \\
\vdots (\exists I) \\
\frac{\Gamma'', b_n : A_1 \vdash b_n : \exists Y_2 \dots \exists Y_m. B_0[Y_1 := B_1]}{\Gamma'' \vdash b_{n-1} : \exists X_n. A_1 \quad \Gamma'', b_n : A_1 \vdash b_n : \exists Y_1 \dots \exists Y_m. B_0} (\exists I) \\
\frac{\Gamma'' \vdash b_{n-1} : \exists X_n. A_1 \quad \Gamma'', b_n : A_1 \vdash b_n : \exists Y_1 \dots \exists Y_m. B_0}{\Gamma'' \vdash b_{n-1} : \exists Y_1 \dots \exists Y_m. B_0} (\exists E)^* \\
\vdots (\exists E)^* \\
\frac{\Gamma' \vdash x : \exists \vec{X}. A_1 \quad \Gamma', b_1 : \exists X_2 \dots \exists X_n. A_1 \vdash b_1 : \exists \vec{Y}. B_0}{\Gamma' \vdash x : \exists \vec{Y}. B_0} (\exists E)^* \\
\vdots
\end{array}$$

Now, we can take a simple derivation with immediate \exists elimination. In such a derivation, all the applications of $(\exists E)^*$ above are permuted to the position only immediate to the judgement $\Gamma, x : \exists X_1 \dots \exists X_n. A_1 \vdash M : A_2$, being the premise of $(\rightarrow I)$ or $(\exists E)^*$ in (5.2).

5

6. TCP AND TIP FOR (\rightarrow, \exists) IN CURRY-STYLE

For the proof of undecidability of TC and TCP we adopt here the technique of Wells [Wel99]. Let $\{A_1 \leq B_1, A_2 \leq B_2\}$ be an instance of SUP [KTU90] built of type variables and \rightarrow (the problem is undecidable for one binary symbol and two equations [KTU90]). Let \mathcal{X} be the set of variables of the instance. The instance has a solution S if and only if $R_1(S(A_1)) = S(B_1)$ and $R_2(S(A_2)) = S(B_2)$ for some substitutions R_1, R_2 .

Let $\gamma, \gamma_1, \gamma_2, \zeta$ be fresh type variables which do not occur in \mathcal{X} . In the system of $(\rightarrow \exists)$, we shorten $A \rightarrow \zeta$ as $\neg A$.

A type in the form of $\exists X_1 \dots \exists X_k. A$ ($k \geq 1$) is called an *existential closure*, simply written by $\exists. A$, if either $\text{FV}(A) = \{X_1, \dots, X_k\}$ or $\text{FV}(A) = \{X_1, \dots, X_k, \zeta\}$ with $X_i \neq \zeta$ for each i .

In the following development, we consider only α -representants of terms such that each bound variable is bound only once and no bound variable is equal to a free one. This is possible with help of renaming of the bound variables. We call this property uniqueness of bound variables. We can establish the following immediate observation:

Observation 6.1 (Subterm condition on $(\exists E)$). **obs:subterm-condition** Assume that $x \in \text{FV}(N)$, and let $P \equiv N[x := M]$ on the application of $(\exists E)$:

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

M is a subterm P_1 of P , such that none of the free variables in M is bound in P .

For a substitution $S = [X_1 := A_1, \dots, X_n := A_n]$, we say that $\text{dom}(S) = \{X_1, \dots, X_n\}$, and we can assume $X_i \notin \text{FV}(A_j)$ for each $i \neq j$ ($1 \leq i, j \leq n$), so that we may write $S = [X_1 := A_1] \dots [X_n := A_n]$ for substitutions.

We say that a type A has *an existential type as one of the results* when it has the form $A \equiv A_1 \rightarrow \dots \rightarrow A_k \rightarrow \exists X. A_{k+1}$ for some A_1, \dots, A_{k+1} .

We reduce SUP to TCP by letting:

$$\begin{aligned} M_1 &\equiv \lambda x. \lambda z. c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) \text{ and} \\ \Gamma &= \{c : \neg \exists. \neg(\neg \neg(B_1 \rightarrow \gamma_1) \rightarrow \neg \neg(\gamma_2 \rightarrow B_2) \rightarrow A_1 \rightarrow A_2)\}. \end{aligned}$$

Theorem 6.2 (SUP \leftrightarrow TCP). **th:typechecking-red** $\{A_1 \leq B_1, A_2 \leq B_2\}$ has a solution if and only if $\Gamma \vdash M_1 : \exists \gamma. (\neg \gamma \rightarrow \neg \gamma)$ is derivable in the system (\rightarrow, \exists) .

Proof. The if part For the if part, assume that the instance $\{A_1 \leq B_1, A_2 \leq B_2\}$ of SUP has a solution, i.e., $R_i(S(A_i)) = S(B_i)$ for $i = 1, 2$. Then we have the following derivations (the reader is encouraged to reconstruct the contexts missing in the displays below):

$$\frac{x : \neg \exists. \neg(SA_1 \rightarrow SA_2) \quad \frac{u_i : \neg(R_i(SA_1) \rightarrow R_i(SA_2))}{u_i : \exists. \neg(SA_1 \rightarrow SA_2)} (\exists I)}{xu_i : \zeta} \quad \frac{}{\lambda u_i. x u_i : \neg \neg(R_i(SA_1) \rightarrow R_i(SA_2))} \quad (6.1)$$

Let $a : \neg(SA_1 \rightarrow SA_2)$, and $y : B$ together with

$B \equiv (\neg \neg(SB_1 \rightarrow R_1(SA_2)) \rightarrow \neg \neg(R_2(SA_1) \rightarrow SB_2) \rightarrow SA_1 \rightarrow SB_2)$ and

$A \equiv (\neg(\neg \neg(B_1 \rightarrow \gamma_1) \rightarrow \neg \neg(\gamma_2 \rightarrow B_2) \rightarrow A_1 \rightarrow B_2))$. Then, one has

$y(\lambda u_1. x u_1)(\lambda u_2. x u_2) : SA_1 \rightarrow SB_2$ under the solution. We continue:

$$\frac{a : \neg(SA_1 \rightarrow SA_2) \quad y(\lambda u_1. x u_1)(\lambda u_2. x u_2) : SA_1 \rightarrow SA_2}{\frac{a(y(\lambda u_1. x u_1)(\lambda u_2. x u_2)) : \zeta}{\lambda y. a(y(\lambda u_1. x u_1)(\lambda u_2. x u_2)) : \neg B} (\rightarrow I)}{c : \Gamma(c) \quad \frac{\lambda y. a(y(\lambda u_1. x u_1)(\lambda u_2. x u_2)) : \exists. \gamma_1. \gamma_2. A}{c(\lambda y. a(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \zeta} (\exists I)}{z : \exists. \neg(SA_1 \rightarrow SA_2) \quad \frac{c(\lambda y. a(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \zeta}{c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \zeta} (\rightarrow E)} (\exists E)$$

Finally, the system $(\rightarrow \exists)$ derives $\Gamma \vdash M_1 : \exists \gamma. (\neg \gamma \rightarrow \neg \gamma)$, as follows:

$$\frac{\frac{\frac{c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \zeta}{\lambda z. c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \neg \exists. \neg(SA_1 \rightarrow SA_2)} (\rightarrow I)}{\lambda x. \lambda z. c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \neg \exists. \neg(SA_1 \rightarrow SA_2) \rightarrow \neg \exists. \neg(SA_1 \rightarrow SA_2)} (\exists I)}{\lambda x. \lambda z. c(\lambda y. z(y(\lambda u_1. x u_1)(\lambda u_2. x u_2))) : \exists \gamma. (\neg \gamma \rightarrow \neg \gamma)}$$

The ‘only-if part’

For the ‘only if’ part, we assume that $\Gamma \vdash M_1 : \exists \gamma. (\neg \gamma \rightarrow \neg \gamma)$ is derivable in the system $(\rightarrow \exists)$ and investigate all of the simple derivations of the judgement.

We divide the analysis of the derivation into several stages. In the course of the analysis we define the required substitutions T, T_1, T_2 such that $T_1(T(A_1)) = T(B_1)$ and $T_2(T(A_2)) = T(B_2)$.

(1) The final rules of the derivation of $M_1 : \exists\gamma.(\neg\gamma \rightarrow \neg\gamma)$.

The final rule of the derivation cannot be $(\exists E)$. If it were such, we would obtain a contradiction in the following way. As the rule $(\exists E)$ must be applied so that $M_1 \equiv N[x_g := M]$ for some N and M , the only possibilities for M and N are (*) $M \equiv M_1$, $N \equiv x_g$ and (**) $M \equiv c$, $N \equiv M_1[c := x_g]$, by Observation 6.1. The case (*) is impossible as M_1 is a λ -abstraction which is forbidden by Corollary 5.5(1). The case (**) is impossible as then the type of c in Γ must be an existential type by Corollary 5.5(1) which is not the case.

As the type of M_1 is an existential type, the only remaining possibility is that the final rule is $(\exists I)$, where the premise is $\Gamma \vdash M_1 : \neg D \rightarrow \neg D$ for some D . Then the only possible rule which can be applied to that judgement is the $(\rightarrow I)^1$ rule as the derivation is simple (we mark the rules with superscript to refer below). The premise of the rule must be $\Gamma, x : \neg D \vdash M_2 : \neg D$ where $M_1 \equiv \lambda x.M_2$.

Now, it is again impossible to apply the $(\exists E)$ rule to the premise since we have, by the Observation 6.1, the same possibilities (*), (**) as in case of M_1 with additional (***) $M \equiv \lambda u_1.xu_1$ and (****) $M \equiv \lambda u_2.xu_2$. All these are, however, again forbidden by Corollary 5.5. Therefore, we can again apply here only the $(\rightarrow I)^2$ rule and obtain a derivation fragment:

$$\frac{\frac{\frac{\Gamma, x : \neg D, z : D \vdash c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2))) : \zeta}{\Gamma, x : \neg D \vdash \lambda z.c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2))) : \neg D} (\rightarrow I)^2}{\Gamma \vdash \lambda x.\lambda z.c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2))) : \neg D \rightarrow \neg D} (\rightarrow I)^1}{\Gamma \vdash \lambda x.\lambda z.c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2))) : \exists\gamma.(\neg\gamma \rightarrow \neg\gamma)} (\exists I)$$

We let $M_3 \equiv c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)))$ and $\Gamma_2 = \Gamma, x : \neg D, z : D$.

(2) A chain of $(\exists E)$ to derive $M_3 \equiv c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2))) : \zeta$

Note that the variable z is used as an operator in an application in M_3 . This, combined with the fact that the derivation we analyse has immediate \exists elimination, implies that $\Gamma_2 \vdash M_3 : \zeta$ is derived by a chain of the applications of $(\exists E)$ which exhausts all the initial existential quantifiers of D :

$$\frac{\frac{\frac{\Gamma_2^{n-1} \vdash a_{n-1} : \exists X_1.D_1 \quad \Gamma_2^n \vdash M_3^n : \zeta}{\Gamma_2^{n-1} \vdash M_3^{n-1} : \zeta} (\exists E)}{\vdots}}{\frac{\Gamma_2 \vdash z : \exists \vec{X}.D_1 \quad \Gamma_2^1 \vdash M_3^1[z := a_1] : \zeta}{\Gamma_2 \vdash M_3 : \zeta} (\exists E)} (6.2)$$

where D_1 does not start with \exists , $\Gamma_2^i = \Gamma_2 \cup \{a_j : \exists X_{n-j} \cdots X_1.D_1 \mid 1 \leq j \leq i\}$, $M_3^i = M_3[z := a_i]$.

We observe now that by Corollary 5.5(1) the only rule which can be used to derive $\Gamma_2^n \vdash M_3^n : \zeta$ is:

$$\frac{\Gamma_2^n \vdash c : \Gamma_2(c) \quad \Gamma_2^n \vdash M_4 : \exists.\exists\gamma_1\gamma_2.D_2}{\Gamma_2^n \vdash M_3^n : \zeta} (\rightarrow E)$$

where

$$\begin{aligned} D_2 &\equiv \neg(\neg\neg(B_1 \rightarrow \gamma_1) \rightarrow \neg\neg(\gamma_2 \rightarrow B_2) \rightarrow A_1 \rightarrow A_2) \text{ and} \\ M_4 &\equiv \lambda y.a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)). \end{aligned}$$

Let $\Gamma_3 = \Gamma_2^n$. Now, the goal for the derivation is $\Gamma_3 \vdash \lambda y.a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \exists.\exists\gamma_1\gamma_2.D_2$.

(3) A derivation for $M_4 \equiv \lambda y.a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \exists.\exists\gamma_1\gamma_2.\neg(\neg\neg(B_1 \rightarrow \gamma_1) \rightarrow \neg\neg(B_2 \rightarrow \tau_2) \rightarrow A_1 \rightarrow A_2)$

Note that we have no variable in the environment which is free in M_4 and has existential type as one of the results. This gives, by Corollary 5.5(1), that no $(\exists E)$ rule can be applied until new object variables enter the environment. This implies that the only rule to apply to obtain $\Gamma_3 \vdash M_4 : \exists.\exists\gamma_1\gamma_2.\neg(\neg\neg(B_1 \rightarrow \gamma_1) \rightarrow \neg\neg(\gamma_2 \rightarrow B_2) \rightarrow A_1 \rightarrow A_2)$ is a sequence of $(\exists I)$ rules. These define a substitution T such that $\text{dom}(T) = \text{FV}(A_1, A_2, B_1, B_2)$ and types \tilde{A}_1, \tilde{A}_2 (substituted for γ_1 and γ_2 respectively). We define now $D_3 \equiv \neg\neg(TB_1 \rightarrow \tilde{A}_1) \rightarrow \neg\neg(\tilde{A}_2 \rightarrow TB_2) \rightarrow TA_1 \rightarrow TA_2$. At the end of the sequence of $(\exists I)$, we obtain an arrow type with which we can employ the $(\rightarrow I)$ rule so that the derivation must have the shape:

$$\frac{\frac{\Gamma_3, y:D_3 \vdash a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \zeta}{\Gamma_3 \vdash \lambda y.a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \neg D_3} (\rightarrow I)}{\Gamma_3 \vdash \lambda y.a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \exists.\exists\gamma_1\gamma_2.D_2} \text{ a seq. of } (\exists I)$$

Let $\Gamma_4 = \Gamma_3, y:D_3$. Currently, the goal is

$$\Gamma_4 \vdash M_5 \equiv a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \zeta.$$

5 **(4) A derivation for $M_5 \equiv a_n(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)) : \zeta$**

As the derivation has immediate \exists elimination and the variable y is freshly introduced to the derivation, the derivation for M_5 must start with a chain of $(\exists E)$ rules which exhausts all the initial existential quantifiers of D_3 . However, D_3 does not start with quantifiers so the only rule that can be used to derive $\Gamma_4 \vdash M_5 : \zeta$ is:

$$\frac{\Gamma_4 \vdash a_n : \Gamma_4(a_n) \equiv D_1 \quad \Gamma_4 \vdash M_6 : D_4}{\Gamma_4 \vdash M_5 : \zeta} (\rightarrow E) \quad (6.3)$$

for some D_4 such that $D_1 = D_4 \rightarrow \zeta$ and $M_6 \equiv y(\lambda u_1.xu_1)(\lambda u_2.xu_2)$.

Currently, the goal is

$$\Gamma_4 \vdash M_6 \equiv y(\lambda u_1.xu_1)(\lambda u_2.xu_2) : D_4.$$

(5) A derivation for $M_6 \equiv y(\lambda u_1.xu_1)(\lambda u_2.xu_2) : D_4$

We cannot apply the $(\exists E)$ rule to obtain the term by Corollary 5.5(1). This is so as there is no individual variable v in Γ_4 which is free in M_6 and has an existential type as one of the results. Therefore, the derivation must start with two $(\rightarrow E)$ rules (written here as one):

$$\frac{\Gamma_4 \vdash y : \Gamma_5(y) \equiv D_3 \quad \Gamma_4 \vdash \lambda u_1.xu_1 : D_5 \quad \Gamma_4 \vdash \lambda u_2.xu_2 : D_6}{\Gamma_4 \vdash y(\lambda u_1.xu_1)(\lambda u_2.xu_2) : D_4} (\rightarrow E) \quad (6.4)$$

Recall that $D_3 \equiv \neg\neg(TB_1 \rightarrow \tilde{A}_1) \rightarrow \neg\neg(\tilde{A}_2 \rightarrow TB_2) \rightarrow TA_1 \rightarrow TA_2$ so $D_5 \equiv \neg\neg(TB_1 \rightarrow \tilde{A}_1)$, $D_6 \equiv \neg\neg(\tilde{A}_2 \rightarrow TB_2)$, and $D_4 \equiv TA_1 \rightarrow TB_2$.

Now, we have to analyse the derivations for the judgements $\Gamma_4 \vdash \lambda u_1.xu_1 : D_5$ and $\Gamma_4 \vdash \lambda u_2.xu_2 : D_6$.

5 **(6) A derivation for $\lambda u_1.xu_1 : D_5$**

Using an argument similar to the one in the previous step, we eliminate the possibility to apply the $(\exists E)$ rule. In this situation the only possibility which remains is the application of $(\rightarrow I)$:

$$\frac{\Gamma_4, u_1 : \neg(TB_1 \rightarrow \tilde{A}_1) \vdash xu_1 : \zeta}{\Gamma_4 \vdash \lambda u_1.xu_1 : \neg\neg(TB_1 \rightarrow \tilde{A}_1)} (\rightarrow I)$$

Now, recall that $\Gamma_4(x) = \neg D$. By the usual argument, $(\exists E)^*$ cannot be applied here. The only other possible next rule is $(\rightarrow E)$:

$$\frac{\Gamma_5 \vdash x : \neg D \quad \Gamma_5 \vdash u_1 : D}{\Gamma_5 \vdash xu_1 : \zeta} (\rightarrow E)$$

where $\Gamma_5 = \Gamma_4 \cup \{u_1 : \neg(TB_1 \rightarrow \tilde{A}_1)\}$. Recall from the analysis of the derivation fragment (6.2) that $D \equiv \exists X_n \cdots X_1.D_1$ where D_1 does not start with \exists . The further derivation of $\Gamma_5 \vdash u_1 : D$ by the (var) rule is possible only provided that $T_1(D_1) \equiv \neg(TB_1 \rightarrow \tilde{A}_1)$ for some substitution T_1 . Recall from the analysis of the derivation fragment (6.3) that $D_1 \equiv \neg D_4$. Therefore, $T_1(D_4) \equiv TB_1 \rightarrow \tilde{A}_1$ must hold. Recall, now, from the analysis of the derivation fragment (6.4) that $D_4 \equiv TA_1 \rightarrow TA_2$. Therefore, we obtain $T_1(TA_1 \rightarrow TA_2) \equiv TB_1 \rightarrow \tilde{A}_1$ and finally

$$T_1(TA_1) \equiv TB_1. \tag{6.5}$$

(6) A derivation for $\lambda u_2.xu_2 : D_6$

Using an argument similar to the one in the previous case we arrive at a substitution T_2 such that $T_2(TA_1 \rightarrow TA_2) \equiv \tilde{A}_2 \rightarrow TB_2$ and finally

$$T_2(TA_2) \equiv TB_2. \tag{6.6}$$

(7) Final recapitulation

In the analysis above, we defined substitutions T, T_1, T_2 such that $T_1(TA_1) \equiv TB_1$, by (6.5), and $T_2(TA_2) \equiv TB_2$, by (6.6). This proves that T, T_1, T_2 are solution to the
10 SUP instance $\{A_1 \leq B_1, A_2 \leq B_2\}$ which concludes the proof for the ‘only if’ part. \square

Theorem 6.3 (TCP (\rightarrow, \exists)). **th:typechecking-imp** *The TCP of the system (\rightarrow, \exists) in the Curry style is undecidable.*

The proof given to the theorem above works as well for the statement of type inference. Let $N' \equiv \lambda x.\lambda z.c(\lambda y.z(y(\lambda u_1.xu_1)(\lambda u_2.xu_2)))$ and $N \equiv bN'$, and

$$\Gamma = \{b : \neg\exists\gamma.(\neg\gamma \rightarrow \neg\gamma), c : \neg\exists.\neg(\neg\neg(B_1 \rightarrow \gamma_1) \rightarrow \neg\neg(\gamma_2 \rightarrow B_2) \rightarrow A_1 \rightarrow A_2)\}.$$

Theorem 6.4 (TIP (\rightarrow, \exists)). **th:typability-imp** $\{A_1 \leq B_1, A_2 \leq B_2\}$ *has a solution if and only if there exists some A such that $\Gamma \vdash N : A$ in the system (\rightarrow, \exists) . Therefore,*
15 TIP of the system (\rightarrow, \exists) in the Curry style is undecidable either.

Proof. It is enough to show that if there exists some A such that $\Gamma \vdash N : A$ then we have a simple derivation of $\Gamma \vdash N : \zeta$. The only ways to apply $(\exists E)$ at the end of the derivation are:

$$\frac{\Gamma \vdash N' : \exists X.B \quad \Gamma, x : B \vdash bx : A}{\Gamma \vdash bx[x := N'] : A} (\exists E) \quad \frac{\Gamma \vdash N : \exists X.B \quad \Gamma, x : B \vdash x : A}{\Gamma \vdash x[x := N] : A} (\exists E)$$

for some $\exists X.B$ as otherwise Observation 6.1 is violated. The former case is impossible by Corollary 5.5(1), since N' is in the form of λ -abstraction. In the latter case, $N : \exists X.B$ must be derived by $(\rightarrow E)$ by Corollary 5.5. Here, however, the type of N cannot be existential but ζ . This implies that $\Gamma \vdash N : A$ is in the last step derived
5 by $(\rightarrow E)$, where $A \equiv \zeta$. \square

7. TCP AND TIP ARE UNDECIDABLE IN CURRY-STYLE (\neg, \wedge, \exists)

Let $\{A_1 \leq B_1, A_2 \leq B_2\}$ be an instance of SUP built of type variables and \wedge as the binary symbol. We use for (\neg, \wedge, \exists) in the Curry style a reduction similar to the one in Theorem 6.2.

10 Note that Lemma 5.7 on derivations with immediate \exists elimination applies to the derivations in the Curry (\neg, \wedge, \exists) alone, as well.

$$\begin{aligned} \text{Let } M_1 &\equiv \lambda x.c \langle \lambda u_1.\pi_1(x)u_1, \lambda u_2.\pi_1(x)u_2, \pi_2(x) \rangle \text{ and} \\ \Gamma &= \{c : \neg\exists.(\neg\neg(B_1 \wedge \gamma_1) \wedge \neg\neg(\gamma_2 \wedge B_2) \wedge \neg(A_1 \wedge A_2))\}. \end{aligned}$$

Theorem 7.1 (TCP (\neg, \wedge, \exists)). **th:typechecking-wedge** $\{A_1 \leq B_1, A_2 \leq B_2\}$ has a solution if and only if $\Gamma \vdash M_1 : \exists\gamma.\neg(\neg\gamma \wedge \gamma)$ in the system (\neg, \wedge, \exists) .

15 *Proof.* **The if part** For the proof of the if part, assume that $R_1(S(A_1)) = S(B_1)$ and $R_2(S(A_2)) = S(B_2)$.

Let

$$\begin{aligned} \Gamma_1 &= \Gamma \cup \{x : (\neg\exists.\neg(D)) \wedge (\exists.\neg(D))\} \quad \text{where } D = SA_1 \wedge SA_2, \\ &\quad \text{and} \\ \Gamma_2 &= \Gamma_1 \cup \{a : \neg D\} \quad \text{and} \\ \Gamma_2^i &= \Gamma_2 \cup \{u_i : \neg(R_i(D))\} \quad \text{where } i = 1, 2. \end{aligned}$$

We can now derive for $i = 1, 2$:

$$\frac{\frac{\Gamma_2^i \vdash \pi_1 x : \neg\exists.\neg(SA_1 \wedge SA_2) \quad \frac{\Gamma_2^i \vdash u_i : \neg(R_i SA_1 \wedge R_i SA_2)}{\Gamma_2^i \vdash u_i : \exists.\neg(SA_1 \wedge SA_2)} (\exists I)}{\Gamma_2^i \vdash \pi_1 x u_i : \perp} (\neg E)}{\Gamma_2 \vdash \lambda u_i.\pi_1 x u_i : \neg\neg(R_i SA_1 \wedge R_i SA_2)} (\neg I) \quad (7.1)$$

It is easy now to derive by double application of $(\wedge I)$ the judgement

$$\Gamma_2 \vdash \langle \lambda u_1.\pi_1 x u_1, \lambda u_2.\pi_1 x u_2, a \rangle : \neg\neg(A_s) \wedge \neg\neg(B_s) \wedge \neg D,$$

where $A_s = SB_1 \wedge R_1 SA_2$, and $B_s = R_2 SA_1 \wedge SB_2$. It is possible to obtain $\neg\neg(A_s)$
20 at the first coordinate and $\neg\neg(B_s)$ at the second by the derivation (7.1) and since the SUP has a solution. The type $\neg D$ is obtained directly by the (var) rule.

Let $A = B_1 \wedge \gamma_2$; $B = \gamma_1 \wedge B_2$; and $C = A_1 \wedge A_2$. We can now further continue:

$$\frac{\Gamma_2 \vdash c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, a \rangle : \neg \neg A_s \wedge \neg \neg B_s \wedge \neg D \quad (\exists I)}{\frac{\Gamma_2 \vdash c : \Gamma(c) \quad \Gamma_1 \vdash \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, a \rangle : \exists. \exists \gamma_1 \gamma_2. P \quad (\neg E)}{\Gamma_2 \vdash c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, a \rangle : \perp}}$$

where $P = \neg \neg A \wedge \neg \neg B \wedge \neg C$.

This can be used as the derivation of the minor premise in:

$$\frac{\frac{\Gamma_1 \vdash \pi_2 x : \exists. \neg(D) \quad \Gamma_2 \vdash c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, a \rangle : \perp \quad (\exists E)}{\Gamma_1 \vdash c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \perp} \quad (\rightarrow I)}{\frac{\Gamma \vdash \lambda x. c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \neg(\neg \exists. \neg(D) \wedge \exists. \neg(D)) \quad (\exists I)}{\Gamma \vdash \lambda x. c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \exists \gamma. \neg(\neg \gamma \wedge \gamma)}}$$

The derivation for the major premise of the initial rule is obtained by the (*var*) rule and the ($\wedge E$) rule.

- 5 **The only-if part** For the proof of the only-if part, assume that $\Gamma \vdash M_1 : \exists \gamma. \neg(\neg \gamma \wedge \gamma)$ is derived using a simple derivation with immediate \exists elimination in the system (\neg, \wedge, \exists) .

(1) **The final rules of the derivation** $M_1 : \exists. \neg(\neg \gamma \wedge \gamma)$

Note that $\Gamma(c)$ is not an existential type, not a conjunction with an existential type in one of the conjuncts. It is impossible to use $(\exists E)^*$ as follows from Observation 6.1 (note that the observation is also valid for the current system). Therefore, $M_1 : \exists. \neg(\neg \gamma \wedge \gamma)$ can be derived only by $(\exists I)$ with γ substituted by some type D . Furthermore, the resulting premise can only be obtained by $(\neg I)$ using the same argument with $\Gamma(c)$. We obtain, therefore, a derivation:

$$\frac{\frac{\Gamma_1 \vdash c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \perp}{\Gamma \vdash \lambda x. c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \neg(\neg D \wedge D)} \quad (\neg I)}{\Gamma \vdash \lambda x. c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle : \exists. \neg(\neg \gamma \wedge \gamma)} \quad (\exists I)$$

where $\Gamma_1 = \Gamma \cup \{x : \neg D \wedge D\}$.

- 10 Let $M_2 \equiv c \langle \lambda u_1. \pi_1 x u_1, \lambda u_2. \pi_1 x u_2, \pi_2 x \rangle$. We are interested now in the form of a derivation for $\Gamma_1 \vdash M_2 : \perp$.

(2) **A chain of $(\exists E)$ to derive $\Gamma_1 \vdash M_2 : \perp$**

Now, Corollary 5.5 combined with the property that the derivations have immediate \exists elimination implies that the judgement is derived by a possibly empty chain of $(\exists E)$ rules. This is possible only if $\pi_2 x : D = \exists \vec{X}. D_1$ for some type D_1 which does not start with \exists . W.l.o.g we may assume that the eliminations exhaust \exists and the

current derivation fragment looks as follows:

$$\frac{\frac{\Gamma_1^{n-1} \vdash a_{n-1} : \exists X_1.D_1 \quad \Gamma_1^n \vdash M_2^n : \perp}{\Gamma_1^{n-1} \vdash M_2^{n-1} : \perp} (\exists E)}{\Gamma_1 \vdash \pi_2 x : \exists \vec{X}.D_1 \quad \Gamma_1^1 \vdash M_2^1 : \perp} (\exists E)}{\Gamma_1 \vdash M_2 : \perp} \quad (7.2)$$

where $\Gamma_1^i = \Gamma_1 \cup \{a_j : \exists X_{n-j} \cdots X_1.D_1 \mid 1 \leq j \leq i\}$ and $M_2^i = M_2^0[x_0 := a_i]$ with $M_2^0[x_0 := \pi_2 x] = M_2$.

Now, there is no variable in Γ_1^n which can be used in an $(\exists E)$ rule, as no variable with an existential type occurs in M_2^n . Therefore, the minor premise of the final rule must be derived using the $(\neg E)$ rule:

$$\frac{\Gamma_1^n \vdash c : \Gamma_1^n(c) \quad \Gamma_1^n \vdash \langle \lambda u_1.\pi_1 x u_1, \lambda u_2.\pi_1 x u_2, a_n \rangle : D_2}{\Gamma_1^n \vdash M_2^n : \perp} (\neg E)$$

where $D_2 \equiv \exists.\exists\gamma_1\gamma_2.(\neg\neg(B_1 \wedge \gamma_1) \wedge \neg\neg(\gamma_2 \wedge B_2) \wedge \neg(A_1 \wedge A_2))$.

Now, let $\Gamma_2 = \Gamma_1^n$ and $M_3 \equiv \langle \lambda u_1.\pi_1 x u_1, \lambda u_2.\pi_1 x u_2, a_n \rangle$.

5 Now, we analyse the possible derivations of $\Gamma_2 \vdash M_3 : D_2$.

(3) A derivation for $M_3 \equiv \langle \lambda u_1.\pi_1 x u_1, \lambda u_2.\pi_1 x u_2, a_n \rangle$:

$\exists.\exists\gamma_1\gamma_2.(\neg\neg(B_1 \wedge \gamma_1) \wedge \neg\neg(\gamma_2 \wedge B_2) \wedge \neg(A_1 \wedge A_2))$

As no new variable occurred in the environment in the previous rule and the derivation has immediate \exists elimination, the only possible step is to apply the $(\exists I)$ rules to eliminate all the existential quantifiers in the head of R_1 :

$$\frac{\Gamma_2 \vdash M_3 : D_3}{\Gamma_2 \vdash M_3 : D_2} \text{ a sequence of } (\exists I)$$

where $D_3 \equiv T(\neg\neg(B_1 \wedge \gamma_1) \wedge \neg\neg(\gamma_2 \wedge B_2) \wedge \neg(A_1 \wedge A_2))$ for some substitution T . As there is no new variable in the environment, the only possible rule which can be applied here is $(\wedge I)$ twice:

$$\frac{\Gamma_2 \vdash \lambda u_1.\pi_1 x u_1 : D_3^1 \quad \Gamma_2 \vdash \lambda u_2.\pi_1 x u_2 : D_3^2 \quad \Gamma_2 \vdash a_n : D_3^3}{\Gamma_2 \vdash \langle \lambda u_1.\pi_1 x u_1, \lambda u_2.\pi_1 x u_2, a_n \rangle : D_3} \text{ twice } (\wedge I)$$

where $D_3 = D_3^1 \wedge D_3^2 \wedge D_3^3$. We have to analyse now the derivations for a_n and $\lambda u_i.\pi_1 x u_i$ where $i = 1, 2$.

10 **(4) A derivation for $\lambda u_i.\pi_1 x u_i$ for $i = 1, 2$**

Recall that $D_3^1 = \neg\neg(TB_1 \wedge T\gamma_1)$ and that γ_1 does not occur in the original SUP instance. As D_3^1 contains no occurrence of \exists we can assume that the following derivation is the only possible derivation of the judgement $\Gamma_2 \vdash \lambda u_1.\pi_1 x u_1 : D_3^1$:

$$\frac{\frac{\Gamma_2^1 \vdash \pi_1 x : \neg D \quad \Gamma_2^1 \vdash u_1 : \exists.D_4}{\Gamma_2, u_1 : \neg(TB_1 \wedge T\gamma_1) \vdash \pi_1 x u_1 : \perp} (\neg E)}{\Gamma_2 \vdash \lambda u_1.\pi_1 x u_1 : \neg\neg(TB_1 \wedge T\gamma_1)} (\neg I)}{\Gamma_2^1 \vdash u_1 : \neg(TB_1 \wedge T\gamma_1) \equiv T_1 D_4} (\exists I)$$

where $\Gamma_2^1 = \Gamma_2 \cup \{u_1 : \neg(TB_1 \wedge T\gamma_1)\}$, T_1 is a substitution and G_1 is a type such that $D = \exists.D_4$ and

$$T_1 D_4 \equiv \neg(TB_1 \wedge T\gamma_1). \quad (7.3)$$

Similarly, we can assume that the following derivation is the only possible derivation of the judgement $\Gamma_2 \vdash \lambda u_2. \pi_1 x u_2 : D_3^2$:

$$\frac{\frac{\Gamma_2^2 \vdash \pi_1 x : \neg D}{\Gamma_2, u_2 : \neg(T\gamma_2 \wedge TB_2) \vdash \pi_1 x u_2 : \perp} (\neg E)}{\Gamma_2 \vdash \lambda u_2. \pi_1 x u_2 : \neg\neg(T\gamma_2 \wedge TB_2)} (\neg I) \quad \frac{\Gamma_2^2 \vdash u_2 : \neg(T\gamma_2 \wedge TB_2) \equiv T_2 D_5}{\Gamma_2^2 \vdash u_2 : \exists.D_5} (\exists I)$$

where $\Gamma_2^2 = \Gamma_2 \cup \{u_2 : \neg(TB_2 \wedge T\gamma_2)\}$, T_2 is a substitution and D_5 is a type such that $D = \exists.D_5$ and

$$T_2 D_5 = \neg(TB_2 \wedge T\gamma_2). \quad (7.4)$$

(5) A derivation for a_n

Recall that the goal is to derive $\Gamma_2 \vdash a_n : D_3^3$ where $D_3^3 = \neg(TA_1 \wedge TA_2)$. Note that $\Gamma_2(a_n) = D_1$ with $D = \exists \vec{X}. D_1$ by derivation (7.2). As D_1 does not start with \exists , we can obtain $\Gamma_2 \vdash a_n : D_3^3$ only by the (*var*) rule and then

$$D_1 = \neg(TA_1 \wedge TA_2). \quad (7.5)$$

(6) Final recapitulation

We observe first that $D = \exists.D_4 \equiv \exists.D_5 = \exists \vec{X}. D_1$. We also know from (7.3) and (7.4) that D_4 and D_5 do not start with \exists . This implies that $D_4 \equiv D_5 \equiv D_1 \equiv \neg(TA_1 \wedge TA_2)$ where the final equality is by (7.5).

In the analysis above, we defined substitutions T, T_1, T_2 such that $T_1(TA_1) \equiv TB_1$ by (7.3) and $T_2(TA_2) \equiv TB_2$ by (7.4). This proves that T, T_1, T_2 are a solution to the SUP instance $\{A_1 \leq B_1, A_2 \leq B_2\}$ which concludes the proof in the ‘only if’ part. \square

The same method can be applied to TIP. Let

$$\begin{aligned} N_1 &\equiv b(\lambda x. c(\lambda u_1. \pi_1(x) u_1, \lambda u_2. \pi_1(x) u_2, \pi_2(x))) \text{ and} \\ \Gamma_1 &= \{b : \neg \exists \gamma. \neg(\neg \gamma \wedge \gamma), c : \neg \exists. (\neg \neg(B_1 \wedge \gamma_1) \wedge \neg \neg(\gamma_2 \wedge B_2) \wedge \neg(A_1 \wedge A_2))\}. \end{aligned}$$

Theorem 7.2 (TIP (\neg, \wedge, \exists)). **th:typability-wedge** $\{A_1 \leq B_1, A_2 \leq B_2\}$ has a solution if and only if there exists A such that $\Gamma_1 \vdash N_1 : A$ in the system (\neg, \wedge, \exists) .

Proof. For the only-if part, assume that there exists A such that $\Gamma_1 \vdash N_1 : A$. Then we obtain $\Gamma_1 \vdash N_1 : \perp$, which follows the proof of TCP. \square

8. PREDICATIVE FRAGMENTS OF EXISTENTIAL SYSTEMS

We show that the undecidability proof methods in the previous sections still work for predicative systems with finitly stratified types. The predicative version of the systems divides the type variables into levels [Lei91]. We write $X^{(k)}$ to mark that X is in the level k . Then the types in the level 0 and k for $k > 0$ are defined as follows:

$$A^{(0)} ::= X^{(0)} \mid \perp \mid \neg A^{(0)} \mid (A^{(0)} \rightarrow A^{(0)}) \mid (A^{(0)} \wedge A^{(0)})$$

$$A^{(k+1)} ::= A^{(k)} \mid X^{(k+1)} \mid \neg A^{(k+1)} \mid (A^{(k+1)} \rightarrow A^{(k+1)}) \mid (A^{(k+1)} \wedge A^{(k+1)}) \mid \exists X^{(k)}.A^{(k+1)}$$

The minimum level of a type A is denoted as $\text{level}(A)$. The inference rules of the predicative type-free system are as usual except for the rules of the quantification:

$$\frac{\Gamma \vdash M : A[X^{(k)} := B] \quad \text{level}(B) \leq k}{\Gamma \vdash \langle \exists, M \rangle : \exists X^{(k)}.A} (\exists I)$$

$$\frac{\Gamma \vdash M_1 : \exists X^{(k)}.A^{(k+1)} \quad \Gamma, x : A \vdash M_2 : A_1^{(k+1)}}{\Gamma \vdash \text{let } \langle \exists, x \rangle = M_1 \text{ in } M_2 : A_1^{(k+1)}} (\exists E)^*$$

And the rules of the predicative Curry system are also defined for stratified types:

$$\frac{\Gamma \vdash M : A[X^{(k)} := B] \quad \text{level}(B) \leq k}{\Gamma \vdash M : \exists X^{(k)}.A} (\exists I)$$

$$\frac{\Gamma \vdash M_1 : \exists X^{(k)}.A^{(k+1)} \quad \Gamma, x : A \vdash M_2 : A_1^{(k+1)}}{\Gamma \vdash M_2[x := M_1] : A_1^{(k+1)}} (\exists E)^*$$

Now expressions for unification problems can be redefined from type variables with level k , and then results in section 4 and 6 still hold for predicative type-free and Curry systems, respectively.

Proposition 8.1. (Predicative type-free system)

prop:Predicative-TF Let $E^{(k)}$ be the equations in the flat form with level k . The problem $E^{(k)}$ is solvable if and only if $x_{\exists} : \exists X^{(k)}.X^{(k)} \vdash \hat{M}_E : \exists X^{(k+1)}.X^{(k+1)}$.

Proof. From the proof of Proposition 4.6, the proof works as well for the predicative type-free system. □

Theorem 8.2. (TCP, TIP for predicative type-free (\rightarrow, \exists))

th:TCP-TIPforTF TCP and TIP are undecidable at level 2 in the predicative type-free system. □

Proof. From Proposition 8.1, we set $k = 0$ so that TCP and TIP are undecidable at level 2 in the predicative type-free system. □

Lemma 8.3. (Predicative enforcing)

lem:predicative-exists Let $M_{\exists} = z_1(z_2 z_{\exists})(z_2(\exists, \lambda x.x))(z_2(\exists^2, z_3))$, where z_1, z_2, z_3 , and z_{\exists} are fresh terms variables. If M_{\exists} is typable in the predicative type-free system, where the derivation contains types only in level $(k+2)$ with $k \geq 0$, then for all types

5 $A^{(k+2)}, B_1^{(k+2)}, B_3^{(k)}$, there exists a context Γ_{\exists} , such that

$$\begin{aligned} \Gamma_{\exists}(z_{\exists}) &= \exists X^{(k+1)}.X^{(k+1)}, \\ \Gamma_{\exists}(z_1) &= B_1^{(k+2)} \rightarrow B_1^{(k+2)} \rightarrow B_1^{(k+2)} \rightarrow A^{(k+2)}, \\ \Gamma_{\exists}(z_2) &= \exists X^{(k+1)}.X^{(k+1)} \rightarrow B_1^{(k+2)}, \text{ and} \\ \Gamma_{\exists}(z_3) &= B_3^{(k)}. \end{aligned}$$

10 *Proof.* Observe that M_{\exists} is typable in the predicative system only when $\Gamma_{\exists}(z_{\exists}) = \exists X^{(l)}.X^{(l)}$ for some $l \geq 1$. The subterms $z_1(\exists, \lambda x.x)$ and $z_1(\exists^2, z_3)$ enforce the shape $\exists X^{(k+1)}.X^{(k+1)}$ on type of x_{\exists} , if $z_3 : B_3^{(k)}$ and $\lambda x.x : B_2^{(k)} \rightarrow B_2^{(k)}$. For types of other free variables, the analysis is straightforward. \square

Theorem 8.4. (TP for predicative type-free (\rightarrow, \exists))

15 **th:TPforTF** TP is undecidable at level 3 in the predicative type-free system.

Proof. Suppose that $\Gamma' \vdash \lambda v.v M_{\exists} \hat{M}_E : B'$ for some Γ' and B' , where the derivation contains types only in level $(k+2)$. Then, from Lemma 8.3, we have $\Gamma_{\exists} \vdash M_{\exists} : B''$ and $\Gamma_{\exists} \vdash \hat{M}_E : B'''$ for some B'' and B''' with level $(k+2)$ in the predicative system, where $\Gamma_{\exists}(x_{\exists}) = \exists X^{(k+1)}.X^{(k+1)}$. Hence, we also have $x_{\exists} : \exists X^{(k+1)}.X^{(k+1)} \vdash \hat{M}_E :$
20 $\exists X^{(k+2)}.X^{(k+2)}$ with level $(k+3)$ types. The inverse direction is clear. Thus, from Proposition 8.1, the problem $E^{(k+1)}$ is solvable if and only if $x_{\exists} : \exists X^{(k+1)}.X^{(k+1)} \vdash \hat{M}_E : \exists X^{(k+2)}.X^{(k+2)}$. Finally, we set $k = 0$ to obtain that TP is undecidable at level 3 in the predicative system. \square

For the predicative Curry system, we consider substitutions of the semi-unification
25 problem, under which levels are closed such as $[X^{(k)} := A^{(k)}]$.

Let M_1 be the term M_1 in Theorem 6.4, and M'_1 be the term M_1 in Theorem 7.1.

Proposition 8.5. (Predicative Curry system)

prop:predicative-Curry $\{A_1^{(k)} \leq B_1^{(k)}, A_2^{(k)} \leq B_2^{(k)}\}$ has a solution if and only if $\Gamma \vdash M_1 : \exists \gamma^{(k+1)}.(\neg \gamma^{(k+1)} \rightarrow \neg \gamma^{(k+1)})$ is derivable in the predicative system (\rightarrow, \exists) .

30 $\{A_1^{(k)} \leq B_1^{(k)}, A_2^{(k)} \leq B_2^{(k)}\}$ has a solution if and only if $\Gamma \vdash M'_1 : \exists \gamma^{(k+1)}. \neg(\neg \gamma^{(k+1)} \wedge \gamma^{(k+1)})$ in the predicative system (\neg, \wedge, \exists) .

Proof. The proofs of Theorem 6.4 and Theorem 7.1, respectively work even for the predicative Curry systems. \square

Theorem 8.6. (Predicative Curry $(\rightarrow, \exists), (\neg, \wedge, \exists)$)

35 **th:TCP-TIPforCurry** TCP and TIP are undecidable at level 2 in the predicative Curry-style systems.

Proof. From Proposition 8.5, we set $k = 0$ so that TCP and TIP are undecidable at level 2 in the predicative Curry systems. \square

9. SUBJECT REDUCTION PROPERTY FOR CURRY SYSTEMS

According to the syntax directed property, the type-free system enjoys the subject reduction property. However, it is known [SU06] that the subject reduction property is broken in the Curry system with both \forall and \exists . Prop. 5.4 is applied to the analysis of the subject reduction property as well. We show a stronger result that the system (\rightarrow, \exists) does not enjoy the subject reduction property. Let $I \equiv \lambda x.x$. We use below implicitly the following lemma:

Lemma 9.1. (identity)

If $\Gamma \vdash I : T$ then:

- $T = \exists \vec{X} Y \vec{Z}. Y$, or
- $T = \exists \vec{X}. T_1 \rightarrow T_2$, or
- $T = T_1 \rightarrow T_1$ for some T_1 .

Proof. The proof is by analysis of all possible derivations. The details are left to an interested reader. \square

Proposition 9.2. (Subject reduction of β)

prop:subject-beta *The subject reduction property of β -reduction is broken with respect to the existential system (\rightarrow, \exists) in Curry-style. Let Γ be $\{f : Z \rightarrow \exists X.(X \rightarrow X), z : Z\}$. Then $\Gamma \vdash \lambda x.(Ifz)(Ifzx) : \exists X.(X \rightarrow X)$ but $\Gamma \not\vdash \lambda x.(fz)(Ifzx) : \exists X.(X \rightarrow X)$.*

Proof. We can derive $\Gamma \vdash \lambda x.(Ifz)(Ifzx) : \exists X.(X \rightarrow X)$ by using $(\exists E)$ rule with major premise $\Gamma \vdash Ifz : \exists Y.(Y \rightarrow Y)$ and the minor premise $\Gamma, y : (Y \rightarrow Y) \vdash \lambda x.y(yx) : \exists X.(X \rightarrow X)$. The derivation for $\Gamma \vdash Ifz : \exists Y.(Y \rightarrow Y)$ is done by the rules which directly result from the structure of the term Ifz . The judgement $\Gamma, y : (Y \rightarrow Y) \vdash \lambda x.y(yx) : \exists X.(X \rightarrow X)$ can be derived using the $(\exists I)$ rule followed by the rules which directly result from the structure of $\lambda x.y(yx)$.

By Lemma 5.7, it is enough to show that no simple derivation with immediate \exists elimination can derive $\Gamma \vdash \lambda x.fz(Ifzx) : \exists X.(X \rightarrow X)$. As the derivation has immediate \exists elimination, it must start with $(\exists E)$ rules. As a variable bound in the resulting term cannot occur in the term in the major premise as a free one, the only possible major premises must derive existential types for the subterms of $\lambda x.fz(Ifzx)$: (1) f (2) z (3) I (4) fz (5) If (6) Ifz (7) $\lambda x.fz(Ifzx)$. By Corollary 5.5 the cases (1)–(3), (7) are impossible. Note now, that the applications of the $(\exists E)$ rule to subsequent minor premises can be permuted. Therefore, we may assume that the outermost application of $(\exists E)$ uses fz in the major premise, the one deeper If in the major premise and the deepest one Ifz in the major premise.

Note now, that the ordering of $(\exists E)$ we introduced here implies that fz cannot be used in the major premise of $(\exists E)$ rule in cases (5), (6). This, however, leads immediately to contradiction as fz can only be typed to an existential type which cannot be used in the major premise of the $(\rightarrow E)$ rule which must be used to derive a type for $fz(Ifzx)$. Therefore, we have to consider further the case (4) only.

In case (4): $\lambda x.fz(Ifzx)$ must be derived by $(\exists E)$

$$\frac{\Gamma \vdash fz : T_1 \quad \Gamma, a_1 : X_1 \rightarrow X_1 \vdash \lambda x.a_1(Ifzx) : T_1}{\Gamma \vdash \lambda x.fz(Ifzx) : T_1} (\exists E)$$

where $T_1 = \exists X.(X \rightarrow X)$. Let $\Gamma_1 = \Gamma \cup \{a_1 : X_1 \rightarrow X_1\}$. In order to obtain the derivation for the minor premise, we can use further an $(\exists E)$ rule. Using a reasoning as above we can limit the possible uses so that the minor premise derives a type for (a) If or (b) Ifz . These cases are analysed below.

5 In case no further $(\exists E)$ rule is applied, we must derive at some point $\Gamma_1, x : U_1 \vdash Ifz : U_2$ for some U_1 and $U_2 = \exists Y.(Y \rightarrow Y)$. The latter, however, implies that Ifz cannot be applied to x as U_2 is not an arrow type.

In case (4a): $\Gamma_1 \vdash \lambda x.a_1(Ifzx) : T_1$ is derived by $(\exists E)$:

$$\frac{\Gamma_1 \vdash If : \exists Y_1.Y_1 \rightarrow T_1 \quad \Gamma_1, a_2 : Y_1 \rightarrow T_1 \vdash \lambda x.a_1(a_2zx) : T_1}{\Gamma_1 \vdash \lambda x.a_1(Ifzx) : T_1} (\exists E)$$

Note now, that the derivation must at some point derive $\Gamma_1, a_2 : Y_1 \rightarrow T_1 \vdash a_2z : U_1$ for some U_1 . This is, however, not possible as $Y_1 \neq Z$.

In case (4b): $\Gamma_1 \vdash \lambda x.a_1(Ifzx) : T_1$ is derived by $(\exists E)$:

$$\frac{\Gamma_1 \vdash Ifz : \exists Y_1.T_2 \quad \Gamma_1, a_2 : T_2 \vdash \lambda x.a_1(a_2x) : T_1}{\Gamma_1 \vdash \lambda x.a_1(Ifzx) : T_1} (\exists E)$$

10 Note that $\Gamma_1 \vdash Ifz : \exists Y_1.T_2$ can only be derived by $(\rightarrow E)$ rule by Corollary 5.5. Therefore, we must derive $\Gamma_1 \vdash If : T_3 \rightarrow \exists Y_1.T_2$ and $\Gamma_1 \vdash z : T_3$ where $T_3 = Z$ by Corollary 5.5. We can exclude the possibility to apply the $(\exists E)$ rule as in the reasoning above (cases (1), (3)). Therefore, $\Gamma_1 \vdash If : Z \rightarrow \exists Y_1.T_2$ must be derived using the $(\rightarrow E)$ rule. This, however, means that $T_2 = Y_1 \rightarrow Y_1$. This, however, excludes the possibility to
15 derive $\Gamma_1, a_2 : T_2 \vdash \lambda x.a_1(a_2x) : T_1$ as the return type of a_2 does not match the argument type of a_1 .

□

Proposition 9.3. (Subject reduction of η)

20 **prop:subject-eta** *The subject reduction property of η -reduction is broken with respect to (\rightarrow, \exists) or (\neg, \exists) .*

Proof. Let $f : \exists X.X \rightarrow W$. Then take $\lambda a.fa : Z \rightarrow W$. Let $g : \neg \exists X.X$, and then take $\lambda a.ga : \neg Z$. Neither $f : (\exists X.X) \rightarrow W \vdash f : Z \rightarrow W$ nor $g : \neg \exists X.X \vdash g : \neg Z$ can be derived in the Curry system by Corollary 5.5(2). □

In this paper, we consider type-free and Curry-style terms. The systems differ in how much of a derivation is preserved in a term — the Curry-style terms omit the information from the rules $(\exists E)$ and $(\exists I)$ while the type-free terms mark applications of the rules, but they omit the types used. This makes a considerable difference as far

as derivation reconstruction problem are concerned. In case of a Curry-style term, we are forced to consider a potentially infinite number of \exists introduction rules (as in the rule $(\exists I)$ from the derivation (6.1) in the proof of Theorem 6.2), while in a type-free term the number of the introduction rules is determined to be the number of occurrences of $\langle \exists, \cdot \rangle$ construct (as in the terms of the first line in the term presented as (4.1) in Definition 4.5).

A solution of a semiunification inequality $X \leq A$ where X is a variable is a pair of substitutions R, S such that $R(S(X)) = S(A)$. The formulation of the problem does not restrict the domain of R in any way. Therefore, this problem matches well the situation we have in the Curry-style system. Still, the unification of the equations in the flat form in its instance $\mathbf{F}X_1 \dots X_n \doteq A$, where X_1, \dots, X_n are unique in the set of equations, requires a fixed number of ‘additions’ to the variable \mathbf{F} . However, the equation in the flat form can be seen as a semiunification inequality in a variant of semiunification where an additional restriction on the substitution R is imposed to have domain of size n . It is indeed so as the variables X_1, \dots, X_n are unique in the whole set of equations.

The difference between the case with the potentially infinite domain of R and with a domain of a fixed size is considerable as the original semiunification problem enjoys the most general solution property while the semiunification with restricted domain of R , as well as the second-order unification, does not. Therefore, it is difficult to devise a direct translation between the two problems and no such translation is known now.

In [FS00], we have studied the type related problems of other systems between Church-style and Curry-style. One of them is known as the domain-free style, and the type inference problem had been shown undecidable for the predicative fragment of domain-free $\lambda 2$, called domain-free ML. For this, we reduced the second-order unification problem for simple instances. However, the same reduction method cannot be applied to the problem of systems in the type-free style, since the previous method essentially refers to type information which is to be erased in the type-free case.

REFERENCES

- [And95] Y. Andou. A normalization-procedure for the first order classical natural deduction with full logical symbols. *Tsukuba Journal of Mathematics*, 19(1):153–162, 1995.
- [Bar93] H. Barendregt. *Handbook of Logic in Computer Science*, volume II, chapter Lambda Calculi with Types. Oxford University Press, USA, 1993.
- [Boe85] H.-J. Boehm. Partial polymorphic type inference is undecidable. In *26th Annual Symposium on Foundations of Computer Science*, pages 339–345. IEEE, October 1985.
- [BS00] Gilles Barthe and Morten Heine Sørensen. Domain-free pure type systems. *J. Funct. Program.*, 10(5):417–452, 2000.
- [eFS09] K. Fujita and A. Schubert. Existential type systems with no types in terms. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings*, volume 5608 of *LNCS*, pages 112–126. Springer-Verlag, 2009.

- [FS00] K. Fujita and A. Schubert. Partially typed terms between Church-style and Curry-style. In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings*, number 1872 in LNCS, pages 505–520, 2000.
- 5 [Fuj10] K. Fujita. CPS-translation as adjoint. *Theoretical Computer Science*, 411: 324–340 (2010).
- [Fuj05] K. Fujita. Galois embedding from polymorphic types into existential types. In Pawel Urzyczyn, editor, *Typed Lambda Calculi and Applications, 7th International Conference, TLCA 2005, Nara, Japan, April 21-23, 2005, Proceedings*, number 3461 in LNCS, pages 194–208, 2005.
- 10 [Has06] Masahito Hasegawa. Relational parametricity and control. *LMCS*, 2(3:3), 1–22, 2006.
- [KN09] Y. Kato and K. Nakazawa. Type checking and inference are equivalent in lambda calculi with existential types. In *WFLP '09: 18th International Workshop on Functional and (Constraint) Logic Programming*, page 31–44, 2009.
- 15 [KTU90] A. J. Kfoury, J. Tiuryn, and P. Urzyczyn. The undecidability of the semi-unification problem. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 468–476, New York, NY, USA, 1990. ACM.
- [KTU93] A. J. Kfoury, J. Tiuryn, and P. Urzyczyn. Type reconstruction in the presence of polymorphic recursion. *ACM Trans. Program. Lang. Syst.*, 15(2):290–311, 1993.
- 20 [Lei91] D. Leivant. Finitely stratified polymorphism. *Information and Computation*, 93 (1): 93–113, 1991.
- [Mai90] Harry G. Mairson. Deciding ML typability is complete for deterministic exponential time. In *POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 382–401, New York, NY, USA, 1990. ACM.
- 25 [MP88] John C. Mitchell and Gordon D. Plotkin. Abstract types have existential type. *ACM Trans. Program. Lang. Syst.*, 10(3):470–502, 1988.
- [NTKN08] K. Nakazawa, M. Tatsuta, Y. Kameyama, and H. Nakano. Undecidability of type-checking in domain-free typed lambda-calculi with existence. In *CSL '08: Proceedings of the 22nd International Workshop on Computer Science Logic*, number 5213 in LNCS, pages 478–492, Berlin, Heidelberg, 2008. Springer-Verlag.
- 30 [Pfe93] F. Pfenning. On the undecidability of partial polymorphic type reconstruction. *Fundamenta Informaticae*, 19(1,2):185–199, September-October 1993.
- [Pra65] D. Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.
- 35 [Sch98] Aleksey Schubert. Second-order unification and type inference for Church-style polymorphism. In *POPL '98: Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 279–288, New York, NY, USA, 1998. ACM.
- [SU06] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism, Volume 149 (Studies in Logic and the Foundations of Mathematics)*. Elsevier Science Inc., New York, NY, USA, 2006.
- 40 [Wel99] J. B. Wells. Typability and type checking in system F are equivalent and undecidable. *Ann. Pure Appl. Logic*, 98(1–3):111–156, 1999.